Alma Mater Studiorum - University of Bologna

PhD in Control System Engineering and Operational Research
Cycle XIV

Settore concorsuale di afferenza: 01/A6 - RICERCA OPERATIVA

Settore scientifico disciplinare: MAT/09 - RICERCA OPERATIVA

Exact Algorithms for Different Classes of Vehicle Routing Problems

Roberto Roberti

Coordinator
Prof. Alberto Caprara

Advisors
Prof. Aristide Mingozzi
Prof. Paolo Toth

Final Exam 2012

Contents

Keywords

- *Vehicle Routing Problems*
- *Traveling Salesman Problems*
- *Time Windows*
- *Exact Algorithms*
- *Column Generation*
- *Dynamic Programming*
- *State-Space Relaxation*
- *Dual-Ascent Heuristics*

Ai miei Genitori

Acknowledgements

I want to express my deeply-felt thanks to my thesis advisors, Professors Aristide Mingozzi and Paolo Toth, for their warm encouragement, thoughtful guidance, and insightful criticism.

I also want to express my gratitude to Professor Roberto Baldacci for his continuous support and sound advice.

Lastly, I wish to thank Dr. Enrico Bartolini, Dr. Valentina Cacchiani, Professor Alberto Caprara, and Professor Roberto Wolfler Calvo.

Chapter 1

Introduction

1.1 On Logistics Problems

Logistics is defined by Eilon and Christofides [1971] as "*the provision of goods and services from a supply point to a demand point*". Despite such a short definition, logistics systems span a large spectrum of decision-making activities. Examples of such activities (see Simchi-Levi et al. [2005]) are as follows:

- **Truck Routing.** A truck must leave a warehouse to deliver products to a set of retailers. The order in which the retailers are visited will determine how long the delivery will take and at what time the vehicle can return to the warehouse. Thus, the vehicle must follow an efficient route.
- **Vehicle Fleet Management.** A warehouse must supply products to a set of retailers using a fleet of vehicles of limited capacity. A dispatcher is in charge of assigning loads to vehicles and determining vehicle routes. Firstly, the dispatcher must decide how to partition the retailers into groups that can be feasibly served by a vehicle (i.e., whose total loads fit in a vehicle). Secondly, the dispatcher must decide what sequence to use so as to minimize the total traveled distance of the vehicles. Therefore, the warehouse incurs a cost that depends on both the assignment of the retailers to the vehicles and the routes performed by the vehicles. Both aspects must be considered simultaneously.
- **Network Configuration.** Several plants produce products to serve a set of retailers. Due, for example, to changes in demand patterns or in plant production or in supply costs, the current set of facilities (i.e., plants and warehouses) is deemed to be inappropriate, and management wants to reorganize the distribution network. Therefore, the management must choose a set of facility locations and capacities, determine production levels for each product at each plant, and

set transportation flows from plants to warehouses and from warehouses to retailers, in such a way that total production, inventory and transportation costs are minimized and service level requirements are met.

The previous three activities are examples of problems that have been intensively studied in the field of operations research in the last few decades.

The first problem, indicated as truck routing, is known as *traveling salesman problem* (TSP). The TSP is to find a routing of a salesman who starts from a home location, visits a prescribed set of cities, and returns to the original location, in such a way that the total distance traveled is minimum and each city is visited exactly once.

A formal definition of the TSP is the following. Let $G = (V, E)$ be a complete undirected graph, where V is the vertex set and E is the edge set. A cost is associated with each edge $e \in E$. Let $\mathscr{H}$ be the family of all Hamiltonian cycles (tours) in graph G, where each tour of the set $\mathscr{H}$ has a cost equal to the sum of the costs of the edges traversed. The TSP is to find a tour of the set $\mathscr{H}$ of minimum cost.

The first work on the TSP is commonly owed to Menger [1932]. Various papers dealing with the TSP were published in the years following, but Robinson [1949] was the first to use the name "traveling salesman problem". An intense study of the TSP started with the seminal paper by Dantzig et al. [1954]. Since then, numerous survey papers and books were published on the TSP and its variants (see Lawler et al. [1985], Reinelt [1994], Gutin and Punnen [2002], and Applegate et al. [2007]). Garey and Johnson [1979] showed that the TSP is $\mathcal{NP}$-hard.

The fields of applications of TSP go beyond the route planning problem of a traveling salesman and spans over several areas from mathematics to computer science, from genetics to electronics. Possible applications of the problem are computer wiring (Lenstra and Rinnooy Kan [1975]), wallpaper cutting (Garfinkel [1977]), dartboard design (Eiselt and Laporte [1991]), crystallography (Bland and Shallcross [1989]), machine scheduling problems, cellular manufacturing, arc routing problems, frequency assignment problems, structuring of matrices, and many more (see Gutin and Punnen [2002] for an exhaustive coverage).

The second problem, referred to as vehicle fleet management, is an example of *capacitated vehicle routing problem* (CVRP), which can be stated as follows. A set of customers, each with a known location and a known requirement for some commodity, must be supplied from a single depot by a fleet of capacitated vehicles. The costs for traveling between each couple of customers and between the depot and each customer are known. The problem is to design the vehicle routes such that each customer is served by a single vehicle, the vehicle capacity is not violated in any of the routes, and the total routing cost is minimized.

The CVRP can be formally described as follows. Let $G = (V', E)$ be an undirected graph, where $V' = \{0, 1, \ldots, n\}$ is the vertex set and E is the edge set. Vertices 1 to n correspond to a set of customers, whereas vertex 0 corresponds to the depot. A nonnegative cost is associated with each edge $e \in E$ and represents the travel cost spent to travel between the two vertices linked by edge e. A set of m identical vehicles, each with capacity Q, is available at the depot. A nonnegative demand is associated with each customer. The CVRP consists of finding a collection of m simple cycles (each corresponding to a vehicle route) of minimum total cost, defined as the sum of the costs of the edges belonging to the cycles, such that each cycle visits the depot, each customer is visited by exactly one cycle, and the sum of the demands of the vertices visited by a cycle does not exceed the vehicle capacity, Q.

The CVRP was first introduced, as the *truck dispatching problem*, by Dantzig and Ramser [1959], who proposed the first mathematical formulation and algorithm approach for the solution of the problem. The first heuristic approach for the CVRP is owed to Clarke and Wright [1964], who proposed an effective greedy heuristic. After these two seminal papers, a lot of effort has been made to model and solve the CVRP and tens of its variants. A number of books dealing with the CVRP have been published, see Eilon and Christofides [1971], Christofides [1985], Golden and Assad [1988], Fischer [1995], Crainic and Laporte [1998], Toth and Vigo [2002], and Golden et al. [2008]. Being a generalization of the TSP, the CVRP is an $\mathcal{NP}$-hard problem.

The CVRP models real-life problems encountered in the physical distribution of goods and appears in many practical situations, such as the collection of mail from mailboxes, the pickup of children by school buses, house-call tour by a doctor, preventive maintenance inspection tours, the delivery of laundry, etc.

The third problem, referred to as network configuration, is commonly known as *uncapacitated facility location problem* (UFL). A set of possible locations at which facilities (such as warehouses) may be built are given. For each location, the cost incurred to build the facility is known. The facilities must be built in order to serve a set of customer locations (such as stores). The cost of serving a customer location from each possible facility location is given. The problem is to decide the locations at which to build facilities and to assign each customer to a facility in such a way that the total cost for building facilities and serving customers is minimum.

The earliest works on the UFL problem date back to the 50's and 60's (see Koopmans and Beckmann [1957], Kuehn and Hamburger [1963], Manne [1964], Efroymson and Ray [1966]). Since then, several books were published on the UFL and other variants of location problems, see Love et al. [1988], Mirchandani and Francis [1990], Drezner [1995], Daskin [1995], Farahani and Hekmatfar [2009], Eiselt and Marianov [2011]. It can easily be shown that the UFL problem is $\mathcal{NP}$-hard.

The UFL problem has several applications, such as bank account allocation, clustering analysis, lock-box location, economic lot sizing, machine scheduling, design of communication networks, and portfolio management. The problem also appears as a subproblem in several contexts, for example network design, vehicle routing, and, of course, location theory with additional constraints.

In this thesis, we deal with some variants of the three aforementioned problems (TSP, VRP, and UFL). We study the mathematical formulations proposed in the literature and review the state-of-the-art exact algorithms presented so far. Foremost, we propose new models for some of the problems considered and develop new exact algorithms that are highly competitive with the other exact algorithms from the literature. The new algorithms proposed share common ingredients: *column generation* (see Desaulniers et al. [2005]), *dynamic programming* (see Bellman [1957], Bertsekas [1995], Bertsekas [2000], Art and Mauch [2007]), and *state-space relaxation* (see Christofides et al. [1981c], Christofides et al. [1981b]).

1.2 Reasons for Research

In the last few decades, optimization packages, based on operations research and mathematical programming techniques, have been used more and more to support managerial decisions in distribution systems. This has been possible for the development of both computer systems (hardware and software) and modeling and algorithmic tools to model and solve real-world instances addressed in the field of logistics.

Today's global markets put enterprises in fierce competition and motivates the continuous evolution of the management of distribution systems.

The study of efficient and innovative methods for supporting managers in taking decisions in all levels (strategic, tactical or operational) is justified by the economical impact of such problems on the firms' revenues. LaLonde and Zinszer [1976] estimated that distribution costs account for around 10% of the all firms' revenues. The Institute of Logistics and Distribution Management [1985] calculated that distribution costs represent more than 45% of the total logistics costs, and, in specific cases, like in the soft drink industry, they represent approximately 70% of the value added costs of goods (see Golden and Wasil [1987]). Recent studies (Eurostat [2009, 2011]) by the statistical office of the European Community (EUROSTAT) estimated that 4,140 billion tonnes of goods per kilometer were moved in 2006, 46% of which were on roads involving 32.2 million vehicles. Therefore, even small savings in distribution costs may have a relevant global impact.

Another crucial factor that justifies the study of methods to support managerial decisions is the environmental impact of the activity of transporting goods. A total GHG (i.e., greenhouse gas) emission of 992.3 million tonnes of CO_2 was attributable to transport in the EU-27 in 2006 (see Eurostat [2009]). Indeed, transport made up a share of 19% of total GHG emissions in the EU-27 - the second largest after the energy industries (31%). Not surprisingly, road transport is the principal polluter in the transport sector, contributing 93.1% to the greenhouse gas emissions. Thus, a smart and sustainable transport policy all over EU-27 is fundamental to ensure that the transport systems meet the citizen's economic, social and environmental needs.

1.3 Contents and Contributions of the Thesis

In this thesis, we deal with five variants of the three problems (TSP, VRP, and UFL) described in §1.1:

- *asymmetric traveling salesman problem*,
- *traveling salesman problem with time windows*,
- *vehicle routing problem with time windows*,
- *multi-trip vehicle routing problem*,
- *two-echelon capacitated vehicle routing problem.*

1.3.1 Asymmetric Traveling Salesman Problem

Many variants of the TSP have been proposed in the literature. Among them, the asymmetric traveling salesman problem (ATSP) is one of the most studied. The ATSP generalizes the TSP by dropping the assumption that the travel distance between two cities is the same in both directions.

Subsuming the TSP as a special case, the ATSP is $\mathcal{NP}$-hard, too.

The ATSP can be formally stated as follows. A complete directed graph $G = (V, A)$ is given, where V and A are the vertex set and arc set, respectively. A cost is associated with each arc $(i, j) \in A$. Let $\mathscr{H}$ be the family of all tours in G, where each tour has a cost equal to the sum of the costs of the traversed arcs. The ATSP is to find a tour of the set $\mathscr{H}$ of minimum cost.

There are many important real-life problems that are naturally modeled as ATSPs. In industrial scheduling, the optimal sequencing of jobs on machines with setup times is an ATSP: more generally, the optimal ordering of any set of tasks or operations with sequence dependent changeover costs is an ATSP or one of its generalizations.

Chapter 2 surveys the most effective mathematical models and exact algorithms proposed for solving the ATSP to optimality. Starting from the fundamental integer linear programming model of Dantzig et al. [1954], classical relaxations (i.e., assignment, shortest spanning r-arborescence, linear programming) are derived. The most effective branch-and-bound and branch-and-cut exact algorithms from the literature are described. A review of the polynomial formulations and a theoretical comparison of their linear relaxations are provided. The 3-node and 2-node transformations of ATSP instances into symmetric TSP instances are also described. Finally, the considered exact algorithms are experimentally compared on a set of benchmark instances.

1.3.2 Traveling Salesman Problem with Time Windows

The traveling salesman problem with time windows (TSPTW) is another well-studied generalization of the TSP, where, other than the travel distance of traveling between two cities, the travel time is known and each city must be visited within a given time interval (*time window*). Throughout the thesis, whenever we talk of TSPTW, we make no assumptions on the symmetry of travel distances and travel times between cities, so we always consider the general case of asymmetric TSPTW.

Being a generalization of the TSP, the TSPTW is $\mathcal{NP}$-hard, too. Indeed, finding a feasible solution of asymmetric TSPTWs is $\mathcal{NP}$-complete (see Savelsbergh [1985]).

The TSPTW is defined on a directed graph $G = (V, A)$. The vertex set V is made up of $n + 1$ vertices $0, 1, \ldots, n$, where vertex 0 represents the city where the salesman starts and returns and the other n vertices are the cities to visit. For each arc (i, j) of the arc set A, the travel distance and the travel time are known. For each vertex $i \in V$, a time window $[e_i, l_i]$ is given. If the salesman arrives at vertex i before the beginning of the time window, e_i, the visit is postponed until time e_i. Let $\mathscr{H}$ be the family of all tours in G, such that each city is visited within its time window. The TSPTW is to find a tour of the set $\mathscr{H}$ of minimum total distance.

The TSPTW has applications in single and multiple vehicle problems. Practical TSPTWs are encountered in a variety of industrial and service sector applications; examples include control of stacker cranes in warehouses (see Ascheuer et al. [2001]), bank deliveries, postal deliveries, and school-bus routing and scheduling.

In Chapter 3, we describe a new exact algorithm for solving the TSPTW. The fundamental ingredient of the new exact method is state-space relaxation, which is a general relaxation procedure, proposed by Christofides et al. [1981b,c], whereby the state-space associated with a given dynamic programming recursion is relaxed in order to compute valid bounds to the original problem. We propose new tour relaxations, called ng and ngL, to compute valid lower bounds on the TSPTW, corresponding to the costs of a problem that seeks a least-cost convex combination of the tours of a given tour relaxation. An optimal TSPTW integer solution is then found by iteratively running a dynamic programming algorithm, that limits the number of states to generate by using bounding functions based on the different tour relaxations proposed and on the lower bounds and corresponding dual solutions previously achieved. An extensive computational analysis on basically all TSPTW benchmark instances (with up to 233 vertices) is reported. The new algorithm is compared with the state-of-the-art exact algorithms (i.e., two branch-and-cut algorithms, a dynamic programming recursion, and a constraint-programming-based method). The computational experiments show that the proposed algorithm solves all but one instance to optimality within short computing times and outperforms the other exact methods considered.

1.3.3 Vehicle Routing Problem with Time Windows

Real-life routing problems usually include a wide range of operational constraints, so many variants of the basic CVRP have been proposed in the literature. The most studied variant of the CVRP is the vehicle routing problem with time windows (VRPTW).

The VRPTW can be formally defined on a directed graph $G = (V', A)$, where $V' = \{0, \ldots, n\}$ is the vertex set and A is the arc set. A travel cost and a travel time are associated with each arc $(i, j) \in A$. A set of m identical vehicles, each with capacity Q, are available at the depot (represented by vertex 0). A nonnegative demand and a time window are associated with each customer $1, \ldots, n$. The VRPTW consists of finding a collection of m simple circuits (each corresponding to a vehicle *route*) of minimum total cost, defined as the sum of the costs of the arcs belonging to the circuits, such that each circuit visits the depot, each customer is visited by exactly one circuit within its time window, and the sum of the demands of the vertices visited by a circuit does not exceed the vehicle capacity, Q.

The VRPTW generalizes not only the CVRP but also the TSPTW, which can be considered as the single-vehicle version of the VRPTW. Since the CVRP is $\mathcal{NP}$-hard, by restriction, the VRPTW is also $\mathcal{NP}$-hard.

The VRPTW has numerous applications in distribution management, such as beverage/food and newspaper delivery. Time windows naturally arise in problems faced by business organizations which work on strict time schedules; specific examples of such problems include bank deliveries, industrial refuse collection, and school-bus routing and scheduling.

Chapter 4 describes an effective exact method for solving the VRPTW that improves on the method proposed by Baldacci et al. [2008] for the CVRP. The new algorithm relies on the set partitioning formulation of the problem and on the state-space relaxations introduced for the TSPTW in Chapter 3. The core of the method is the dual-ascent heuristics developed to find near-optimal dual solutions of the linear relaxation of the set partitioning model; these dual solutions and the corresponding lower bounds are computed via column generation by solving the pricing problem with dynamic programming and state-space relaxations and by solving the master problem through subgradient optimization. Another main ingredient of the exact method is a column-and-cut generation algorithm that strengthens the lower bounds achieved by the dual-ascent heuristics by adding valid inequalities; the pricing problem is solved with dynamic programming but adopting a new strategy of combining different feasible dual solutions to limit the state-space graph. An optimal VRPTW integer solution is finally found by solving, with a general purpose integer programming solver, a restricted master problem generated by taking into account all of the feasible dual solutions computed by the different bounding procedures previously run. We show that the proposed algorithm can be properly tailored to obtain an effective solution method for the CVRP. The last part of Chapter 4 compares the performance of our exact method on both VRPTW and CVRP benchmark instances with the state-of-the-art exact algorithms (branch-and-cut and branch-and-cut-and-price) from the literature. The proposed method solves four of the five open Solomon VRPTW instances and significantly improves the average running times of the state-of-the-art algorithms for both the VRPTW and the CVRP.

1.3.4 Multi-Trip Vehicle Routing Problem

The multi-trip vehicle routing problem (MTVRP) is a generalization of the CVRP, where each vehicle is allowed to perform a set of routes, called a *schedule*, of total duration not exceeding a maximum driving time.

The MTVRP is defined on an undirected graph $G = (V', E)$. The vertex set V' is partitioned as $V' = \{0\} \cup V$, where vertex 0 represents the depot and the set $V = \{1, \dots, n\}$ represents n customers, each one requiring q_i units of product from the depot. A travel cost and a travel time are associated with each edge $\{i, j\} \in E$. A fleet of m identical vehicles is located at the depot; each vehicle has capacity Q and maximum driving time T. The MTVRP calls for the design of a set of m schedules of minimum total cost such that each customer is visited exactly once by the routes of the schedules and each schedule does not exceed the maximum driving time.

The reason for studying the MTVRP is that, in many practical applications, the assumption that a vehicle can perform at most a route per day is unrealistic. This is especially true whenever the vehicle capacity is small with respect to the quantities requested by customers or whenever the planning period is large. In urban areas, for example, travel times are rather small, and it is often the case that after performing short tours vehicles are reloaded and used again. A practical application of MTVRP is the distribution of perishable goods.

Despite its practical importance, little attention has been given to the MTVRP. Though a few heuristics have been proposed in the last two decades, no exact algorithm can be found in the literature, except for an exact algorithm proposed for an MTVRP with time windows and unlimited maximum driving time.

In Chapter 5, we present an exact method for solving the MTVRP. The method is based on two set-partitioning-like formulations. The first formulation has a binary variable for each feasible route and each vehicle, whereas the second formulation has a binary variable for each feasible schedule. The linear relaxations of the two formulations and the relation between them are studied. We propose four column-and-cut generation bounding procedures to solve the linear relaxations of the two formulations enforced with valid inequalities derived from the CVRP and with a new class of valid inequalities specific for the problem. Furthermore, we present a simple though effective method to improve the lower bounds achieved by the different bounding procedures that is based on the use of multiple feasible dual solutions. The exact method we propose executes, in sequence, the four bounding procedures and then finds an optimal MTVRP solution by solving, with a general purpose integer programming solver, a restricted master problem containing a reduced set of either routes or schedules belonging to any optimal solution. The computational results show that the proposed method can solve, to optimality, MTVRP benchmark instances involving up to 120 customers.

1.3.5 Two-Echelon Capacitated Vehicle Routing Problem

The two-echelon capacitated vehicle routing problem (2E-CVRP) models two-level distribution systems in which freight arrives at a central depot, is transported to intermediate facilities, called *satellites*, and is finally delivered to customers from the satellites. Therefore, two distribution levels are involved. The distribution system takes into account routing costs, handling costs for the operations of unloading and loading freight at satellites, and fixed costs for using satellites. Limits on the maximum number of first and second level vehicles to use, on the maximum number of vehicles to route at each satellite, and on the maximum quantity of freight to handle at each satellite are imposed. The objective is to minimize the total cost of the system.

The 2E-CVRP concerns both vehicle fleet management and network configuration. Indeed, the 2E-CVRP subsumes both the CVRP and the UFL, so it is an $\mathcal{NP}$-hard problem.

A common application of the 2E-CVRP is *city logistics* (see Taniguchi [2001], Taniguchi and Thompson [2008]), which is an area of urban study and urban management. The goal of city logistics is to establish efficient, safe and environmentally friendly urban freight transport systems. Space in city centers is often limited and must be shared between private and public passenger transport as well as parking facilities. Other than producing congestion, pollution and noise, large vehicles usually have low average loads and perform numerous empty trips. Therefore, officials want to reduce the number of vehicles, especially freight vehicles, in city centers and want to switch from large to smaller vehicles.

In Chapter 6, we introduce a new mathematical formulation of the 2E-CVRP that is used to derive both integer and continuous relaxations. We present a new bounding procedure based on dynamic programming and an exact algorithm that decomposes the 2E-CVRP into a set of CVRPs with multiple depots and side constraints. The new algorithm is tested on five sets of instances from the literature and a new set of instances with up to 100 customers and 6 satellites. Computational results show that 144 out of 153 instances from the literature were solved, 97 of which for the first time. The comparisons with the state-of-the-art exact methods show that new exact method compares favorably with the other exact methods from the literature.

Chapter 2

Asymmetric Traveling Salesman Problem

[1]

This chapter surveys the most effective mathematical models and exact algorithms proposed for finding the optimal solution of the well-known *asymmetric traveling salesman problem* (ATSP). The fundamental *integer linear programming* (ILP) model proposed by Dantzig, Fulkerson and Johnson is first presented, its classical (Assignment, Shortest Spanning r-Arborescence, Linear Programming) relaxations are derived, and the most effective branch-and-bound and branch-and-cut algorithms are described. The polynomial ILP formulations proposed for the ATSP are then presented and analyzed. The considered algorithms and formulations are finally experimentally compared on a set of benchmark instances.

2.1 Introduction

Let $G = (V, A)$ be a complete digraph, where $V = \{1, \dots, n\}$ is the vertex set and $A = \{(i, j) : i, j \in V\}$ the arc set, and let c_{ij} be a cost associated with arc $(i, j) \in A$ (with $c_{ii} = +\infty$, for $i \in V$). A *Hamiltonian circuit* (*tour*) of graph G is a circuit that visits each vertex of the vertex set V exactly once. The *asymmetric traveling salesman problem* (ATSP) is to find a Hamiltonian circuit $G^* = (V, A^*)$ of graph G whose cost $\sum_{(i,j)\in A^*} c_{ij}$ is minimum. If the considered graph G is undirected, the corresponding problem is denoted as *symmetric traveling salesman problem* (STSP).

The ATSP is known to be $\mathcal{NP}$-hard in the strong sense and has been intensively studied in the last six decades. In this chapter, we consider and experimentally compare the most effective *integer linear programming* (ILP) models and exact algorithms proposed

[1]This chapter is based on Roberti and Toth [2012]

for solving the ATSP. Previous surveys on the subject were presented by Balas and Toth [1985], Fischetti et al. [2004], Öncan et al. [2009], D'Ambrosio et al. [2010]. Several books dealing with the STSP and its variations have been published; among them, we mention those by Lawler et al. [1985], Reinelt [1994], Gutin and Punnen [2002], Applegate et al. [2007].

2.2 The Dantzig-Fulkerson-Johnson Formulation and its Relaxations

Dantzig et al. [1954] proposed the following ILP model (hereafter DFJ) with n^2 binary variables x_{ij}

$$(DFJ) \quad \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

$$s.t. \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, \dots, n, \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \dots, n, \tag{2.3}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1, \quad S \subset V \, : \, S \neq \varnothing, \tag{2.4}$$

$$x_{ij} \in \{0, 1\}, \qquad (i, j) \in A, \tag{2.5}$$

where variable x_{ij} is equal to 1 if and only if arc $(i, j) \in A$ is in the optimal tour. Constraints (2.2) and (2.3) impose that the in-degree and out-degree, respectively, of each vertex is equal to one, whereas constraints (2.4) are *subtour elimination constraints* (SECs) and impose that no partial circuit exists.

Moreover, it is well-known that one can halve the number of SECs (2.4) by replacing them with

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1, \quad S \subset V \setminus \{r\} \, : \, S \neq \varnothing,$$

where r is any vertex of vertex set V.

Because of constraints (2.2) and (2.3), constraints (2.4) can be equivalently written as *connectivity constraints*

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \ge 1, \quad S \subset V \, : \, S \neq \varnothing. \tag{2.6}$$

Also in this case, one can halve the number of connectivity constraints (2.6) by replacing them with

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1, \quad S \subset V : r \in S \tag{2.7}$$

or with

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1, \quad S \subset V : S \neq \varnothing, r \notin S \tag{2.8}$$

where $r \in V$ is any fixed vertex.

A valid lower bound on the ATSP can be obtained by optimally solving the *linear programming* (LP) relaxation of the previous models (2.1)-(2.5) or (2.1)-(2.3) plus constraints (2.5) and (2.7), obtained by replacing constraints (2.5) with constraints

$$x_{ij} \geq 0, \quad (i, j) \in A. \tag{2.9}$$

Although the considered ILP models require an exponential number of SECs or connectivity constraints, their LP relaxations can be efficiently solved in polynomial time by using the effective polynomial *separation procedure* proposed by Padberg and Rinaldi [1990a] for the STSP.

Additional lower bounds can be obtained by considering the different substructures of the ATSP, each associated with a subset of constraints defining a well-structured relaxation.

Constraints (2.2), (2.3) and (2.9), with objective function (2.1), define the well-known *min-sum assignment problem* (AP). Such a problem always has an integer optimal solution and requires the finding of a minimum-cost collection of vertex-disjoint subtours visiting all vertices of graph G. Relaxation AP can be solved in $O(n^3)$ time (see, e.g. Lawler [1976], and Carpaneto and Toth [1987] for an efficient implementation).

Constraints (2.2), (2.7) and (2.9), with objective function (2.1), define the well-known *shortest spanning r-arborescence problem* (r-SAP). Such a problem always has an integer optimal solution and corresponds to finding a minimum-cost spanning subdigraph $\bar{G} = (V, \bar{A})$ of graph G such that (i) the in-degree of each vertex is exactly one, and (ii) each vertex can be reached from the root vertex r. Relaxation r-SAP can be solved in $O(n^2)$ time by finding the shortest spanning arborescence rooted at vertex r (see, e.g. Edmonds [1967], Tarjan [1977], and Fischetti and Toth [1993] for an efficient implementation) and adding the minimum-cost arc entering vertex r.

A third substructure, corresponding to constraints (2.3), (2.8) and (2.9), with objective function (2.1), defines the *shortest spanning r-antiarborescence problem* (r-SAAP). Such a problem can easily be transformed into an r-SAP by simply transposing the input cost matrix, so it can be solved in $O(n^2)$ time. Different choices of the root vertex

r generally produce different values of the lower bounds corresponding to relaxations r-SAP and r-SAAP. Moreover, these relaxations can be strengthened by considering the associated Lagrangean relaxations, obtained by embedding, in a Lagrangean fashion, the relaxed constraints (2.3) for r-SAP, and (2.2) for r-SAAP, in the objective function (2.1). Near-optimal Lagrangean multipliers, leading to good lower bounds, can be obtained by applying the well-known *subgradient optimization* procedure proposed by Held and Karp [1970] and Held and Karp [1971] for the STSP.

The lower bounds corresponding to relaxations AP, r-SAP and r-SAAP can also be improved (see Fischetti and Toth [1992]) by combining the associated substructures with the *additive approach* introduced by Fischetti and Toth [1989].

2.3 Review of Polynomial Formulations

In this section, we consider the papers presenting polynomial formulations for the ATSP. For each paper, we focus on the formulation producing the tightest LP-relaxation lower bound. Unlike the exact algorithms described in §2.4, the polynomial formulations can be directly solved by a general-purpose ILP solver. Classifications and comparisons of the polynomial formulations for the ATSP have been recently presented in Öncan et al. [2009] and Godinho et al. [2011].

The earliest polynomial formulation (i.e., formulation requiring a number of constraints polynomial in the number of vertices n) of the ATSP is owed to Miller et al. [1960] (hereafter MTZ) and is given by (2.1)-(2.3), (2.5) plus the following $(n-1)^2$ constraints to break subtours

$$u_i - u_j + (n-1)x_{ij} \leq n-2, \quad i,j = 2,\ldots,n, \tag{2.10}$$

where u_i, $i = 2,\ldots,n$, is an arbitrary real number representing the order of vertex i in the optimal tour. Miller et al. originally proposed their formulation with no bounds on variables u_i. Later on, simple bounds (e.g., $1 \leq u_i \leq n-1$, $i = 2,\ldots,n$) were introduced to restrict the range of variables u_i. This does not affect the LP bound of MTZ and, in our computational experiments, has shown to increase the computing time, so we leave variables u_i unrestricted.

Gavish and Graves [1978] proposed another formulation (hereafter GG) having LP relaxation stronger than that of MTZ (see Wong [1980] and Padberg and Sung [1991]) but weaker than that of DFJ (see Gouveia [1995]). Formulation GG is a *single-commodity flow formulation* where subtours are broken by introducing n^2-n nonnegative variables g_{ij}, $i = 2,\ldots,n$, $j = 1,\ldots,n$. Formulation GG consists of constraints (2.1)-(2.3), (2.5) and the following constraints

$$\sum_{j=1}^{n} g_{ij} - \sum_{j=2}^{n} g_{ji} = 1, \qquad i = 2, \ldots, n, \tag{2.11}$$

$$0 \leq g_{ij} \leq (n-1)x_{ij}, \quad i = 2, \ldots, n, \; j = 1, \ldots, n, \tag{2.12}$$

where variables g_{ij} can be interpreted as the number of arcs on the path from vertex 1 to arc (i, j) in the optimal tour (see Gouveia and Pires [1999]).

For fixed values of the variables x_{ij}, constraints (2.11) and (2.12) form a network flow problem, so variables g_{ij} take integer values. Langevin et al. [1990] showed that the LP relaxation of GG is equivalent to that of the *two-commodity flow formulation* proposed by Finke et al. [1984] - hereafter FCG.

Fox et al. [1980] proposed three formulations for the *time-dependent traveling salesman problem* that are valid for the ATSP, as well. These formulations present n^3 binary variables, r_{ijk}, that are equal to 1 if and only if arc $(i, j) \in A$ is in position k in the optimal tour. The first formulation (therein P1) has $4n$ constraints, whereas the second formulation (hereafter P1b) has $3n$ constraints and is obtained from the first one by dropping a set of n constraints. Thus, the LP relaxation of the first formulation is stronger than that of the second one; Gouveia and Voß [1995] showed that the LP relaxations of both formulations are stronger than that of GG. The third formulation proposed by Fox et al. (therein P2) has $n+1$ constraints and is obtained from the first formulation by surrogating the first $3n$ constraints. Öncan et al. [2009] showed that the LP relaxation of the third formulation is weaker than that of the second one, whereas Padberg and Sung [1991] proved that the LP relaxation of the third formulation is also weaker than that of DFJ. We have not reported the three formulations because, in our computational experiments, they proved to be ineffective in solving the ATSP.

The first *multi-commodity flow* (MCF) formulation was proposed by Wong [1980] (hereafter WONG). WONG considers $2n - 2$ commodities and introduces $2n^3 - 2n^2$ nonnegative continuous variables and $4n^3 - 2n^2 - 2n$ constraints. This formulation was later modified by Langevin [1988] and by Loulou [1988] to obtain two additional MCF formulations which provide LP relaxations equivalent to that of WONG.

Another MCF formulation with only $n - 1$ commodities was proposed by Claus [1984]. This formulation (hereafter CLAUS) introduces $n^3 - n^2$ nonnegative continuous variables w_{ij}^k, $i, j = 1, \ldots, n$, $k = 2, \ldots, n$, and $2n^3 - n^2 - n$ constraints, and consists of constraints (2.1)-(2.3), (2.5) plus the following constraints to break subtours

$$
\begin{aligned}
&\sum_{j=1}^{n} w_{ij}^{k} - \sum_{j=1}^{n} w_{ji}^{k} = 0, && i,k = 2,\dots,n \;:\; i \neq k,\\
&\sum_{j=2}^{n} w_{1j}^{k} - \sum_{j=2}^{n} w_{j1}^{k} = -1, && k = 2,\dots,n,\\
&\sum_{j=1}^{n} w_{ij}^{i} - \sum_{j=1}^{n} w_{ji}^{i} = 1, && i = 2,\dots,n,\\
&0 \leq w_{ij}^{k} \leq x_{ij}, && i,j = 1,\dots,n,\; k = 2,\dots,n,
\end{aligned}
$$

where variable w_{ij}^{k} is equal to 1 if and only if the commodity going from vertex 1 to vertex k flows on arc (i,j). Langevin et al. [1990] proved that the LP relaxation of CLAUS is equivalent to that of WONG, whereas Padberg and Sung [1991] proved that the LP relaxation of CLAUS is equivalent to that of DFJ, as well.

Formulation MTZ was strengthened by Desrochers and Laporte [1990], who proposed formulation DL having LP relaxation stronger than that of MTZ and obtained from MTZ by replacing constraints (2.10) with the following $n^2 - 1$ lifted constraints

$$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n-2, \quad i,j = 2,\dots,n, \tag{2.13}$$

$$-u_i + (n-3)x_{i1} + \sum_{j=2}^{n} x_{ji} \leq -1, \quad i = 2,\dots,n, \tag{2.14}$$

$$u_i + (n-3)x_{1i} + \sum_{j=2}^{n} x_{ij} \leq n-1, \quad i = 2,\dots,n. \tag{2.15}$$

Gouveia and Pires [1999] presented four formulations (therein called RMTZ, L1RMTZ, L2RMTZ, and L3RMTZ). The LP relaxations of all of these formulations are stronger than that of MTZ. Here, we present formulation L3RMTZ (hereafter GP), whose LP relaxation is stronger than those of L1RMTZ and L2RMTZ, which in turn are stronger than that of RMTZ. Gouveia and Pires [1999] also showed that the LP relaxations of formulations L1RMTZ and L2MTZ are weaker than those of MCF formulations, such as WONG and CLAUS. Formulation GP introduces $(n-1)^2$ additional nonnegative continuous variables v_{ij}, $i,j = 2,\dots,n$, that are equal to 1 if and only if vertex i is in the path from vertex 1 to vertex j. Formulation GP consists of constraints (2.1)-(2.3), (2.5) plus the following $2n^3 - 10n^2 + 18n - 10$ constraints to break subtours

$$
\begin{aligned}
&x_{ij} - v_{ij} \leq 0, && i,j = 2,\dots,n,\\
&x_{ij} + v_{ji} \leq 1, && i,j = 2,\dots,n,\\
&x_{ji} + x_{ij} + v_{ki} - v_{kj} \leq 1, && i,j,k = 2,\dots,n \;:\; i \neq j \neq k,\\
&x_{kj} + x_{ik} + x_{ij} + v_{ki} - v_{kj} \leq 1, && i,j,k = 2,\dots,n \;:\; i \neq j \neq k.
\end{aligned}
$$

Gouveia and Pires [2001] presented other formulations, among them a polynomial formulation (therein MCF+) whose LP relaxation is stronger than those of CLAUS and GP. As the LP relaxation of MCF+ is weaker than that of formulation SST (which will be introduced later in this section), as shown by Öncan et al. [2009], and MCF+ has more constraints than SST, we do not report a detailed description of MCF+.

Sherali and Driscoll [2002] strengthened formulation DL by applying a reformulation-linearization technique and by introducing $(n-1)^2$ additional nonnegative continuous variables y_{ij}, $i, j = 2, \ldots, n$, where variable y_{ij} represents the order of arc (i, j) in the optimal tour. The resulting formulation (hereafter SD) replaces constraints (2.13), (2.14) and (2.15) with the following $4n^2 - 4n$ constraints

$$
\begin{aligned}
&\sum_{j=2}^{n} y_{ij} + (n-1)x_{i1} - u_i = 0, && i = 2, \ldots, n,\\
&\sum_{i=2}^{n} y_{ij} - u_j = -1, && j = 2, \ldots, n,\\
&x_{ij} - y_{ij} \leq 0, && i, j = 2, \ldots, n,\\
&y_{ij} - (n-2)x_{ij} \leq 0, && i, j = 2, \ldots, n,\\
&u_j + (n-2)x_{ij} + (n-1)x_{ji} - y_{ij} - y_{ji} \leq n-1, && i, j = 2, \ldots, n,\\
&y_{ij} + y_{ji} - u_j - x_{ji} \leq -1, && i, j = 2, \ldots, n,\\
&-x_{1j} + (n-3)x_{j1} - u_j \leq -2, && j = 2, \ldots, n,\\
&(n-3)x_{1j} - x_{j1} + u_j \leq n-2, && j = 2, \ldots, n.
\end{aligned}
$$

Recently, Öncan et al. [2009] showed that the LP relaxation of SD is also stronger than that of GG.

Sarin et al. [2005] studied the asymmetric traveling salesman problem with and without precedence constraints and proposed five polynomial formulations (therein ATSPxy, L1ATSPxy, SL1ATSPxy, L2ATSPxy and ML1ATSPxy) for the ATSP, whose LP relaxations are stronger than that of RMTZ. Moreover, Sarin et al. [2005] showed that the LP relaxation of L1ATSPxy is stronger than that of SL1ATSPxy whose LP relaxation is stronger than that of ATSPxy; the LP relaxations of L1ATSPxy and L2ATSPxy are also stronger than those of formulations L1RMTZ and L2RMTZ by Gouveia and Pires [1999], respectively. Here we report formulation L2ATSPxy (hereafter SSB) only, which, in our computational experiments, was shown to be the best performer. Formulation SSB introduces $(n-1)^2$ nonnegative continuous variables d_{ij}, $i, j = 2, \ldots, n$, and $n^3 - n^2 - n + 1$ constraints, and consists of constraints (2.1)-(2.3), (2.5) and the following constraints to break subtours

$$d_{ij} - x_{ij} \geq 0, \qquad i,j = 2,\ldots,n, \tag{2.16}$$

$$d_{ij} + d_{ji} = 1, \qquad i,j = 2,\ldots,n \,:\, i \neq j, \tag{2.17}$$

$$x_{1j} + x_{j1} \leq 1, \qquad j = 2,\ldots,n, \tag{2.18}$$

$$x_{ij} + d_{jk} + x_{kj} + d_{ki} + x_{ik} \leq 2, \qquad i,j,k = 2,\ldots,n. \tag{2.19}$$

Although variable d_{ij}, $i,j = 2,\ldots,n$, is continuous, it has a binary connotation and is equal to 1 if and only if vertex i precedes (not necessarily immediately) vertex j in the optimal tour. Godinho et al. [2011] noticed that the meaning of variables d_{ij} is basically the same of variables v_{ij} introduced in formulation GP. An analysis of all the formulations involving variables x_{ij} and v_{ij} (or d_{ij}) can be found in Gouveia and Pesneau [2006].

Sherali et al. [2006] proposed several polynomial formulations. Here, we present formulation SST (therein ATSP6) only, which uses $(n-1)^3$ nonnegative continuous variables t_{ij}^k, $i,j,k = 2,\ldots,n$, and consists of (2.1)-(2.3), (2.5), (2.17) and the following constraints

$$\begin{aligned}
& d_{ij} + x_{ji} + d_{jk} + d_{ki} \leq 2, && i,j,k = 2,\ldots,n,\\
& d_{ij} - x_{1i} \geq 0, && i,j = 2,\ldots,n,\\
& d_{ji} - x_{i1} \geq 0, && i,j = 2,\ldots,n,\\
& 0 \leq t_{ij}^k \leq x_{ik}, && i,j,k = 2,\ldots,n : i \neq j \neq k,\\
& x_{ij} + \sum_{k=2;\, k\neq j}^{n} t_{ij}^k = d_{ij}, && i,j = 2,\ldots,n,\\
& x_{1k} + \sum_{i=2;\, i\neq j}^{n} t_{ij}^k = d_{kj}, && k,j = 2,\ldots,n,
\end{aligned}$$

where t_{ij}^k is equal to 1 if and only if, in the optimal tour, arc (i,k) is used and vertex k precedes vertex j. Godinho et al. [2011] pointed out that variables t_{ij}^k can be interpreted in the same way as the flow variables w_{ij}^k used in formulation CLAUS. Öncan et al. [2009] showed that the LP relaxation of SST is stronger than those of MCF+, whereas Sherali et al. [2006] proved that the LP relaxation of SST is stronger than that of L1ATSPxy.

In Figure 2.1, we summarize the relationships among the linear relaxations of the polynomial formulations reviewed in this section. A link going from formulation A to formulation B means that the LP relaxation of B is stronger than that of A, whereas a dashed line connecting two formulations means that the relative LP relaxations are equivalent.

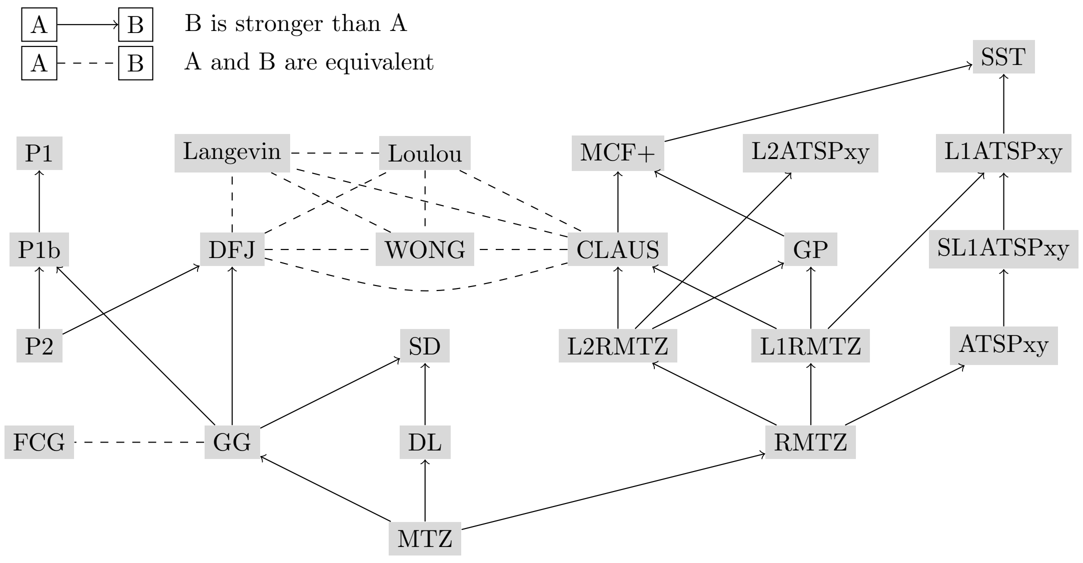

FIGURE 2.1: Relations among the linear relaxations of the polynomial formulations and of formulation DFJ

2.4 Exact Algorithms

Many branch-and-bound algorithms have been proposed to find the optimal solution of the ATSP. After the seminal paper by Little et al. [1963], where for the first time the term "*branch-and-bound*" was coined, other algorithms were proposed by Bellmore and Malone [1971], Garfinkel [1973], Smith et al. [1977], Carpaneto and Toth [1980], Balas and Christofides [1981], Miller and Pekny [1989], Pekny and Miller [1992], Fischetti and Toth [1992], Carpaneto et al. [1995]. In the following, two of the most effective branch-and-bound algorithms for the ATSP (i.e., those proposed by Carpaneto et al. [1995] and Fischetti and Toth [1992]), are briefly reviewed. The algorithm proposed by Pekny and Miller [1992] exhibits, on the whole, a performance comparable with that of the approach described in Carpaneto et al..

The algorithm proposed by Carpaneto et al. [1995] is a lowest-first branch-and-bound method based on the AP relaxation and the *subtour elimination branching scheme*. At the root node of the decision tree, the AP relaxation of the original problem is solved, the *patching heuristic* algorithm proposed by Karp [1979] is applied to determine an initial tour of cost z^*, and a *reduction procedure* based on the AP reduced costs, c'_{ij}, is executed to transform the original complete graph into a sparse one (by setting $x_{ij} = 0$ if $V(AP) + c'_{ij} \geq z^*$, where $V(AP)$ is the value of the optimal solution of the AP relaxation). At each of the other nodes of the decision tree, the AP relaxation of the subproblem associated with the considered node is solved, through an effective parametric technique, in $O(n^2)$ time. If $V(AP) \geq z^*$, the node is fathomed. Otherwise, if the AP solution contains no subtour (i.e., a feasible solution has been found) the best solution thus far is updated, z^* is set equal to $V(AP)$, and the node is fathomed. If neither of the two previous cases occur, the subtour elimination branching scheme proposed by Carpaneto and Toth [1980] is applied: the subtour S of the AP solution having the minimum number, say h, of not "imposed" arcs is selected, and h descending nodes are generated so as to forbid, by "imposing" and "excluding" proper arc subsets, subtour S for each descending node.

The algorithm proposed by Fischetti and Toth [1992] is a lowest-first branch-and-bound method based on the branching scheme introduced by Carpaneto and Toth [1980], and, at each node of the decision tree, computes the corresponding lower bound by applying the additive approach combining the AP, r-SAP, and r-SAAP relaxations.

More recently, two effective branch-and-cut algorithms for the ATSP have been proposed by Fischetti and Toth [1997] and Fischetti et al. [2003].

The algorithm proposed by Fischetti and Toth [1997] is based on the DFJ model (2.1)-(2.5) and exploits additional classes of facet-inducing inequalities for the ATSP polytope that proved to be of crucial importance for the solution of some real-world instances.

An ATSP inequality $\alpha x \leq \alpha_0$ is called *symmetric* when $\alpha_{ij} = \alpha_{ji}$ for each arc $(i, j) \in A$. Symmetric inequalities can be thought of as derived from valid inequalities for the STSP defined on the complete undirected graph $G' = (V, E)$. Indeed, let y_e be equal to 1 if edge $e \in E$ belongs to the optimal STSP solution, $y_e = 0$ otherwise. Every inequality $\sum_{e \in E} \alpha_e y_e \leq \alpha_0$ valid for the STSP can be transformed into a valid ATSP inequality by simply replacing y_e with $x_{ij} + x_{ji}$ for all edges $e = \{i, j\} \in E$. This produces the symmetric inequality $\alpha x \leq \alpha_0$, where $\alpha_{ij} = \alpha_{ji} = \alpha_{(i,j)}$ for all couples of vertices $i, j \in V$, $i \neq j$. Conversely, every symmetric ATSP inequality $\alpha x \leq \alpha_0$ corresponds to the valid STSP inequality $\sum_{\{i,j\} \in E} \alpha_{ij} y_{(i,j)} \leq \alpha_0$. The above correspondence implies that every separation algorithm for the STSP can be used, as a "black box", for the ATSP, as well. Several exact/heuristic separation algorithms for the STSP have been proposed in recent years, all of which can be used for the ATSP. Only two such separation tools are used in Fischetti and Toth [1997], namely (i) the Padberg and Rinaldi [1990a] exact algorithm for SECs; and (ii) the simplest heuristic scheme for 2-matching constraints, i.e., for combs with 2-node teeth, where each component H of the graph induced by the edges $e \in E$ with fractional y^*_e is heuristically considered, in turn, as the handle of the comb. Having fixed H, the most violated 2-matching constraint with handle H is easily found by sorting the edges having one extreme node in H and the other extreme node in $V \setminus H$ by non increasing y^*_e, and by taking the first k such edges to act as teeth, for $k = 1, 3, 5, \ldots$

In addition, Fischetti and Toth [1997] considered the D^+_k and D^-_k inequalities proposed by Grötschel and Padberg [1985] and the odd *close alternative trail* (odd CAT) inequalities proposed by Balas [1989] (and analyzed by Fischetti [1991]). The separation problem for the classes of the D^+_k and D^-_k inequalities is a combinatorial optimization problem that can be effectively solved in practice by an implicit enumeration scheme enhanced by suitable pruning conditions (see Fischetti and Toth [1997]). As for the detection of violated odd CAT inequalities, Balas [1989] showed that these inequalities correspond to odd cycles in an auxiliary undirected "incompatibility" graph. An effective heuristic separation algorithm, based on the computation of a minimum-weight odd cycle going through a given edge, was proposed by Fischetti and Toth [1997]. In addition, *clique lifting* (see Fischetti and Balas [1993]) and *shrinking* (see Padberg and Rinaldi [1990b]) procedures are applied to simplify the considered separation problems. A detailed analysis of the polyhedral structure of the ATSP can be found in Balas and Toth [1985].

Pricing is an important ingredient of branch-and-cut codes because it allows one to effectively handle LP relaxations involving a huge number of variables. In order to keep the size of the LP relaxation as small as possible, the following pricing scheme is commonly used. We determine a (small) *core* set of arcs, say $\tilde{A}$, and decide to temporarily fix $x_{ij} = 0$ for each arc $(i, j) \in A \setminus \tilde{A}$. We then solve the corresponding restricted LP problem, compute the associated LP reduced costs $\bar{c}_{ij}$ and check whether $\bar{c}_{ij} \geq 0$ for all $(i, j) \in A \setminus \tilde{A}$. If this is the case, then the LP relaxation has been solved to optimality. Otherwise, the current core set $\tilde{A}$ is enlarged by adding (some of) the arcs with negative reduced cost, and the whole procedure is iterated.

Fischetti and Toth [1997] proposed an improved pricing technique, called *AP pricing*, in which the pricing condition is strengthened by exploiting the fact that any feasible solution of the current LP relaxation cannot arbitrarily select the arcs of negative reduced cost because the degree equations - among other constraints - must be fulfilled.

The exact algorithm proposed by Fischetti and Toth [1997] is a lowest-first branch-and-cut method. At each node of the branching tree, the LP relaxation is initialized by taking all of the constraints of the last LP solved at the father node (for the root node, only the degree equations are taken). As for the variables, one retrieves, from a scratch file, the optimal basis associated with the last LP solved at the father node and initializes the core variable set, $\tilde{A}$, by taking the arcs belonging to this basis (for the root node, $\tilde{A}$ contains the $2n-1$ variables in the optimal AP basis found by solving AP on the original costs c_{ij}). In addition, $\tilde{A}$ contains all the arcs of the best known ATSP solution. Starting with the above advanced basis, one iteratively solves the current LP relaxation, applies the AP pricing procedure and repeats if needed.

On exit of the pricing loop, the separation algorithms are applied to find, if any, ATSP inequalities that cut off the current LP optimal solution x^*. When violated cuts are found, one adds them to the current LP relaxation and repeats.

When separation fails and x^* is integer, the current best ATSP solution is updated, and a backtracking step occurs. If x^* is fractional, the current LP basis is saved in a file, and one branches on the variable x_{ij} with $0 < x^*_{ij} < 1$ that maximizes the score $\sigma(i, j) = c_{ij} \min\{x^*_{ij}, 1 - x^*_{ij}\}$. As a heuristic rule, a large priority is given to the variables with $0.4 \leq x^*_{ij} \leq 0.6$ (if any) so as to produce a significant change in both descending nodes.

This branching scheme has been enhanced by Fischetti et al. [2003] through the *fractional persistency* mechanism proposed by Fischetti et al. [2001] for the solution of crew scheduling and vehicle scheduling problems. The corresponding branch-and-cut algorithm will be denoted as FLT in §2.6.

2.5 Transformation of ATSP Instances into STSP Instances

Any code for the ATSP can be invoked to solve STSP instances. In fact, the reverse also stands by means of the following two transformations.

- The *3-node* transformation proposed by Karp [1972]. A complete undirected graph with $3n$ vertices is obtained from the original complete directed one by adding two copies, $n+i$ and $2n+i$, of each vertex $i \in V$, and by (i) setting to 0 the cost of edges $\{i, n+i\}$ and $\{n+i, 2n+i\}$ for each vertex $i \in V$, (ii) setting the cost of edge $\{2n+i, j\}$ to c_{ij}, for each couple of vertices $i, j \in V$, and (iii) setting to $+\infty$ the costs of all remaining edges;
- The *2-node* transformation proposed by Jonker and Volgenant [1983] (see also Jünger et al. [1995]). A complete undirected graph with $2n$ vertices is obtained from the original complete directed one by adding a copy, $n+i$, of each vertex $i \in V$, and by (i) setting to 0 the cost of the edge $\{i, n+i\}$ for each vertex $i \in V$, (ii) setting the cost of edge $\{n+i, j\}$ to $c_{ij} + M$, for each couple of vertices $i, j \in V$, where M is a sufficiently large positive value, and (iii) setting to $+\infty$ the costs of all the remaining edges. The transformation value nM must be subtracted from the STSP optimal cost.

The most effective branch-and-cut algorithm for the STSP is currently the one by Applegate et al. [2007], and the corresponding code (Concorde by Applegate et al. [1999]), is publicly available. In Fischetti et al. [2004], this code was used to test the effectiveness of the approach based on the ATSP-to-STSP transformation. The code has been used with default parameters. The results have shown that the 2-node transformation is, in general, more effective than the 3-node one.

2.6 Computational Results

As a testbed for our computational experiments, we took the 27 ATSP instances collected in the TSPLIB by Reinelt [1991] and 5 real-world instances provided by Balas [2000]. All instances have integer nonnegative costs.

TABLE 2.1: Computing times for solving the LP relaxations of 8 polynomial formulations by using the primal, dual or barrier method

	Primal		*Dual*		*Barrier*	
Formulation	*Solved*	T_{LP}	*Solved*	T_{LP}	*Solved*	T_{LP}
MTZ	10	0.16	10	0.03	10	16.30
GG	10	0.25	10	0.56	10	0.14
CLAUS	3	1,018.11	10	353.84	10	95.05
DL	10	0.30	10	0.05	10	17.23
GP	10	48.01	10	3.48	10	232.78
SD	10	7.45	10	5.10	10	0.75
SSB	9	360.49	10	6.68	10	103.59
SST	0	1,200.00	1	1,128.43	3	1,004.75

Tables 2.1 and 2.2 report on the computational performance of eight polynomial formulations (namely, MTZ, GG, CLAUS, DL, GP, SD, SSB, and SST) for the ATSP, described in §2.3, on 10 small-size instances from the TSPLIB (namely, instances ftv33, ftv35, ftv38, ftv44, ftv47, ftv55, ftv64, ftv70, ft70, and ft53), which is the set of instances used in the computational study of Öncan et al. [2009]. The computational results reported in Tables 2.1 and 2.2 were obtained on an Intel Core2 Duo@2.26 GHz by running CPLEX11.2 as LP and ILP solver. All computing times reported in the tables are expressed in seconds.

In Table 2.1, we compare, on the ten considered instances, the performance of the eight polynomial formulations for computing the corresponding LP relaxations by using three different LP methods (namely *Primal*, *Dual* and *Barrier*). For each method and each formulation, Table 2.1 reports the number of LP-bounds computed within the time limit of $1,200$ seconds (column *Solved*) and the average computing time (column T_{LP}). In the computation of the average computing time, we considered a computing time of $1,200$ seconds whenever the LP relaxation of an instance could not be computed within the time limit or the instance could not be run due to exceeded memory limits for CPLEX.

Table 2.1 shows that, for a given formulation, the computing time may significantly vary by using different LP methods. The dual method turned out to be the best performing method when solving MTZ, DL, GP, and SSB, whereas the barrier method was the best performing one when solving GG, CLAUS, SD, and SST. All formulations but SST could solve the LP relaxation of all 10 instances within the time limit imposed.

Table 2.2 reports, for each of the eight formulations and each of the ten instances, the percentage gap (computed with respect to the optimal solution value) left after solving the LP relaxation (column *LP*), the percentage gap left at the root note of the search tree of CPLEX (column *Rt*), and the computing times for solving the LP relaxation (column T_{LP}), the root node (column T_{Rt}), and the ILP formulation itself (column T_{ILP}). For each formulation, we used the LP-method that turned out to be the best one in the experiments summarized in Table 2.1. The time limit for solving the ILP formulation was set equal to 1,800 seconds. In the rows labeled “Avg”, Table 2.2 reports average values and, in parenthesis, the number of instances solved to optimality within the time limit. In the computation of the average computing times, we considered a computing time equal to 1,200 or 1,800 seconds whenever the LP model or the ILP model, respectively, could not be solved within the time limit. For formulation SST the averages refer to the first three instances since the remaining ones could not be run due to exceeded memory limits for CPLEX.

As to the LP-bounds, Table 2.2 shows that SST obtains tight lower bounds but requires larger computing times than the other formulations. Moreover, SST was able to provide the lower bounds, within the time limit, on only 3 of the 10 instances considered. Among the other formulations, the one that provides the tightest LP-bounds is CLAUS, followed by SSB and GP. The other four formulations (i.e., MTZ, GG, DL and SD) obtain, on average, worse LP-bounds than the previous ones but within shorter computing times (less than a second on average). GG was the only formulation able to solve, to optimality, all of the 10 instances within the time limit imposed, whereas MTZ, DL and SD could solve, to optimality, all but one of the instances. By considering the results reported in Table 2.2, we can conclude that, for what concerns the overall computing time, the most effective formulations are MTZ, GG and DL.

The results reported in Table 2.2 show that formulations MTZ, GG, and DL are the best formulations to be directly used within CPLEX. This is probably due to two main reasons: (i) the limited number of constraints required to break subtours allows CPLEX to compute the LP-bounds effectively, and (ii) the large variety of cuts embedded in CPLEX lead to root lower bounds competitive with those that can be obtained with other formulations having stronger LP-bounds but within much shorter computing times. We stress that the computational analysis reported in Table 2.2 is aimed at comparing the suitability of each formulation to be directly solved with CPLEX. Therefore, any conclusion that can be drawn from the results reported in Table 2.2 cannot consider the fitness of each polynomial formulation as the starting point for more complicated exact algorithms for solving the ATSP. We also stress that, for each class of formulations, we report only the results obtained by the formulation providing, on average, the tightest LP-bounds because the other formulations (i.e., those appearing in Figure 2.1 but neglected in this section) were computationally outperformed by at least one of the eight formulations considered in this study.

In Table 2.3, we compare, of the 27 ATSP instances from TSPLIB and the 5 instances provided by Balas, the performance of (a) the branch-and-bound algorithm by Carpaneto et al. [1995] (hereafter CDT) described in §2.4, (b) the branch-and-bound algorithm based on the additive bounding procedure by Fischetti and Toth [1992] (hereafter FT) described in §2.4, (c) the branch-and-cut algorithm by Fischetti et al. [2003] (hereafter FLT) described in §2.4, (d) the branch-and-cut algorithm (hereafter

Concorde) solving the STSP instances obtained from the transformation described in §2.5, and (e) the polynomial formulations MTZ, GG and DL (see §2.3). For each instance and each algorithm, we report the percentage gap of the corresponding lower bound, or the value of the LP relaxation of the considered formulation, computed at the root node of the decision tree (column LB) and the computing time for solving the instance to optimality (column T). The rows labeled "Avg" report average values and, in parenthesis, the number of instances solved to optimality within the corresponding time limit.

Different time limits were imposed on the algorithms: $1,000$ seconds for CDT and FT, $10,000$ seconds for FLT and Concorde, $1,800$ seconds for MTZ, GG and DL. For this reason, in the computation of the average computing time, only the instances solved to optimality within the imposed time limit are considered. The rows labeled "Avg 1,000 sec" report the average computing time and, in parenthesis, the number of instances solved to optimality within the time limit of 1,000 seconds (a computing time equal to 1,000 seconds is considered when this time limit is reached).

The computational results reported for CDT, FT, FLT and Concorde are taken from Fischetti et al. [2004] and were obtained on a Digital Alpha 533 MHz with CPLEX6.5.3 as LP solver, whereas the computational results relative to formulations MTZ, GG and DL were obtained on an Intel Core2 Duo@2.26 GHz by running CPLEX11.2. We have experimentally found that the latter machine is approximately 10-12 times faster than that used by Fischetti et al.; therefore, the computing times reported for MTZ, GG and DL were multiplied by 10 in order to have a fair comparison.

Table 2.3 shows that FLT and Concorde were the only two exact methods able to solve, within the imposed time limit, all of the 32 instances to optimality and are clearly better performing than the other 5 exact methods considered. Although, on the considered instances, Concorde generally obtains better lower bounds at the root node, FLT is always faster. Formulations MTZ, GG and DL proved not to be competitive with either FLT or Concorde because they solved only 20, 11 and 19 instances, respectively, and, on the solved instances, their computing times are always much longer. As for the branch-and-bound algorithms CDT and FT, for which a time limit of 1,000 seconds was imposed, it can be noted that they dominate, with regards to the instances solved to optimality within this time limit and the computing times, the polynomial formulations MTZ, GG and DL. By considering the values of the lower bounds at the root node, Table 2.3 shows that: (i) the lower bound of CDT (given by the value of the AP relaxation) is only slightly worse than that of MTZ (i.e., the addition of constraints (2.10) to the AP relaxation only marginally improves the corresponding LP relaxation); (ii) the lower bound of FT (corresponding to the additive bounding procedure) is always better that those of MTZ and GG, and globally better than that of DL.

Table 2.2: Performance of 8 polynomial formulations on 10 ATSP instances from the TSPLIB

	MTZ					*GG*					*CLAUS*					*DL*				
Inst	*LP*	T_{LP}	*Rt*	T_{Rt}	T_{ILP}	*LP*	T_{LP}	*Rt*	T_{Rt}	T_{ILP}	*LP*	T_{LP}	*Rt*	T_{Rt}	T_{ILP}	*LP*	T_{LP}	*Rt*	T_{Rt}	T_{ILP}
ftv33	7.64	0.02	2.77	0.22	4.91	7.03	0.06	0.78	0.83	5.51	0.00	7.20	0.00	9.44	14.88	5.35	0.03	4.58	0.11	3.40
ftv35	6.12	0.02	4.15	0.09	6.27	5.59	0.06	0.63	3.51	11.93	1.06	9.17	1.00	19.02	79.36	4.04	0.05	3.58	0.16	5.23
ftv38	5.87	0.02	2.91	0.14	9.02	5.44	0.08	0.80	2.29	21.46	1.02	12.50	0.96	31.01	74.88	3.45	0.03	3.14	0.17	5.71
ftv44	5.55	0.02	2.68	0.14	16.40	5.18	0.10	1.74	2.47	18.02	1.74	23.71	1.27	77.38	tl	2.43	0.03	2.10	0.20	3.17
ftv47	6.77	0.02	2.39	0.28	36.04	6.54	0.11	1.84	1.83	58.27	1.54	37.43	1.52	256.06	tl	2.83	0.06	2.08	0.23	12.00
ftv55	10.55	0.03	3.89	0.27	43.74	10.31	0.16	4.97	1.19	91.16	1.49	87.38	1.43	324.61	tl	6.05	0.06	3.89	0.39	19.48
ftv64	6.31	0.05	4.21	0.28	114.50	5.84	0.22	4.00	1.98	322.56	1.71	182.06	1.63	1,368.27	tl	4.24	0.06	4.03	0.67	32.53
ftv70	9.26	0.05	3.94	0.59	168.70	8.75	0.27	7.06	2.51	1,125.57	2.10	285.50	1.99	tl	tl	4.69	0.06	4.05	0.87	52.40
ft70	1.77	0.06	1.16	0.64	tl	1.10	0.25	0.85	3.74	326.07	0.05	237.47	0.04	1,500.99	tl	0.88	0.08	0.64	0.89	65.22
ft53	14.04	0.05	10.88	0.44	312.38	12.45	0.14	0.00	3.65	37.95	0.00	68.13	0.00	255.02	265.67	12.93	0.05	10.97	0.38	tl
Avg	7.39	0.03	3.90	0.31	251.20	6.82	0.14	2.27	2.40	201.85	1.07	95.05	0.98	564.18	1,123.48	4.69	0.05	3.91	0.41	199.91
					(9)					(10)					(4)					(9)

	GP					SD					SSB					SST				
Inst	*LP*	T_{LP}	*Rt*	T_{Rt}	T_{ILP}	*LP*	T_{LP}	*Rt*	T_{Rt}	T_{ILP}	*LP*	T_{LP}	*Rt*	T_{Rt}	T_{ILP}	*LP*	T_{LP}	*Rt*	T_{Rt}	T_{ILP}
ftv33	0.00	0.85	0.00	1.97	8.21	4.78	0.25	4.53	0.61	26.91	0.00	0.41	0.00	0.71	0.77	0.00	277.58	0.00	496.90	496.90
ftv35	1.29	0.85	1.29	3.32	914.01	3.90	0.28	3.35	0.70	49.75	1.09	0.46	1.02	2.64	14.88	0.65	632.53	0.65	tl	tl
ftv38	1.24	1.32	1.24	5.38	tl	3.26	0.39	3.15	1.22	81.67	1.05	0.65	0.98	2.48	31.50	0.64	737.40	0.64	tl	tl
ftv44	1.84	1.58	1.26	8.24	tl	2.43	0.44	2.03	0.97	97.31	1.74	0.88	0.92	21.36	91.24	-	-	-	-	-
ftv47	1.86	1.63	1.53	6.27	tl	2.75	0.72	2.71	1.51	109.82	1.86	2.52	1.54	32.08	1,247.54	-	-	-	-	-
ftv55	2.33	3.85	1.12	83.12	tl	5.89	0.53	5.62	4.18	1,105.22	2.33	2.80	1.41	261.42	999.26	-	-	-	-	-
ftv64	3.30	6.64	2.99	189.60	tl	4.00	0.84	3.68	4.79	309.85	2.88	14.92	2.70	691.29	tl	-	-	-	-	-
ftv70	3.73	8.11	3.37	53.06	tl	4.64	2.22	4.45	6.99	525.08	3.12	27.54	2.63	476.80	tl	-	-	-	-	-
ft70	1.15	7.24	1.09	44.94	tl	0.80	1.36	0.74	7.46	236.81	0.55	14.66	0.48	38.21	tl	-	-	-	-	-
ft53	10.69	2.75	9.33	16.73	tl	11.39	0.48	11.26	1.47	tl	10.63	1.94	9.22	8.18	tl	-	-	-	-	-
Avg	2.74	3.48	2.32	41.26	1,532.22	4.38	0.75	4.15	2.99	434.24	2.53	6.68	2.09	153.51	958.52	0.43	549.17	0.43	1,365.63	1,365.63
					(2)					(9)					(6)					(1)

TABLE 2.3: Comparison between branch-and-bound algorithms, branch-and-cut algorithms and polynomial formulations

	CDT		*FT*		*FLT*		*Concorde*		*MTZ*		*GG*		*DL*	
Inst	*LB*	*T*	*LB*	*T*	*LB*	*T*	*LB*	*T*	*LB*	*T*	*LB*	*T*	*LB*	*T*
br17	100.00	3.6	0.00	0.0	0.00	0.0	0.00	0.2	94.23	10.9	68.91	8.2	43.59	34.6
ft53	14.11	tl	1.56	0.2	0.00	0.1	0.00	0.6	14.04	3,123.8	12.45	379.5	12.93	tl
ft70	1.80	0.4	0.57	0.3	0.02	0.2	0.01	3.2	1.77	tl	1.10	3,260.7	0.88	652.2
ftv33	7.85	0.0	3.73	0.1	0.00	0.0	0.00	0.3	7.64	49.1	7.03	55.1	5.35	34.0
ftv35	6.25	0.0	3.53	0.2	0.88	0.4	0.68	9.0	6.12	62.7	5.59	119.3	4.04	52.3
ftv38	6.01	0.0	3.01	0.3	0.85	0.6	0.52	14.5	5.87	90.2	5.44	214.6	3.45	57.1
ftv44	5.70	0.0	4.46	0.1	0.37	0.5	0.12	9.1	5.55	164.0	5.18	180.2	2.43	31.7
ftv47	6.98	0.1	3.49	0.4	1.01	0.5	0.62	23.4	6.77	360.4	6.54	582.7	2.83	120.0
ftv55	10.76	1.1	6.59	1.5	0.81	1.4	0.44	9.0	10.55	437.4	10.31	911.6	6.05	194.8
ftv64	6.42	0.8	4.89	1.3	1.36	2.6	0.33	20.8	6.31	1,145.0	5.84	3,225.6	4.24	325.3
ftv70	9.44	3.3	6.92	3.7	0.92	1.1	0.26	17.8	9.26	1,687.0	8.75	11,255.7	4.69	524.0
ftv90	6.33	0.7	3.17	2.5	0.25	0.5	0.06	14.7	6.24	340.4	6.21	tl	1.72	498.1
ftv100	6.60	16.6	3.91	29.6	0.39	2.2	0.00	12.6	6.53	1,083.1	6.49	tl	3.11	1,237.5
ftv110	5.87	5.0	4.49	38.4	0.77	7.4	0.05	25.6	5.81	5,964.9	5.71	tl	2.41	tl
ftv120	6.51	50.1	5.77	99.4	0.97	13.1	0.28	54.4	6.44	8,755.0	6.32	tl	2.18	tl
ftv130	4.46	6.4	3.77	16.6	0.35	1.6	0.00	16.6	4.42	1,938.2	4.34	tl	1.31	600.7
ftv140	4.92	13.9	4.26	38.8	0.25	2.1	0.00	25.6	4.87	1,127.6	4.85	tl	1.41	287.3
ftv150	3.91	3.1	3.03	17.0	0.27	2.6	0.00	27.0	3.87	920.6	3.86	tl	1.31	402.7
ftv160	4.58	93.8	4.03	316.2	0.67	3.8	0.30	55.7	4.55	12,731.4	4.54	tl	2.35	6,000.6
ftv170	4.50	tl	4.14	tl	0.87	4.1	0.40	41.9	4.48	tl	4.47	tl	2.05	tl
kro124p	6.22	tl	2.73	135.7	0.04	1.0	0.00	9.9	6.13	2,296.6	5.47	tl	3.46	2,316.8
p43	97.37	tl	0.37	tl	0.16	9.3	0.16	22.7	97.34	tl	85.23	tl	96.16	tl
rbg323	0.00	0.1	0.00	0.3	0.00	0.4	0.00	23.9	0.00	tl	0.00	tl	0.00	8,481.1
rbg358	0.00	0.1	0.00	0.5	0.00	0.5	0.00	29.3	0.00	tl	0.00	tl	0.00	tl
rbg403	0.00	0.1	0.00	1.0	0.00	1.3	0.00	49.3	0.00	tl	0.00	tl	0.00	tl
rbg443	0.00	0.1	0.00	1.2	0.00	1.4	0.00	34.5	0.00	tl	0.00	tl	0.00	tl
ry48p	13.21	tl	2.94	20.3	0.53	0.8	0.35	22.9	12.88	1,316.0	11.17	tl	4.25	3,417.3
balas84	14.07	tl	5.53	986.6	1.01	15.7	1.01	78.0	13.98	tl	12.34	tl	9.67	tl
balas108	25.00	tl	9.87	tl	1.97	89.0	2.63	1,416.0	24.80	tl	18.14	tl	19.50	tl
balas120	21.68	tl	13.29	tl	1.05	1,276.3	1.05	7,186.9	21.43	tl	18.58	tl	16.86	tl
balas160	19.40	tl	11.34	tl	1.26	671.1	1.26	7,848.0	19.18	tl	16.16	tl	18.83	tl
balas200	15.63	tl	8.68	tl	1.24	1,712.8	0.74	2,294.2	15.55	tl	12.86	tl	15.31	tl
Avg	13.61	9.06	4.06	65.85	0.57	119.51	0.35	606.18	13.33	2,180.21	11.37	1,835.75	9.14	1,329.90
		(22)		(26)		(32)		(32)		(20)		(11)		(19)
Avg	-	318.7	-	241.0	-	88.6	-	145.4	-	794.9	-	826.6	-	681.7
$1,000$ sec		(22)		(26)		(30)		(28)		(9)		(8)		(14)

Chapter 3

Traveling Salesman Problem with Time Windows

[1]

The *traveling salesman problem with time windows* (TSPTW) is the problem of finding, in a weighted digraph, a least-cost tour starting from a selected vertex, visiting each vertex of the graph exactly once within a given time window, and returning to the starting vertex. This chapter introduces new tour relaxations to compute valid lower bounds on the TSPTW. The optimal integer TSPTW solution is computed with a dynamic programming algorithm that uses bounding functions based on the different tour relaxations and the dual solutions obtained. An extensive computational analysis on basically all TSPTW instances (involving up to 233 vertices) from the literature is reported. The results show that the proposed algorithm solves all but one instances and outperforms all exact methods published in the literature so far.

3.1 Introduction

The *traveling salesman problem with time windows* (TSPTW) is defined on a digraph $G = (V', A)$, where V' is the vertex set and A is the arc set. The vertex set V' is equal to $V \cup \{p, q\}$, where the set V contains n vertices $1, 2, \ldots, n$, and vertices p and q are two special vertices. We indicate with $\Gamma_i \subseteq V'$ the set of *successors* (i.e., $\Gamma_i = \{j \in V' : (i, j) \in A\}$) and with $\Gamma_i^{-1} \subseteq V'$ the set of *predecessors* (i.e., $\Gamma_i^{-1} = \{j \in V' : (j, i) \in A\}$) of vertex $i \in V'$ in graph G. We assume that $\Gamma_p^{-1} = \Gamma_q = \varnothing$ and $(p, q) \notin A$.

A time window $[e_i, l_i]$ is associated with each vertex $i \in V'$, where e_i and l_i represent the earliest and latest time to visit vertex i, respectively. A travel cost d_{ij} and a travel

[1]This chapter is based on Baldacci et al. [2011b]

time t_{ij} are associated with each arc $(i, j) \in A$; the travel time t_{ij} includes the service time at vertex i. Travel times t_{ij} and time windows $[e_i, l_i]$ are assumed integer values.

A *salesman tour* is a path in G that starts from vertex p at time e_p, visits each vertex $i \in V$ within its time window, and ends at vertex q before time l_q. The salesman is allowed to arrive at a vertex $i \in V$ before time e_i, but, in this case, the service of vertex i is postponed until time e_i. Hereafter, we assume that graph G contains at least one salesman tour. The cost of a salesman tour is the sum of the travel costs of the arcs traversed. The TSPTW consists of finding a minimum-cost salesman tour.

The problem is $\mathcal{NP}$-hard because it generalizes the classical *traveling salesman problem* (TSP). Even finding a feasible solution is $\mathcal{NP}$-complete (see Savelsbergh [1985]).

The TSPTW has many practical applications, such as in single and multiple vehicle problems, control of stacker cranes in warehouses, bank deliveries, and postal deliveries.

3.2 Literature Review

The first exact algorithms are due to Christofides et al. [1981a] and Baker [1983]. They proposed branch-and-bound methods to solve a variant of the problem where the total schedule time must be minimized. The algorithm of Christofides et al. [1981a] was based on the technique introduced by Christofides et al. [1981c] called *state-space relaxation* (SSR), whereby the state-space associated with a given *dynamic programming* (DP) recursion is relaxed into a space of smaller cardinality in such a way that the solution of the relaxed recursion provides a lower bound to the original problem. Christofides et al. [1981a] and Baker [1983] reported solutions of TSPTW instances with up to 50 vertices.

Langevin et al. [1993] proposed a branch-and-bound algorithm based on a two-commodity flow formulation and reported solutions of instances with up to 60 vertices.

Algorithms based on DP were proposed by Dumas et al. [1995], Mingozzi et al. [1997], Balas and Simonetti [2001], and Li [2009]. Dumas et al. [1995] described a DP algorithm that applies sophisticated elimination tests to reduce the state-space and the number of state transitions; they reported the solutions of instances involving up to 200 vertices. Mingozzi et al. [1997] presented a DP algorithm for the TSP with time windows and precedence constraints based on the SSR technique and presented computational results for instances with up to 120 vertices. Balas and Simonetti [2001] considered a special case of the TSPTW where, for some initial ordering of the vertices, vertex i precedes vertex j if $j \geq i + k$ (for some value $k > 0$), and described a DP algorithm that is linear in the number of vertices and exponential in k. Li [2009] presented a DP algorithm based on a bi-directional resource-bounded label correcting algorithm (see Righini and Salani [2006]); Li reported the solutions of instances involving up to 233 vertices and solved a number of open instances to optimality.

Branch-and-cut methods for the TSPTW were proposed by Ascheuer et al. [2001] and, recently, by Dash et al. [2012]. Ascheuer et al. [2001] described heuristic algorithms and branch-and-cut methods to solve a real-world application, called *stacker crane optimization*, that can be modeled as TSPTW. The cutting plane algorithms described by Ascheuer et al. [2001] use three alternative integer programming formulations of the TSPTW and are based on the formulations and polyhedral analysis presented in Ascheuer et al. [2000]. Ascheuer et al. [2001] reported computational results for real-world instances with up to 233 vertices, showing that most TSPTW instances with up to 70 vertices can be solved to optimality by any of the three models. Dash et al. [2012] introduced an extended formulation for the TSPTW based on partitioning the time windows into sub-windows, called *buckets*; they described a branch-and-cut algorithm that uses the valid inequalities described in Ascheuer et al. [2000, 2001] and new valid inequalities specific for the bucket formulation. Dash et al. reported the solutions of instances with up to 233 vertices and solved a number of open instances.

Focacci et al. [2002] proposed a hybrid approach for solving the TSPTW merging constraint programming algorithms and optimization techniques to compute bounds on the optimal solution value. Their algorithm solved instances with up to 69 vertices.

Heuristic algorithms were proposed, among others, by Gendreau et al. [1998], Wolfler Calvo [2000], Ohlmann and Thomas [2007], López Ibáñez and Blum [2010], and da Silva and Urrutia [2010].

3.3 An Exact Algorithm for the TSPTW

We assume that matrix t_{ij} satisfies the triangle inequality, so the time windows and the arc set A can be reduced by using the reduction rules described by Dash et al. [2012] and summarized in the following.

We denote by $\vec{G} = (V', \vec{A})$ the precedence digraph, where arc $(i, j) \in \vec{A}$ indicates that vertex $i \in V'$ must precede vertex $j \in V'$ in any salesman tour. The arc set $\vec{A}$ is obtained by (i) setting $\vec{A} = \Gamma_p \cup \Gamma_q^{-1}$ and adding arc $(i, j) \in A$ to $\vec{A}$ for any couple of vertices $i, j \in V$ such that $e_j + t_{ji} > l_i$; (ii) adding, to $\vec{A}$, any arc $(i, j) \in A$ if there exists a vertex $k \in V \setminus \{i, j\}$ such that both arcs (i, k) and (k, j) belong to $\vec{A}$. The latter operation is repeated until no more arcs are added to $\vec{A}$. Then, for any arc $(i, j) \in \vec{A}$, the arc (j, i) is removed from the arc set A of graph G because it cannot belong to any salesman tour.

We indicate with $\vec{\Gamma}_i \subseteq V'$, $i \in V'$, the set of successors and with $\vec{\Gamma}_i^{-1} \subseteq V'$, $i \in V'$, the set of predecessors of vertex $i \in V'$ in graph $\vec{G}$. With each arc $(i, j) \in A$, we associate a time $\vec{t}_{ij}$, which represents a lower bound on the minimum time spent between the visits to vertices i and j in any salesman tour, computed as the cost of the longest path in $\vec{G}$ from vertex i to vertex j using travel time t_{ij} as cost of arc $(i,j) \in \vec{A}$. We assume $\vec{t}_{ii} = 0$, $i \in V$.

A *forward path* $F = (p, i_1, \ldots, i_k = \sigma_F)$ is an elementary path that starts from vertex p at time e_p, visits vertices $V(F) = \{p, i_1, \ldots, i_k\}$ within their time windows, and ends at vertex σ_F at time $t_F \in [e_{\sigma_F}, l_{\sigma_F}]$. The cost, $c(F)$, of a forward path F is the sum of the costs of the traversed arcs. Let $f(S,t,i)$ be the cost of a least-cost forward path F that visits the set of vertices $S \subseteq V'$ and ends at vertex $i \in S$ at time $t \in [e_i, l_i]$. The cost z^* of a TSPTW optimal solution is given by

$$z^* = \min_{t \in [e_q, l_q]} \{f(V', t, q)\}.$$

Let $\Omega(t,j,i)$, $j,i \in V'$, $t \in [e_i, l_i]$, be the set of departure times from vertex j to arrive at vertex i at time t if j is visited immediately before i. The set $\Omega(t,j,i)$ is defined as

$$\Omega(t,j,i) = \begin{cases} t' \,:\, t' \in [e_j, \min\{l_j, t - t_{ji}\}]\}, & \text{if } t = e_i, \\ t - t_{ji} \,:\, t - t_{ji} \in [e_j, l_j], & \text{if } t \in (e_i, l_i]. \end{cases}$$

Functions $f(S,t,i)$ can be computed with DP on a graph $\mathscr{G}_F = (\mathscr{F}, \mathscr{A}_F)$ as follows. The vertex set $\mathscr{F}$ represents the states of the DP recursion and is defined as

$$\mathscr{F} = \{(S,t,i) \,:\, S \subseteq V',\, i \in S,\, t \in [e_i, l_i]\}.$$

The arc set $\mathscr{A}_F$ represents the transitions of the DP recursion and is defined as

$$\mathscr{A}_F = \{((S',t',j),(S,t,i)) \,:\, (S',t',j),(S,t,i) \in \mathscr{F},\, S' = S \backslash \{i\},\, t' \in \Omega(t,j,i),\, j \in \Gamma_i^{-1} \cap S\}.$$

A possible DP recursion for computing functions $f(S,t,i)$ is

$$f(S,t,i) = \min_{(S',t',j) \in \mathscr{F} \,:\, ((S',t',j),(S,t,i)) \in \mathscr{A}_F} \{f(S',t',j) + d_{ji}\}, \quad (S,t,i) \in \mathscr{F}. \tag{3.1}$$

It is required to initialize $f(\{p\}, e_p, p) = 0$ and $f(\{p\}, t, p) = \infty$, for each $t \in (e_p, l_p]$.

The size of the path set $\mathscr{F}$ can be reduced by removing any state $(S,t,i) \in \mathscr{F}$ that cannot lead to any optimal TSPTW solution according to the following rules.

Dominance 1. Let (S,t,i), $(S,t',i) \in \mathscr{F}$ be two states such that $f(S,t,i) \geq f(S,t',i)$ and $t > t'$, then (S,t,i) is dominated by (S,t',i).

Fathoming 1. Any state $(S,t,i) \in \mathscr{F}$ cannot lead to any feasible solution if (i) there exists a vertex $j \in V' \setminus S$ such that $t + \vec{t}_{ij} > l_j$ or (ii) $\vec{\Gamma}_i^{-1} \not\subseteq S$.

Fathoming 2. Let z_{UB} be an upper bound on the TSPTW, and let $b(S,t,i)$ be a lower bound on the cost of a least-cost path that starts from vertex $i \in S$ at time $t \in [e_i, l_i]$, visits each vertex in $V' \setminus S$ within its time window, and ends at vertex q before time

l_q. Any state (S, t, i) such that $f(S, t, i) + b(S, t, i) \geq z_{UB}$ cannot lead to any optimal solution.

In §3.6, we describe three methods for computing bounding functions $b(S, t, i)$ based on three SSRs of recursion (3.1) and on a valid lower bound LB described in §3.5.

To avoid the a priori computation of an upper bound z_{UB}, we use an iterative algorithm where, at iteration h, recursion (3.1) is computed and Fathoming 2 is applied using a not necessarily valid upper bound z^h_{UB}. The exact algorithm can be outlined as follows.

Step 1. Compute a lower bound LB as described in §3.5, and set $z^0_{UB} = LB$. Initialize $h = 1$, $\theta = \max\{\lceil 10^{-3}LB \rceil, 1\}$, and $z^1_{UB} = \lceil LB \rceil + \theta$.

Step 2. Compute DP recursion (3.1) by using the bounding functions $b(S, t, i)$ described in §3.6 and z^h_{UB} instead of z_{UB} in Fathoming 2.

Step 3. There are two possible cases.

i) *No state* (V', t, q)*, with* $t \in [e_q, l_q]$*, was generated* (this happens if $z^* > z^h_{UB}$). Set $h = h + 1$, $z^h_{UB} = z^{h-1}_{UB} + \theta$, and go to Step 2.

ii) *At least a state* (V', t, q)*, with* $t \in [e_q, l_q]$*, was generated.* The optimal TSPTW solution cost is $z^* = \min_{t \in [e_q, l_q]}\{f(V', t, q)\}$.

The algorithm can terminate prematurely at iteration h, while computing recursion (3.1) at Step 2, if the number of generated states exceeds the computer memory available. In this case, z^{h-1}_{UB} represents a valid lower bound on the TSPTW.

3.4 Relaxations of the TSPTW

We describe three relaxations of salesman tours, called t-tour, ng-tour and ngL-tour relaxations, that are used to compute valid lower bounds on the TSPTW and bounding functions $b(S, t, i)$ used by Fathoming 2.

3.4.1 The t-Tour Relaxation

The t-tour relaxation was introduced by Christofides et al. [1981c]. This relaxation provides function $f(t, i)$ that corresponds to the cost of a least-cost nonnecessarily elementary path, called (t, i)-path, that starts from vertex p at time e_p, visits a set of vertices (without two-vertex loops) within their time windows, and ends at vertex $i \in V'$ at time $t \in [e_i, l_i]$. The DP recursion for computing functions $f(t, i)$ is described in Christofides et al. A t-tour is the (t^*, q)-path such that $f(t^*, q) = \min_{t \in [e_q, l_q]}\{f(t, q)\}$.

3.4.2 The ng-Tour Relaxation

In general, the t-tour relaxation does not provide any detailed knowledge of the path of cost $f(t, i)$, so additional conditions to ensure that such path provides a feasible solution to the original problem cannot be imposed. To alleviate this drawback, we introduce the ng-path relaxation that consists of partitioning the set of all possible (t, i)-paths ending at vertex $i \in V'$ at time $t \in [e_i, l_i]$ according to a mapping function that associates with each (t, i)-path a subset of the visited vertices depending on the order in which they are visited. The subset of vertices associated with each ng-path is used in the DP recursion to impose partial elementarity. The ng-tour relaxation can be described as follows.

Let $N_i \subseteq V'$ be a set of selected vertices for vertex $i \in V'$ (according to some criterion) such that $i \in N_i$ and $|N_i| \leq \Delta(N_i)$, where $\Delta(N_i)$ is a parameter. The sets N_i allow us to associate, with each forward path $F = (p, i_1, \ldots, i_k = \sigma_F)$, the set $\Pi_F \subseteq V(F)$ containing vertex i_k and every vertex $i_r \in V(F)$, $r = 1, .., k-1$, that belongs to all sets $N_{i_{r+1}}, \ldots, N_{i_k}$ associated with the vertices $i_{r+1}, \ldots, i_k$ visited after i_r in path F. Formally, the set Π_F is defined as

$$\Pi_F = \{i_r \in V(F) \setminus \{i_k\} \, : \, i_r \in \bigcap_{s=r+1}^{k} N_{i_s}\} \bigcup \{i_k\}.$$

A *forward ng-path* (NG, t, i) is a nonnecessarily elementary path $F = (p, i_1, \ldots, i_{k-1}, i_k = i)$ that starts from vertex p at time e_p, visits a set of vertices (each once or more) within their time windows, ends at vertex $i \in V'$ at time $t \in [e_i, l_i]$, and such that $NG = \Pi_F$ and $i \notin \Pi_{F'}$, where $F' = (p, i_1, \ldots, i_{k-1})$ is an ng-path.

Notice that an ng-path can contain a loop $(i_r = j,\ i_{r+1}, \ldots, i_{r+s} = j)$ for $s \geq 2$ if and only if there exists at least one index k such that $2 \leq k \leq s-1$ and $j \notin N_{i_{r+k}}$.

An example of ng-path is the following. Consider path $F = (p, 1, 2, 3, 4, 1)$. The corresponding path F' is $F' = (p, 1, 2, 3, 4)$. Suppose that sets N_i, $i = 1, 2, 3, 4$, are defined as $N_1 = \{1, 3, 4\}$, $N_2 = \{1, 2, 5\}$, $N_3 = \{1, 3, 4\}$, $N_4 = \{2, 3, 4\}$. The set $\Pi_{F'}$ contains vertices 3 and 4 only because $1 \notin N_2 \cap N_3 \cap N_4$ and $2 \notin N_3 \cap N_4$ whereas $3 \in N_4$ and 4 is the last vertex visited by path F'. Being elementary, path F' is clearly an ng-path. Thus, because $1 \notin \Pi_{F'}$, path F is an ng-path. Moreover, the set Π_F contains vertices 1, 3 and 4 only because $2 \notin N_3 \cap N_4 \cap N_1$ whereas $3 \in N_4 \cap N_1$, $4 \in N_1$ and 1 is the last vertex visited by path F; this means that path F can be propagated towards all vertices but 1, 3 and 4.

We denote by $f(NG, t, i)$ the cost of a least-cost forward ng-path (NG, t, i). Any (NG, t, q)-path ending at vertex q at time $t \in [e_q, l_q]$ is called ng-tour.

A valid lower bound LB on the TSPTW is given by

$$LB = \min_{t \in [e_q, l_q],\ NG \subseteq N_q} \{f(NG, t, q)\} \leq z^*. \tag{3.2}$$

Functions $f(NG, t, i)$ can be computed with DP on a graph $\widehat{\mathscr{G}_F} = (\widehat{\mathscr{F}}, \widehat{\mathscr{A}_F})$. The vertex set $\widehat{\mathscr{F}}$ represents the states of the DP recursion and is defined as

$$\widehat{\mathscr{F}} = \{(NG, t, i) \, : \, i \in V', \ t \in [e_i, l_i], \, \forall NG \subseteq N_i \text{ s.t. } NG \ni i\}.$$

The arc set $\widehat{\mathscr{A}_F}$ represents the transitions of the DP recursion and is defined as

$$\widehat{\mathscr{A}_F} = \{((NG', t', j), (NG, t, i)) \, : \, (NG', t', j), (NG, t, i) \in \widehat{\mathscr{F}},$$

$$j \in \Gamma_i^{-1}, \, t' \in \Omega(t, j, i), \, \forall NG' \subseteq N_j \text{ s.t. } NG' \ni j \text{ and } NG' \cap N_i = NG \setminus \{i\}\}.$$

A possible DP recursion for computing functions $f(NG, t, i)$ is

$$f(NG, t, i) = \min_{(NG', t', j) \in \widehat{\mathscr{F}} \, : \, ((NG', t', j), (NG, t, i)) \in \widehat{\mathscr{A}_F}} \{f(NG', t', j) + d_{ji}\}, \quad (NG, t, i) \in \widehat{\mathscr{F}}. \tag{3.3}$$

Notice that the condition $NG' \cap N_i = NG \setminus \{i\}$ in the definition of $\widehat{\mathscr{A}_F}$ imposes that functions $f(NG, t, i)$ are computed by propagating functions $f(NG', t', j)$ such that $i \notin NG'$. Moreover, notice that the following inequality holds for each forward path F

$$c(F) \geq \min_{NG \subseteq V(F) \cap N_{\sigma_F}} \{f(NG, t_F, \sigma_F)\}.$$

It is required to initialize $f(\{p\}, e_p, p) = 0$ and $f(\{p\}, t, p) = \infty$, for each $t \in (e_p, l_p]$.

An optimal ng-tour is the ng-path (NG^*, t^*, q) such that

$$f(NG^*, t^*, q) = \min_{(NG, t, q) \in \widehat{\mathscr{F}}} \{f(NG, t, q)\}. \tag{3.4}$$

The size of the state set $\widehat{\mathscr{F}}$ can be reduced by removing, via the following dominance rule, any state that cannot lead to an optimal ng-tour.

Dominance 2. Let $(NG, t, i), (NG', t', i) \in \widehat{\mathscr{F}}$ such that $f(NG, t, i) \geq f(NG', t', i)$, $NG \supseteq NG'$, $t > t'$. State (NG, t, i) is dominated by (NG', t', i).

Notice that a stronger version of Dominance 2 can be obtained by replacing the condition $t < t'$ with $t \leq t'$. However, we found it to be computationally convenient to apply Dominance 2 instead of this stronger version.

The *decremental state-space relaxation* introduced by Righini and Salani [2008] and the *partial elementarity relaxation* introduced by Desaulniers et al. [2008] are special cases of the ng-path relaxation. These relaxations can be obtained by setting $N_i = \widehat{V}$, for each vertex $i \in V'$, where $\widehat{V} \subseteq V'$ is a selected set of vertices.

Choosing the sets N_i of selected vertices. Lower bound LB computed by expression (3.2) depends on the sets N_i, $i \in V'$. If $N_i = V'$, for each vertex $i \in V'$, functions $f(NG, t, i)$ provide the costs of least-cost elementary paths and the optimal ng-tour provided by expression (3.4) corresponds to an optimal TSPTW solution.

The computational results of §3.7 were obtained by defining the sets N_i, $i \in V'$, as follows. We set $N_p = \{p\}$ and $N_q = \{q\}$, and define $T_i = \{j \in \Gamma_i \cap \Gamma_i^{-1} : t_{ij} = 0 \text{ or } t_{ji} = 0\}$, for each vertex $i \in V$ (notice that the set T_i contains vertex i). First, we set $N_i = T_i$, for each vertex $i \in V$. Then, for each vertex $i \in V$ such that $|N_i| < \Delta(N_i)$, we add, to N_i, the $(\Delta(N_i) - |T_i|)$-nearest vertices to i belonging to the set $V \setminus T_i$ according to travel times t_{ij}.

Notice that recursion (3.3) allows ng-paths to contain two-vertex loops, which can be eliminated with the method described by Christofides et al. [1981c]. Nonetheless, we do not implement this method because of the additional memory required and because, in practice, whenever sets N_i, $i \in V'$, are defined as above and parameter $\Delta(N_i)$ is equal to 11 or 13, the resulting (NG, t, i)-paths rarely contain two-vertex loops.

Implementation issues. How the DP recursion (3.3) is computed and how Dominance 2 is tested affect the effectiveness of the whole solution process. We decided to compute (3.3) with a forward DP recursion using the variable state t (i.e., the time) as stage. At stage t, the order in which states are propagated is defined by a queue where states having time t are maintained.

To apply Dominance 2, we use an additional function $F_t(NG, i)$, $i \in V$, $NG \subseteq N_i$ s.t. $NG \ni i$, that represents the cost of a least-cost state $(NG, t', i) \in \widehat{\mathscr{F}}$ with $t' < t$ (i.e., $F_t(NG, i) = \min_{t' \in [e_i, t)}\{f(NG, t', i) : (NG, t', i) \in \widehat{\mathscr{F}}\}$). Let $\mathscr{B}(NG, i) = \{NG' \subseteq NG : NG' \ni i\}$, for each vertex $i \in V$ and each set $NG \subseteq N_i$ s.t. $NG \ni i$. Because we use values of parameter $\Delta(N_i)$ less than or equal to 13, the sets $\mathscr{B}(NG, i)$ can be computed by complete enumeration before starting the DP recursion. From the definitions of $F_t(NG, i)$ and $\mathscr{B}(NG, i)$, we derive that a state (NG, t, i) is dominated according to Dominance 2 if $f(NG, t, i) \geq \min_{NG' \in \mathscr{B}(NG, i)}\{F_t(NG', i)\}$. Notice that function $F_t(NG, i)$ can be recursively computed as

$$F_t(NG, i) = \min\{F_{t-1}(NG, i), f(NG, t-1, i)\}, \quad i \in V,\ NG \subseteq N_i \text{ s.t. } NG \ni i.$$

3.4.3 The ngL-Tour Relaxation

In the case of the TSPTW, the ng-tour relaxation can be enhanced by forcing any ng-tour to visit each vertex of a selected subset of vertices exactly once while satisfying the precedence constraints imposed for such vertices by the precedence digraph $\vec{G}$.

Consider a path $L = (p = i_0, i_1, \ldots, i_h = q)$ in $\vec{G}$ from vertex p to vertex q. Because any arc $(i,j) \in \vec{A}$ of graph $\vec{G}$ implies that vertex i must be visited before vertex j in any salesman tour, the h arcs of path L decompose any salesman tour in h subpaths $P_{i_{k-1}i_k}$, $k = 1, \ldots, h$, where $P_{i_{k-1}i_k}$ corresponds to the subpath of the salesman tour that starts from vertex i_{k-1}, visits each vertex of subset $S_k \subseteq V' \setminus V(L)$ exactly once, and ends at vertex i_k. Thus, any salesman tour is made up of any collection of h subpaths $P_{i_{k-1}i_k}$, $k = 1, \ldots, h$, such that $\cup_{k=1}^{h} S_k = V' \setminus V(L)$. For a given path L of $\vec{G}$ defined as above, we can force any ng-tour to visit, exactly once, each vertex $i \in V(L)$ as described in the following.

Let V_k be the set of vertices that must or can be visited in between vertices i_{k-1} and i_k $(i_{k-1}, i_k \in L)$. For each $k = 1, \ldots, h$, we have

$$V_k = \{j \in V \setminus \{i_{k-1}\} \, : \, e_{i_{k-1}} + \vec{t}_{i_{k-1}j} \leq l_j \text{ and } \max\{e_{i_{k-1}} + \vec{t}_{i_{k-1}j}, e_j\} + \vec{t}_{ji_k} \leq l_{i_k}\}.$$

Notice that $V_k \cap V(L) = i_k$.

A *forward ngL(k)-path* is an ng-path (NG, t, i) that ends at vertex $i \in V_k$ at time $t \in [e_i, l_i]$, and such that each vertex $j \in \{i_0, i_1, \ldots, i_{k-1}\}$ is visited exactly once. An ngL-tour is an $ngL(h)$-path (NG, t, q).

Let $\widehat{\mathscr{F}_k} = \{(NG, t, i) \in \widehat{\mathscr{F}} \, : \, i \in V_k\}$ be the subset of states corresponding to the forward $ngL(k)$-paths, and let $f_k(NG, t, i)$ be the cost of a least-cost $ngL(k)$-path $(NG, t, i) \in \widehat{\mathscr{F}_k}$.

Functions $f_k(NG, t, i)$, $k = 1, \ldots, h$, $i \in V'$, $NG \subseteq N_i$ s.t. $NG \ni i$, $t \in [e_i, l_i]$, can be computed with the following iterative procedure that, for each $k = 1, \ldots, h$, performs the following operations

a) Initialize $f_k(NG, t, i_{k-1}) = f_{k-1}(NG, t, i_{k-1})$, for each state $(NG, t, i_{k-1}) \in \widehat{\mathscr{F}_{k-1}}$, and $f_k(NG, t, i) = \infty$, for each state $(NG, t, i) \in \widehat{\mathscr{F}_k}$. We assume $f_0(\{p\}, e_p, p) = 0$ and $f_0(\{p\}, t, p) = \infty$, for each $t \in (e_p, l_p]$;

b) Compute $f_k(NG, t, i)$, $(NG, t, i) \in \widehat{\mathscr{F}_k}$, as follows

$$f_k(NG, t, i) = \min_{\substack{(NG',t',j) \in \widehat{\mathscr{F}_k} \, : \, j \in \Gamma_i^{-1}, \, t' \in \Omega(t,j,i), \\ NG' \subseteq N_j \text{ s.t. } NG' \ni j, \, NG' \cap N_i = NG \setminus \{i\}}} \{f_k(NG', t', j) + d_{ji}\}. \tag{3.5}$$

Because of the initialization and of the definition of the sets $V_k, k = 1, \ldots, h$, any $ngL(h)$-path corresponding to $f_h(NG, t, q)$ visits every vertex of the set $V(L)$ exactly once.

An optimal ngL-tour is the $ngL(h)$-path $(NG^*, t^*, q) \in \widehat{\mathscr{F}_h}$ such that

$$f_h(NG^*, t^*, q) = \min_{(NG,t,i) \in \widehat{\mathscr{F}_h}} \{f_h(NG, t, i)\}.$$

The value $f_h(NG^*, t^*, q^*)$ depends on the path L of $\vec{G}$ used. In our computational experiments, the path L was chosen as the path in $\vec{G}$ from vertex p to vertex q of maximal cardinality in order to maximize the number of vertices visited exactly once by any ngL-tour.

The following dominance rule, similar to Dominance 2 described in §3.4.2 for the ng-tour relaxation, can be used to reduce the size of each state sets $\widehat{\mathscr{F}}_k$, $k = 1, \dots, h$.

Dominance 3. Let $(NG, t, i), (NG', t', i) \in \widehat{\mathscr{F}}_k$ be such that $f_k(NG, t, i) \geq f_k(NG', t', i)$, $NG' \supseteq NG$, $t > t'$. State (NG, t, i) is dominated by (NG', t', i) and can be removed from $\widehat{\mathscr{F}}_k$.

3.5 A Valid Lower Bound LB on the TSPTW

In this section, we describe lower bounds on the TSPTW based on the three tour relaxations described in §3.4.

Let $\widehat{\mathscr{H}}$ be the index set of the tours of a given tour relaxation of G (i.e., t-tour or ng-tour or ngL-tour). For each tour $\ell \in \widehat{\mathscr{H}}$, we denote by $a_{i\ell}$ the number of times vertex $i \in V$ is visited by tour $\ell \in \widehat{\mathscr{H}}$ and by c_ℓ its cost. Associate a penalty $u_i \in \mathbb{R}$ with each vertex $i \in V$, and let $\boldsymbol{u} = (u_1, \dots, u_n)$. For a given penalty vector $\boldsymbol{u}$, a valid lower bound $LR(\boldsymbol{u})$ on the TSPTW is given by

$$LR(\boldsymbol{u}) = \min_{\ell \in \widehat{\mathscr{H}}} \left\{ c_\ell - \sum_{i \in V} a_{i\ell} u_i \right\} + \sum_{i \in V} u_i.$$

A better lower bound LB on the TSPTW can be computed by using subgradient optimization to solve the following Lagrangean dual problem LD

$$(LD) \quad z(LD) = \max_{\boldsymbol{u} \in \mathbb{R}^n} \{LR(\boldsymbol{u})\}.$$

Solving problem LD by usual subgradient optimization requires to find, at each iteration, an optimal value of problem $LR(\boldsymbol{u})$ for a given penalty vector $\boldsymbol{u}$. The optimal value of problem $LR(\boldsymbol{u})$ is computed with the DP recursion associated with the tour relaxation chosen by replacing arc costs d_{ij} with $d'_{ij} = d_{ij} - u_j$, for each arc $(i, j) \in A \setminus A_q$, and $d'_{iq} = d_{iq}$, for each arc $(i, q) \in A_q$, where $A_q = \{(i, q) : i \in \Gamma_q^{-1}\}$.

Problem LD is equivalent to the following linear program D

$$(D) \quad z(D) = \max\, u_0 + \sum_{i \in V} u_i, \tag{3.6}$$

$$s.t.\ u_0 \leq c_\ell - \sum_{i \in V} a_{i\ell} u_i, \quad \ell \in \widehat{\mathscr{H}}, \tag{3.7}$$

$$u_0 \in \mathbb{R}, \tag{3.8}$$

$$u_i \in \mathbb{R}, \qquad i \in V. \tag{3.9}$$

Dualizing D, we obtain the following problem P

$$(P) \quad z(P) = \min \sum_{\ell \in \widehat{\mathscr{H}}} c_\ell x_\ell, \tag{3.10}$$

$$s.t. \sum_{\ell \in \widehat{\mathscr{H}}} a_{i\ell} x_\ell = 1, \quad i \in V, \tag{3.11}$$

$$\sum_{\ell \in \widehat{\mathscr{H}}} x_\ell = 1, \tag{3.12}$$

$$x_\ell \geq 0, \qquad \ell \in \widehat{\mathscr{H}}. \tag{3.13}$$

Problem P seeks a minimum-weight convex combination of tours such that each vertex $i \in V$ has, on average, degree one. Notice that variables $u_i \in \mathbb{R}$, $i \in V$, and $u_0 \in \mathbb{R}$ of problem D represent the dual variables of constraints (3.11) and (3.12), respectively.

Problem P can be solved by column generation where, at each iteration, tours of negative reduced cost with respect to the dual solution of the current master problem must be computed. In our computational experiments, we found that this method of solving P requires fewer iterations than solving LD with subgradient optimization.

To solve P, we use a column generation procedure that differs from standard column generation methods because the master problem is not solved with the simplex but using a dual-ascent heuristic procedure. The procedure computes lower bound LB_1 corresponding to a P dual solution $\boldsymbol{u^1} = (u_0^1, u_1^1, \ldots, u_n^1)$.

Below we describe how we initialize the master problem, the method for solving the master problem, and the procedure for solving the pricing problem.

3.5.1 Initializing the Master Problem

The initial master problem contains a small subset of elementary but nonnecessarily Hamiltonian tours that are generated as follows. The procedure consists of executing a limited number of subgradient iterations to solve problem LD. The optimal tour found at each iteration is made elementary and then added to the master. At the end, the procedure provides a lower bound LB_0 and a corresponding solution of problem LD. The procedure can be described as follows.

- Set $LB_0 = 0$. Initialize the Lagrangean penalties $\mu_i \in \mathbb{R}$, $i \in V$, as $\mu_i = 0$.
- Perform $Maxit0$ iterations of the following subgradient method.
  a) Define $d'_{ij} = d_{ij} - \mu_j$, for each arc $(i,j) \in A \setminus A_q$, and $d'_{iq} = d_{iq}$, for each arc $(i,q) \in A_q$.
  b) Find the tour $\ell^* \in \widehat{\mathscr{H}}$ (i.e., t-tour or ng-tour or ngL-tour) of minimum cost, c'_{ℓ^*}, with respect to the modified arc costs d'_{ij}.
  c) If $c'_{\ell^*} + \sum_{i\in V} \mu_i > LB_0$, update $LB_0 = c'_{\ell^*} + \sum_{i\in V} \mu_i$, and set $\mu_i^0 = \mu_i$, for each vertex $i \in V$.
  d) Add, to the master problem, the elementary but nonnecessarily Hamiltonian tour derived, from tour of index ℓ^*, by removing the visits to the vertices that have already been visited in the tour.
  e) Because in any salesman tour each vertex $i \in V$ must be visited exactly once, update penalties μ_i, $i \in V$, using usual subgradient expressions setting $z_{UB} = 1.2\ (c'_{\ell^*} + \sum_{i\in V} \mu_i)$ in computing the step size.

Notice that $\boldsymbol{u^0} = (u_0^0, u_1^0, \ldots, u_n^0)$, where $u_i^0 = \mu_i^0$, $i \in V$, and $u_0^0 = LB_0 - \sum_{i\in V} \mu_i$, represents a feasible solution of problem D of cost LB_0.

3.5.2 Solving the Master Problem

Instead of solving the master problem with the simplex, we use a dual-ascent heuristic based on the following theorem.

Theorem 1. Let $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \ldots, \lambda_n)$ be a vector of $n+1$ penalties, where penalties $\lambda_i \in \mathbb{R}$, $i \in V$, are associated with constraints (3.11) and penalty λ_0 with constraint (3.12). A feasible dual solution $\boldsymbol{u}$ of problem P of cost $z(P(\boldsymbol{\lambda})) = u_0 + \sum_{i\in V} u_i$ is obtained as

$$\left.\begin{array}{l} u_i = \min_{\ell\in\widehat{\mathscr{H}}:\, a_{i\ell}\geq 1} \left\{ \frac{c_\ell - \lambda_0 - \sum_{i\in V} a_{i\ell}\lambda_i}{\sum_{i\in V} a_{i\ell}} \right\} + \lambda_i,\ i \in V \\ u_0 = \lambda_0 \end{array}\right\}. \tag{3.14}$$

Proof. Consider a tour $\ell \in \widehat{\mathscr{H}}$. From expression (3.14), we derive

$$u_i \leq \frac{c_\ell - \lambda_0 - \sum_{i\in V} a_{i\ell}\lambda_i}{\sum_{i\in V} a_{i\ell}} + \lambda_i, \quad i \in V \ :\ a_{i\ell} \geq 1.$$

Thus, for each tour $\ell \in \widehat{\mathscr{H}}$, the following relation holds

$$\sum_{i\in V} a_{i\ell} u_i \leq \sum_{i\in V} \frac{a_{i\ell}(c_\ell - \lambda_0 - \sum_{i\in V} a_{i\ell}\lambda_i)}{\sum_{i\in V} a_{i\ell}} + \sum_{i\in V} a_{i\ell}\lambda_i = c_\ell - \lambda_0 = c_\ell - u_0,$$

which corresponds to constraint (3.7) for tour $\ell \in \widehat{\mathscr{H}}$

$$\sum_{i \in V} a_{i\ell} u_i \leq c_\ell - u_0 . \square$$

To find a near-optimal solution $\boldsymbol{u'}$ of the current master problem, we perform $Maxit1$ subgradient iterations to modify the penalty vector $\boldsymbol{\lambda}$. Baldacci et al. [2008] showed that a valid subgradient of function $z(P(\boldsymbol{\lambda}))$ at point $\boldsymbol{\lambda}$ can be computed by associating with the current solution $\boldsymbol{u''}$ a nonnecessarily feasible solution $\boldsymbol{x''}$ of the current master problem such that $z(P(\boldsymbol{\lambda})) = \sum_{\ell \in \widehat{\mathscr{H}}} c_\ell x''_\ell$ defined as follows.

Let $\ell(i) \in \widehat{\mathscr{H}}$ be the index of the tour associated with u_i, $i \in V$, in expressions (3.14). Define variables $\xi^i_\ell \in \{0, 1\}$, $\ell \in \widehat{\mathscr{H}}$, $i \in V$, as $\xi^i_{\ell(i)} = 1$, for each vertex $i \in V$, and $\xi^i_\ell = 0$, for each vertex $i \in V$ and each tour $\ell \in \widehat{\mathscr{H}} \setminus \{\ell(i)\}$. The solution $\boldsymbol{x''}$ is computed as $x''_\ell = \sum_{i \in V} \frac{a_{i\ell} \xi^i_\ell}{\sum_{i \in V} a_{i\ell}}$, for each tour $\ell \in \widehat{\mathscr{H}}$.

Let α_i, $i \in V$, and α_0 denote the values of the left-hand-side of constraints (3.11) and (3.12) with respect to solution $\boldsymbol{x''}$, respectively. The values of α_i, $i \in V$, and α_0 are used to update the penalty vector $\boldsymbol{\lambda}$ by means of the usual subgradient expressions, where $z_{UB} = 1.2\ z(P(\boldsymbol{\lambda}))$. We denote by $\boldsymbol{u'}$ the final dual solution of the current master problem of cost $z(P(\boldsymbol{\lambda}))$ achieved by the procedure described above.

3.5.3 Solving the Pricing Problem

The pricing problem consists of finding a tour of minimum reduced cost with respect to the current dual solution $\boldsymbol{u'}$ of the master. Define the modified arc cost d'_{ij} as

$$d'_{ij} = \begin{cases} d_{ij} - u'_j, & (i,j) \in A \setminus A_q, \\ d_{iq} - u'_0, & (i,q) \in A_q. \end{cases}$$

The reduced cost $c'_\ell = c_\ell - \sum_{i \in V} a_{i\ell} u'_i - u'_0$ of tour $\ell \in \widehat{\mathscr{H}}$ with respect to $\boldsymbol{u'}$ is equal to the cost of the tour (i.e., t-tour or ng-tour or ngL-tour) using the modified arc costs d'_{ij} instead of d_{ij}. Therefore, to compute the tour of minimum reduced cost with respect to $\boldsymbol{u'}$, it is sufficient to compute the associated DP recursion using d'_{ij} instead of d_{ij}.

If the cost of the minimum reduced cost tour is negative, the tour is added to the master problem and the bounding procedure performs another iteration; otherwise, $\boldsymbol{u'}$ is a feasible dual solution of problem P of cost $z(P(\boldsymbol{\lambda}))$, so the best-known lower bound LB_1 is update as $LB_1 = z(P(\boldsymbol{\lambda}))$ and $\boldsymbol{u^1} = \boldsymbol{u'}$ if $LB_1 \leq z(P(\boldsymbol{\lambda}))$. The bounding procedure terminates after a number of macro-iteration.

3.6 Computing Bounding Functions $b(S, t, i)$

In this section, we describe how to compute bounding functions $b(S, t, i)$, introduced in §3.3, to reduce the state-space graph associated with the DP recursion (3.1).

We define a *backward path* $B = (\sigma_B = i_k, i_{k+1}, \dots, i_h, q)$ as a path that starts from vertex σ_B at time $t_B \in [e_{\sigma_B}, l_{\sigma_B}]$, visits each vertex in $V(B) = \{i_k, i_{k+1}, \dots, i_h, q\}$ within its time window, and ends at vertex q before time l_q.

Consider the forward path F associated with $f(S, t, i)$. The least-cost salesman tour containing F is obtained by adding, to F, a least-cost backward path B of cost $c(B)$ that starts from vertex i at time $t' \in [t, l_i]$ and visits all vertices $V' \setminus S$.

The method to compute functions $b(S, t, i)$ is based on the following proposition. For a D solution $(\boldsymbol{u}^{\mathbf{1}}, w^1)$ providing lower bound LB_1, define the modified arc costs d^1_{ij} as

$$d^1_{ij} = \begin{cases} d_{ij} - u^1_j, & (i, j) \in A \setminus A_q, \\ d_{iq}, & (i, q) \in A_q. \end{cases}$$

Proposition 1. Let $lb(S, t, i)$ be a lower bound on the cost, using arc costs d^1_{ij} instead of d_{ij}, of any backward path that starts from vertex $i \in V$ at time $t' \in [t, l_i]$. A valid lower bound on the cost $c(B)$ of a least-cost backward path B that starts from vertex $i \in V$ at time $t' \in [t, l_i]$ and visits vertices $V' \setminus S$ is given by

$$b(S, t, i) = lb(S, t, i) + \sum_{j \in V \setminus S} u^1_i. \tag{3.15}$$

Proof. Let $c^1(B)$ be the cost of backward path B using arc costs d^1_{ij}. From the definition of $lb(S, t, i)$, we have

$$lb(S, t, i) \leq c^1(B). \tag{3.16}$$

Because B visits each vertex $j \in V \setminus S$ exactly once, then

$$c^1(B) = c(B) - \sum_{j \in V \setminus S} u^1_i. \tag{3.17}$$

From expressions (3.15), (3.16) and (3.17) we obtain $b(S, t, i) \leq c(B)$. □

In the following, we describe three methods to compute lower bound $lb(S, t, i)$ on $c(B)$ that are based on the three relaxations described in §3.4.

A *backward (t,i)-path* is a nonnecessarily elementary path B that starts from vertex $i \in V'$ at time $t \in [e_i, l_i]$, visits a subset of vertices (each once or more) within their time windows, and ends at vertex q before time l_q. We denote by $f^{-1}(t, i)$ the cost of a least-cost backward (t, i)-path without 2-vertex loops.

A *backward ng-path (NG,t,i)* is a nonnecessarily elementary path $B = (i = i_k, i_{k+1}, \ldots, i_h, i_q)$ that starts from vertex $i \in V'$ at time $t \in [e_i, l_i]$, visits a subset of vertices (each once or more) within their time windows such that $NG = \Pi^{-1}(B)$, ends at vertex q before l_q, and such that $i \notin \Pi^{-1}(B')$, where path B' is $B' = (i_{k+1}, \ldots, i_h, i_q)$ and must be a backward ng-path. For each backward path B, the set $\Pi^{-1}(B)$ is defined as

$$\Pi^{-1}(B) = \{i_r \, : \, i_r \in \bigcap_{s=k}^{r-1} N_{i_s}, r = k+1, \ldots, h\} \bigcup \{i_k\}.$$

We denote by $f^{-1}(NG, t, i)$ the cost of a least-cost backward ng-path (NG, t, i).

An example of backward ng-path (NG,t,i) is the following. Let $B = (9, 6, 7, 8, 9, 0)$ be a backward path that starts from vertex 9, and let $N_6 = \{6, 7, 8\}$, $N_7 = \{6, 7, 8\}$, $N_8 = \{7, 8, 9\}$ and $N_9 = \{6, 7, 9\}$. Thus, path B' is $B' = (6, 7, 8, 9, 0)$ and $\Pi(B') = \{6, 7, 8\}$ because 6 is the first visited vertex and $7 \in N_6$, $8 \in N_6 \cap N_7$, and $9 \notin N_6 \cap N_7 \cap N_8$. Because B' is an ng-path and $9 \notin \Pi(B')$, B is an ng-path.

For a given longest path $L = (i_0 = p, i_1, \ldots, i_k, i_{k+1}, \ldots, i_h, i_{h+1} = q)$, let $\bar{V}_k$ be the subset of vertices that must or can be visited in between vertices i_k and i_{k+1}, $k = 0, \ldots, h$, that is

$$\bar{V}_k = \{j \in V \setminus \{i_{k+1}\} \, : \, e_{i_k} + \vec{t}_{i_k j} \leq l_j \text{ and } \max\{e_{i_k} + \vec{t}_{i_k j}, e_j\} + \vec{t}_{j i_{k+1}} \leq l_{i_{k+1}}\}.$$

A *backward ngL(k)-path (NG,t,i)* is a backward ng-path that starts from vertex $i \in \bar{V}_k$ and visits the vertices $\{i_{k+1}, \ldots, i_h, i_q\}$ exactly once. We denote by $f_k^{-1}(NG, t, i)$ the cost of a least-cost backward $ngL(k)$-path.

Functions $f^{-1}(t, i)$, $f^{-1}(NG, t, i)$ and $f_k^{-1}(NG, t, i)$ can be computed with the same DP recursions used to compute $f(t, i)$, $f(NG, t, i)$ and $f(NG, k, t, i)$ on the TSPTW instance resulting from the following operations: (i) for each vertex $i \in V'$, replace time window $[e_i, l_i]$ with time window $[l_q - l_i, l_q - e_i]$; (ii) replace the cost and time matrices $[d_{ij}]$ and $[t_{ij}]$ with their transposed matrices $[d_{ij}]^T$ and $[t_{ij}]^T$.

Let functions $f^{-1}(t, i)$, $f^{-1}(NG, t, i)$ and $f_k^{-1}(NG, t, i)$ be computed replacing the arc costs d_{ij} with d^1_{ij}. These functions allow for the computing of the lower bound $lb(S, t, i)$ as follows

a) From functions $f^{-1}(t, i)$, we derive

$$lb(S, t, i) = \min_{t' \in [t, l_i]} \left\{ f^{-1}(t', i) \right\}.$$

b) From functions $f^{-1}(NG, t, i)$, we derive

$$lb(S, t, i) = \min_{\substack{NG \subseteq N_i \, : \, NG \ni i, NG \cap S = \{i\}, \\ t' \in [t, l_i]}} \left\{ f^{-1}(NG, t', i) \right\}.$$

c) From functions $f_k^{-1}(NG, t, i)$, we derive

$$lb(S,t,i) = \min_{\substack{NG\subseteq N_i \,:\, NG\ni i,\, NG\cap S=\{i\},\\ t'\in[t,l_i]}} \left\{ f_{k'}^{-1}(NG, t', i) \right\},$$

where k' is defined as follows

i) $i \notin V(L)$: k' is the index of the first vertex $i_{k'}$ of the longest path L such that $i_{k'} \notin S$;

ii) $i \in V(L)$: k' is the index of the vertex of L such that $i_{k'} = i$.

3.7 Computational Results

This section reports on the computational experiments of our algorithm, which was implemented in C and compiled with Visual Studio 2008. The runs were performed on a Sony Vaio P8400 (Intel Core 2 Duo, 2.26 GHz), equipped with 4 GB of RAM.

We denote by BMR.ng the exact algorithm of §3.3 using either the ng-tour or the ngL-tour relaxations, according to the criterion described in §3.7.1, in computing the lower bound and the bounding functions $b(S, t, i)$. We denote by BMR.t the exact method when the t-tour relaxation is used in computing the lower bound and the bounding functions.

We tested both BMR.ng and BMR.t on the seven classes of instances available at `http://iridia.ulb.ac.be/~manuel/tsptw-instances` and the class proposed by Mingozzi et al. [1997]. The classes proposed by Langevin et al. [1993], Dumas et al. [1995], and Mingozzi et al. [1997] were easily solved (the most difficult instance was solved in 12 seconds by BMR.ng), so relative results are omitted. The results on the other five classes are reported in the following and compared with the following state-of-the-art exact methods:

- the branch-and-cut of Ascheuer et al. [2001] (hereafter AFG) - runs performed on a Sun Sparc Station 10 with a time limit (tl) of 5 hours (i.e., tl = 18,000 seconds);

- the constraint programming based method of Focacci et al. [2002] (hereafter FLM) - Pentium III 700 MHz with 128 MB of RAM (tl = 1,800 seconds). The results reported in the tables are, on each instance, the best achieved by the two versions of their method (i.e., *AP-bound* and *Lagrangean-bound*).

- the branch-and-cut of Dash et al. [2012] (hereafter DGLT) - workstation Intel 2.40 GHz running under Suse Linux 10.1 (tl = 18,000 seconds);

TABLE 3.1: Results of BMR.ng on a representative set of instances

Inst	$\|V'\|$	$\|L\|$	*Prec*	z^*	*Rel*	$\Delta(N_i)$	*LB*	*%LB*	T_{LB}	$\|\mathscr{F}\|$	T_{tot}
n150w120.002	152	13	77.8	677	*ng*	11	663.3	97.98	69.7	-	-
					ng	13	663.8	98.05	124.8	-	-
					ngL	11	668.0	98.66	77.2	14,473	211.3
					ngL	13	668.4	98.73	130.5	12,149	260.6
n200w120.005	202	14	79.6	840	*ng*	11	827.5	98.51	155.0	-	-
					ng	13	828.9	98.68	195.1	-	-
					ngL	11	831.5	98.99	204.9	-	-
					ngL	13	833.2	99.19	237.4	29,072	574.3
rbg152.3	152	12	81.3	1,539	*ng*	11	1,536.2	99.82	47.3	22,015	192.5
					ng	13	1,536.7	99.85	64.7	17,387	206.7
					ngL	11	1,537.7	99.92	73.3	17,191	201.9
					ngL	13	1,538.2	99.95	81.5	14,151	213.2
rbg233.2	233	22	90.9	2,188	*ng*	11	2,187.0	99.95	72.1	9,200	124.8
					ng	13	2,187.0	99.95	92.2	8,906	136.9
					ngL	11	2,187.0	99.95	231.1	6,750	364.3
					ngL	13	2,187.0	99.95	253.3	5,265	359.6

- the DP algorithm of Li [2009] (hereafter LI) - Lenovo Thinkstation with 4 Intel processors at 2 GHz, 4 BG of RAM and a Linux operating system (no time limit imposed).

These exact methods were not tested on all five instance classes. In the tables of this section, “tl” indicates that the time limit was reached.

According to SPEC (`http://www.spec.org/benchmarks.html`), our machine is 10% slower than that of Dash et al. [2012] and 10% faster than that of Li [2009]. No benchmarks are available to compare our machine with those used by Ascheuer et al. [2001] and Focacci et al. [2002]. Nevertheless, our machine is clearly faster (say, 6-10 times faster) than these two machines.

3.7.1 Parameter Setting

To find a parameter setting for BMR.ng on all instances, we performed preliminary tests applying both ng-tour and ngL-tour relaxations and varying parameter $\Delta(N_i)$.

In Table 3.1, we present such results on a selected set of representative instances. The columns report the instance name (*Inst*), the number of vertices of graph G ($|V'|$), the length of the longest path L of the precedence graph $\widehat{G}$ ($|L|$), the percentage of precedences (*Prec*) computed as $Prec = 100\,\frac{2|\widehat{A}|}{(n+2)(n+1)}$, and the optimal solution cost (z^*). Then, we indicate the tour relaxation used (*Rel*), which can be either ng-tour or ngL-tour, and the setting of parameter $\Delta(N_i)$. Finally, the lower bound achieved (LB) is reported with the percentage ($\%LB = 100LB/z^*$), the time (in seconds) to compute the lower bound (T_{LB}), the cardinality (in thousands) of the set $\mathscr{F}$ generated by the exact algorithm of §3.3 ($|\mathscr{F}|$), and the total computing time in seconds (T_{tot}).

TABLE 3.2: Results on easy Ascheuer instances

			BMR.ng						BMR.t		AFG		FLM		DGLT		LI
Inst	$\|V'\|$	z^*	*Prec*	*LB*	$\%LB$	T_{LB}	$\|\mathscr{F}\|$	T_{tot}	$\%LB$	T_{tot}	$\%LB$	T_{tot}	$\%LB$	T_{tot}	$\%LB$	T_{tot}	T_{tot}
rbg010a	12	149	72.7	149.0	100.0	0.1		0.1	99.5	0.8	99.3	0.1	99.3	0.0	100.0	0.0	0.1
rbg017	17	148	72.1	148.0	100.0	0.1		0.1	97.2	0.7	100.0	0.8	93.2	0.1	99.3	0.0	0.1
rbg017.2	17	107	41.2	107.0	100.0	0.1		0.1	100.0	0.6	100.0	0.0	93.5	0.0	100.0	0.0	19.9
rbg016a	18	179	83.0	179.0	100.0	0.1		0.1	100.0	2.2	98.9	0.2	93.3	0.1	100.0	0.0	0.1
rbg016b	18	142	56.9	142.0	100.0	0.2		0.2	95.9	4.8	93.7	8.8	88.7	0.1	97.2	0.2	0.1
rbg017a	19	146	49.1	146.0	100.0	0.3		0.3	100.0	0.4	100.0	0.1	100.0	0.1	100.0	0.0	10.3
rbg019a	21	217	91.9	217.0	100.0	0.2		0.2	100.0	0.4	100.0	0.0	100.0	0.0	100.0	0.0	0.1
rbg019b	21	182	61.4	182.0	100.0	0.3		0.3	99.4	2.8	98.9	54.5	96.2	0.2	99.5	0.3	0.2
rbg019c	21	190	45.7	190.0	100.0	0.5		0.5	98.5	1.4	95.8	8.7	94.2	0.3	96.8	0.9	35.3
rbg019d	21	344	78.1	344.0	100.0	0.3		0.3	98.8	2.6	99.7	0.7	98.3	0.0	100.0	0.2	0.1
rbg021	21	190	45.7	190.0	100.0	0.5		0.5	98.6	2.5	95.8	8.7	94.2	0.3	96.8	0.9	35.5
rbg021.2	21	182	43.3	182.0	100.0	0.4		0.4	100.0	0.8	100.0	0.2	98.4	0.2	100.0	0.1	48.0
rbg021.3	21	182	42.9	182.0	100.0	1.4		1.4	98.9	3.0	97.8	27.1	91.2	0.4	98.4	2.7	101.2
rbg021.4	21	179	40.5	179.0	100.0	1.2	1	1.4	99.7	2.2	98.9	5.8	92.7	0.3	100.0	0.2	692.8
rbg021.5	21	169	39.5	169.0	100.0	1.1		1.1	99.6	1.9	98.8	6.6	94.7	0.2	100.0	0.3	1,320.5
rbg021.6	21	134	24.3	134.0	100.0	0.6	1	0.7	100.0	1.4	99.3	1.3	95.5	0.7	100.0	0.3	7,876.4
rbg021.7	21	133	19.5	133.0	100.0	1.5	1	1.7	98.5	2.1	96.2	4.3	94.0	0.6	100.0	0.6	-
rbg021.8	21	132	18.6	132.0	100.0	1.6	15	1.8	98.4	2.3	97.7	17.4	97.0	0.6	98.5	2.8	-
rbg021.9	21	132	18.6	132.0	100.0	3.1	18	3.3	98.0	2.2	97.0	26.1	94.7	0.8	98.5	2.8	-
rbg020a	22	210	53.7	210.0	100.0	0.7		0.7	100.0	0.9	100.0	0.2	98.6	0.0	100.0	0.0	10.5
rbg027a	29	268	43.1	267.9	100.0	1.3	1	1.5	99.5	7.3	99.3	2.2	97.0	0.2	99.3	1.4	0.7
rbg031a	33	328	75.8	328.0	100.0	0.5		0.5	99.1	7.4	100.0	1.7	97.3	2.7	100.0	0.2	0.1
rbg033a	35	433	77.6	433.0	100.0	0.5		0.5	99.3	6.5	100.0	1.8	92.8	1.0	99.8	0.9	0.2
rbg034a	36	403	72.4	402.9	100.0	3.2	5	3.5	99.2	9.4	99.5	0.9	97.0	55.2	100.0	1.9	1.0
rbg035a	37	254	76.3	254.0	100.0	0.6		0.6	98.8	12.7	100.0	1.8	87.0	3.5	100.0	0.2	0.1
rbg035a.2	37	166	70.9	166.0	100.0	23.5	6	24.1	98.4	20.1	95.2	64.8	93.4	36.8	100.0	5.3	-
rbg038a	40	466	78.1	466.0	100.0	2.0		2.0	100.0	6.2	100.0	4,232.2	100.0	0.2	100.0	1.5	0.5
rbg040a	42	386	79.2	384.9	99.7	1.3	1	1.5	97.7	5.7	92.0	751.8	73.1	738.1	96.6	3.6	0.8
rbg050a	52	414	44.2	414.0	100.0	16.1	4	16.6	99.7	24.4	100.0	18.6	98.8	95.6	100.0	24.5	-
rbg055a	57	814	84.9	814.0	100.0	1.8	1	2.1	99.3	7.3	99.9	6.4	98.2	2.5	100.0	3.5	2.4
rbg067a	69	1,048	88.8	1,048.0	100.0	1.9	1	2.2	99.4	17.6	99.9	5.9	98.6	4.0	100.0	3.6	3.2
rbg125a	127	1,409	92.8	1,409.0	100.0	5.6	2	6.4	99.3	71.0	99.5	229.8	98.2	tl	100.0	9.6	3.5
Avg					100.0	2.3		2.4	99.1	7.2	98.5	171.5	95.0	30.5	99.4	2.1	376.4
Solved								32		32		32		31		32	27

Table 3.1 shows that increasing parameter $\Delta(N_i)$ lets us compute better lower bounds and generate smaller state-space graphs when solving the problem to optimality. Nonetheless, because the complexity of computing the recursions 3.3 and 3.5 and testing Fathoming 1 increases, the computing time to compute the lower bound and the total computing time may increase.

The ngL-tour relaxation dominates the ng-tour relaxation but is more time consuming. At the same time, the lower bound that can be achieved with the ngL-tour relaxation is sometimes just slightly better or the same bound that can be achieved with the ng-tour relaxation. In particular, we can see that the higher is the number of arcs in the precedence digraph $\widehat{G}$ (i.e., $Prec$), the lower is the increment of the lower bound when applying ngL-tour relaxation instead of the ng-tour relaxation.

Notice that the ngL-tour relaxation allows us to solve a few instances that cannot be solved with the ng-tour relaxation. Thus, BMR.ng uses the ng-tour relaxation if $Prec \geq 80$ and the ngL-tour relaxation otherwise.

In computing the lower bound LB (see §3.5.3), parameter $\Delta(N_i)$ was set equal to 11 when using the ng-tour relaxation and equal to 13 when using ngL-tour relaxation. In computing bounding functions $b(S, t, i)$ (see §3.6), parameter $\Delta(N_i)$ was set equal to 13 for both relaxations.

Finally, in the column generation method described in §3.5, for both BMR.ng and BMR.t we set $Maxit0 = 200$ and $Maxit1 = 150$. In the exact method, due to memory limits, the cardinality of the state set $\mathscr{F}$ is limited to 10^8. Both BMR.ng and BMR.t terminate prematurely when this memory limit is reached.

3.7.2 Ascheuer Instances

The Ascheuer class consists of 50 asymmetric real-world instances with up to 233 vertices. Both travel costs and times are integer values. Travel times satisfy the triangle inequality.

Like Dash et al. [2012], we present the results on this class dividing the 50 instances into 32 easy instances (those solved by Ascheuer et al. [2001] within the time limit of 5 hours) and the remaining 18 hard instances.

Table 3.2 presents the computational results on easy instances. The columns mean the same as in Table 3.1. For AFG, FLM and DGLT, column (%LB) reports the percentage deviation of the lower bound at the root node of the decision tree. The last two lines of the table indicate averages on the percentage deviation of the lower bounds, the computing times of the instances solved to optimality and the number of instances solved by each method.

TABLE 3.3: Results on hard Ascheuer instances

			BMR.ng						BMR.t		AFG	FLM		DGLT		LI
Inst	$\|V'\|$	z^*	*Prec*	*LB*	*%LB*	T_{LB}	$\|\mathscr{F}\|$	T_{tot}	*%LB*	T_{tot}	*%LB*	*%LB*	T_{tot}	*%LB*	T_{tot}	T_{tot}
rbg041a	43	402	77.3	402.0	100.0	2.0		2.0	95.4	4.2	89.8	87.6	tl	97.5	146.8	2.2
rbg042a	44	411	71.6	411.0	100.0	6.6		6.6	97.7	5.0	95.9	90.0	149.8	98.3	188.3	35.6
rbg048a	50	487	47.8	487.0	100.0	12.3		12.3	100.0	14.7	93.2	91.2	tl	100.0	129.2	-
rbg049a	51	**484**	63.4	472.1	97.5	19.5	6	20.4	95.7	13.7	84.3	83.3	tl	95.9	tl	-
rbg050b	52	**512**	61.1	504.8	98.6	19.1	2	19.9	96.5	8.1	87.3	87.1	tl	96.3	tl	-
rbg050c	52	**526**	49.8	522.4	99.3	20.3	93	21.7	98.5	23.1	96.4	95.8	tl	98.3	tl	-
rbg086a	88	1,051	91.7	1,050.0	99.9	2.1	1	2.5	99.1	14.5	99.1	97.3	tl	99.8	4.9	1.9
rbg092a	94	1,093	89.8	1,091.6	99.9	6.7	3	7.3	99.7	14.2	99.2	98.8	tl	99.8	90.4	9.0
rbg132.2	132	1,083	88.2	1,080.2	99.7	45.5	294	47.9	99.4	80.5	97.2	n.a.		99.8	2,761.1	-
rbg132	132	1,360	94.2	1,360.0	100.0	10.6	1	11.3	98.9	48.9	97.3	n.a.		99.5	37.6	6.1
rbg152.3	152	1,539	81.3	1,536.2	99.8	47.3	20,971	166.9	99.4	238.8	98.8	n.a.		99.9	10,353.3	-
rbg152	152	1,783	94.0	1,783.0	100.0	18.8	3	19.8	99.3	36.8	98.7	n.a.		99.8	43.7	11.5
rbg172a	174	1,799	93.9	1,799.0	100.0	37.1	4	38.4	99.2	455.7	98.8	n.a.		99.8	425.5	544.3
rbg193.2	193	**2,017**	89.1	2,015.3	99.9	50.8	414	54.4	99.6	76.3	97.6	n.a.		99.7	tl	-
rbg193	193	2,414	94.5	2,414.0	100.0	46.4	16	47.9	99.8	65.9	98.8	n.a.		100.0	159.6	3,165.7
rbg201a	203	2,189	94.8	2,189.0	100.0	54.3	4	56.0	99.3	716.2	98.6	n.a.		99.9	462.7	581.1
rbg233.2	233	**2,188**	90.9	2,187.0	100.0	72.1	8,969	120.6	99.8	124.4	98.1	n.a.		99.7	tl	-
rbg233	233	2,689	95.4	2,688.0	100.0	78.9	35	81.0	99.5	319.3	98.0	n.a.		100.0	749.4	1,433.3
Avg					99.7	30.6		40.9	98.7	125.6	95.9	91.4	149.8	99.1	1,196.3	579.0
Solved								18		18			1		13	10

n.a. Data not available

Table 3.2 shows that the lower bound computed by BMR.ng closes the gap on all but 3 instances. BMR.ng solves all 32 instances with the same performance of DGLT.

Table 3.3 reports the results on hard Ascheuer instances. In column z^* the optimal value is in bold when instances were solved for the first time by BMR.ng. Table 3.3 shows that BMR.ng solves the 5 open instances and outperforms the other exact methods.

3.7.3 Pesant Instances

The Pesant class consists of 27 symmetric instances proposed by Pesant et al. [1998] with 21 to 46 vertices. Travel costs and times are the Euclidean distances truncated to four decimal places. Travel times include service times and satisfy the triangle inequality.

Table 3.4 reports the results on Pesant instances. The lower bounds computed by BMR.ng are of excellent quality. BMR.ng and BMR.t solve all instances of this class and clearly outperform FLM, DGLT and LI.

3.7.4 Potvin Instances

The Potvin class consists of 28 symmetric instances introduced by Potvin and Bengio [1996] with 15 to 47 vertices (we neglect instances with fewer than 10 vertices). The instances feature x-y coordinates. Travel times are Euclidean and include service time at the starting vertex of each arc. Travel costs and travel times coincide.

In Table 3.5, we compare BMR.ng and BMR.t with LI. In column z^*, the optimal solution cost is rounded to two decimal places. The last column indicates the computing time of LI (a “-” is reported when the instance was not solved).

From Table 3.5, both BMR.ng and BMR.t compare favorably with LI. BMR.ng is clearly superior to both BMR.t and LI because it solves all instances.

3.7.5 Gendreau Instances

The Gendreau class is made up of 28 groups of 5 instances proposed by Gendreau et al. [1998]. Each group consists of five instances having the same number of vertices and the same time windows width. The instances are obtained from the instances of Dumas et al. [1995] extending the time windows and resulting in time windows ranging from 80 to 200 time units in increments of 20. The number of vertices ranges from 22 to 102.

TABLE 3.4: Results on Pesant instances

			BMR.ng						BMR.t		FLM		DGLT		LI
Inst	$\lvert V'\rvert$	z^*	*Prec*	*LB*	*%LB*	T_{LB}	$\lvert\mathscr{F}\rvert$	T_{tot}	*%LB*	T_{tot}	*%LB*	T_{tot}	*%LB*	T_{tot}	T_{tot}
rc201.0	27	378.62	84.3	378.62	100.0	0.1		0.1	100.0	0.1	100.0	0.2	100.0	0.0	0.1
rc201.1	30	374.70	85.3	374.70	100.0	0.2		0.2	99.0	0.3	68.4	1.8	100.0	0.0	0.1
rc201.2	30	427.65	83.7	427.65	100.0	0.2		0.2	100.0	0.1	99.0	0.4	100.0	0.0	0.1
rc201.3	21	232.54	82.9	232.54	100.0	0.1		0.1	99.9	0.1	93.7	0.1	98.7	0.1	0.1
rc202.0	27	246.22	45.3	246.22	100.0	0.4		0.4	100.0	0.2	98.2	1.0	100.0	0.2	63.0
rc202.1	24	206.53	55.1	206.53	100.0	0.4		0.4	94.8	0.4	71.1	3.8	100.0	0.1	1.1
rc202.2	29	341.77	61.3	341.77	100.0	0.2		0.2	94.6	0.8	82.5	3.1	100.0	0.1	0.4
rc202.3	28	367.85	50.5	367.85	100.0	0.3		0.3	97.9	0.6	76.8	33.1	100.0	0.2	5.1
rc203.0	37	377.45	24.0	376.22	99.7	4.2	2	4.6	83.4	31.0	79.0	tl	95.4	3,437.7	-
rc203.1	39	356.99	25.1	356.66	99.9	14.1	1	14.5	89.1	20.4	84.4	tl	97.6	2,722.7	-
rc203.2	30	337.47	42.3	337.47	100.0	0.5		0.5	94.1	2.3	87.5	94.3	100.0	0.9	392.7
rc204.0	34	221.45	16.8	221.45	100.0	1.7		1.7	93.1	2.0	96.8	352.8	100.0	2.8	-
rc204.1	30	205.37	14.7	205.37	100.0	4.3		4.3	92.3	9.0	98.9	3.4	99.0	14.9	-
rc204.2	42	**378.40**	14.6	374.77	99.0	33.5	7	36.4	92.3	29.4	87.3	tl	96.6	tl	-
rc205.0	28	251.65	60.1	251.65	100.0	0.3		0.3	100.0	1.4	91.7	7.9	100.0	0.1	1.9
rc205.1	24	271.22	67.8	271.22	100.0	0.2		0.2	98.9	0.2	88.7	0.2	100.0	0.0	0.4
rc205.2	30	434.70	68.5	434.70	100.0	0.4		0.4	92.7	1.2	67.8	1,289.1	100.0	0.3	0.3
rc205.3	26	361.24	64.6	361.24	100.0	0.2		0.2	100.0	0.2	72.4	4.7	100.0	0.0	0.1
rc206.0	37	485.23	62.6	485.23	100.0	0.8		0.8	90.3	1.7	89.1	338.1	97.5	73.3	1.6
rc206.1	35	334.73	57.8	334.73	100.0	1.0		1.0	96.3	0.8	96.0	22.9	98.2	85.0	42.5
rc206.2	34	335.37	55.8	335.37	100.0	0.9		0.9	98.7	0.9	96.0	24.1	98.8	84.8	189.5
rc207.0	38	436.69	38.7	436.69	100.0	0.6		0.6	95.1	2.0	75.9	572.0	99.5	19.0	18,691.8
rc207.1	35	396.36	37.8	396.36	100.0	1.4		1.4	94.4	1.6	94.4	321.7	98.4	34.9	10,864.5
rc207.2	32	246.41	33.3	246.41	100.0	0.6		0.6	98.4	0.7	96.7	15.1	97.2	116.4	-
rc208.0	46	**380.56**	8.6	380.56	100.0	38.1		38.1	92.8	63.4	85.1	tl	93.4	tl	-
rc208.1	29	239.04	13.5	239.04	100.0	6.9		6.9	96.5	0.9	96.1	34.2	96.1	990.6	-
rc208.2	31	213.92	12.7	213.92	100.0	1.3		1.3	94.4	1.2	100.0	1.4	100.0	7.9	-
Avg					99.9	4.2		4.3	95.5	6.4	87.9	135.9	98.8	303.7	1,680.8
Solved								27		27		23		25	18

The name of each instance indicates the number of vertices (excluding p and q), the width of the time windows, and the instance number in the group (e.g. n20w120.1 is the first instance of the group of 5 instances with 22 vertices and time windows of 120 units). In some papers, two groups of instances (n100w80.x and n100w100.x) are included in the Dumas class instead of the Gendreau class, but the instances coincide.

Travel times and costs coincide and are first computed as truncated integer Euclidean distances and then modified to satisfy triangle inequality by iteratively setting $c_{ij} = t_{ij} = t_{ik} + t_{kj}$, if $t_{ik} + t_{kj} < t_{ij}$, until no violation is identified.

Because t-tour relaxation requires strictly positive travel times, BMR.t was not tested on the Gendreau instances such that $t_{ij} = 0$ for some arc $(i, j) \in A$.

In Table 3.6, we report detailed computational results on Gendreau instances. BMR.ng and BMR.t are compared with LI. Table 3.6 shows that BMR.ng is much faster than LI and solves all 140 instances while LI solves 46 of them. BMR.t solves 18 more instances than LI. The comparison of BMR.t and LI on the instances solved by both methods shows that BMR.t outperforms LI.

TABLE 3.5: Results on Potvin instances

			BMR.ng						BMR.t		LI
Inst	$\lvert V'\rvert$	z^*	*Prec*	*LB*	*%LB*	T_{LB}	$\lvert\mathscr{F}\rvert$	T_{tot}	*%LB*	T_{tot}	T_{tot}
rc201.1	21	444.54	80.0	444.54	100.0	0.1		0.1	99.6	0.7	0.0
rc201.2	27	711.54	86.0	711.54	100.0	0.2		0.2	100.0	1.3	0.1
rc201.3	33	790.61	86.7	790.61	100.0	0.2		0.2	98.7	4.0	0.1
rc201.4	27	793.64	86.0	793.64	100.0	0.2		0.2	99.4	3.3	0.0
rc202.1	34	771.78	52.2	771.64	100.0	1.0	1	1.3	92.9	9.0	164.5
rc202.2	15	304.14	33.3	304.14	100.0	0.2		0.2	95.5	3.4	0.6
rc202.3	30	837.72	76.6	835.87	99.8	0.2	1	0.4	97.2	5.9	0.0
rc202.4	29	793.03	49.8	791.54	99.8	5.1	1	5.3	98.1	7.2	94.8
rc203.1	20	453.48	40.5	453.48	100.0	0.2		0.2	99.4	3.2	2.4
rc203.2	34	**784.16**	32.4	781.64	99.7	1.7	1	2.1	93.4	11.1	-
rc203.3	38	**817.53**	33.1	810.46	99.1	9.3	10	9.9	94.9	71.6	-
rc203.4	16	314.29	28.3	314.29	100.0	0.4		0.4	100.0	0.8	11.1
rc204.1	47	**878.64**	14.8	872.62	99.3	42.6	10	43.8	93.2	*581.6*[a]	-
rc204.2	34	**662.16**	16.2	650.94	98.3	21.3	151	23.1	90.5	*473.4*[a]	-
rc204.3	25	**455.03**	17.0	455.03	100.0	9.9		9.9	95.0	9.0	-
rc205.1	15	343.21	67.6	343.21	100.0	0.1		0.1	98.3	4.8	0.0
rc205.2	28	755.93	72.0	755.93	100.0	0.3		0.3	98.8	3.7	0.2
rc205.3	36	825.06	57.3	825.06	100.0	2.2		2.2	97.7	8.5	197.7
rc205.4	29	760.47	76.1	756.95	99.5	0.2	1	0.4	97.5	4.1	0.1
rc206.2	38	828.06	59.9	826.66	99.8	6.4	1	6.8	95.3	9.7	20.5
rc206.3	26	574.42	52.3	574.42	100.0	0.6		0.6	90.0	6.3	5.5
rc206.4	39	831.67	59.0	827.54	99.5	1.8	1	2.2	90.5	13.4	69.6
rc207.1	35	**732.68**	42.9	731.57	99.8	9.0	1	9.3	93.3	11.9	-
rc207.2	32	**701.25**	33.7	694.22	99.0	53.1	5	54.8	89.4	47.2	-
rc207.3	34	**682.40**	30.8	677.23	99.2	1.9	1	2.3	91.8	15.6	-
rc208.1	39	**789.25**	10.4	785.69	99.5	46.7	2	47.5	90.0	*435.9*[a]	-
rc208.2	30	**533.78**	13.1	533.78	100.0	30.1		30.1	93.8	25.2	-
rc208.3	37	**634.44**	10.7	622.48	98.1	82.0	24	86.8	94.1	*532.6*[a]	-
Avg					99.7	11.7		12.2	95.3	11.7	33.3
Solved								28		24	17

[a] BMR.t runs out of memory

3.7.6 Olhmann Instances

The Olhmann class was proposed by Ohlmann and Thomas [2007] and consists of 25 instances divided in five groups. The instances have 152 or 202 vertices and are obtained from the Dumas instances by extending the time windows by 100 time units. Travel times and travel costs are computed as for the Gendreau class.

To our knowledge, no exact method has been tested on this class so far. In Table 3.7, we report the computational results of BMR.ng. BMR.t was not tested on these instances as the requirement of strictly positive travel times is not satisfied by any of the instances.

The table shows that all but one instance can be solved in a reasonable amount of computing time. Instance n200w140.4 could not be solved as BMR.ng ran out of memory (i.e., $|\mathscr{F}| > 10^8$). BMR.ng ran out of memory after 2,413.1 seconds at iteration

TABLE 3.6: Results on Gendreau instances

		BMR.ng			BMR.t		LI			BMR.ng			BMR.t		LI
Inst	z^*	*Prec*	*%LB*	T_{tot}	*%LB*	T_{tot}	T_{tot}	*Inst*	z^*	*Prec*	*%LB*	T_{tot}	*%LB*	T_{tot}	T_{tot}
n20w120.1	267	62.8	100.0	0.4	100.0	0.2	0.2	n60w200.1	**410**	47.2	100.0	14.7	89.2	*714.0*^{b}	-
n20w120.2	218	52.4	100.0	0.5	96.6	0.2	0.3	n60w200.2	**414**	52.1	100.0	7.4	a		-
n20w120.3	303	63.2	100.0	0.5	94.2	0.2	0.1	n60w200.3	**455**	46.1	97.6	192.1	a		-
n20w120.4	300	63.2	100.0	0.3	97.2	0.1	0.1	n60w200.4	**431**	52.2	99.7	18.4	96.6	14.7	-
n20w120.5	240	60.2	100.0	0.3	99.3	0.3	0.3	n60w200.5	**427**	47.5	99.6	20.0	a		-
n20w140.1	176	50.2	100.0	0.4	95.3	0.6	0.5	n80w100.1	**541**	74.2	100.0	3.8	a		-
n20w140.2	272	56.7	100.0	0.4	95.9	0.3	0.1	n80w100.2	**567**	80.4	100.0	1.1	a		-
n20w140.3	236	54.5	100.0	0.3	92.9	0.5	0.1	n80w100.3	**578**	75.7	100.0	3.4	a		-
n20w140.4	255	51.1	100.0	1.2	96.3	0.5	0.3	n80w100.4	**648**	82.0	99.6	20.4	98.3	1.2	-
n20w140.5	225	58.9	100.0	0.3	88.4	0.6	0.1	n80w100.5	**532**	76.4	100.0	6.1	a		-
n20w160.1	241	51.9	100.0	0.6	92.5	0.4	1.6	n80w120.1	**498**	70.8	99.9	27.0	a		-
n20w160.2	201	52.8	100.0	0.3	95.0	0.2	0.1	n80w120.2	**577**	73.6	99.9	16.0	a		-
n20w160.3	201	60.2	100.0	0.3	98.5	0.3	0.1	n80w120.3	**540**	73.1	100.0	6.4	a		-
n20w160.4	203	42.0	100.0	0.7	90.8	0.4	3.4	n80w120.4	**501**	68.7	98.1	61.4	a		-
n20w160.5	245	50.6	100.0	0.3	87.7	0.7	0.3	n80w120.5	**591**	72.2	99.3	39.6	a		-
n20w180.1	253	51.5	100.0	0.3	97.7	0.3	0.4	n80w140.1	**511**	68.0	100.0	7.5	a		-
n20w180.2	265	50.2	100.0	0.3	99.4	0.4	0.3	n80w140.2	**470**	67.8	99.8	77.8	a		-
n20w180.3	271	44.2	100.0	1.0	94.1	0.5	0.4	n80w140.3	**580**	68.7	99.1	46.5	a		-
n20w180.4	201	42.9	100.0	0.4	96.4	0.3	3.2	n80w140.4	**422**	67.8	99.6	38.8	a		-
n20w180.5	193	35.9	100.0	0.3	91.7	0.7	27.2	n80w140.5	**545**	68.4	100.0	8.1	94.5	282.7	-
n20w200.1	233	38.1	100.0	1.1	93.1	0.4	12.0	n80w160.1	**506**	62.6	100.0	93.7	a		-
n20w200.2	203	31.6	100.0	1.1	94.9	0.6	55.7	n80w160.2	**548**	60.6	99.9	194.8	a		-
n20w200.3	249	40.3	100.0	0.8	89.5	0.8	4.5	n80w160.3	**521**	60.4	100.0	58.2	a		-
n20w200.4	293	41.6	98.8	2.4	90.6	0.8	3.0	n80w160.4	**509**	63.5	100.0	50.8	a		-
n20w200.5	227	34.2	100.0	0.4	99.5	0.3	26.4	n80w160.5	**438**	63.3	100.0	58.1	92.8	*576.8*^{b}	-
n40w120.1	434	70.4	100.0	2.1	a		15.7	n80w180.1	**551**	57.8	99.9	75.4	91.9	*702.3*^{b}	-
n40w120.2	444	66.7	100.0	1.9	95.4	1.7	51.5	n80w180.2	**478**	58.5	100.0	61.5	a		-
n40w120.3	357	65.9	100.0	2.1	98.8	1.5	39.1	n80w180.3	**524**	55.3	100.0	84.3	89.7	*967.6*^{b}	-
n40w120.4	303	63.8	100.0	1.9	88.7	2.4	28.6	n80w180.4	**479**	59.6	100.0	47.4	a		-
n40w120.5	350	68.3	100.0	1.3	98.9	0.9	2.4	n80w180.5	**470**	61.5	100.0	49.0	91.9	*602.8*^{b}	-
n40w140.1	328	65.2	100.0	2.0	97.3	1.0	42.9	n80w200.1	**490**	47.5	99.3	177.4	a		-
n40w140.2	383	63.8	100.0	1.5	92.4	3.2	83.2	n80w200.2	**488**	55.1	99.9	32.1	a		-
n40w140.3	398	66.9	100.0	2.2	a		5.5	n80w200.3	**464**	50.0	99.6	39.5	a		-
n40w140.4	342	59.9	100.0	1.7	93.1	2.1	230.5	n80w200.4	**526**	54.1	99.1	63.3	a		-
n40w140.5	371	56.3	100.0	6.2	95.7	1.5	1,288.1	n80w200.5	**439**	49.0	99.8	46.9	a		-
n40w160.1	348	56.8	100.0	1.4	94.3	1.0	117.3	n100w80.1	**670**	84.3	100.0	4.5	a		-
n40w160.2	337	56.1	100.0	1.2	95.7	0.9	1,331.4	n100w80.2	**666**	86.4	99.9	2.5	98.3	2.9	-
n40w160.3	346	53.7	100.0	3.5	a		474.4	n100w80.3	**691**	85.4	100.0	1.4	97.9	7.0	-
n40w160.4	**288**	47.5	100.0	12.9	90.6	196.6	-	n100w80.4	**700**	84.1	100.0	2.8	a		-
n40w160.5	315	51.5	100.0	6.9	92.6	4.5	9,369.0	n100w80.5	**603**	82.3	100.0	2.6	a		-
n40w180.1	**337**	51.3	100.0	25.9	87.7	9.7	-	n100w100.1	**643**	79.7	100.0	18.4	a		-
n40w180.2	**347**	47.5	100.0	14.9	a		-	n100w100.2	**618**	81.0	100.0	2.5	96.3	37.3	-
n40w180.3	**279**	51.7	100.0	1.6	a		-	n100w100.3	**685**	80.5	100.0	4.5	96.3	15.8	-
n40w180.4	**354**	52.7	99.6	7.5	96.6	1.1	-	n100w100.4	**684**	79.9	99.5	83.6	a		-
n40w180.5	**335**	43.9	100.0	4.8	86.2	48.0	-	n100w100.5	**572**	78.2	100.0	18.6	a		-
n40w200.1	**330**	39.7	100.0	8.8	a		-	n100w120.1	**629**	78.2	100.0	34.5	a		-
n40w200.2	**303**	39.6	99.7	17.1	a		-	n100w120.2	**540**	77.4	99.2	24.1	a		-
n40w200.3	**339**	42.0	95.0	25.8	a		-	n100w120.3	**615**	74.2	100.0	61.6	97.1	33.2	-
n40w200.4	**301**	47.6	99.7	12.9	87.4	12.5	-	n100w120.4	**662**	79.7	99.7	91.4	a		-
n40w200.5	**296**	35.0	100.0	8.9	93.0	4.6	-	n100w120.5	**537**	74.9	100.0	42.1	a		-
n60w120.1	384	69.7	100.0	17.6	96.1	3.0	8,431.4	n100w140.1	**603**	71.2	100.0	38.1	a		-
n60w120.2	426	72.6	100.0	4.1	99.0	0.9	419.3	n100w140.2	**613**	74.3	100.0	44.4	a		-
n60w120.3	407	69.1	100.0	6.0	a		20,411.3	n100w140.3	**481**	73.9	100.0	51.5	a		-
n60w120.4	490	70.1	100.0	4.8	96.5	5.0	13,702.2	n100w140.4	**533**	74.3	100.0	24.7	a		-
n60w120.5	547	70.7	100.0	9.1	96.9	1.9	1,455.1	n100w140.5	**509**	73.1	99.1	66.9	90.5	*725.4*^{b}	-
n60w140.1	**423**	65.8	99.6	8.4	a		-	n100w160.1	**582**	69.4	100.0	117.4	92.0	*310.0*^{b}	-
n60w140.2	462	67.3	99.8	11.5	96.0	2.8	4,497.5	n100w160.2	**530**	68.8	99.8	99.5	a		-
n60w140.3	**427**	66.0	100.0	11.8	94.6	10.1	-	n100w160.3	**495**	71.5	100.0	52.6	a		-
n60w140.4	**488**	67.6	100.0	14.2	93.7	2.9	-	n100w160.4	**580**	69.8	100.0	116.6	a		-
n60w140.5	460	66.1	99.0	33.4	90.6	8.5	5,148.5	n100w160.5	**586**	71.9	99.9	72.9	a		-
n60w160.1	**560**	63.5	99.3	42.6	90.9	458.2	-	n100w180.1	**568**	60.6	99.4	170.4	a		-
n60w160.2	**423**	62.3	100.0	7.7	98.7	2.2	-	n100w180.2	**503**	60.8	100.0	40.4	a		-
n60w160.3	**434**	54.5	99.7	34.8	92.8	*184.0*^{b}	-	n100w180.3	**574**	61.9	99.0	135.2	a		-
n60w160.4	**401**	62.7	100.0	4.0	96.5	6.2	-	n100w180.4	**526**	61.7	100.0	128.4	a		-
n60w160.5	**501**	63.0	100.0	7.2	91.6	356.0	-	n100w180.5	**501**	60.0	98.6	117.9	a		-
n60w180.1	**411**	56.7	99.1	103.9	a		-	n100w200.1	**549**	55.7	100.0	424.3	a		-
n60w180.2	**399**	53.7	100.0	8.4	95.4	13.3	-	n100w200.2	**502**	55.9	100.0	86.9	a		-
n60w180.3	**444**	54.7	99.1	28.5	90.9	*624.0*^{b}	-	n100w200.3	**557**	57.9	99.5	264.6	a		-
n60w180.4	**456**	54.9	99.9	51.6	93.5	20.8	-	n100w200.4	**521**	57.3	99.4	199.5	a		-
n60w180.5	**395**	50.3	98.4	39.3	a		-	n100w200.5	**486**	55.5	99.8	104.7	a		-
								Avg			99.8	36.7	94.2	24.9	1,462.7
								Solved				140		64	46

a: not attempted as $t_{ij} = 0$ for some $(i, j) \in A$

^{b} BMR.t runs out of memory

TABLE 3.7: Results on Olhmann instances

		BMR.ng					
Inst	z^*	*Prec*	*LB*	*%LB*	T_{LB}	$\lvert\mathscr{F}\rvert$	T_{tot}
n150w120.1	**732**	77.5	725.5	99.1	243.8	243	248.6
n150w120.2	**677**	77.8	668.4	98.7	130.5	12,149	260.6
n150w120.3	**747**	76.5	746.4	99.9	160.0	2	163.2
n150w120.4	**762**	78.0	761.6	99.9	172.8	1	176.0
n150w120.5	**689**	76.9	684.7	99.4	113.1	373	118.9
n150w140.1	**762**	72.5	754.0	98.9	198.2	14,993	350.2
n150w140.2	**753**	74.5	752.0	99.9	439.0	7	442.1
n150w140.3	**613**	73.3	608.5	99.3	420.6	1,671	440.9
n150w140.4	**676**	73.7	676.0	100.0	104.8		104.8
n150w140.5	**663**	73.1	662.0	99.8	365.9	3	369.4
n150w160.1	**704**	70.4	701.4	99.6	262.9	48	266.5
n150w160.2	**711**	69.3	709.7	99.8	213.1	5	216.4
n150w160.3	**608**	71.9	603.2	99.2	483.0	352	491.8
n150w160.4	**672**	69.9	672.0	100.0	570.6	12	577.1
n150w160.5	**658**	71.0	655.0	99.5	169.5	44	173.4
n200w120.1	**795**	78.3	793.3	99.8	303.8	57	308.8
n200w120.2	**721**	80.2	713.9	99.0	110.4	20,223	257.1
n200w120.3	**879**	80.4	868.6	98.8	151.1	57,625	958.6
n200w120.4	**777**	79.1	775.8	99.8	412.3	13	417.1
n200w120.5	**840**	79.6	833.2	99.2	237.4	29,072	574.3
n200w140.1	**830**	77.1	826.2	99.5	1,217.8	6,261	1,274.9
n200w140.2	**760**	76.3	756.2	99.5	614.5	407	624.5
n200w140.3	**758**	76.3	756.0	99.7	250.7	133	259.0
n200w140.4	816[a]	75.3	807.1	98.9	592.9	-	*2,413.1*[b]
n200w140.5	**822**	76.6	819.6	99.7	515.2	90	521.0
Avg				99.5	338.2		399.8
Solved							24

[a] Best known upper bound found by López Ibáñez and Blum [2010]

[b] BMR.ng ran out of memory after 2,413.1 seconds. This time is not included in the final average computing time

$h = 7$ with $z^7_{UB} = 814$. Thus, $z^6_{UB} = 813$ is a valid lower bound on the optimal solution cost.

3.7.7 More Details on the Computational Results

Table 3.8 shows average results of the exact method, BMR.ng, using either the *ng*-tour or *ngL*-tour relaxations on the 6 classes of instances. Columns *%LB* show that the *ngL*-tour relaxation closes half of the gap left by the *ng*-tour relaxation but is more time-consuming. Columns "*Solved*" show the effectiveness of the *ngL*-tour relaxation in solving the Ohlmann instances. BMR.ng using the *ng*-tour relaxation cannot solve 4 of the instances ($n150w120.2$, $n150w140.1$, $n200w120.5$, $n200w140.1$) solved by using the *ngL*-tour relaxation.

Table 3.9 shows the effectiveness of the dominance and fathoming rules applied in Step 2 of the exact method (see §3.3). For a selected set of difficult instances, the table

Table 3.8: Comparison of ng-tour and ngL-tour relaxations

		BMR.ng with ng-tour				BMR.ng with ngL-tour			
Class	*#instances*	*Solved*	*%LB*	T_{LB}	T_{tot}	*Solved*	*%LB*	T_{LB}	T_{tot}
Ascheuer Easy	32	32	99.00	1.4	1.7	32	99.99	2.6	2.8
Ascheuer Hard	18	18	99.60	23.5	33.4	18	99.70	89.2	114.1
Pesant	27	27	99.95	3.1	3.3	27	99.95	4.2	4.3
Potvin	28	28	99.50	8.9	9.7	28	99.67	11.7	12.2
Gendreau	140	140	99.59	19.1	37.7	140	99.78	35.6	36.9
Olhmann	25	20	99.25	197.1	242.6	24	99.55	340.1	404.7
Avg			99.52	28.0	42.1		99.78	56.8	65.0
Sum	270	265				269			

Table 3.9: Effectiveness of the Dominance and Fathoming Rules

	Fath 1		*Dom* 1		*Fath* 2		*Undominated*	
Inst	n_1	$\%n_1$	n_2	$\%n_2$	n_3	$\%n_3$	n_4	$\%n_4$
rbg049a	71	54.7	4	5.4	50	89.4	6	4.5
rbg050b	13	40.2	1	2.5	18	92.9	2	4.1
rbg050c	1,275	52.5	80	6.9	979	91.3	93	3.8
rbg193.2	6,394	70.6	1,027	38.6	1,219	74.6	414	4.6
rbg233.2	166,371	75.0	24,763	44.7	21,671	70.7	8,969	4.0
n150w120.2	369,823	76.5	6,281	5.5	95,468	88.7	12,149	2.5
n150w140.1	579,249	79.9	3,506	2.4	126,949	89.4	14,993	2.1
n200w120.2	629,067	77.7	5,461	3.0	154,832	88.4	20,223	2.5
n200w120.3	2,338,415	80.5	23,374	4.1	486,025	89.4	57,625	2.0
n200w120.5	1,389,485	81.8	9,951	3.2	270,429	90.3	29,072	1.7
n200w140.1	320,984	82.8	1,673	2.5	58,525	90.3	6,261	1.6

$\%n_1 = n_1/(n_1+n_2+n_3+n_4)*100$ $\%n_2 = n_2/(n_2+n_3+n_4)*100$
$\%n_3 = n_3/(n_3+n_4)*100$ $\%n_4 = n_4/(n_1+n_2+n_3+n_4)*100$

reports the following columns: the number of states fathomed by Fathoming 1 (n_1), the percentage of times Fathoming 1 is successfully applied ($\%n_1$), the number of states dominated by Dominance 1 (n_2), the percentage of times Dominance 1 is successfully applied ($\%n_2$), the number of states fathomed by Fathoming 2 (n_3), the percentage of times Fathoming 2 is successfully applied ($\%n_3$), the number of undominated states (n_4), and the percentage of undominated states on the number of states generated ($\%n_4$).

From Table 3.9, the effectiveness of the dominance and fathoming rules is noticeable. In particular, it is remarkable that 70 to 90% of the states that are not discarded by Fathoming 1 and Dominance 1 can be fathomed with Fathoming 2.

3.8 Conclusions

In this chapter, we described an exact dynamic programming algorithm for the Traveling Salesman Problem with Time Windows (TSPTW) based on two tour relaxations called ng-tour and ngL-tour relaxations.

We reported extensive computational results on several classes of both real-world and random instances taken from the literature and used them to test both exact and heuristic methods involving up to 233 vertices.

The exact algorithm solved all but one instance and outperforms all previously published exact methods for the TSPTW, solving 136 out of 270 instances for the first time.

Chapter 4

Vehicle Routing Problem with Time Windows

[1]

In this chapter, we describe an effective exact method for solving both the *capacitated vehicle routing problem* (CVRP) and the *vehicle routing problem with time windows* (VRPTW) that improves the method proposed by Baldacci et al. [2008] for the CVRP. The proposed algorithm is based on the set partitioning formulation of the problem. The *ng*-route relaxation, introduced in Chapter 3, is used by different dual-ascent heuristics to find near-optimal dual solutions of the linear relaxation of the set partitioning model. We describe a column-and-cut generation algorithm strengthened by valid inequalities that uses a new strategy for solving the pricing problem. The *ng*-route relaxation and the different dual solutions achieved allow us to generate a reduced set partitioning problem containing all routes of any optimal solution that is finally solved by an integer programming solver. The proposed method solves 4 of the 5 open Solomon VRPTW instances and significantly improves the running times of state-of-the-art algorithms for both VRPTW and CVRP.

4.1 Introduction

The problem of supplying customers with vehicles based at a central depot is generally known as *vehicle routing problem* (VRP). The solution of a VRP calls for the design of a set of *routes*, each performed by a vehicle that starts and ends at the depot, such that all customers are serviced, a set of operational constraints are satisfied, and the total routing cost is minimized.

[1]This chapter is based on Baldacci et al. [2010, 2011a, 2012a]

The two most studied members of the VRP family are the *capacitated vehicle routing problem* (CVRP) and the *vehicle routing problem with time windows* (VRPTW). In the CVRP, a fleet of identical capacitated vehicles located at a central depot must be optimally routed to supply a set of customers with known demands. Each vehicle can perform at most one route, and the total demand delivered by a route cannot exceed the vehicle capacity. The VRPTW generalizes the CVRP by imposing that each customer must be visited within a specified time interval, called *time window*. The CVRP is $\mathcal{NP}$-hard, so is the VRPTW.

4.2 Literature Review

The most effective exact algorithms for the CVRP are owed to Baldacci et al. [2004], Lysgaard et al. [2004], Fukasawa et al. [2006], and Baldacci et al. [2008]. Baldacci et al. [2004] described a *branch-and-cut* (BC) algorithm based on a two-commodity network flow formulation of the problem. Lysgaard et al. [2004] proposed a BC algorithm that enhances the method of Augerat et al. [1995]. The method of Fukasawa et al. [2006] combines a BC based on the algorithm of Lysgaard et al. [2004] with a new *branch-and-cut-and-price* (BCP) algorithm based on the two-index and the *set partitioning* (SP) formulations. The lower bound is computed with a column-and-cut generation method that uses k-cycle-free q-routes (with k up to 4) instead of feasible CVRP routes and the valid inequalities used by Lysgaard et al. [2004]. The algorithm decides at the root node to use either a pure BC or the BCP algorithm. Baldacci et al. [2008] presented an algorithm based on the SP model strengthened with capacity and clique inequalities and were the first to compute lower bounds based on elementary routes. Their algorithm could not solve three instances solved by Fukasawa et al. [2006] but is faster with regards to the problems solved by both methods.

The first exact algorithm for the VRPTW based on the SP formulation was the *branch-and-price* (BP) algorithm of Desrochers et al. [1992]. This method was improved by Kohl et al. [1999] by adding 2-path inequalities to linear relaxation of the SP formulation. Kohl and Madsen [1997] proposed a branch-and-bound algorithm where the lower bounds were computed using subgradient and bundle methods. These methods were based on 2-cycle elimination algorithms. Irnich and Villeneuve [2006] described a BP algorithm where the pricing subproblem is solved with a k-cycle elimination procedure. Algorithms based on elementary routes were proposed by Feillet et al. [2004] and Chabrier [2006]. Jepsen et al. [2008] described a BCP algorithm based on the SP model and Subset-Row (SR) inequalities. This method was improved by Desaulniers et al. [2008] by adding both SR and generalized k-path inequalities and by using a tabu search heuristic, instead of dynamic programming (DP) algorithms, to rapidly generate negative reduced cost routes. Their method outperforms all other algorithms, remarkably decreasing the computational time, and solving 5 of the 10 open Solomon instances.

In general, any exact algorithm for the VRPTW based on the SP model can be easily adapted to solve the CVRP by simply relaxing the time window constraints in the pricing algorithm. Nonetheless, such a simple adaptation might not be effective, and none of the methods published in the literature so far for the VRPTW have been proven to effectively solve the CVRP.

The method of Baldacci et al. [2008] for the CVRP is based on an exact solution framework that can be tailored to solve the VRPTW as well as several other variants of the VRP. The method has the following steps: (i) use different bounding procedures to find near-optimal dual solutions of the linear relaxation of the SP model strengthened by valid inequalities; (ii) use a column-and-cut generation procedure to reduce the integrality gap by adding, in a cutting plane fashion, both strengthened capacity and clique constraints; (iii) use the final dual solution achieved at Step (ii) to generate a reduced SP problem containing only the routes of reduced cost less than or equal to the gap between a known upper bound and the lower bound obtained; (iv) solve the resulting reduced problem with an *integer programming* (IP) solver.

The key components of this method are the bounding procedures of Step (i) that are based on the state-space relaxation technique, introduced in Christofides et al. [1981c], to extend the route set with a relaxation of feasible routes easier to compute, and the use of bounding functions to reduce the state-space graph computed by DP when solving the pricing problem and generating the final SP model.

In this chapter, we describe an exact method to solve both the VRPTW and the CVRP that improves the method of Baldacci et al. [2008]. We show that the *ng*-route relaxation (described in Chapter 3 for the TSPTW) improves other nonelementary route relaxations proposed for the CVRP and VRPTW. This relaxation is particularly effective for difficult VRPTW instances with wide time windows and loose capacity constraints. The *ng*-route relaxation is used at Step (i) to derive a new dual-ascent heuristic and at Steps (ii) and (iii) to reduce the state-space graph of the DP algorithm in generating elementary routes. We also describe a new family of valid inequalities, called *weak subset-row inequalities*, that are a relaxation of SR inequalities. The main advantage of these new inequalities is that their duals can be implicitly considered while solving the pricing problem. Moreover, we propose new ideas to improve the pricing algorithm in the column-and-cut procedure based on the use of the dual solution achieved at Step (i) to eliminate routes of negative reduced costs with respect to the current dual solution that cannot be in any optimal solution. Finally, we report computational results for both VRPTW and CVRP showing that the proposed method solves 4 of the 5 open Solomon VRPTW instances and significantly improves the running times of state-of-the-art algorithms for both VRPTW and CVRP.

4.3 Mathematical Formulation and its Relaxations

The VRPTW is defined on a complete digraph $G = (V', A)$, where $V' = \{0, 1, \dots, n\}$ is a set of $n + 1$ vertices and A is the arc set. Vertex 0 represents the depot, and the vertex subset $V = V' \setminus \{0\}$ corresponds to n customers. A demand q_i and a time window $[e_i, l_i]$ are associated with each vertex $i \in V'$, where e_i and l_i represent the earliest and the latest time to visit vertex i. Time windows are "hard", meaning that if a vehicle visits a customer before time e_i, the service is postponed until time e_i. We assume that $q_0 = 0$. A travel cost d_{ij} and a travel time $t_{ij} > 0$ are associated with each arc $(i, j) \in A$; without loss of generality, we assume that travel time t_{ij} includes the service time at vertex i, so the departure time from any customer $i \in V$ coincides with the end of its service. We also assume that matrices d_{ij} and t_{ij} satisfy the triangle inequality; therefore, time windows, travel times, and customer demands can be used to properly reduce the arc set A as described in Desrochers et al. [1992]. For each vertex $i \in V'$, we indicate with $\Gamma_i \subseteq V'$ the set of *successors* of vertex i in graph G (i.e., $\Gamma_i = \{j \in V' : (i, j) \in A\}$) and with $\Gamma_i^{-1} \subseteq V'$ the set of *predecessors* of vertex i in graph G (i.e., $\Gamma_i^{-1} = \{j \in V' : (j, i) \in A\}$).

A fleet of m identical vehicles of capacity Q stationed at the depot must fulfill customer demands. A vehicle route $R = (0, i_1, \dots, i_r, 0)$, with $r \geq 1$, is a simple circuit in graph G that passes through the depot, visits vertices $V(R) = \{0, i_1, \dots, i_r\} \subseteq V'$, within their time windows, leaves the depot 0 at time e_0, returns to the depot 0 before time l_0, and such that the total demand of visited customers does not exceed the vehicle capacity Q (i.e., $\sum_{i \in V(R)} q_i \leq Q$). The cost of route R is the sum of the travel costs of the arc set, $A(R)$, traversed by route R.

The VRPTW consists of designing at most m routes of minimum total cost such that each customer is visited exactly once by exactly one route.

The following notation is used in the rest of this chapter. Given a set of vertices $S \subseteq V$, we indicate with $q(S) = \sum_{i \in S} q_i$ the total demand of the customers of the set S and with $k(S)$ the minimum number of vehicles needed to serve all customers of the set S. We also denote with $q_{min} = \min\{\min_{i \in V}\{q_i\},\ q(V) - (m - 1)Q\}$ the minimum customer demand delivered by any feasible route. Finally, we denote with z_{UB} a valid upper bound on the optimal solution cost of the VRPTW.

We also define a *forward path* $F = (0, i_1, \dots, i_{k-1}, i_k)$ as an elementary path that starts from depot 0 at time e_0, visits vertices $V(F) = \{0, i_1, \dots, i_{k-1}, i_k\}$ within their time windows, and ends at customer $\sigma_F = i_k$ at time $t_F \in [e_{\sigma_F}, l_{\sigma_F}]$. We denote by $A(F)$ the set of arcs traversed by path F and by $c(F) = \sum_{(i,j) \in A(F)} d_{ij}$ the cost of path F. Similarly, a *backward path* $B = (\sigma_B = i_k, i_{k+1}, \dots, i_h, 0)$ is a path that starts from vertex σ_B at time t_B, visits customers $V(B) = \{i_k, i_{k+1}, \dots, i_h, 0\}$ within their time windows, and ends at the depot 0 before time l_0.

4.3.1 Set Partitioning Formulation

Let $\mathscr{R}$ be the index set of all feasible routes in graph G, and let $a_{i\ell}$, $i \in V'$, $\ell \in \mathscr{R}$, be a binary coefficient equal to 1 if $i \in V(R_\ell)$, 0 otherwise (we assume that $a_{0\ell} = 1$, for each route $\ell \in \mathscr{R}$). Each route $\ell \in \mathscr{R}$ has an associated cost c_ℓ. Let x_ℓ, $\ell \in \mathscr{R}$, be a binary variable equal to 1 if and only if route ℓ is in the optimal solution (0 otherwise). The SP formulation of the VRPTW is

$$(P) \quad z(P) = \min \sum_{\ell \in \mathscr{R}} c_\ell x_\ell, \tag{4.1}$$

$$s.t. \sum_{\ell \in \mathscr{R}} a_{i\ell} x_\ell = 1, \quad i \in V, \tag{4.2}$$

$$\sum_{\ell \in \mathscr{R}} x_\ell \leq m, \tag{4.3}$$

$$x_\ell \in \{0,1\}, \qquad \ell \in \mathscr{R}. \tag{4.4}$$

Constraints (4.2) specify that each customer $i \in V$ must be visited by exactly one route. Constraint (4.3) requires that at most m routes are selected.

4.3.2 Relaxation LP

The linear relaxation of formulation P can be strengthened with the following valid inequalities.

i) **Capacity constraints (CCs)**. Let $\mathscr{S} = \{S : S \subseteq V, |S| \geq 2\}$ be the set of all subsets of vertices of cardinality greater than 1, and let $\rho_\ell(S) = |\{(i,j) \in A(R_\ell) : i \in V' \setminus S, j \in S\}|$ be the number of times route $\ell \in \mathscr{R}$ enters the set of vertices S. The capacity constraints (CCs) are

$$\sum_{\ell \in \mathscr{R}} \rho_\ell(S) x_\ell \geq k(S), \quad S \in \mathscr{S}. \tag{4.5}$$

In solving the VRPTW, we ignore CCs, whereas in solving the CVRP we consider only a subset of CCs a priori generated as described in Baldacci et al. [2008].

ii) **Strengthened capacity constraints (SCs)**. These inequalities, introduced by Baldacci et al. [2004], lift CCs and are given by inequalities (4.5) where the route coefficient $\rho_\ell(S)$ is equal to 1 if $V(R_\ell) \cap S \neq \varnothing$ and 0 otherwise.

iii) **Subset-row inequalities (SR3s)**. Let $\mathscr{C} = \{C \subseteq V : |C| = 3\}$ be the set of all customer triplets, and let $\mathscr{R}(C) \subseteq \mathscr{R}$ be the subset of routes serving at least two customers in C (i.e., $\mathscr{R}(C) = \{\ell \in \mathscr{R} : |V(R_\ell) \cap C| \geq 2\}$). Subset-row inequalities (SR3s) are

$$\sum_{\ell \in \mathscr{R}(C)} x_\ell \leq 1, \quad C \in \mathscr{C}. \tag{4.6}$$

Inequalities SR3s correspond to a subset of SRs and clique inequalities used by Jepsen et al. [2008] for solving the VRPTW and by Baldacci et al. [2008] for solving the CVRP, respectively. Hereafter, with $C \in \mathscr{C}$ we refer to both the index and the customer triplet of an SR3 inequality.

iv) **Weak subset-row inequalities (WSR3s).** These inequalities are a relaxed form of the SR3s (4.6) where, given the triplet $C \in \mathscr{C}$, the route set $\mathscr{R}(C)$ contains only those routes traversing at least one arc (i, j) with $i, j \in C$. The reason for using WSR3s instead of SR3s is that WSR3 dual variables can be implicitly considered in solving the pricing problem. The SR3s (4.6) and, thus, the WSR3s are separated by complete enumeration.

We denote by LP the linear relaxation of formulation P strengthened with inequalities (4.5) and (4.6) and by $z(LP)$ its optimal solution cost. Moreover, we denote by D the dual of problem LP. The dual variables are given by the vectors $\boldsymbol{u} = (u_0, u_1, \ldots, u_n)$, $\boldsymbol{v} = (v_1, v_2, \ldots, v_{|\mathscr{S}|})$, and $\boldsymbol{g} = (g_1, g_2, \ldots, g_{|\mathscr{C}|})$, where $u_1, \ldots, u_n \in \mathbb{R}$ are the dual variables of constraints (4.2), $u_0 \in \mathbb{R}_-$ is the dual variable associated with constraint (4.3), $\boldsymbol{v} \in \mathbb{R}_+^{|\mathscr{S}|}$ are the dual variables of inequalities (4.5), and $\boldsymbol{g} \in \mathbb{R}_-^{|\mathscr{C}|}$ are the dual variables of inequalities (4.6).

By enlarging the route set $\mathscr{R}$ to contain also nonnecessarily elementary routes, we can design efficient dual-ascent heuristic procedures to find near-optimal solutions of D. In §4.5, we describe three bounding procedures, called H^1, H^2 and H^3, where procedure H^k provides lower bound LB_k corresponding to the cost of both a feasible D solution $(\boldsymbol{u^k}, \boldsymbol{v^k}, \boldsymbol{g^k})$ and a nonnecessarily feasible P solution $\boldsymbol{x^k}$. In §4.6, we describe a column-and-cut generation bounding procedure, called H^4, for solving problem LP, that computes lower bound LB_4 corresponding to the D solution $(\boldsymbol{u^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$. In the following, we denote with c_ℓ^k the reduced cost of route $\ell \in \mathscr{R}$ with respect to the D solution $(\boldsymbol{u^k}, \boldsymbol{v^k}, \boldsymbol{g^k})$.

Procedure H^4 differs from classical column-and-cut generation methods for the new strategy for solving the pricing problem and the use of the D solution $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3})$ to reduce the size of the route set $\mathscr{R}$ by removing any route such that $c_\ell^3 > z_{UB} - LB_3$. The usage of the D solution $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3})$ has the main benefits of (i) eliminating routes of negative reduced cost when solving the pricing problem in H^4 and (ii) improving the final lower bound LB_4. Procedures H^1, H^2, H^3 and H^4 are executed in sequence, and the solution $(\boldsymbol{u^k}, \boldsymbol{v^k}, \boldsymbol{g^k})$ of H^k is used to hot-start procedure H^{k+1}, $k = 1, 2, 3$.

4.3.3 *ng*-Route Relaxation for the CVRP

In §3.4.2, we introduced the *ng*-route relaxation for the TSPTW. The same route relaxation can be used for obtaining valid lower bounds to the VRPTW, as well. In particular, it is used in bounding procedures H^2, H^3, and H^4 for solving the pricing subproblems. In this chapter, as route relaxations of VRPTW elementary routes, we use (t,i)-routes and (NG,t,i)-routes, which correspond exactly to t-tours and ng-tours described in §§3.4.1 and 3.4.2. As to the CVRP, we use the (q,i)-path and the ng-route relaxations that can be described as follows.

A (q,i)-path is a nonnecessarily elementary path that starts from the depot, visits a set of customers of total demand equal to q, and ends at vertex $i \in V'$. The cost $f(q,i)$ of a least-cost (q,i)-path can be computed as described by Christofides et al. [1981b]. A (q,i)-route is a $(q,0)$-path.

A *forward ng-path* (NG,q,i) is a nonnecessarily elementary path $F=(0,i_1,\ldots,i_{k-1},$ $i_k=i)$ that starts from the depot 0, visits a set of customers (each once or more) of total demand equal to q such that $NG=\Pi_F$, ends at customer $i \in V'$, and such that $i \notin \Pi_{F'}$, where $F'=(0,i_1,\ldots,i_{k-1})$.

We denote by $f(NG,q,i)$ the cost of a least-cost forward ng-path (NG,q,i). An (NG,q,i)-route is an $(NG,q,0)$-path. Functions $f(NG,q,i)$ can be computed using DP recursions similar to (3.3) as follows.

Define the state-space graph $\widehat{\mathscr{G}_F}=(\widehat{\mathscr{F}},\widehat{\mathscr{A}_F})$, where the vertex set is defined as

$$\widehat{\mathscr{F}}=\{(NG,q,i)\,:\,i\in V',q\in[q_i,Q],\forall NG\subseteq N_i \text{ s.t. } NG\ni i \text{ and } \sum_{j\in NG} q_j\le q\},$$

and the arc set as

$$\widehat{\mathscr{A}_F}=\{((NG',q',j),(NG,q,i))\,:\,(NG',q',j),(NG,q,i)\in\widehat{\mathscr{F}},$$

$$j\in\Gamma_i^{-1},q'=q-q_i,\forall NG'\subseteq N_j \text{ s.t. } NG'\ni j \text{ and } NG'\cap N_i=NG\setminus\{i\}\}.$$

The DP recursion for computing functions $f(NG,q,i)$ is

$$f(NG,q,i)=\min_{(NG',q',j)\,:\,((NG',q',j),(NG,q,i))\in\widehat{\mathscr{A}_F}}\{f(NG',q',j)+d_{ji}\},\quad (NG,q,i)\in\widehat{\mathscr{F}}. \tag{4.7}$$

By using the transpose of the cost matrix $[d_{ij}]$, we can compute the reverse functions $f^{-1}(q,i)$ and $f^{-1}(NG,q,i)$ with recursions similar to those used for $f(q,i)$ and $f(NG,q,i)$. For symmetric CVRPs, we have $f^{-1}(q,i)=f(q,i)$ and $f^{-1}(NG,q,i)=f(NG,q,i)$.

4.4 An Exact Algorithm

In this section, we describe an exact algorithm for solving both the VRPTW and the CVRP. The algorithm generates a reduced problem $\widetilde{P}$ obtained from P by replacing the route set $\mathscr{R}$ with a smaller route set $\widetilde{\mathscr{R}}\subseteq\mathscr{R}$ containing any optimal solution and solves $\widetilde{P}$ with an IP solver. The core of the algorithm is the bounding procedures H^1, H^2, H^3, and H^4 described in §§4.5 and 4.6. The algorithm can be described as follows.

Step 1. Execute in sequence bounding procedures H^1, H^2 and H^3. If the LP solution $\boldsymbol{x^k}$ corresponding to lower bound LB_k is a feasible P solution, for some $k \in \{1, 2, 3\}$, $\boldsymbol{x^k}$ is an optimal solution, stop.

Step 2. Call procedure `GenR` (see §4.5.5) to generate the largest route set $\mathscr{R}^3 \subseteq \mathscr{R}$ such that (i) $c^3_\ell \leq z_{UB} - LB_3$, $\ell \in \mathscr{R}^3$, and (ii) $|\mathscr{R}^3| \leq \Delta(\mathscr{R})$, where $\Delta(\mathscr{R})$ is a parameter. If $|\mathscr{R}^3| < \Delta(\mathscr{R})$, $\mathscr{R}^3$ contains the routes of any optimal solution of cost less than or equal to z_{UB} and is defined *optimal*. Otherwise, $\mathscr{R}^3$ is defined *nonoptimal*.

Step 3. Call procedure H^4 to compute lower bound LB_4 corresponding to the cost of the D solution $(\boldsymbol{u^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$. If the optimal LP solution $\boldsymbol{x^4}$ is integer, stop.

Step 4. We have two cases

(a) If the route set $\mathscr{R}^3$ is optimal, select the subset $\widetilde{\mathscr{R}} \subseteq \mathscr{R}^3$ of routes such that $c^4_\ell \leq z_{UB} - LB_4$.

(b) If the route set $\mathscr{R}^3$ is nonoptimal, call `GenRF` (see §4.7) to compute the set of routes $\widetilde{\mathscr{R}}$ such that

$$\left.\begin{array}{ll} c^3_\ell \leq z_{UB} - LB_3 & \ell \in \widetilde{\mathscr{R}} \\ c^4_\ell \leq z_{UB} - LB_4 & \ell \in \widetilde{\mathscr{R}} \\ |\widetilde{\mathscr{R}}| \leq \Delta(\mathscr{R}) & \end{array}\right\} \tag{4.8}$$

If $|\widetilde{\mathscr{R}}| < \Delta(\mathscr{R})$, then $\widetilde{\mathscr{R}}$ contains the routes of any optimal solution and is defined *optimal*.

Step 5. Solve problem $\widetilde{P}$ derived from P by replacing the route set $\mathscr{R}$ with the route set $\widetilde{\mathscr{R}}$ and by adding the subsets of CCs and SR3s saturated by the optimal LP solution produced by H^4. Let $z(\widetilde{P})$ be the cost of the optimal solution $\tilde{\boldsymbol{x}}$ of $\widetilde{P}$ (we assume $z(\widetilde{P}) = \infty$ if no feasible solution exists). If the route set $\widetilde{\mathscr{R}}$ is optimal, then $\tilde{\boldsymbol{x}}$ is an optimal solution; otherwise, $\min\{z(\widetilde{P}), z_{UB}\}$ is a valid upper bound to $z(P)$.

4.5 Bounding Procedures H^1, H^2 and H^3

Bounding procedures H^1, H^2 and H^3 use the same column-and-cut generation method, called CCG, to find lower bounds LB_1, LB_2 and LB_3 corresponding to the cost of three near-optimal D solutions. Procedures H^1 and H^2 are based on two different route relaxations and add CCs. Procedure H^3 is based on elementary routes and adds both SCs and WSR3s.

4.5.1 Algorithm CCG

Algorithm CCG differs from standard column-and-cut generation methods based on the simplex algorithm because it uses a dual-ascent heuristic algorithm to solve the master problem. Procedure *CCG* was proposed by Baldacci et al. [2008] for the CVRP and used by Boschetti et al. [2008] for the set partitioning problem, who showed that *CCG* is faster than the simplex because it is not affected by the typical degeneracy of the simplex.

Define $A(C) = \{(i,j) \in A : i,j \in C\}$, and let $\mathscr{C}_\ell = \{C \in \mathscr{C} : |A(C) \cap A(R_\ell)| \geq 1\}$ be the subset of WSR3s involving the route ℓ. Algorithm *CCG* is based on the following theorem, which is a straightforward extension of Theorem 1 described in §3.5.2.

Theorem 1. Let $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \ldots, \lambda_n)$, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{|\mathscr{S}|})$, $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_{|\mathscr{C}|})$ be three vectors of penalties, where penalty $\lambda_0 \in \mathbb{R}_-$ is associated with constraint (4.3), penalties $\lambda_i \in \mathbb{R}$, $i \in V$, with constraints (4.2), penalties $\mu_S \in \mathbb{R}_+$, $S \in \mathscr{S}$, with constraints (4.5) in the form of either SCs or CCs, and penalties $\omega_C \in \mathbb{R}_-$, $C \in \mathscr{C}$, with constraints (4.6) in the relaxed form of WSR3s. Let $\widehat{\mathscr{R}}$ be the index set of nonnecessarily elementary routes given by some SSR. A feasible dual solution $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g})$ of problem *LP* of cost $z(D(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}))$ is obtained as

$$\left.\begin{array}{ll} u_i = b_i + \lambda_i & i \in V \\ u_0 = \lambda_0 & \\ v_S = \mu_S & S \in \mathscr{S} \\ g_C = \omega_C & C \in \mathscr{C} \end{array}\right\}, \tag{4.9}$$

where b_i, $i \in V$, is defined as

$$b_i = q_i \min_{\ell \in \widehat{\mathscr{R}} : a_{i\ell} \geq 1} \left\{ \frac{c_\ell - \sum_{i \in V'} a_{i\ell}\lambda_i - \sum_{S \in \mathscr{S}} \rho_\ell(S)\mu_S - \sum_{C \in \mathscr{C}_\ell} \omega_C}{\sum_{i \in V} a_{i\ell} q_i} \right\}. \tag{4.10}$$

Proof. Consider a route $\ell \in \widehat{\mathscr{R}}$. From the definition of the dual solution $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g})$, we derive

$$u_i \leq q_i \frac{c_\ell - \sum_{i \in V'} a_{i\ell}\lambda_i - \sum_{S \in \mathscr{S}} \rho_\ell(S)\mu_S - \sum_{C \in \mathscr{C}_\ell} \omega_C}{\sum_{i \in V} a_{i\ell} q_i} + \lambda_i, \quad i \in V : a_{i\ell} \geq 1.$$

Thus, for each route $\ell \in \widehat{\mathscr{R}}$, the following relation holds

$$\sum_{i \in V'} a_{i\ell} u_i \leq \sum_{i \in V'} a_{i\ell} q_i \frac{c_\ell - \sum_{i \in V'} a_{i\ell}\lambda_i - \sum_{S \in \mathscr{S}} \rho_\ell(S)\mu_S - \sum_{C \in \mathscr{C}_\ell} \omega_C}{\sum_{i \in V} a_{i\ell} q_i} + \sum_{i \in V'} a_{i\ell}\lambda_i$$

$$= c_\ell - \sum_{S\in\mathscr{S}} \rho_\ell(S)\mu_S - \sum_{C\in\mathscr{C}_\ell} \omega_C,$$

which corresponds to the constraint of problem D for route $\ell \in \widehat{\mathscr{R}}$

$$\sum_{i\in V'} a_{i\ell}u_i + \sum_{S\in\mathscr{S}} \rho_\ell(S)v_S + \sum_{C\in\mathscr{C}_\ell} g_C \leq c_\ell. \square$$

Algorithm CCG is a column-and-cut generation-like method for solving the problem

$$LCG = \max_{(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\omega})} \{z(D(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\omega}))\}. \tag{4.11}$$

Procedure CCG executes a number of macro-iterations to compute a dual solution $(\boldsymbol{u'},\boldsymbol{v'},\boldsymbol{g'})$ of the master problem, defined by the route subset $\mathscr{R}' \subseteq \mathscr{R}$, solving problem (4.11) with a predefined number $Maxit2$ of subgradient iterations to modify the penalty vectors $(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\omega})$.

Baldacci et al. [2008] have shown that a valid subgradient of function $z(D(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\omega}))$ at point $(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\omega})$ can be computed by associating, with the current D solution, a nonnecessarily feasible LP solution $\boldsymbol{x}$ of cost $z(LP) = z(D(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\omega}))$ defined as follows. Let $\widetilde{\mathscr{R}}$ be the index set of the distinct routes producing b_i, $i \in V$, in expressions (4.10), and let $\ell(i)$ be the index of the route in $\widetilde{\mathscr{R}}$ associated with b_i, $i \in V$. Solution $\boldsymbol{x}$ is computed as $x_\ell = \sum_{i\in V} \frac{a_{i\ell}q_i}{\sum_{i\in V} a_{i\ell}q_i}\xi^i_\ell$, $\ell \in \widetilde{\mathscr{R}}$, by setting $\xi^i_{\ell(i)} = 1$ and $\xi^i_\ell = 0$, for each route $\ell \in \widetilde{\mathscr{R}} \setminus \{\ell(i)\}$, and each customer $i \in V$. The values of the left-hand-side of constraints (4.2), (4.3), (4.5), and (4.6) with respect to $\boldsymbol{x}$ are used to update penalty vectors $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, and $\boldsymbol{\omega}$ by means of the usual subgradient expressions, where we use $z_{UB} = 1.2z(LP)$ instead of a feasible upper bound.

The set $\mathscr{S}$ of CCs (or SCs) is generated a priori (see §4.3). After computing the master dual solution $(\boldsymbol{u'},\boldsymbol{v'},\boldsymbol{g'})$, CCG adds to the master a subset of WSR3 inequalities violated by the LP solution $\boldsymbol{x}$. Moreover, CCG generates a subset $\mathscr{N}$ of routes of negative reduced cost with respect to $(\boldsymbol{u'},\boldsymbol{v'},\boldsymbol{g'})$. If $\mathscr{N} \neq \varnothing$, then CCG sets $\mathscr{R}' = \mathscr{R}' \cup \mathscr{N}$; otherwise, $(\boldsymbol{u'},\boldsymbol{v'},\boldsymbol{g'})$ is a feasible D solution. CCG terminates after $Maxit1$ macro iterations ($Maxit1$ defined a priori).

We denote by $(\boldsymbol{u^*},\boldsymbol{v^*},\boldsymbol{g^*})$ and $\boldsymbol{x^*}$ the final D and LP solutions of cost LCG achieved by CCG using penalty vectors $(\boldsymbol{\lambda^*},\boldsymbol{\mu^*},\boldsymbol{\omega^*})$, respectively.

A step-by-step description of procedure CCG is the following.

Step 1. *Initialization.* A route set $\mathscr{R}' \subseteq \widehat{\mathscr{R}}$ is generated to initialize the master problem which corresponds to LP, where $\mathscr{R}$ is replaced with $\mathscr{R}'$. We assume that $\mathscr{R}'$ contains at least one route passing through each customer $i \in V$. The route set $\mathscr{R}'$ and the subsets $\mathscr{S}$ and $\mathscr{C}$ of SCs and WSR3s are generated, and $(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\omega})$

are initialized as described in §§4.5.2, 4.5.3, and 4.5.4. Set $LCG = 0$ and $iter = 1$.

Step 2. *Find a master dual solution* $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$ *of cost* z'. A near-optimal dual solution $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$ of cost z' of the master problem is computed by an iterative procedure that initializes $z' = 0$ and performs $Maxit2$ iterations of the following operations:

(i) Compute a dual solution $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g})$ of the master of cost z by means of expressions (4.9) and (4.10), where $\widehat{\mathscr{R}}$ is replaced with $\mathscr{R}'$, using the current vectors $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})$. Let $\widetilde{\mathscr{R}}$ be the index set of the distinct routes producing b_i, $i \in V$, in expressions (4.10), and let $\ell(i)$ be the index of the route in $\mathscr{R}'$ associated with b_i, $i \in V$. Define a nonnecessarily feasible LP solution $\boldsymbol{x}$ as $x_\ell = \sum_{i \in V} \frac{a_{i\ell} q_i}{\sum_{i \in V} a_{i\ell} q_i} \xi^i_\ell$, $\ell \in \widetilde{\mathscr{R}}$, by setting $\xi^i_{\ell(i)} = 1$, and $\xi^i_\ell = 0$, for each route $\ell \in \widetilde{\mathscr{R}} \setminus \{\ell(i)\}$, and each customer $i \in V$. If $z > z'$, update $z' = z$, $\boldsymbol{x'} = \boldsymbol{x}$, $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'}) = (\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g})$.

(ii) Update the penalty vectors $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})$ as follows. Compute

- $\alpha_0 = \displaystyle\sum_{\ell \in \widetilde{\mathscr{R}}} x_\ell$ and $\alpha_i = \displaystyle\sum_{\ell \in \widetilde{\mathscr{R}}} a_{i\ell} x_\ell$, $i \in V$;
- $\beta_S = \displaystyle\sum_{\ell \in \widetilde{\mathscr{R}}} b_\ell(S) x_\ell$, $S \in \mathscr{S}$;
- $\delta_C = \displaystyle\sum_{\ell \in \mathscr{R}(C) \cap \widetilde{\mathscr{R}}} \left\lfloor \frac{\sum_{i \in C} a_{i\ell}}{2} \right\rfloor x_\ell$, $C \in \mathscr{C}$.

Then, vectors $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})$ are modified as follows

$$\left.\begin{array}{ll} \lambda_i = \lambda_i - \epsilon\gamma(\alpha_i - 1), & i \in V \\ \lambda_0 = \min\{0, \lambda_0 - \epsilon\gamma(\alpha_0 - m)\} & \\ \mu_S = \max\{0, \mu_S - \epsilon\gamma(\beta_S - k(S)\}, & S \in \mathscr{S} \\ \omega_C = \min\{0, \omega_C - \epsilon\gamma(\delta_C - 1)\}, & C \in \mathscr{C} \end{array}\right\}$$

where ϵ is a positive constant and

$$\gamma = \frac{0.2z'}{\sum_{i \in V} (\alpha_i - 1)^2 + (\alpha_0 - m)^2 + \sum_{S \in \mathscr{S}} (\beta(S) - k(S))^2 + \sum_{C \in \mathscr{C}} (\delta_C - 1)^2}.$$

Step 3. *Check if* $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$ *is a feasible* D *solution.* Generate (see §§4.5.2, 4.5.3, and 4.5.4) the largest subset $\mathscr{N} \subseteq \widehat{\mathscr{R}}$ of routes of negative reduced cost with respect to the current dual master solution $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$ and such that $|\mathscr{N}| \leq \Delta(\mathscr{N})$ ($\Delta(\mathscr{N})$ is an a priori defined parameter). If $\mathscr{N} = \varnothing$ and z' is greater than LCG, then $LCG = z'$, $(\boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{g}^*) = (\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$, $\boldsymbol{x}^* = \boldsymbol{x'}$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\omega}^*) = (\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})$; otherwise, $\mathscr{R}' = \mathscr{R}' \cup \mathscr{N}$ is updated.

Step 4. *Termination criterion.* Set $iter = iter + 1$. If $iter = Maxit1$, stop.

Step 5. *Separate WSR3s.* Find the set $\mathscr{C}'$ of the WSR3s most violated by the current LP solution $\boldsymbol{x'}$ obtained at Step 2 and such that $|\mathscr{C}'| \leq \Delta(\mathscr{C})$. WSR3s are separated by complete enumeration.

4.5.2 Bounding Procedure H^1

Bounding procedure H^1 enlarges the route set $\mathscr{R}$ with the t-routes (to solve the VRPTW) or the q-routes (in the CVRP, see Christofides et al. [1981b]). The initial route set $\mathscr{R}'$ of the master problem contains all single-customer routes $(0, i, 0)$, $i \in V$. WSR3s are ignored (i.e., we set $\mathscr{C} = \varnothing$); solving the VRPTW, we initialize $\mathscr{S} = \varnothing$, whereas in the CVRP the set $\mathscr{S}$ contains CCs and is generated a priori as described in Baldacci et al. [2008]. We initialize $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}) = (\mathbf{0}, \mathbf{0}, \mathbf{0})$.

Define the modified arc cost $d'_{ij} = d_{ij} - \frac{1}{2}(u'_i + u'_j) - \sum_{S \in \mathscr{S}_{ij}} v'_S$, $(i, j) \in A$, with respect to the current dual solution $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$, where $\mathscr{S}_{ij} = \{S \in \mathscr{S} : (i, j) \in A,\, i \in V' \setminus S,\, j \in S\}$. The set $\mathscr{N}$ is computed as follows. If we use t-routes, we compute functions $f(t, i)$ using the modified arc costs d'_{ij} instead of d_{ij}. Let $h(i) = \min_{t \in [e_i, l_i]}\{f(t, i) + d'_{i0}\}$. The set $\mathscr{N}$ contains any t-route corresponding to $h(i) < 0$, $i \in V$. Similarly, we compute $\mathscr{N}$ when $\widehat{\mathscr{R}}$ contains q-routes.

At the end, H^1 sets $(\boldsymbol{u^1}, \boldsymbol{v^1}, \boldsymbol{g^1}) = (\boldsymbol{u^*}, \boldsymbol{v^*}, \boldsymbol{g^*})$, $\boldsymbol{x^1} = \boldsymbol{x^*}$, $(\boldsymbol{\lambda^1}, \boldsymbol{\mu^1}, \boldsymbol{\omega^1}) = (\boldsymbol{\lambda^*}, \boldsymbol{\mu^*}, \boldsymbol{\omega^*})$, $LB_1 = LCG$.

4.5.3 Bounding Procedure H^2

Procedure H^2 enlarges the route set $\mathscr{R}$ with the ng-routes and adds to LP the same set $\mathscr{S}$ of CCs used by H^1. WSR3s (4.6) are ignored. We initialize $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}) = (\boldsymbol{\lambda^1}, \boldsymbol{\mu^1}, \boldsymbol{\omega^1})$, define $d^1_{ij} = d_{ij} - \frac{1}{2}(u^1_i + u^1_j) - \sum_{S \in \mathscr{S}_{ij}} v^1_S$, $(i, j) \in A$, and compute N_i to be the $\Delta(N_i)$ nearest customers to i according to d^1_{ij}. We compute functions $f(NG, t, i)$ using d^1_{ij} instead of d_{ij} in recursions (4.7) and the costs $h(i) = \min_{(NG,t,i) \in \widehat{\mathscr{F}}}\{f(NG, t, i) + d^1_{i0}\}$, $i \in V$, of the least-cost ng-route visiting i immediately before arriving at the depot. The route set $\mathscr{R}'$ contains the ng-routes corresponding to $h(i) < 0$, $i \in V$.

At each iteration, to generate the set $\mathscr{N}$, we compute functions $f(NG, t, i)$ with the modified arc cost d'_{ij}, $(i, j) \in A$, and N_i contains the $\Delta(N_i)$ nearest customers to i according to d'_{ij}. The route set $\mathscr{N}$ contains every ng-route corresponding to $h(i) < 0$, $i \in V$.

At the end, procedure H^2 sets $(\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2}) = (\boldsymbol{u^*}, \boldsymbol{v^*}, \boldsymbol{g^*})$, $\boldsymbol{x^2} = \boldsymbol{x^*}$, $(\boldsymbol{\lambda^2}, \boldsymbol{\mu^2}, \boldsymbol{\omega^2}) = (\boldsymbol{\lambda^*}, \boldsymbol{\mu^*}, \boldsymbol{\omega^*})$, and $LB_2 = LCG$.

4.5.4 Bounding Procedure H^3

Procedure H^3 uses the set of elementary routes $\mathscr{R}$. The sets $\mathscr{R}'$ and $\mathscr{N}$ are generated using the procedure `GenR` described in §4.5.5. Given a D solution $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$ and two parameters Δ and γ, `GenR` generates at most Δ routes of reduced cost less than or equal to γ with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$.

The initial route set $\mathscr{R}'$ is obtained by setting $\Delta = \Delta^a$, $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2})$, and $\gamma = z_{UB} - LB_2$, and adding all single-customer routes to $\mathscr{R}'$. Moreover, we initialize $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}) = (\boldsymbol{\lambda^2}, \boldsymbol{\mu^2}, \boldsymbol{\omega^2})$. In generating the route set $\mathscr{N}$, we set $\Delta = \Delta^b$, $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$, and $\gamma = -\varepsilon$ (say $\varepsilon = 10^{-6}$).

Bounding procedure H^3 adds, to LP, the set $\mathscr{S}$ of CCs used by H^1 and H^2 in the form of SCs and separates WSR3s.

At the end, procedure H^3 sets $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3}) = (\boldsymbol{u^*}, \boldsymbol{v^*}, \boldsymbol{g^*})$, $\boldsymbol{x^3} = \boldsymbol{x^*}$, $(\boldsymbol{\lambda^3}, \boldsymbol{\mu^3}, \boldsymbol{\omega^3}) = (\boldsymbol{\lambda^*}, \boldsymbol{\mu^*}, \boldsymbol{\omega^*})$, and $LB_3 = LCG$.

4.5.5 Procedure `GenR`

Procedure `GenR` is a DP algorithm that generates elementary routes using bounding functions based on the ng-path relaxation. `GenR` is an extension of the algorithm described by Baldacci et al. [2008] for the symmetric CVRP that, in turn, is based on the method proposed by Mingozzi et al. [1994] and adapted by Baldacci et al. [2006] for the asymmetric CVRP on a multigraph. Similar methods have been used by Righini and Salani [2008], Jepsen et al. [2008], and Desaulniers et al. [2008].

In §4.3, we gave the definition of forward and backward paths. In addition, for a forward path $F = (0, i_1, \ldots, i_{k-1}, i_k)$, we refer to i_{k-1} with π_F and set $q_F = \sum_{i \in V(F)} q_i$, and, for a backward path $B = (i_k, i_{k+1}, \ldots, i_h, 0)$, we refer to i_{k+1} with π_B and set $q_B = \sum_{i \in V(B)} q_i$.

Let τ be a time such that $\tau \in (e_0, l_0)$ (say, $\tau = \lfloor (l_0 - e_0)/2 \rfloor$). We define $\mathscr{F}$ as the set of all forward paths such that π_F is visited at time less than τ, $F \in \mathscr{F}$, and $\mathscr{B}$ as the set of all backward paths such that π_B is visited at time greater than τ, $B \in \mathscr{B}$, plus all backward paths $B = (k, 0)$, $k \in V$. Procedure `GenR` is based on the observation that all feasible VRPTW routes can be obtained combining any pair of paths (F, B), $F \in \mathscr{F}$, $B \in \mathscr{B}$, satisfying the following feasibility conditions

$$\sigma_F = \sigma_B, \quad V(F) \cap V(B) = \{0, \sigma_F\}, \quad t_F \leq t_B, \quad q_F + q_B - q_{\sigma_F} \leq Q. \tag{4.12}$$

`GenR` is a two-phase algorithm. Given a D solution $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$, it generates at most Δ routes of reduced cost less than or equal to γ. In Phase 1, `GenR` generates the path sets $\mathscr{F}$ and $\mathscr{B}$ using a procedure called `GenP`; in Phase 2, it derives feasible routes by combining $\mathscr{F}$ and $\mathscr{B}$ using a procedure called `Combine`. `GenR` is based on the following lemma.

Lemma 1. Let $\tilde{d}_{ij} = d_{ij} - \frac{1}{2}(\tilde{u}_i + \tilde{u}_j) - \sum_{S \in \mathscr{S}_{ij}} \tilde{v}_s$ be the arc reduced costs with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$, and let $\tilde{c}(F) = \sum_{(i,j) \in A(F)} \tilde{d}_{ij}$, $F \in \mathscr{F}$, and $\tilde{c}(B) = \sum_{(i,j) \in A(B)} \tilde{d}_{ij}$, $B \in \mathscr{B}$. Let $\ell \in \mathscr{R}$ be the route of reduced cost $\tilde{c}_\ell = c_\ell - \sum_{i \in V(R_\ell)} \tilde{u}_i - \sum_{S \in \mathscr{S}} \rho_\ell(S)\tilde{v}_S - \sum_{C \in \mathscr{C}_\ell} \tilde{g}_C$, where $\mathscr{S}$ corresponds to SCs, resulting from a given pair of paths F and B satisfying conditions (4.12). The following inequality holds $\tilde{c}(F) + \tilde{c}(B) \leq \tilde{c}_\ell$.

Proof. Because paths F and B satisfy conditions (4.12) and route ℓ is the combination of F and B, we have

$$\sum_{(i,j) \in A(F)} d_{ij} + \sum_{(i,j) \in A(B)} d_{ij} = c_\ell \tag{4.13}$$

and

$$\frac{1}{2} \sum_{(i,j) \in A(F)} (\tilde{u}_i + \tilde{u}_j) + \frac{1}{2} \sum_{(i,j) \in A(B)} (\tilde{u}_i + \tilde{u}_j) = \sum_{i \in V(R_\ell)} \tilde{u}_i. \tag{4.14}$$

From the definitions of $\tilde{d}_{ij}$, $\tilde{c}(F)$ and $\tilde{c}(B)$ and from expressions (4.13) and (4.14), we derive

$$\tilde{c}(F) + \tilde{c}(B) = c_\ell - \sum_{i \in V(R_\ell)} \tilde{u}_i - \sum_{(i,j) \in A(R_\ell)} \sum_{S \in \mathscr{S}_{ij}} \tilde{v}_S. \tag{4.15}$$

Because $b_\ell(S) \in \{0, 1\}$, $S \in \mathscr{S}$, as $\mathscr{S}$ corresponds to SCs and $\tilde{v}_S \geq 0$, we have

$$\sum_{(i,j) \in A(R_\ell)} \sum_{S \in \mathscr{S}_{ij}} \tilde{v}_S \geq \sum_{S \in \mathscr{S}} b_\ell(S)\tilde{v}_S. \tag{4.16}$$

From expressions (4.15) and (4.16), we obtain

$$\tilde{c}(F) + \tilde{c}(B) \leq c_\ell - \sum_{i \in V(R_\ell)} \tilde{u}_i - \sum_{S \in \mathscr{S}} b_\ell(S)\tilde{v}_S. \tag{4.17}$$

Because $\tilde{g}_C \leq 0$, $C \in \mathscr{C}$, from the definition of $\tilde{c}_\ell$ and expression (4.17), we derive the lemma.□

`GenR` is called (i) at the beginning of H^3 to generate the route set $\mathscr{R}'$ of the initial master problem setting $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2})$ and $\gamma = z_{UB} - LB_2$; (ii) in H^3 to generate the route set $\mathscr{N}$ of negative reduced cost routes setting $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$ and $\gamma = -\varepsilon$; (iii) at Step 2 of the exact method (see §4.4) to generate the route set $\mathscr{R}^3$ setting $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3})$ and $\gamma = z_{UB} - LB_3$.

4.5.5.1 Procedure `GenP`.

Procedure `GenP` is Phase 1 of `GenR` and computes the path sets $\mathscr{F}$ and $\mathscr{B}$ using bounding functions based on the *ng*-path relaxation.

In generating the set $\mathscr{B}$, `GenP` imposes that any path $B \in \mathscr{B}$ is *undominated* by any other path $B' \in \mathscr{B}$ such that $V(B) = V(B')$, $\sigma(B) = \sigma(B')$, $c(B) \geq c(B')$, and $t(B) \leq t(B')$.

Let $\tilde{lb}(B)$ be a lower bound on the reduced cost $\tilde{c}_\ell$, with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$, of any route ℓ containing path B. Procedure `GenP` is a Dijkstra-like algorithm (see Baldacci et al. [2008]) generating $\mathscr{B}$ as a sequence of *undominated* paths $(B^1, \dots, B^h)$, with $h \leq \Delta(\mathscr{B})$, such that $\tilde{lb}(B^1) \leq \dots \leq \tilde{lb}(B^h) \leq \gamma$, where $\Delta(\mathscr{B})$ is a parameter. To compute $\tilde{lb}(B)$ we have to consider two cases

Case 1: $\tilde{\boldsymbol{g}} = \mathbf{0}$ (i.e., $\mathscr{C} = \varnothing$), functions $f(NG, t, i)$ are computed with the modified costs $\tilde{d}_{ij}$ and the subsets N_i, $i \in V$, contain the $\Delta(N_i)$ nearest customers to i according to $\tilde{d}_{ij}$. Value $\tilde{lb}(B)$ is given by

$$\tilde{lb}(B) = \tilde{c}(B) + \min_{NG \subseteq N_i \,:\, NG \cap V(B) = \{\sigma_B\},\, t' \leq t_B} \{f(NG, t', \sigma_B)\}.$$

Case 2: $\tilde{\boldsymbol{g}} \neq \mathbf{0}$, $\tilde{lb}(B)$ is computed extending the *ng*-route relaxation to consider the dual variables $\tilde{\boldsymbol{g}}$. We define $N_i = \bigcup_{C \in \mathscr{C} \,:\, i \in C} C$, $i \in V$, and, for all $i \in V$ such that $|N_i| < \Delta(N_i)$, we add to N_i the $\Delta(N_i) - |N_i|$ nearest customers to i according to $\tilde{d}_{ij}$ not in N_i. We derive an expanded state-space graph $\widetilde{\mathscr{G}}_F = (\widetilde{\mathscr{F}}, \widetilde{\mathscr{A}_F})$ from $\widehat{\mathscr{G}}_F$ by partitioning all paths represented by state $(NG, t, i) \in \widehat{\mathscr{F}}$ in $|\Gamma_i^{-1}|$ partitions, where each partition is identified by the last vertex j visited before i and is represented by a state $(NG, t, j, i) \in \widetilde{\mathscr{F}}$. The sets $\widetilde{\mathscr{F}}$ and $\widetilde{\mathscr{A}_F}$ are defined as follows

$$\widetilde{\mathscr{F}} = \{(NG, t, j, i) \,:\, i \in V,\, j \in \Gamma_i^{-1},\, t \in [e_i, l_i],\, \forall NG \subseteq N_i \text{ s.t. } NG \ni i\},$$

$$\widetilde{\mathscr{A}_F} = \{((NG', t', k, j), (NG, t, j, i)) \,:\, (NG', t', k, j), (NG, t, j, i) \in \widetilde{\mathscr{F}},$$
$$j \in \Gamma_i^{-1},\, k \in \Gamma_j^{-1},\, t' \in \Omega(t, j, i),\, \forall NG' \subseteq N_j \text{ s.t. } NG' \ni j \text{ and } NG' \cap N_i = NG \setminus \{i\}\},$$

where $\Omega(t, j, i)$ is the set of departure times from vertex j to arrive at vertex i at time t when j is visited immediately before i and is defined as

$$\Omega(t, j, i) = \begin{cases} t' \,:\, t' \in [e_j, \min\{l_j, t - t_{ji}\}]\}, & \text{if } t = e_i, \\ t - t_{ji} \,:\, t - t_{ji} \in [e_j, l_j], & \text{if } t \in (e_i, l_i]. \end{cases}$$

Let $\mathscr{C}_{ij} = \{C \in \mathscr{C} \,:\, (i, j) \in A(C)\}$. A possible DP recursion for computing the cost $f(NG, t, j, i)$ of a least cost *ng*-path $(NG, t, j, i) \in \widetilde{\mathscr{F}}$ is

$$f(NG, t, j, i) = \min_{\substack{(NG', t', k, j) \,:\\ ((NG', t', k, j), (NG, t, j, i)) \in \widetilde{\mathscr{A}_F}}} \{f(NG', t', k, j) + \tilde{d}_{ji} - \sum_{C \in \mathscr{C}_{ji} \setminus \mathscr{C}_{kj}} \tilde{g}_C\}.$$

The following lemma gives a method for computing $\tilde{lb}(B)$.

Lemma 2. Let $i = \sigma_B$ and $k = \pi_B$. Lower bound $\tilde{lb}(B)$ can be computed as follows

$$\tilde{lb}(B) = \tilde{c}(B) - \sum_{\substack{C \in \mathscr{C}: \\ |A(C) \cap A(B)| \geq 1}} \tilde{g}_C + \min_{\substack{NG \subseteq N_i : NG \cap V(B) = \{i\}, \\ t' \leq t(B), j \in \Gamma_i^{-1}}} \{f(NG, t', j, i) + \tilde{g}_{\{jik\}}\}, \tag{4.18}$$

where $\tilde{g}_{\{jik\}}$ is the dual of inequality (4.6) corresponding to $C = \{j, i, k\}$ ($\tilde{g}_{\{jik\}} = 0$ if $\{j, i, k\} \notin \mathscr{C}$).

Proof. Let R be the route of minimum reduced cost with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$ containing the backward path B, and let F be the forward path producing R once combined with B. Let $\mathscr{W}(F)$, $\mathscr{W}(B)$ and $\mathscr{W}(R)$ be the subsets of the WSR3s of $\mathscr{C}$ covered by F, B and R, respectively. Notice that $\mathscr{W}(F) \cap \mathscr{W}(B)$ is either empty or contains the WSR3 inequality $C^* \in \mathscr{C}$ such that $A(C^*)$ contains both the first arc (i, k) of B and the terminal arc $(\pi(P), i)$ of F, that is $C^* = \{\pi(P), i, k\}$. If $\{\pi(P), i, k\} \in \mathscr{C}$, let $\tilde{g}_{\{\pi(P),i,k\}}$ be the dual of inequality (4.6) corresponding to $\{\pi(P), i, k\}$; otherwise ($\{\pi(P), i, k\} \notin \mathscr{C}$), set $\tilde{g}_{\{\pi(P),i,k\}} = 0$. Because

$$\tilde{c}(R) = \tilde{c}(F) + \tilde{c}(B) - \sum_{C \in \mathscr{W}(R)} \tilde{g}_C$$

and

$$\sum_{C \in \mathscr{W}(R)} \tilde{g}_C = \sum_{C \in \mathscr{W}(F)} \tilde{g}_C + \sum_{C \in \mathscr{W}(B)} \tilde{g}_C - \tilde{g}_{\{\pi(F),i,k\}},$$

we have

$$\tilde{c}(R) = \tilde{c}(B) - \sum_{C \in \mathscr{W}(B)} \tilde{g}_C + \tilde{c}(F) - \sum_{C \in \mathscr{W}(F)} \tilde{g}_C + \tilde{g}_{\{\pi(F),i,k\}}. \tag{4.19}$$

Notice that

$$\min_{\substack{NG \subseteq N_i \text{ s.t. } NG \cap V(B) = \{i\}, \\ t' \leq t(B), \, j \in \Gamma_i^{-1}}} \{f(NG, t', j, i) + \tilde{g}_{\{j,i,k\}}\} \leq \tilde{c}(F) - \sum_{C \in \mathscr{W}(F)} \tilde{g}_C + \tilde{g}_{\{\pi(P),i,k\}}. \tag{4.20}$$

From (4.19) and (4.20), we derive that $\tilde{lb}(B)$ computed according to expression (4.18) provides a valid lower bound on $\tilde{c}(R)$. □

Similarly, `GenP` generates the path set $\mathscr{F}$ as a sequence of *undominated* paths (F^1, $\dots$, F^h), with $h \leq \Delta(\mathscr{F})$, such that $\tilde{lb}(F^1) \leq \dots \leq \tilde{lb}(F^h) \leq \gamma$, where $\Delta(\mathscr{F})$ is a parameter. Bound $\tilde{lb}(F)$ is computed similarly as described above for $\mathscr{B}$ using the reverse functions $f^{-1}(NG, t, i)$ and $f^{-1}(NG, t, j, i)$.

4.5.5.2 Procedure `Combine`.

This procedure combines the path sets $\mathscr{F}$ and $\mathscr{B}$ to derive at most Δ routes of reduced cost with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$ less than or equal to γ using the iterative method described by Baldacci et al. [2006].

Because of Lemma 1, routes of reduced cost less than or equal to γ can be generated combining only path pairs (F, B) such that $\tilde{c}(F) + \tilde{c}(B) \leq \gamma$. Procedure `Combine` dynamically generates a sequence of pairs (F^{r_1}, B^{s_1}), ..., (F^{r_k}, B^{s_k}), ..., (F^{r_h}, B^{s_h}) such that each pair satisfies conditions (4.12) and $\tilde{c}(F^{r_1}) + \tilde{c}(B^{s_1}) \leq \ldots \leq \tilde{c}(F^{r_k}) + \tilde{c}(B^{s_k}) \leq \ldots \leq \tilde{c}(F^{r_h}) + \tilde{c}(B^{s_h}) \leq \gamma$. The pool of routes generated by `Combine` contains any route R resulting from the pairs of paths in the sequence such that $\tilde{c}(R) \leq \gamma$ and R is *undominated* by any other route R' previously generated (i.e., R' *dominates* R if $V(R') = V(R)$ and $\tilde{c}(R') \leq \tilde{c}(R)$).

The procedure terminates as soon as Δ routes have been found or all pairs have been considered.

4.5.5.3 `GenR` for the CVRP.

Unlike the VRPTW, the path sets $\mathscr{F}$ and $\mathscr{B}$ are defined as in Baldacci et al. [2008]: $\mathscr{F}$ contains any forward path F such that $q_F \leq \frac{Q}{2} + q_{\sigma_F}$ and $\mathscr{B}$ contains any backward path B such that $q_B \leq \frac{Q}{2} + q_{\sigma_B}$ (for symmetric CVRPs $\mathscr{F}$ and $\mathscr{B}$ coincide).

Moreover, the lower bound $\tilde{lb}(B)$ is computed with bounding functions $f(NG, q, i)$, if $\tilde{\boldsymbol{g}} = \mathbf{0}$, or $f(NG, q, j, i)$, if $\tilde{\boldsymbol{g}} \neq \mathbf{0}$. Functions $f(NG, q, j, i)$ are derived by expanding $f(NG, q, i)$ so that j is the vertex preceding i similarly as described in §4.5.5.1 for functions $f(NG, t, j, i)$.

4.6 Column-and-Cut Generation Procedure H^4

Procedure H^4 is a column-and-cut generation procedure based on the simplex algorithm to solve relaxation LP. H^4 differs from the methods of Jepsen et al. [2008], Desaulniers et al. [2008], and Baldacci et al. [2008] for the pricing algorithm.

Before starting H^4 (see Step 2 of the exact algorithm in §4.4) an attempt is made to generate the set $\mathscr{R}^3$ of all routes such that $c^3_\ell \leq z_{UB} - LB_3$. The set $\mathscr{R}^3$ is generated by procedure `GenR` imposing that $|\mathscr{F}| \leq \Delta(\mathscr{F})$, $|\mathscr{B}| \leq \Delta(\mathscr{B})$, and that at most $\Delta(\mathscr{R})$ are generated. We call $\mathscr{F}^3$ the set $\mathscr{F}$, $\mathscr{B}^3$ the set $\mathscr{B}$, and $\mathscr{R}^3$ the set $\mathscr{R}$ generated by `GenR`. Define each set $\mathscr{F}^3$, $\mathscr{B}^3$, and $\mathscr{R}^3$ as *optimal* if $|\mathscr{F}^3| < \Delta(\mathscr{F})$, $|\mathscr{B}^3| < \Delta(\mathscr{B})$, and $|\mathscr{R}^3| < \Delta(\mathscr{R})$. The set $\mathscr{R}^3$ is optimal if and only if both $\mathscr{F}^3$ and $\mathscr{B}^3$ are optimal.

The route set $\mathscr{R}'$ of the initial master problem of H^4 is obtained by extracting the Δ^a routes of minimum reduced cost with respect to $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3})$ from $\mathscr{R}^3$ and by adding all single-customer routes. We initialize the set $\mathscr{C}$ of SR3s as $\mathscr{C} = \varnothing$ and use the same set $\mathscr{S}$ of SCs of H^3.

At each iteration, a set $\mathscr{N}$ of at most Δ^b negative reduced cost routes with respect to the current dual solution $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ is produced by procedure `GenR4` (see §4.6.1). The set $\mathscr{N}$ is either extracted from $\mathscr{R}^3$ or generated by combining sets $\mathscr{F}^3$ and $\mathscr{B}^3$. The sets $\mathscr{F}^3$ and $\mathscr{B}^3$ are newly generated when necessary if they are not-optimal.

At each iteration, H^4 adds the set $\mathscr{C}'$ of at most $\Delta(\mathscr{C})$ SR3s most violated by the current LP solution. H^4 ends if $\mathscr{N} = \varnothing$ and $\mathscr{C}' = \varnothing$ and achieves a D solution $(\boldsymbol{u^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$ of cost LB_4.

4.6.1 Procedure `GenR4`

Procedure `GenR4` is called at each iteration of H^4 to generate the route set $\mathscr{N}$ of at most Δ^b routes of negative reduced cost with respect to the current dual solution $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ of the master problem.

`GenR4` performs the following steps

Step 1. Extract from $\mathscr{R}^3$ a set of at most Δ^b routes of negative reduced cost and add them to $\mathscr{N}$. If $\mathscr{N} \neq \varnothing$ or $\mathscr{R}^3$ is optimal, stop.

Step 2. Call procedure `Combine4` that combines the sets $\mathscr{F}^3$ and $\mathscr{B}^3$ to derive the set $\mathscr{N}$. If $\mathscr{N} \neq \varnothing$ or both sets $\mathscr{F}^3$ and $\mathscr{B}^3$ are optimal, stop.

Step 3. If $\mathscr{F}^3$ is not-optimal, call procedure `GenP4` to generate a new path set $\mathscr{F}$ and set $\mathscr{F}^3 = \mathscr{F}$ ($\mathscr{F}^3$ is still not-optimal).

Step 4. Similarly to the previous step, if $\mathscr{B}^3$ is not-optimal, call procedure `GenP4` to generate a new path set $\mathscr{B}$ and set $\mathscr{B}^3 = \mathscr{B}$ ($\mathscr{B}^3$ is still not-optimal).

Step 5. Call `Combine4` that combines the sets $\mathscr{F}^3$ and $\mathscr{B}^3$ to derive the set $\mathscr{N}$.

If `GenP4` or `Combine4` run out of memory, H^4 and the exact method terminate prematurely.

4.6.2 Procedure `GenP4`

Procedure `GenP4` generates sets $\mathscr{F}$ and/or $\mathscr{B}$ in Steps 3 and 4 of `GenR4`. `GenP4` is similar to procedure `GenP` but applies different fathoming rules and an additional dominance rule introduced by Jepsen et al. [2008]. In generating $\mathscr{B}$, procedure `GenP4` applies the following rules.

Let $lb^3(B)$, $B \in \mathscr{B}$, be a lower bound to the reduced cost with respect to the dual solution $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3})$ found by H^3 of any route R containing path B. Lower bound $lb^3(B)$ corresponds to $\tilde{lb}(B)$ described in §4.5.5.1 when $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$ is replaced with $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3})$.

Fathoming 1. Any path $B \in \mathscr{B}$ such that $lb^3(B) > z_{UB} - LB_3$ can be fathomed because B cannot generate any route of any VRPTW solution of cost less than or equal to z_{UB}.

Let $\bar{lb}(B)$, $B \in \mathscr{B}$, be a lower bound to the reduced cost with respect to the dual solution $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ of any route containing path B.

Fathoming 2. Any path B such that $\bar{lb}(B) \geq 0$ is fathomed.

Lower bound $\bar{lb}(B)$ can be computed as described in the following lemma.

Lemma 3. Let functions $f(NG, t, i)$ be computed with the arc costs $\bar{d}_{ij} = d_{ij} - \frac{1}{2}(\bar{u}_i + \bar{u}_j) - \sum_{S \in \mathscr{S}_{ij}} \bar{v}_S$. Let $\bar{c}(B)$ be the cost of path B using arc costs $\bar{d}_{ij}$, and let $i = \sigma_B$. Lower bound $\bar{lb}(B)$ can be computed as follows

$$\bar{lb}(B) = \bar{c}(B) - \sum_{\substack{C \in \mathscr{C}: \\ |C \cap V(B) \setminus \{i\}| \geq 2}} \bar{g}_C + \min_{\substack{NG \subseteq N_i: \\ NG \cap V(B) = \{i\}, t' \leq t_B}} \{f(NG, t', i) - \sum_{\substack{C \in \mathscr{C}: \\ |C \cap NG| \geq 2}} \bar{g}_C\}. \tag{4.21}$$

Proof. Let R be the route of minimum reduced cost with respect to $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ containing the backward path B, and let F be the forward path producing R once combined with B. Let $\mathscr{C}(S) = \{C \in \mathscr{C} : |C \cap S| \geq 2\}$. We have

$$\bar{c}(R) = \bar{c}(B) - \sum_{C \in \mathscr{C}(V(B) \setminus \{i\})} \bar{g}_C + \bar{c}(F) - \sum_{C \in \mathscr{C}(V(F))} \bar{g}_C - \sum_{C \in \bar{\mathscr{C}}(V(R))} \bar{g}_C$$

where $\bar{\mathscr{C}}(V(R)) = \mathscr{C}(V(R)) \setminus (\mathscr{C}(V(B) \setminus \{i\}) \cup \mathscr{C}(V(F)))$. Because $\bar{\boldsymbol{g}} \leq \boldsymbol{0}$ and

$$\min_{\substack{NG \subseteq N_i : NG \cap V(B) = \{i\}, \\ t' \leq t_B}} \{f(NG, t', i) - \sum_{\substack{C \in \mathscr{C}: \\ |C \cap NG| \geq 2}} \bar{g}_C\} \leq \bar{c}(F) - \sum_{C \in \mathscr{C}(V(F))} \bar{g}_C,$$

we have

$$\bar{c}(R) \geq \bar{c}(B) - \sum_{C \in \mathscr{C}(V(B) \setminus \{i\})} \bar{g}_C + \min_{\substack{NG \subseteq N_i: \\ NG \cap V(B) = \{i\}, t' \leq t_B}} \{f(NG, t', i) - \sum_{\substack{C \in \mathscr{C}: \\ |C \cap NG| \geq 2}} \bar{g}_C\},$$

thus proving that expression (4.21) provides a valid lower bound $\bar{lb}(B)$ on $\bar{c}(R)$.□

The dominance rule of Jepsen et al. [2008] when applied to $\mathscr{B}$ is as follows.

Dominance 1. Let B, $B' \in \mathscr{B}$ be two backward paths such that $\sigma_{B'} = \sigma_B$, $t_{B'} \geq t_B$, $q_{B'} \geq q_{min}$, $q_B \geq q_{min}$, and $V(B') \subseteq V(B)$. Path B' dominates path B if

$$\bar{c}(B') \leq \bar{c}(B) - \sum_{C \in \mathscr{C}\,:\,|C \cap V(B) \setminus V(B')| \geq 2} \bar{g}_C.$$

Similar rules are applied by procedure `GenP4` to generate the path set $\mathscr{F}$.

4.6.3 Procedure `Combine4`

This procedure generates the route set $\mathscr{N}$ of negative reduced costs with respect to the dual solution $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ of the master problem combining the path sets $\mathscr{F}^3$ and $\mathscr{B}^3$ as defined in §4.6.1. Procedure `Combine4` corresponds to procedure `Combine` (see §4.5.5.2) replacing $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$ with $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$, setting $\gamma = -\varepsilon$, and applying the following fathoming rule to reduce the set $\mathscr{N}$.

Fathoming 3. Let $c^3(R)$ be the reduced cost of route R with respect to $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3})$ computed as $c^3(R) = c(R) - \sum_{i \in V(R)} u_i^3 - \sum_{S \in \mathscr{S}} \rho_\ell(S) v_S^3 - \sum_{C \in \mathscr{C}\,:\,|A(C) \cap A(R)| \geq 1} g_C^3$. A route R of negative reduced cost with respect to $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ cannot belong to $\mathscr{N}$ if $c^3(R) > z_{UB} - LB_3$ as route R cannot be in any solution of cost less than or equal to z_{UB}.

4.6.4 Procedure `GenR4` for the CVRP

The lower bound $lb^3(B)$ is computed using bounding functions $f(NG, q, j, i)$ as described in §4.5.5.3 for $\tilde{lb}(B)$ whereas the lower bound $\bar{lb}(B)$ is computed according to expression (4.21) but using bounding functions $f(NG, q, i)$ instead of $f(NG, t, i)$.

4.7 Procedure `GenRF` for Generating Route Set $\widetilde{\mathscr{R}}$

The exact method described in §4.4 at Step 4 asks to generate the largest subset $\widetilde{\mathscr{R}} \subseteq \mathscr{R}$ of routes satisfying conditions (4.8), whenever $\mathscr{R}^3$ is not-optimal.

For this purpose, we use a two-phase procedure, called `GenRF`, similar to `GenR` (see §4.5.5). In the first phase, two sets $\mathscr{F}$ and $\mathscr{B}$ of forward and backward paths are computed as described below. In the second phase, these sets are combined by procedure `CombineF` (see §4.7.2) to derive the final route set $\widetilde{\mathscr{R}}$. In the first phase, there are four cases

Case 1. Both $\mathscr{F}^3$ and $\mathscr{B}^3$ generated by `GenR` are optimal. We set $\mathscr{F} = \mathscr{F}^3$ and $\mathscr{B} = \mathscr{B}^3$.

Case 2. $\mathscr{B}^3$ is optimal but $\mathscr{F}^3$ is not. We set $\mathscr{B} = \mathscr{B}^3$ and call `GenPF` (§4.7.1) to compute $\mathscr{F}$.

Case 3. $\mathscr{F}^3$ is optimal but $\mathscr{B}^3$ is not. We set $\mathscr{F} = \mathscr{F}^3$ and call `GenPF` to compute $\mathscr{B}$.

Case 4. Both $\mathscr{F}^3$ and $\mathscr{B}^3$ are not optimal. We call `GenPF` to compute both $\mathscr{F}$ and $\mathscr{B}$.

In the end, $\widetilde{\mathscr{R}}$ is defined *optimal* (i.e., contains any optimal VRPTW solution) if and only if $\mathscr{F}$, $\mathscr{B}$, and $\widetilde{\mathscr{R}}$ are such that $|\mathscr{F}| < \Delta(\mathscr{F})$, $|\mathscr{B}| < \Delta(\mathscr{B})$ and $|\widetilde{\mathscr{R}}| < \Delta(\widetilde{\mathscr{R}})$.

In the following §§4.7.1 and 4.7.2, we describe procedure `GenPF` and `CombineF` for the VRPTW. It is quite obvious how to adapt them for the CVRP.

4.7.1 Procedure `GenPF`

Procedure `GenPF` generates one or both sets $\mathscr{F}$ and $\mathscr{B}$ required by `GenRF` (see cases (2), (3) and (4) in §4.7). `GenPF` is similar to procedure `GenP` but applies different fathoming rules.

In generating the set $\mathscr{B}$, `GenPF` applies the following fathoming rules. Let $lb^3(B)$ and $lb^4(B)$ be the lower bounds on the reduced costs, $c^3(R)$ and $c^4(R)$, of any route R containing path $B \in \mathscr{B}$ with respect to $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3})$ and $(\boldsymbol{u^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$, respectively. Bound $lb^3(B)$ is computed as described in §4.6.2, and $lb^4(B)$ is computed using expression (4.21) where $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ is replaced with $(\boldsymbol{u^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$.

Fathoming 4. Any path $B \in \mathscr{B}$ such that $lb^3(B) > z_{UB} - LB_3$ or $lb^4(B) > z_{UB} - LB_4$ can be fathomed.

Fathoming 5. Let $d^4_{ij} = d_{ij} - \frac{1}{2}(u^4_i + u^4_j) - \sum_{S \in \mathscr{S}_{ij}} v^4_s$, $(i,j) \in A$. Whenever $q_{min} = 0$, any backward path B such that

$$\sum_{(i,j)\in A(B)} d^4_{ij} - \sum_{C \in \mathscr{C}\,:\,|C \cap V(B) \setminus \{i\}| \geq 2} g^4_C - d^4_{i0} > z_{UB} - LB_4 \tag{4.22}$$

can be fathomed as it cannot produce any route R of reduced cost $c^4(R) \leq z_{UB} - LB_4$.

Proof. Let R be the route of minimum reduced cost $c^4(R)$ with respect to $(\boldsymbol{u^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$ obtained by combining B starting from $\sigma_B = i$ with a forward path F ending at vertex $\sigma_F = i$. Let $\mathscr{C}(S) = \{C \in \mathscr{C} : |C \cap S| \geq 2\}$. We have

$$c^4(R) = \sum_{(i,j)\in A(R)} d^4_{ij} - \sum_{C \in \mathscr{C}(V(R))} g^4_C.$$

Since $\mathscr{C}(V(R)) \supseteq \mathscr{C}(V(B) \setminus \{i\}) \cup \mathscr{C}(V(F))$ and $g^4_C \leq 0$, $C \in \mathscr{C}$, we have

$$c^4(R) \geq \sum_{(i,j)\in A(B)} d^4_{ij} - \sum_{C \in \mathscr{C}(V(B)\setminus\{i\})} g^4_C + \sum_{(i,j)\in A(F)} d^4_{ij} - \sum_{C \in \mathscr{C}(V(F))} g^4_C. \tag{4.23}$$

Consider the route R' obtained by adding arc $(i,0)$ to F. As $c^4(R') \geq 0$, we have

$$\sum_{(i,j)\in A(F)} d^4_{ij} - \sum_{C\in\mathscr{C}(V(F))} g^4_C \geq -d^4_{i0}. \tag{4.24}$$

From inequalities (4.23) and (4.24), we obtain

$$c^4(R) \geq \sum_{(i,j)\in A(B)} d^4_{ij} - \sum_{C\in\mathscr{C}(V(B)\setminus\{i\})} g^4_C - d^4_{i0}.$$

Thus, if B satisfies inequality (4.22), then $c^4(R) > z_{UB} - LB_4$. So route R cannot be in any optimal solution of cost less than or equal to z_{UB}. □

Similar fathoming rules are used to generate $\mathscr{F}$.

4.7.2 Procedure `CombineF`

This procedure generates the route set $\widetilde{\mathscr{R}}$ combining the path sets $\mathscr{F}$ and $\mathscr{B}$ defined in §4.7. `CombineF` is similar to `Combine` (see §4.5.5.2) but imposes that any route $R \in \widetilde{\mathscr{R}}$ is such that $c^3(R) \leq z_{UB} - LB_3$ and $c^4(R) \leq z_{UB} - LB_4$.

4.8 Computational Results

This section reports on the computational results of the exact method (hereafter called BMR) described in this chapter. All algorithms were coded in Fortran 77 and compiled with Intel Fortran 11.0. Cplex 12.1 was used as the LP solver in procedure H^4 and the IP solver in the exact method. All tests were run on an IBM Intel Xeon X7350 Server (2.93 GHz - 16 GB of RAM).

4.8.1 Computational Results on the VRPTW

Our exact method BMR for the VRPTW was tested on Solomon instances Solomon [1987], which are divided into six classes (classes C1, RC1 and R1 with tight time windows and strict vehicle capacity, and classes C2, RC2 and R2 with wide time windows and loose vehicle capacity). We considered all 100-customer instances and instances with 50 customers of classes C2, RC2 and R2.

The travel costs d_{ij} are computed as $d_{ij} = \lfloor 10e_{ij} \rfloor / 10$, where e_{ij} is the Euclidean distance between vertices i and j; the travel times t_{ij} are integer values computed as $t_{ij} = 10(d_{ij} + s_i)$, where s_i is the service time at vertex i.

Because instances of classes C1, RC1 and R1 have tight time windows, we did not find it worth running procedure H^2. Thus, on such instances H^2 was skipped. In addition, we ignored SCs and WSR3s for all classes of instances.

BMR uses the best upper bounds reported in Ropke [2005] and Danna and Pape [2005]. Such upper bounds are obtained by running the heuristic of Pisinger and Ropke [2007] with different parameter settings (Ropke [2010]). Whenever BMR uses the upper bound, its computing time is added to the total computing time of BMR. For instance R211 with 100 customers, the upper bound used was found by Desaulniers et al. [2008].

BMR uses the following parameter setting

- in H^1: $Maxit1 = 100$ and $Maxit2 = 50$;
- in H^2: $\Delta(N_i) = 8$, $Maxit1 = 100$, and $Maxit2 = 50$;
- in H^3: $\Delta(N_i) = 10$, $\Delta^b = 300$, $\Delta(\mathscr{F}) = \Delta(\mathscr{B}) = 5 \times 10^7$, $\Delta(\mathscr{R}) = 1.5 \times 10^6$, $Maxit1 = 100$, and $Maxit2 = 50$;
- in H^4, $\Delta^a = 1 \times 10^4$, $\Delta^b = 300$, $\Delta(\mathscr{F}) = \Delta(\mathscr{B}) = 5 \times 10^7$, $\Delta(\mathscr{R}) = 1.5 \times 10^6$, $\Delta(\mathscr{C}) = 20$.

We compare BMR with the methods of Jepsen et al. [2008] and Desaulniers et al. [2008], hereafter called JPSP and DHL, respectively. Desaulniers et al. [2008] presented three versions of their algorithm. We consider the version labeled "ESPPRC SRC" as it could solve more instances than the others. According to SPEC (`http://www.spec.org/benchmarks.html`), our machine is three times faster than the Intel Pentium 4 3.0 GHz PC of JPSP and twice as fast as the Linux PC Dual Core AMD Opteron at 2.6 GHz of DHL.

In Tables 4.1, 4.2, and 4.3, we report on detailed computational results on the Solomon instance considered. The columns of three tables report the instance name ($Inst$), the optimal value (z^* - in bold if solved for the first time by BMR), the upper bound used (z_{UB}) and the time to compute it (T) in columns under heading *Upper Bound*. For each bounding procedure H^k, $k = 1, \ldots, 4$ (if run), we report the lower bound (LB_k) and the cumulative computing time spent up to H^k. Columns $|\mathscr{F}^3|$, $|\mathscr{B}^3|$, and $|\mathscr{R}^3|$ report the cardinalities (in thousands) of the sets $\mathscr{F}^3$, $\mathscr{B}^3$ and $\mathscr{R}^3$; if we could not completely generate a set, an empty circle is displayed. The number of SR3s inequalities ($SR3$) added in H^4 is shown. The number of routes ($|\widetilde{\mathscr{R}}|$) in the final reduced problem $\widetilde{P}$ and the time taken by Cplex to solve it (T_{cpx}) are shown. Finally, the total computing time in seconds (T_{tot}) of the methods compared are reported in the last three columns of the table. T_{tot} under BMR is equal to the sum of the time to compute the upper bound, the time spent up to H^4 and T_{cpx}.

Table 4.3 shows that BMR was able to solve four instances open so far. The only open Solomon instance is R.208.100, where BMR ran out of memory. Notice that the lower bounds achieved using the *ng*-routes in algorithm *CCG* are close to the bounds achieved using elementary routes and the time taken to perform H^2 is limited.

Table 4.4 compares BMR, JPSP, and DHL. For each class, Table 4.4 reports the class name (*Class*), the number of customers (n), the number of instances (NP), the number of instances solved by each of the three methods (*Solved*) and the average computing time in seconds (T) (*n.a.* means data are not available). In the last three rows, the average computing time of the methods over all instances, the instances solved by JPSP, and the instances solved by DHL are shown.

Table 4.4 shows that BMR outperforms JPSP and DHL: all instances solved by the other methods were solved by BMR and the average time is significantly lower.

Table 4.5 reports the optimal solutions BMR found for the four 100-customer instances (RC204, RC208, R204, and R211) open before this paper. We give the optimal solution value, the number of vehicles used, and, for each route, the cost and the sequence of visited customers.

We also report a computational analysis about some components of the algorithm proposed (i.e., parameter $\Delta(N_i)$ and dominance and fathoming rules). Table 4.6 reports the results obtained on a selected set of instances by varying parameter $\Delta(N_i)$ in procedure H^2. In particular, the table shows the value of lower bound LB_2 using $\Delta(N_i) = 5, 8, 10, 12$, and the corresponding computing time, compared with LB_1, LB_3, and LB_4. The last line of the table reports the average percentage deviations of the different lower bounds. The table shows that using $\Delta(N_i) = 8$ gives a good trade-off between quality of the lower bound and computing time. Indeed, the lower bound is on average about 5 percent greater than LB_1 and about 0.5 percent lower than LB_3.

Tables 4.7 and 4.8 report statistics on the number of states fathomed by the dominance and fathoming rules applied in procedures `GenP4` (called by procedure H^4) and `GenPF` (called to generate the final route sets $\widetilde{\mathscr{R}}$). The tables show the following columns: total number of forward (backwards) states in millions generated in computing $\mathscr{F}$ ($\mathscr{B}$) (*States*), percentage of states eliminated by Dominance 1 (%*Dom*1), percentage of states fathomed by Fathoming rule x (%*Fath* x, $x = 1, 2, 4, 5$), final cardinality (in millions) of the set $\mathscr{F}$ ($\mathscr{B}$) generated ($|\mathscr{F}|$, $|\mathscr{B}|$). The dominance and fathoming rules are reported in the tables using the same order of application in BMR.

Regarding procedure `GenP4`, table 4.7 shows that the dominance and fathoming rules are very effective in reducing the number of states. Indeed, on average about 90% of the states generated (for both sets $\mathscr{F}$ and $\mathscr{B}$) are eliminated. In particular, the new Fathoming 1 and Fathoming 2 rules eliminates on average 80% of the states not dominated by Dominance 1.

Concerning procedure `GenPF`, table 4.8 shows that both Fathoming 5 and 4 eliminate on average about 90% of the states generated.

4.8.2 Computational Results on the CVRP

We considered classes A, B, E, F, M, and P available at `http://branchandcut.org/VRP/data`. Cost d_{ij} is an integer value computed as $d_{ij} = \lfloor e_{ij} + 0.5 \rfloor$, where e_{ij} is the Euclidean distance between i and j. As done by Fukasawa et al. [2006], we impose that exactly m vehicles are used in the solution by simply transforming constraint (4.3) into an equality constraint.

For computational convenience we skip bounding procedure H^2, so the sequence of the bounding procedures is H^1, H^3, and H^4. WSR3s are used whenever $\lceil n/m \rceil \geq 12$.

If BMR cannot solve a problem in Step 1, we use the upper bound used by Fukasawa et al. [2006] and Baldacci et al. [2008], but, while generating the route set $\mathscr{R}^3$, whenever in `GenP` $|\mathscr{F}^3| > 1 \times 10^6$, we run our implementation of the tabu search algorithm of Gendreau et al. [1994] with a time limit of 180 seconds. The computing time of the Tabu Search is considered in the total computing time of BMR whereas the computing time of the initial upper bound is ignored (as done by Fukasawa et al. [2006] and Baldacci et al. [2008]).

In our tests, we use the following parameter setting

- in H^1 and H^2: $Maxit1 = 150$ and $Maxit2 = 100$;
- in H^3: $\Delta(N_i) = 10$, $\Delta^b = 300$, $\Delta(\mathscr{F}) = 1 \times 10^7$, $\Delta(\mathscr{R}) = 1 \times 10^7$, $\Delta(\mathscr{C}) = 100$, $Maxit1 = 150$, and $Maxit2 = 100$;
- in H^4: $\Delta^a = 1 \times 10^4$, $\Delta^b = 200$, $\Delta(\mathscr{F}) = 1 \times 10^7$, $\Delta(\mathscr{R}) = 1 \times 10^6$, $\Delta(\mathscr{C}) = 10$, $Maxit1 = 150$, and $Maxit2 = 100$.

BMR is compared with the methods of Baldacci et al. [2008] (BCM), Fukasawa et al. [2006] (FLL) and Lysgaard et al. [2004] (LLE). According to SPEC, our machine is three times faster than the Pentium 4 2.6 GHz PC of BCM and the Pentium 4 2.4 GHZ PC of FLL and ten times faster than the Intel Celeron 700 MHz PC of LLE.

Tables 4.9, 4.10, 4.11, 4.12, and 4.13 show the same columns of Tables 4.1, 4.2, and 4.3 reported for the VRPTW. Column z_{UB} reports the initial upper bound whereas column UB_4 reports the upper bound computed by our tabu search heuristic (when run). Under FLL, column (s) indicates that the BCP algorithm based on s-cycle-free q-routes was used, whereas $(-)$ indicates that the BC was used instead of the BCP. The total time of BMR is the sum of the time spent up to H^4 (that includes the time spent to compute UB_4) and T_{cpx}, but, if H^k, for some $k \in \{1, 2, 3\}$ succeeds in solving the problem to optimality, the time is equal to the time spent up to H^k.

Tables 4.9-4.13 show that BMR solves three problems not solved by BCM but does not solve problem F-n135-k7 that is solved by FLL using the BC algorithm. It is worth mentioning that problem F-n135-k7 was solved for the first time by Augerat [1995].

Table 4.14 summarizes the results of the methods on the six CVRP classes considered. For each class, the name ($Class$), the number of instances (NP) and, for each exact method, the number of instances solved to optimality (Opt), the average percentage deviation of the lower bound ($\%LB$) and the average computing time in seconds (T) are reported. Under FLL, column Opt_{BCP} and Opt_{BC} report the number of instances solved using BCP and BC, respectively. The last two rows indicate average values of $\%LB$ and T and the number of problems solved by each method.

Table 4.14 clearly shows that BMR outperforms the other methods on all classes but class F, where the BC algorithm of Lysgaard et al. [2004] outperforms the other methods.

Table 4.15 reports the results obtained by varying parameter $\Delta(N_i)$ in procedure H^2 and by using different types of inequalities in procedures H^1 and H^3 on a set of selected instances. For procedure H^1, the value of lower bound LB_1 and the corresponding computing time without (no CCs) and with (CCs) CCs are shown. The table shows the value of lower bound LB_2 using $\Delta(N_i) = 5, 8, 10, 12$, and the corresponding computing time. For procedure H^3, we show the value of lower bound LB_3 and the corresponding computing time without (no WSR3s) and with (WSR3s) WSR3s. The last line of the last reports the average percentage deviations of the different lower bounds.

The results show that: (i) increasing parameter $\Delta(N_i)$ slightly improves lower bound LB_2; (ii) LB_1 with CCs is very close to LB_3 without WSR3s; (iii) the increase of LB_2 with respect to LB_1 achieved by H^1 with CCs is very small and is not worth the extra computing time required by procedure H^2; (iv) CCs and WSR3s substantially increase lower bounds LB_1 and LB_3, respectively. It is worth mentioning that on instance E-n101-k14 lower bound LB_1 with CCs is greater than lower bound LB_2. This happens as 2-vertex loops are allowed in the ng-route relaxation while they are forbidden in the q-route relaxation.

4.9 Conclusions

In this chapter, we described an exact method for solving the VRPTW and CVRP based on the set partitioning formulation strengthened with valid inequalities. We introduced a new route relaxation, called ng-route, that improves other nonelementary route relaxations proposed in the literature and a new strategy for solving the pricing problem in a column-and-cut generation procedure that involve the use of multiple dual solutions.

We reported computational results showing that the proposed method solves 4 of the 5 open Solomon VRPTW instances and is significantly faster than the state-of-the-art algorithms for both VRPTW and CVRP.

Table 4.1: Detailed results on VRPTW Solomon instances with 50 customers and wide time windows

		Upper Bound		*Proc.* H^1		*Proc.* H^2		*Proc.* H^3					*Proc.* H^4			*Problem* $\widetilde{P}$		T_{tot}	
Inst	z^*	z_{UB}	T	LB_1	T	LB_2	T	LB_3	T	$\lvert\mathscr{F}^3\rvert$	$\lvert\mathscr{B}^3\rvert$	$\lvert\mathscr{R}^3\rvert$	$SR3$	LB_4	T	$\lvert\widetilde{\mathscr{R}}\rvert$	T_{cpx}	BMR	JPSP
C.201.50	360.2	360.2	83	360.2	3	360.2	4											4	42
C.202.50	360.2	360.2	89	360.2	3	360.2	6											6	67
C.203.50	359.8	359.8	89	359.8	3	359.8	7											7	214
C.204.50	350.1	350.1	96	350.1	3	350.1	8											8	-
C.205.50	359.8	359.8	99	358.2	4	359.8	7											7	64
C.206.50	359.8	359.8	86	344.2	4	359.8	10											10	38
C.207.50	359.6	359.6	89	356.6	4	359.6	9											9	72
C.208.50	350.5	350.5	92	340.4	4	350.5	9											9	55
RC.201.50	684.8	684.8	40	670.1	3	684.8	8											8	3
RC.202.50	613.6	613.6	40	504.0	3	613.5	6	613.6	7									7	10
RC.203.50	555.3	555.3	50	409.2	3	555.3	15											15	190
RC.204.50	444.2	444.2	63	314.4	3	444.2	27											27	-
RC.205.50	630.2	630.2	40	541.4	3	630.2	9											9	5
RC.206.50	610.0	610.0	43	441.1	3	610.0	9											9	8
RC.207.50	558.6	558.6	53	390.7	3	558.2	7	558.6	8									8	21
RC.208.50	476.7	481.8	79	316.2	4	468.5	8	472.3	22	7	22	22	10	476.7	50			129	1,639
R.201.50	791.9	791.9	43	788.4	3	791.7	4	791.9	5									5	4
R.202.50	698.5	698.5	46	692.7	3	698.5	8											8	9
R.203.50	605.3	605.9	50	590.5	3	598.2	6	598.5	7	12	8	8	35	605.3	12			62	50
R.204.50	506.4	506.4	79	474.0	5	499.8	23	502.2	36	326	232	232	35	506.4	52			131	-
R.205.50	690.1	696.7	50	666.2	3	681.2	5	682.3	7	49	28	28	45	690.1	12			62	15
R.206.50	632.4	632.4	53	609.2	3	622.8	9	624.8	12	63	15	15	85	632.4	21			74	190
R.207.50	575.5	575.5	53	482.2	3	561.4	11	564.1	15	786	136	o	70	575.5	101			154	34,406
R.208.50	487.7	487.7	96	461.4	6	476.3	28	481.3	94	7,385	3,622	1,282	55	487.7	441			537	-
R.209.50	600.6	600.6	50	582.1	3	598.5	6	599.2	8	1	1	1	5	600.6	10			60	16
R.210.50	645.6	645.6	53	623.6	4	633.9	8	635.7	11	103	34	34	145	645.3	40	128	1	94	18,545
R.211.50	535.5	543.3	83	507.6	4	526.7	8	528.3	11	507	443	443	90	535.5	96			179	10,543
Avg				91.1		99.4		99.6						100.0				61	2,879
Tot	27																	27	23

TABLE 4.2: Detailed results on VRPTW Solomon instances with 100 customers and tight time windows

		Upper Bound		*Proc. H^1*		*Proc. H^3*						*Proc. H^4*		*Problem $\widetilde{P}$*		T_{tot}		
Inst	z^*	z_{UB}	T	LB_1	T	LB_3	T	$\lvert\mathscr{F}^3\rvert$	$\lvert\mathscr{B}^3\rvert$	$\lvert\mathscr{R}^3\rvert$	$SR3$	LB_4	T	$\lvert\widetilde{\mathscr{R}}\rvert$	T_{cpx}	BMR	JPSP	DHL
C.101.100	827.3	827.3	96	827.3	7											7	3	2
C.102.100	827.3	827.3	106	820.3	8	827.3	8									8	12	8
C.103.100	826.3	826.3	112	809.3	9	826.3	11									11	33	28
C.104.100	822.9	822.9	119	795.6	10	822.7	31	10	15	10		822.9	39		1	159	4,113	86
C.105.100	827.3	827.3	99	821.2	7	827.3	8									8	5	3
C.106.100	827.3	827.3	102	817.7	8	827.3	10									10	7	4
C.107.100	827.3	827.3	102	818.9	7	827.3	8									8	6	4
C.108.100	827.3	827.3	106	818.9	7	827.3	8									8	14	7
C.109.100	827.3	827.3	112	803.2	8	827.3	10									10	20	16
RC.101.100	1,619.8	1,619.8	92	1,576.2	5	1,581.1	5	24	31	298	95	1,619.8	13			105	12	19
RC.102.100	1,457.4	1,463.5	99	1,398.6	5	1,405.5	8	428	432	o	150	1,457.4	39			138	76	120
RC.103.100	1,258.0	1,267.0	102	1,214.7	5	1,224.5	8	811	794	o	342	1,257.6	237	118,992	96	435	2,705	541
RC.104.100	1,132.3	1,132.6	109	1,070.3	6	1,100.4	35	1,610	1,371	o	255	1,129.8	442	5,029	10	561	65,806	11,773
RC.105.100	1,513.7	1,513.8	99	1,467.6	5	1,471.7	7	81	116	o	60	1,513.7	17			116	26	33
RC.106.100	1,372.7	1,373.9	96	1,301.2	5	1,318.5	8	293	392	o	428	1,367.2	267	40,162	50	413	15,891	3,916
RC.107.100	1,207.8	1,209.3	99	1,156.4	6	1,182.7	8	136	195	o	105	1,207.8	28			127	153	161
RC.108.100	1,114.2	1,114.2	102	1,049.7	6	1,073.0	11	1,460	1,529	o	185	1,114.2	213			315	3,365	635
R.101.100	1,637.7	1,637.7	99	1,628.5	5	1,630.4	6	1	1	5	10	1,634.0	9	46		108	1	8
R.102.100	1,466.6	1,467.6	109	1,462.0	5	1,466.2	7	1	1	1		1,466.6	10			119	4	3
R.103.100	1,208.7	1,208.7	112	1,201.7	5	1,203.2	6	4	8	24	30	1,208.7	11			123	23	20
R.104.100	971.5	976.0	112	947.8	6	955.7	12	1,124	1,060	o	449	971.3	224	119,830	7	343	32,343	3,103
R.105.100	1,355.3	1,355.3	102	1,343.8	5	1,345.0	6	6	8	47	103	1,355.1	11	80		113	43	36
R.106.100	1,234.6	1,234.6	109	1,224.3	5	1,225.7	7	24	40	237	160	1,234.6	20			129	75	87
R.107.100	1,064.6	1,064.6	109	1,049.9	5	1,052.3	7	76	163	703	300	1,064.3	84	193		193	1,310	416
R.108.100	932.1	933.7	119	905.7	6	913.0	10	1,882	2,066	o	280	932.1	225			344	5,911	891
R.109.100	1,146.9	1,146.9	102	1,129.3	5	1,133.5	9	30	52	329	233	1,144.1	53	4,953	1	156	1,432	1,127
R.110.100	1,068.0	1,075.9	109	1,047.8	6	1,054.9	9	384	401	o	398	1,068.0	143	225,584	7	259	1,068	426
R.111.100	1,048.7	1,048.7	109	1,030.5	6	1,033.9	10	127	304	o	281	1,045.8	106	10,620	40	255	83,931	5,738
R.112.100	948.6	948.6	116	916.6	6	926.2	11	2,115	4,311	o	539	946.6	721	6,214	28	865	202,803	16,073
Avg				97.7		98.7						99.9				188	14,524	1,562
Tot	29															29	29	29

Table 4.3: Detailed results on VRPTW Solomon instances with 100 customers and wide time windows

		Upper Bound		*Proc.* H^1		*Proc.* H^2		*Proc.* H^3					*Proc.* H^4			*Problem* $\widetilde{P}$		T_{tot}		
Inst	z^*	z_{UB}	T	LB_1	T	LB_2	T	LB_3	T	$\lvert\mathscr{F}^3\rvert$	$\lvert\mathscr{B}^3\rvert$	$\lvert\mathscr{R}^3\rvert$	$SR3$	LB_4	T	$\lvert\widetilde{\mathscr{R}}\rvert$	T_{cpx}	BMR	JPSP	DHL
C.201.100	589.1	589.1	228	589.1	6													6	203	9
C.202.100	589.1	589.1	244	589.1	14													14	3,483	49
C.203.100	588.7	588.7	264	588.6	14	588.7	32											32	13,070	122
C.204.100	588.1	588.1	277	577.5	15	587.5	170	588.1	182									182	-	16,416
C.205.100	586.4	586.4	251	586.4	12													12	416	15
C.206.100	586.0	586.0	238	576.8	13	586.0	28											28	594	24
C.207.100	585.8	585.8	244	581.8	13	585.8	22											22	1,241	84
C.208.100	585.8	585.8	244	581.3	13	585.8	22											22	555	26
RC.201.100	1,261.8	1,262.6	139	1,240.0	7	1,254.4	10	1,255.4	12	6	7	34	55	1,261.7	18	585		157	229	92
RC.202.100	1,092.3	1,095.8	152	1,003.8	7	1,084.7	11	1,086.2	13	91	71	655	45	1,092.3	31			183	312	89
RC.203.100	923.7	923.7	185	814.3	7	916.0	16	919.5	22	119	29	269	30	923.7	67			252	14,917	324
RC.204.100	**783.5**	783.5	150	687.4	20	771.1	235	778.4	455	1,375	339	o	50	783.5	902			1,052	-	-
RC.205.100	1,154.0	1,154.0	149	1,054.0	7	1,144.0	12	1,145.8	14	25	22	117	30	1,154.0	23			172	221	111
RC.206.100	1,051.1	1,051.1	172	951.5	8	1,034.4	13	1,037.7	16	276	177	o	40	1,051.1	55			227	339	344
RC.207.100	962.9	966.6	182	865.7	9	940.6	25	945.8	62	o	o	o	490	962.9	26,721			26,903	-	91,405
RC.208.100	**776.1**	777.3	215	703.9	20	761.3	114	765.8	168	3,909	3,306	o	110	776.1	980	6,959		1,195	-	-
R.201.100	1,143.2	1,148.5	149	1,135.2	8	1,140.0	15	1,140.3	17	62	44	o	100	1,143.2	31			180	139	78
R.202.100	1,029.6	1,036.9	178	1,008.6	8	1,020.6	19	1,021.2	22	o	o	o	115	1,027.3	1,890	575,375	1,338	3,406	8,282	1,663
R.203.100	870.8	872.4	198	845.7	9	862.8	91	865.8	165	o	564	o	100	870.8	1,941			2,139	54,187	641
R.204.100	**731.3**	731.3	221	688.8	22	721.3	64	724.2	194	o	o	o	130	731.3	216,146			216,367	-	-
R.205.100	949.8	949.8	191	916.0	11	934.1	27	938.0	31	3,779	920	o	553	948.3	1,240	19,034	2	1,433	-	6,904
R.206.100	875.9	880.6	201	834.0	13	860.2	34	866.3	76	o	o	o	145	875.9	6,474			6,675	-	60,608
R.207.100	794.0	794.0	238	746.9	16	781.6	120	789.9	368	o	o	o	35	794.0	1,163			1,401	-	11,228
R.208.100	701.2	701.2	284	660.9	21	686.4	156	690.3	2,405	o	o	o		m.o.				-	-	-
R.209.100	854.8	855.7	136	818.7	15	838.0	28	840.6	59	o	o	o	180	854.3	4,355	120,577	5	4,496	78,560	22,514
R.210.100	900.5	900.8	136	848.9	17	883.8	30	888.2	57	o	o	o	523	900.4	39,572	53,704	3	39,711	-	400,904
R.211.100	**746.7**	751.7	168	704.9	21	729.3	133	734.1	219	o	o	o	140	746.7	10,825			10,993	-	-
Avg				95.4		99.0		99.3						99.9				12,202	11,047	27,893
Tot	27																	26	16	22

m.o.: `GenP4` runs out of memory

Table 4.4: VRPTW Solomon instances: summary

			Solved			*T*		
Class	*n*	*NP*	BMR	JPSP	DHL	BMR	JPSP	DHL
C2	50	8	8	7	n.a.	8	79	n.a.
RC2	50	8	8	7	n.a.	27	268	n.a.
R2	50	11	11	9	n.a.	124	7,086	n.a.
C1	100	9	9	9	9	25	468	18
RC1	100	8	8	8	8	276	11,004	2,150
R1	100	12	12	12	12	251	27,412	2,327
C2	100	8	8	7	8	40	2,795	2,093
RC2	100	8	8	5	6	3,767	3,204	15,394
R2	100	11	10	4	8	28,680	35,292	63,068
Avg						3,955	9,767	12,920
Solved by JPSP						261	9,767	
Solved by DHL						1,825		12,920

Table 4.5: Optimal solutions of 100-customer VRPTW instances solved by BMR for the first time

Cost	Route
	Solution of RC204. Total cost = 783.5. Number of vehicles = 4.
174.6	80 92 95 62 50 34 31 29 27 26 28 30 32 33 89 76 63 85 51 84 56 91
139.5	81 54 41 39 42 44 43 40 36 35 37 38 72 71 93 67 94 96
221.4	69 98 12 14 47 17 16 15 11 10 9 87 13 86 74 59 97 75 58 77 25 24 22 83 65 90
248.0	66 64 20 49 19 18 48 21 23 57 52 99 82 53 60 78 73 79 7 8 46 45 5 3 1 4 6 2 88 55 100 70 61 68
	Solution of RC208. Total cost = 776.1. Number of vehicles = 4.
132.5	61 42 44 39 38 36 35 37 40 43 41 72 71 93 96 54 81
218.7	90 65 82 99 52 83 64 49 19 18 48 21 23 25 77 58 75 97 59 87 74 86 57 24 22 20 66
226.6	69 98 88 2 6 7 79 73 78 12 14 47 17 16 15 13 9 11 10 53 60 8 46 4 45 5 3 1 70 100 55 68
198.3	94 92 95 67 62 50 34 31 29 27 26 28 30 32 33 76 89 63 85 51 84 56 91 80
	Solution of R204. Total cost = 731.3. Number of vehicles = 5.
8.8	53
211.6	27 69 1 50 76 3 79 33 81 9 51 20 66 65 71 35 34 78 29 24 55 25 54 80 68 77 12 26 28
150.5	2 57 42 43 15 41 22 75 56 23 67 39 4 72 74 73 21 40 58
136.2	6 94 95 92 98 85 91 44 14 38 86 16 61 93 99 96 59 97 87 37 100 13
224.2	52 31 88 7 82 48 19 11 62 10 70 30 32 90 63 64 49 36 47 46 8 45 17 84 5 60 83 18 89
	Solution of R211. Total cost = 746.7. Number of vehicles = 4.
173.1	53 40 21 73 41 22 75 23 67 39 25 55 24 29 68 80 54 4 56 74 72 26
214.0	27 69 31 88 7 82 46 47 36 49 64 63 90 32 66 71 65 35 34 78 79 3 77 50 1
194.4	52 18 83 5 99 95 92 98 85 61 16 86 38 44 14 42 43 15 57 2 87 97 37 100 91 93 59 96 94 13 58
165.2	28 12 76 33 81 9 51 20 30 70 10 62 11 19 48 8 45 17 84 60 6 89

TABLE 4.6: VRPTW: impact of parameter $\Delta(N_i)$ in procedure H^2

		H^1		H^2								H^3		H^4	
				$\Delta(N_i) = 5$		$\Delta(N_i) = 8$		$\Delta(N_i) = 10$		$\Delta(N_i) = 12$					
Inst	z*	LB_1	T	LB_2	T	LB_2	T	LB_2	T	LB_2	T	LB_3	T	LB_4	T
R.207.50	575.5	482.2	3	558.5	7	561.4	11	561.6	13	562.4	16	564.1	15	575.5	101
R.208.50	487.7	461.4	6	475.1	22	476.3	28	476.6	37	477.3	57	481.3	94	487.7	441
R.108.100	932.1	905.7	6	909.8	7	910.3	10	910.3	14	910.8	26	913.0	10	932.1	225
R.112.100	948.6	916.6	6	924.2	7	925.0	10	925.0	18	925.0	39	926.2	11	946.6	721
RC.203.100	923.7	814.3	7	914.6	11	916.0	16	916.5	29	917.4	43	919.5	22	923.7	67
RC.204.100	783.5	687.4	20	753.9	91	771.1	235	772.9	638	773.0	976	778.4	455	783.5	902
RC.206.100	1,051.1	951.5	8	1,022.8	13	1,034.4	13	1,034.9	15	1,035.4	16	1,037.7	16	1,051.1	55
RC.207.100	962.9	865.7	9	918.4	23	940.6	25	940.9	38	941.6	48	945.8	62	962.9	26,721
RC.208.100	776.1	703.9	20	740.9	42	761.3	114	762.6	223	763.8	352	765.8	168	776.1	980
R.202.100	1,029.6	1,008.6	8	1,018.6	9	1,020.6	19	1,020.6	23	1,020.7	24	1,021.2	22	1,027.3	1,890
R.203.100	870.8	845.7	9	858.0	64	862.8	91	864.3	108	865.0	149	865.8	165	870.8	1,941
R.204.100	731.3	688.8	22	719.1	54	721.3	64	722.2	180	722.2	298	724.2	194	731.3	216,146
R.205.100	949.8	916.0	11	928.0	14	934.1	27	934.6	31	935.0	32	938.0	31	948.3	1,240
R.206.100	875.9	834.0	13	857.6	21	860.2	34	860.7	36	861.6	53	866.3	76	875.9	6,474
R.207.100	794.0	746.9	16	780.4	83	781.6	120	781.6	191	782.9	236	789.9	368	794.0	1,163
R.209.100	854.8	818.7	15	833.8	16	837.0	28	837.6	30	838.8	34	840.6	59	854.3	4,355
R.210.100	900.5	848.9	17	880.6	18	883.8	30	884.8	37	884.8	49	888.2	57	900.4	39,572
R.211.100	746.7	704.9	21	727.1	102	729.3	133	731.3	189	731.3	417	734.1	219	746.7	10,825
Avg		93.3		97.5		98.2		98.3		98.4		98.7		99.9	

Table 4.7: VRPTW: effectiveness of the Dominance and Fathoming Rules applied in `GenP4`

	Generating $\mathscr{F}$					*Generating $\mathscr{B}$*				
Inst	*States*	%Dom1	%Fath1	%Fath2	$\|\mathscr{F}\|$	*States*	%Dom1	%Fath1	%Fath2	$\|\mathscr{B}\|$
RC.207.100	41	38.4	48.7	1.8	5	21	37.9	49.2	1.7	2
R.202.100	11	49.0	41.4	1.6	1	7	49.2	39.7	1.1	1
R.203.100	32	46.9	42.3	1.5	3					
R.204.100	180	43.8	43.6	0.4	22	283	37.1	48.3	0.4	40
R.206.100	235	58.8	32.1	1.2	19	179	67.4	24.2	1.3	13
R.207.100	81	54.4	38.6	1.0	5	43	54.8	36.4	0.9	3
R.209.100	102	55.2	37.9	1.3	6	31	56.1	38.0	0.8	2
R.210.100	184	57.5	33.7	0.9	15	222	59.7	31.4	1.0	18
R.211.100	496	68.6	24.1	0.6	33	371	56.7	34.4	0.3	32
Avg		52.5	38.0	1.1			52.4	37.7	0.9	

Table 4.8: VRPTW: effectiveness of the Dominance and Fathoming Rules applied in `GenPF`

	Generating $\mathscr{F}$				*Generating $\mathscr{B}$*			
Inst	*States*	%Fath5	%Fath4	$\|\mathscr{F}\|$	*States*	%Fath5	%Fath4	$\|\mathscr{B}\|$
R.202.100	10	37.1	51.8	1	3	47.1	46.9	1
R.209.100	339	50.0	42.2	26	113	51.3	43.4	6
R.210.100	108	45.8	46.0	9	210	55.1	35.4	20
Avg		44.3	46.7			51.2	41.9	

Table 4.9: Results on CVRP instances of class A

			Proc. H^1		*Proc. H^3*				*Proc. H^4*				Problem $\widetilde{P}$		T_{tot}			
Inst	z_{UB}	z^*	LB_1	T	LB_3	T	$\lvert\mathscr{F}^3\rvert$	$\lvert\mathscr{R}^3\rvert$	$SR3$	LB_4	UB_4	T	$\lvert\widetilde{\mathscr{R}}\rvert$	T_{cpx}	BMR	BCM	FLL (s)	LLE
A-n37-k5	669	669	664.7	3	666.1	5	5	1	3	669.0		6			6	13	19 (-)	25
A-n37-k6	949	949	929.8	3	937.9	7	9	10	37	941.6		8	1,968	< 0.1	8	12	379 (3)	531
A-n38-k5	730	730	717.9	4	722.4	7	10	10	41	728.7		7	267	< 0.1	7	18	26 (-)	116
A-n39-k5	822	822	815.1	4	818.7	7	8	2	14	822.0		7			7	19	167 (3)	138
A-n39-k6	831	831	820.5	4	823.7	6	5	4	31	831.0		7	149	< 0.1	7	12	98 (3)	109
A-n44-k6	937	937	932.4	2	935.3	5	2	1	2	937.0		5	124	< 0.1	5	102	90 (2)	620
A-n45-k6	944	944	933.0	6	941.1	11	4	1	4	944.0		12	114	< 0.1	12	77	170 (3)	157
A-n45-k7	1,146	1,146	1,135.7	5	1,141.4	10	5	4	62	1,146.0		11	316	< 0.1	11	35	331 (3)	19,414
A-n46-k7	914	914	912.6	6	914.0	8	1	1	0	914.0		8			8	11	92 (2)	50
A-n48-k7	1,073	1,073	1,065.9	9	1,071.8	14	2	1	1	1,073.0		15	129	< 0.1	15	20	166 (3)	372
A-n53-k7	1,010	1,010	1,001.1	11	1,003.8	21	15	12	78	1,009.7		24	364	< 0.1	24	28	611 (3)	363
A-n54-k7	1,167	1,167	1,148.7	12	1,156.6	25	84	207	91	1,166.1		35	709	< 0.1	35	86	1,409 (3)	7,246
A-n55-k9	1,073	1,073	1,064.2	9	1,067.8	18	10	5	30	1,070.7		19	771	< 0.1	19	19	84 (3)	468
A-n60-k9	1,354	1,354	1,339.0	17	1,343.9	32	78	201	108	1,351.7		44	2,704	2	46	111	3,080 (3)	-
A-n61-k9	1,034	1,034	1,016.0	13	1,023.2	31	45	45	80	1,032.0		37	1,009	< 0.1	37	34	1,883 (3)	68,636
A-n62-k8	1,290	1,288	1,271.8	19	1,280.5	42	229	468	106	1,285.4		65	13,099	31	96	1,342	3,102 (3)	-
A-n63-k9	1,616	1,616	1,596.8	18	1,608.6	37	20	17	32	1,616.0		40	303	< 0.1	40	44	1,046 (3)	-
A-n63-k10	1,315	1,314	1,291.9	20	1,302.4	42	58	80	60	1,309.5		47	4,616	4	51	129	4,988 (3)	-
A-n64-k9	1,402	1,401	1,376.7	16	1,387.5	36	146	271	84	1,395.3		49	7,810	14	63	120	11,254 (3)	-
A-n65-k9	1,174	1,174	1,161.0	14	1,165.6	29	20	24	29	1,174.0		32	294	< 0.1	32	38	516 (3)	1,324
A-n69-k9	1,159	1,159	1,136.7	20	1,143.9	43	131	363	84	1,156.8		55	1,659	< 0.1	55	124	7,171 (3)	-
A-n80-k10	1,763	1,763	1,746.2	30	1,755.5	63	126	80	89	1,760.7		78	2,513	4	82	194	6,464 (3)	-
Avg			98.8		99.4					99.9					30	118	1,961	6,638
Solved		22													22	22	22	15

Table 4.10: Results on CVRP instances of class B

			Proc. H^1		*Proc.* H^3				*Proc.* H^4				Problem $\widetilde{P}$		T_{tot}			
Inst	z_{UB}	z^*	LB_1	T	LB_3	T	$\lvert\mathscr{F}^3\rvert$	$\lvert\mathscr{R}^3\rvert$	$SR3$	LB_4	UB_4	T	$\lvert\widetilde{\mathscr{R}}\rvert$	T_{cpx}	BMR	BCM	FLL (s)	LLE
B-n38-k6	805	805	801.5	3	803.8	4	2	1	4	805.0		4			4	16	14 (-)	37
B-n39-k5	549	549	549.0	5	549.0	6	1	1	-	549.0		6			6	7	3 (-)	9
B-n41-k6	829	829	827.7	3	828.7	4	1	1	2	829.0		4			4	15	18 (-)	42
B-n43-k6	742	742	735.7	4	736.8	5	88	29	91	740.0		20	9,591	3	23	37	29 (-)	125
B-n44-k7	909	909	909.0	2	909.0	3	1	1	-	909.0		3			3	24	9 (-)	8
B-n45-k5	751	751	748.8	6	750.9	8	2	1	-	751.0		9			9	16	16 (-)	46
B-n45-k6	678	678	675.3	5	677.0	8	11	1	-	678.0		10			10	61	279 (3)	299
B-n50-k7	741	741	741.0	3	741.0	4	1	1	-	741.0		4			4	24	6 (-)	11
B-n50-k8	1,312	1,312	1,291.1	11	1,302.6	24	1,011	334	92	1,308.8	1,312	118	23,073	29	147	662	2,845 (3)	31,026
B-n51-k7	1,032	1,032	1,025.8	8	1,026.7	11	339	27	71	1,032.0		32	8,793	2	34	52	46 (-)	209
B-n52-k7	747	747	746.1	7	746.1	7	2	1	3	747.0		8			8	29	9 (-)	25
B-n56-k7	707	707	704.3	7	704.8	9	28	1	-	705.0		11	1,653	¡0.1	11	73	22 (-)	46
B-n57-k7	1,153	1,153	1,149.9	16	1,152.8	22	6	1	-	1,153.0		25			25	184	168 (-)	441
B-n57-k9	1,598	1,598	1,593.7	11	1,596.0	15	9	1	30	1,598.0		19			19	60	193 (3)	1,366
B-n63-k10	1,496	1,496	1,483.8	15	1,486.5	25	472	278	31	1,496.0		52	10,360	3	55	94	682 (-)	6,513
B-n64-k9	861	861	859.7	11	860.2	13	17	1	3	861.0		15			15	70	86 (-)	42
B-n66-k9	1,316	1,316	1,301.7	17	1,307.7	27	2,547	1,331	137	1,315.2	1,330	283	13,442	9	292	227	1,778 (3)	24,424
B-n67-k10	1,032	1,032	1,025.8	12	1,027.4	18	67	14	85	1,032.0		41	9,400	2	43	287	568 (-)	3,309
B-n68-k9	1,275	1,272	1,261.1	23	1,263.2	34	2,267	1,103	141	1,267.6	1,272	336	55,097	190	526	6,168	87,436 (3)	-
B-n78-k10	1,221	1,221	1,208.4	23	1,214.9	36	266	71	52	1,221.0		91	11,131	6	97	229	1,053 (3)	87,408
Avg			99.5		99.7					99.9					67	417	4,763	8,178
Solved		20													20	20	20	19

TABLE 4.11: Results on CVRP instances of classes E and M

			Proc. H^1		*Proc.* H^3				*Proc.* H^4				Problem $\widetilde{P}$		T_{tot}			
Inst	z_{UB}	z^*	LB_1	T	LB_3	T	$\lvert\mathscr{F}^3\rvert$	$\lvert\mathscr{R}^3\rvert$	$SR3$	LB_4	UB_4	T	$\lvert\widetilde{\mathscr{R}}\rvert$	T_{cpx}	BMR	BCM	FLL (s)	LLE
E-n51-k5	521	521	516.9	3	517.6	7	7	7	14	521.0		8			8	12	65 (-)	59
E-n76-k7	682	682	667.2	5	669.8	28	910	o	193	680.6		151	3,597	1	152	3,370	46,520 (2)	118,683
E-n76-k8	735	735	724.3	13	725.1	45	218	1,249	160	734.0		81	1,765	1	82	873	22,891 (2)	-
E-n76-k10	830	830	815.4	18	816.5	54	317	1,476	143	826.2		81	10,696	33	114	174	80,722 (3)	-
E-n76-k14	1,021	1,021	1,004.3	15	1,007.0	36	109	255	93	1,014.7		42	8,965	10	52	44	48,637 (3)	-
E-n101-k8	815	815	801.8	30	808.8	218	1,584	o	118	815.0	818	579			579	-	801,963 (3)	-
E-n101-k14	1,071	1,067	1,049.6	10	1,052.9	45	2,272	o	198	1,062.3	1,067	319	56,692	134	453	1,230	116,284 (3)	-
M-n101-k10	820	820	819.9	18	820.0	34	50	1	-	820.0		35			35	47	119 (-)	33
M-n121-k7	1,034	1,034	1,028.5	94	1,032.5	611	2,961	12	36	1,033.4	1,035	1,247	12,440	2	1,249	2,448	25,678 (3)	-
M-n151-k12	1,015		993.4	131	1,004.3	380	o	o		m.o.					-	-	-	-
M-n200-k16	-		1,240.3	109	1,256.6	319	o	o		m.o.					-	-	-	-
M-n200-k17	1,275		1,243.1	139	1,258.7	436	o	o		m.o.					-	-	-	-
Avg			98.5		99.0					99.8					303	1,025	126,987	39,592
Solved		12													9	8	9	3

m.o.: `GenP` runs out of memory

TABLE 4.12: Results on CVRP instances of class F

			Proc. H^1		*Proc.* H^3				*Proc.* H^4				Problem $\widetilde{P}$		T_{tot}			
Inst	z_{UB}	z^*	LB_1	T	LB_3	T	$\lvert\mathscr{F}^3\rvert$	$\lvert\mathscr{R}^3\rvert$	$SR3$	LB_4	UB_4	T	$\lvert\widetilde{\mathscr{R}}\rvert$	T_{cpx}	BMR	BCM	FLL (s)	LLE
F-n45-k4	724	724	723.9	20	724.0	23	4	1	0	724.0		23			23		8 (-)	6
F-n72-k4	237	237	233.0	249	236.2	301	228	10	1	237.0		304			304		121 (-)	40
F-n135-k7	1,162	1,162	1,158.0	1,057	m.o.										-		7,065 (-)	3,092
Avg			99.3		99.8					100.0					164		2,398	1,046
Solved		3													2		3	3

m.o.: `GenP` runs out of memory

TABLE 4.13: Results on CVRP instances of class P

			Proc. H^1		*Proc.* H^3				*Proc.* H^4				Problem $\widetilde{P}$		T_{tot}			
Inst	z_{UB}	z^*	LB_1	T	LB_3	T	$\lvert\mathscr{F}^3\rvert$	$\lvert\mathscr{R}^3\rvert$	$SR3$	LB_4	UB_4	T	$\lvert\widetilde{\mathscr{R}}\rvert$	T_{cpx}	BMR	BCM	FLL (s)	LLE
P-n16-k8	450	450	448.0	< 0.1	448.0	1	1	1	1	450.0		1			1	1	1 (2)	10
P-n19-k2	212	212	210.7	< 0.1	212.0	1	1	1	-	212.0		1			1	1	1 (-)	5
P-n20-k2	216	216	212.8	< 0.1	215.6	1	1	1	2	216.0		1			1	2	1 (-)	15
P-n21-k2	211	211	211.0	< 0.1	211.0	1	1	1	-	211.0		1			1	1	1 (-)	3
P-n22-k2	216	216	214.8	1	215.4	1	1	1	1	216.0		1			1	3	2 (-)	16
P-n22-k8	603	603	602.9	< 0.1	603.0	1	1	1	-	603.0		1			1	12	3 (2)	44
P-n23-k8	529	529	529.0	< 0.1	529.0	1	1	1	-	529.0		1			1	1	18 (2)	69
P-n40-k5	458	458	456.1	1	456.6	2	1	1	11	458.0		2			2	26	34 (-)	19
P-n45-k5	510	510	504.2	3	504.8	6	1	1	19	510.0		6			6	21	194 (3)	76
P-n50-k7	554	554	548.7	4	549.7	8	6	5	6	554.0		9			9	16	143 (3)	805
P-n50-k8	649	631	613.4	6	615.9	15	47	70	113	625.1	631	64	2,993	4	68	596	9,272 (3)	-
P-n50-k10	696	696	685.4	4	689.2	8	6	6	45	694.9		8	393	1	9	10	304 (3)	73,016
P-n51-k10	741	741	730.0	4	734.0	10	6	5	49	740.9		10	202	1	11	9	105 (3)	82,469
P-n55-k7	568	568	556.5	7	559.1	12	50	121	97	565.7		16	2,410	1	17	205	4,649 (2)	11,178
P-n55-k8	588	588	577.3	6	579.3	13	39	98	103	584.7		16	3,851	2	18	74	1,822 (2)	
P-n55-k10	699	694	677.8	5	680.6	15	112	314	84	689.1		19	18,061	10	29	66	9,076 (3)	-
P-n55-k15	993	989	966.6	7	970.7	25	20	18	46	984.5		26	1,501	1	27	8	1,944 (3)	-
P-n60-k10	756	744	736.8	9	739.3	22	125	395	57	743.1		28	53,850	2	30	59	570 (3)	-
P-n60-k15	1,033	968	959.8	6	963.2	15	3	2	45	967.8	968	73	241	< 0.1	73	8	442 (3)	-
P-n65-k10	792	792	784.5	5	787.9	13	6	6	27	792.0		14			14	16	422 (3)	-
P-n70-k10	834	827	811.5	13	813.6	34	147	483	129	823.0	827	150	7,925	16	166	774	24,039 (3)	-
P-n76-k4	593	593	587.0	13	590.0	89	174	569	65	593.0		118			118	2,211	572 (-)	535
P-n76-k5	627	627	614.1	5	619.5	87	905	3,707	137	627.0		282	401	< 0.1	282	-	14,546 (-)	10,970
P-n101-k4	681	681	668.5	13	678.7	73	4,475	8,811	106	681.0	681	1,154	458	1	1,155	-	1,253 (-)	281
Avg			98.8		99.2					99.8					85	187	2,892	11,219
Solved		24													24	22	24	16

Table 4.14: CVRP instances: summary

		BMR			BCM			FLL					LLE		
Class	NP	*Opt*	*%LB*	*T*	*Opt*	*%LB*	*T*	*Opt*	Opt_{BCP}	Opt_{BC}	*%LB*	*T*	*Opt*	*%LB*	*T*
A	22	22	99.9	30	22	99.8	118	22	20	2	99.2	1,961	15	97.9	6,638
B	20	20	99.9	67	20	99.8	417	20	6	14	99.5	4,763	19	99.4	8,178
E-M	12	9	99.8	303	8	99.4	1,025	9	7	2	98.9	126,987	3	97.7	39,592
F	3	2	100.0	164				3	0	3	99.9	2,398	3	99.9	1,046
P	24	24	99.8	85	22	99.7	187	24	16	8	99.2	2,892	16	97.7	11,219
Avg			99.9	92		99.7	323				99.3	17,409		98.4	9,935
Tot	81	77			72			78	49	29			56		

Table 4.15: CVRP: impact of parameter $\Delta(N_i)$ and of different types of inequalities

		H^1				H^2								H^3				H^4	
		no CCs		CCs		$\Delta(N_i) = 5$		$\Delta(N_i) = 8$		$\Delta(N_i) = 10$		$\Delta(N_i) = 12$		no WSR3s		WSR3s			
Inst	z^*	LB_1	T	LB_1	T	LB_2	T	LB_2	T	LB_2	T	LB_2	T	LB_3	T	LB_3	T	LB_4	T
B-n50-k8	1,312.0	1,229.0	2	1,291.1	11	1,302.6	20	1,302.6	21	1,302.6	21	1,302.6	22	1,302.6	24	1,303.0	39	1,308.8	118
B-n66-k9	1,316.0	1,223.8	3	1,301.7	17	1,307.0	20	1,307.4	22	1,307.4	23	1,307.4	24	1,307.7	27	1,309.7	70	1,315.2	283
B-n68-k9	1,272.0	1,166.8	3	1,261.1	23	1,263.2	28	1,263.2	29	1,263.2	29	1,263.2	31	1,263.2	34	1,264.3	103	1,267.6	336
B-n78-k10	1,221.0	1,126.4	4	1,208.4	23	1,212.7	27	1,213.9	29	1,214.3	30	1,214.5	32	1,214.9	36	1,218.0	67	1,221.0	91
E-n76-k7	682.0	667.2	2	667.2	5	668.5	20	668.9	22	669.2	23	669.3	23	669.8	28	673.3	74	680.6	151
E-n76-k8	735.0	720.3	2	724.3	13	724.7	33	724.8	34	724.9	36	724.9	38	725.1	45	729.1	72	734.0	81
E-n76-k10	830.0	811.3	1	815.4	18	815.9	46	815.9	46	816.1	49	816.2	52	816.5	54	820.9	73	826.2	81
E-n76-k14	1,021.0	999.2	1	1,004.3	15	1,006.1	26	1,006.7	28	1,007.0	33	1,007.0	36	1,007.0	36	1,009.3	40	1,014.7	42
E-n101-k8	815.0	796.5	5	801.8	30	802.1	32	802.1	32	802.4	34	802.4	35	806.4	35	808.8	218	815.0	579
E-n101-k14	1,067.0	1,044.6	5	1,049.6	10	1,047.5	33	1,047.6	35	1,047.7	40	1,047.8	40	1,052.9	45	1,056.7	191	1,062.3	319
F-n135-k7	1,162.0	1,098.2	140	1,158.0	1,057	1,158.9	1,359	1,159.1	1,396	1,159.3	1,405	1,159.3	1,423	-		-		-	
M-n121-k7	1,034.0	1,013.2	9	1,028.5	94	1,031.1	173	1,031.8	178	1,032.0	191	1,032.0	197	1,032.2	220	1,032.5	611	1,033.4	1,247
P-n50-k8	631.0	612.3	< 0.1	613.4	6	614.6	10	615.3	11	615.5	13	615.9	13	615.9	15	618.6	27	625.1	64
P-n55-k10	694.0	676.9	1	677.8	5	679.7	9	680.6	11	680.6	12	680.6	12	680.6	15	682.6	25	689.1	19
P-n70-k10	827.0	807.9	1	811.5	13	812.3	24	812.9	28	813.2	31	813.3	32	813.6	34	817.6	60	823.0	150
P-n76-k5	627.0	614.1	3	614.1	5	615.3	24	615.7	27	615.9	27	616.1	28	616.8	32	619.5	87	627.0	282
P-n101-k4	681.0	668.5	13	675.7	55	676.0	58	676.0	59	676.4	60	676.4	60	676.9	62	678.7	73	681.0	1,154
Avg		96.4		98.5		98.7		98.7		98.8		98.8		98.8		99.1		99.7	

Chapter 5

Multi-Trip Vehicle Routing Problem

[1]

The *multi-trip vehicle routing problem* (MTVRP) is a variant of the *capacitated vehicle routing problem* where each vehicle can perform a subset of routes, called a vehicle schedule, subject to maximum driving time constraints. Despite its practical importance, the MTVRP has received little attention in the literature. Few heuristics have been proposed, and only an exact algorithm has been presented for a variant of the MTVRP with customer time window constraints and unlimited driving time for each vehicle. We describe two set-partitioning-like formulations of the MTVRP. The first formulation requires the generation of all feasible routes, whereas the second formulation is based on the generation of all feasible schedules. We study valid lower bounds, based on the linear relaxations of both formulations enforced with valid inequalities, that are embedded into an exact solution method. The computational results show that the proposed exact algorithm can solve MTVRP instances taken from the literature, with up to 120 customers.

5.1 Introduction

The *capacitated vehicle routing problem* (CVRP) and its many variations play an important role in the management of many distribution systems. In most of the studied models, the vehicles are identical and each vehicle is allowed to perform, at most, a single route.

In many contexts, a distribution company uses leased vehicles to service the customers and incurs a significant cost for each vehicle used. Whenever the planning period

[1]This chapter is based on Mingozzi et al. [2012]

is large with respect to the route duration, and some vehicles can perform several routes in the period, the primary concern of the company is to minimize the number of vehicles used. The *multi-trip vehicle routing problem* (MTVRP) is an extension of the CVRP where each vehicle can perform multiple routes during its working period. The MTVRP can be described as follows.

A complete undirected graph $G = (V', E)$ is given, where V' is the set of vertices and E is the set of edges. We have $V' = \{0\} \cup V$, where vertex 0 represents the *depot* and the set $V = \{1, \dots, n\}$ represents n *customers*, each one requiring q_i units of product from the depot. A fleet $M = \{1, \dots, m\}$ of m identical vehicles is located at the depot. Each vehicle has capacity Q and maximum driving time T. A *travel cost* α_{ij} and a *travel time* τ_{ij} are associated with each edge $\{i, j\} \in E$. All input data are assumed to be nonnegative.

A *route* of a vehicle is a least-cost elementary cycle in G that passes through the depot and a subset of the customers such that the total demand of the customers visited does not exceed the vehicle capacity Q. The cost (duration) of a route is equal to the sum of the travel costs (travel times) of the edges traversed.

A *schedule* of a vehicle is a subset of routes whose total duration is less than or equal to the maximum driving time T. The cost of a schedule is equal to the sum of the costs of its routes.

The MTVRP calls for the design of a set of m schedules of minimum total cost such that each customer is visited exactly once by the routes of the schedules.

5.2 Literature Review

To the best of our knowledge, no exact algorithm for the MTVRP has been presented in the literature so far. The main heuristic methods proposed are surveyed in the following. Fleischmann [1990] addressed the problem first. He proposed a modification of the well-known saving algorithm and by using a bin packing heuristic to assign the routes to the vehicles. Taillard et al. [1996] proposed a three-phase algorithm. In the first phase, a set of routes satisfying the capacity constraints are designed. Next, the routes are combined to derive different CVRP solutions. Finally, the routes of each CVRP solution are assigned to the vehicles by solving a bin packing problem. Petch and Salhi [2004] described a multi-phase heuristic that constructs many feasible CVRP solutions and, for each one, assigns routes to vehicles with a bin packing heuristic. Salhi and Petch [2007] investigated a genetic algorithm approach. Olivera and Viera [2007] described an adaptive memory heuristic algorithm. Different tabu search algorithms were proposed by Alonso et al. [2008], Brandão and Mercer [1997, 1998]. A real life application of the MTVRP is described in Gribkovskaia et al. [2006].

Azi et al. [2010] addressed a variant of the MTVRP where: (i) a time window and a revenue are associated with each customer, (ii) the duration of each route is limited, (iii) the maximum driving time of the vehicles is unlimited, and (iv) it is not required to service all customers. Thus, the customers must be chosen based on their associated revenue minus the traveling cost to reach them. Azi et al. described a branch-and-price algorithm based on a set packing formulation that can routinely solve instances with 25 customers and a few instances with up to 50 customers. When the time windows are removed, the problem becomes the well-known distance-constrained CVRP. In this case, the algorithm was able to solve two instances with 25 customers.

5.3 Mathematical Formulations and Relaxations

In this section, we describe two set-partitioning-like formulations, called $F1$ and $F2$, and a valid integer relaxation, called $RF1$, of the MTVRP. Formulation $F1$ requires the a priori generation of all feasible routes, whereas formulation $F2$ is based on the generation of all feasible schedules. Relaxation $RF1$ is an IP problem similar to the set partitioning formulation of the CVRP.

5.3.1 Formulation $F1$

Let $\mathscr{R}$ be the index set of all feasible routes on graph G, and let $\mathscr{R}_i \subseteq \mathscr{R}$ be the index subset of the routes visiting customer $i \in V$. A cost d_ℓ and a duration τ_ℓ are associated with each route $\ell \in \mathscr{R}$. In the following, we use R_ℓ and $E(R_\ell)$ to indicate the set of customers visited and the edges traversed by route $\ell \in \mathscr{R}$, respectively.

Let ξ_ℓ^j be a binary variable equal to 1 if and only if route $\ell \in \mathscr{R}$ is assigned to vehicle $j \in M$. The mathematical formulation $F1$ is as follows

$$(F1) \quad z(F1) = \min \sum_{\ell \in \mathscr{R}} d_\ell \sum_{j \in M} \xi_\ell^j \tag{5.1}$$

$$s.t. \sum_{\ell \in \mathscr{R}_i} \sum_{j \in M} \xi_\ell^j = 1, \qquad i \in V, \tag{5.2}$$

$$\sum_{\ell \in \mathscr{R}} \tau_\ell \xi_\ell^j \leq T, \qquad j \in M, \tag{5.3}$$

$$\xi_\ell^j \in \{0,1\}, \qquad j \in M, \ \ell \in \mathscr{R}. \tag{5.4}$$

Constraints (5.2) impose that each customer is visited exactly once, and constraints (5.3) define a feasible schedule for each vehicle used.

We denote by $LF1$ the LP-relaxation of formulation $F1$ and by $z(LF1)$ its optimal solution cost.

5.3.2 Formulation $F2$

Let $\mathscr{H}$ be the index set of all schedules. For each schedule $k \in \mathscr{H}$, we denote by $\Omega_k \subseteq \mathscr{R}$ the index subset of the routes in the schedule, by $c_k = \sum_{\ell \in \Omega_k} d_\ell$ its cost, and by $\tau(\Omega_k) = \sum_{\ell \in \Omega_k} \tau_\ell$ its total duration. Moreover, we denote by $V(\Omega_k) = \cup_{\ell \in \Omega_k} R_\ell$ and $E(\Omega_k) = \cup_{\ell \in \Omega_k} E(R_\ell)$ the set of customers visited and the set of edges traversed by schedule $k \in \mathscr{H}$, respectively.

We assume that $\mathscr{H}$ contains *undominated* schedules only (i.e., given $k \in \mathscr{H}$, there exists no $k' \in \mathscr{H} \setminus \{k\}$ such that $V(\Omega_k) = V(\Omega_{k'})$ and $c_k > c_{k'}$). Notice that whenever $m = 1$, $\mathscr{H}$ contains only schedules corresponding to optimal integer solutions of the CVRP, where the total duration of the routes is less than or equal to T.

Let y_k be a binary variable equal to 1 if and only if schedule $k \in \mathscr{H}$ is assigned to a vehicle. Formulation $F2$ of the MTVRP is as follows

$$(F2) \quad z(F2) = \min \sum_{k \in \mathscr{H}} c_k y_k \tag{5.5}$$

$$s.t. \quad \sum_{k \in \mathscr{H} \,:\, i \in V(\Omega_k)} y_k = 1, \qquad i \in V, \tag{5.6}$$

$$\sum_{k \in \mathscr{H}} y_k \le m, \tag{5.7}$$

$$y_k \in \{0,1\}, \qquad k \in \mathscr{H}. \tag{5.8}$$

Constraints (5.6) specify that each customer $i \in V$ must be visited exactly once. Constraint (5.7) imposes the upper bound on the number of vehicles used.

We denote by $LF2$ the LP-relaxation of $F2$ and by $z(LF2)$ its optimal solution cost.

5.3.3 Comparing Relaxations $LF1$ and $LF2$

The following proposition shows the relation between relaxations $LF1$ and $LF2$.

Proposition 1. The inequality $z(LF1) \le z(LF2)$ holds and can be strict.

Proof. We show that any optimal dual solution of $LF1$ of cost $z(LF1)$ is a feasible dual solution of $LF2$ of cost $z(LF1)$, as well. Let $\boldsymbol{u} = (u_1, \dots, u_n)$ and $\boldsymbol{w} = (w_1, \dots, w_m)$

be the vectors of the dual variables associated with constraints (5.2) and (5.3), respectively. The dual of $LF1$ is

$$(DF1) \quad z(DF1) = \max \sum_{i\in V} u_i + T \sum_{j\in M} w_j \tag{5.9}$$

$$s.t. \sum_{i\in R_\ell} u_i + \tau_\ell w_j \le d_\ell, \quad j \in M,\ \ell \in \mathscr{R}, \tag{5.10}$$

$$u_i \in \mathbb{R}, \qquad i \in V, \tag{5.11}$$

$$w_j \le 0, \qquad j \in M. \tag{5.12}$$

Consider a vehicle schedule Ω_k. Adding inequalities (5.10) over the routes of Ω_k and all vehicles of M and dividing by m, we obtain

$$\sum_{\ell\in\Omega_k}\sum_{i\in R_\ell} u_i + \sum_{\ell\in\Omega_k}\frac{\tau_\ell}{m}\sum_{j\in M} w_j \le \sum_{\ell\in\Omega_k} d_\ell. \tag{5.13}$$

Define $\bar{u}_0 = \frac{T}{m}\sum_{j\in M} w_j$. Because $\sum_{\ell\in\Omega_k}\tau_\ell \le T$ and $w_j \le 0$, we have

$$\bar{u}_0 \le \sum_{\ell\in\Omega_k}\frac{\tau_\ell}{m}\sum_{j\in M} w_j \le 0. \tag{5.14}$$

From inequalities (5.13) and (5.14), we derive

$$\sum_{\ell\in\Omega_k}\sum_{i\in R_\ell} u_i + \bar{u}_0 \le c_k,$$

which corresponds to the dual constraint associated with variable y_k of problem $LF2$, thus showing that $\bar{\boldsymbol{u}} = (\bar{u}_0, u_1, \ldots, u_n)$ is a dual solution of $LF2$, where u_i, $i \in V$, are the dual variables of constraints (5.6) and $\bar{u}_0$ is the dual variable of constraint (5.7). The cost of the dual solution $\bar{\boldsymbol{u}}$ of $LF2$ defined above is

$$\sum_{i\in V} u_i + m\bar{u}_0 = \sum_{i\in V} u_i + \frac{mT}{m}\sum_{j\in M} w_j = z(LF1).\square$$

The following example shows that $z(LF1)$ can be strictly smaller than $z(LF2)$.

Example 1. Consider an MTVRP instance with $n = 4$, $m = 2$, $T = 100$, $Q = 2$, $q_1 = 2$, $q_2 = 1$, $q_3 = 1$, and $q_4 = 2$. Travel costs and times coincide; in particular, we have $\alpha_{01} = \tau_{01} = 30$, $\alpha_{02} = \tau_{02} = 15$, $\alpha_{03} = \tau_{03} = 15$, $\alpha_{04} = \tau_{04} = 30$, and $\alpha_{23} = \tau_{23} = 20$.

Formulation $F1$ has the following five feasible routes (i.e., $|\mathscr{R}| = 5$) for each vehicle: $R_1 = \{1\}$, $d_1 = 60$; $R_2 = \{2\}$, $d_2 = 30$; $R_3 = \{3\}$, $d_3 = 30$; $R_4 = \{4\}$, $d_4 = 60$; and $R_5 = \{2, 3\}$, $d_5 = 50$. Solving $LF1$, we obtain $z(LF1) = 170$ corresponding to $\xi^1_1 = 1$, $\xi^1_5 = \frac{1}{2}$, $\xi^2_4 = 1$, $\xi^2_5 = \frac{1}{2}$ (the other ξ^j_ℓ variables having a null value).

Formulation $F2$ has the following nine schedules (i.e., $|\mathscr{H}| = 9$): $\Omega_1 = \{1\}$, $c_1 = 60$; $\Omega_2 = \{2\}$, $c_2 = 30$; $\Omega_3 = \{3\}$, $c_3 = 30$; $\Omega_4 = \{4\}$, $c_4 = 60$; $\Omega_5 = \{5\}$, $c_5 = 50$; $\Omega_6 = \{1, 2\}$, $c_6 = 90$; $\Omega_7 = \{1, 3\}$, $c_7 = 90$; $\Omega_8 = \{2, 4\}$, $c_8 = 90$; and $\Omega_9 = \{3, 4\}$, $c_9 = 90$. Solving $LF2$, we obtain $z(LF2) = 180 > z(LF1)$ corresponding to solution $y_6 = 1$, $y_9 = 1$, and $y_k = 0$, for any $k \neq 6, 9$.

5.3.4 Relaxation $RF1$

Problem $RF1$ is an integer problem derived from $F1$ and is used, in the bounding procedures described in the next sections, to compute three lower bounds to the MTVRP. Problem $RF1$ is obtained from $F1$ as follows.

1. Replace constraints (5.3) with the following surrogate constraint
$$\sum_{\ell \in \mathscr{R}} \tau_\ell \sum_{j \in M} \xi_\ell^j \leq mT. \tag{5.15}$$
2. Set $x_\ell = \sum_{j \in M} \xi_\ell^j$, $\ell \in \mathscr{R}$. Notice that, because of constraints (5.2), we have $x_\ell \in \{0, 1\}$.
3. Replace the term $\sum_{j \in M} \xi_\ell^j$ with x_ℓ into expressions (5.1), (5.2), and (5.15).

The resulting relaxed problem $RF1$ involves only the binary variables x_ℓ, $\ell \in \mathscr{R}$, where x_ℓ is equal to 1 if and only if route ℓ is in the solution.

$$(RF1) \quad z(RF1) = \min \sum_{\ell \in \mathscr{R}} d_\ell x_\ell \tag{5.16}$$
$$s.t. \sum_{\ell \in \mathscr{R}_i} x_\ell = 1, \qquad i \in V, \tag{5.17}$$
$$\sum_{\ell \in \mathscr{R}} \tau_\ell x_\ell \leq mT, \tag{5.18}$$
$$x_\ell \in \{0, 1\}, \qquad \ell \in \mathscr{R}. \tag{5.19}$$

In the following, we refer to problem $RF1$ as the CVRP relaxation of the MTVRP (or as the CVRP associated to the MTVRP) and denote by $LRF1$ the LP-relaxation of $RF1$ and by $z(LRF1)$ its optimal solution cost.

Proposition 2. The following equality holds

$$z(LF1) = z(LRF1). \tag{5.20}$$

Proof. Equality (5.20) follows from the observation that any $LF1$ solution $\boldsymbol{\xi}$ can be transformed into an $LRF1$ solution $\boldsymbol{x}$ of the same cost, and vice versa.

The following algorithm transforms a solution $\boldsymbol{x}$ of $LRF1$ into a feasible $LF1$ solution $\boldsymbol{\xi}$ of cost $z(LRF1)$.

1. Initialize $j = 0$.
2. Set $j = j + 1$ and $wt = 0$.
3. Let $\ell^* = \min\{\ell \in \mathscr{R} : x_{\ell^*} > 0\}$. If $\tau_{\ell^*} x_{\ell^*} \leq T - wt$, set $wt = wt + \tau_{\ell^*} x_{\ell^*}$, $\xi^j_{\ell^*} = x_{\ell^*}$, $x_{\ell^*} = 0$, and repeat Step 3. If $\tau_{\ell^*} x_{\ell^*} > T - wt$, define $p = \frac{(T-wt)}{\tau_{\ell^*} x_{\ell^*}}$, and set $\xi^j_{\ell^*} = p$, $x_{\ell^*} = 1 - p$, and return to Step 2.

Any $LF1$ solution $\boldsymbol{\xi}$ can be transformed into a feasible solution $\boldsymbol{x}$ of $LRF1$ of cost $z(LF1)$ by simply setting $x_\ell = \sum_{j \in M} \xi^j_\ell$. □

5.4 Improving Relaxations $LRF1$ and $LF2$ and Outline of the Exact Method

In this section, we describe two relaxations, called $\overline{LRF1}$ and $\overline{LF2}$, derived respectively from $LRF1$ and $LF2$ by adding some families of valid inequalities. These relaxations are used to derive valid lower bounds on the MTVRP that are embedded in the exact method proposed in §5.8.

5.4.1 Relaxation $\overline{LRF1}$

Relaxation $\overline{LRF1}$ derives from $LRF1$ by adding two types of valid inequalities for the set partitioning formulation of the CVRP (see Chapter 4), called *strengthened capacity inequalities* and *subset row inequalities*, and replacing inequality (5.18) with a new type of inequalities, called *working time inequalities*.

Strengthened Capacity Inequalities. Let $\mathscr{S}$ be the set of all the subsets of customers of cardinality greater than or equal to 2 (i.e., $\mathscr{S} = \{S \subseteq V : |S| \geq 2\}$), and let $\delta(S)$ be the cutset of $S \in \mathscr{S}$. Let $\rho_\ell(S)$ be the number of edges of $\delta(S)$ traversed by route $\ell \in \mathscr{R}$ (i.e., $\rho_\ell(S) = |E(R_\ell) \cap \delta(S)|$). The strengthened capacity (SC) inequalities are

$$\sum_{\ell \in \mathscr{R}} \rho_\ell(S) x_\ell \geq 2k(S), \quad S \in \mathscr{S},$$

where $k(S)$ is a lower bound on the number of vehicles required to service the customer set S and can be computed as $\lceil \sum_{i \in S} q_i / Q \rceil$.

Subset Row Inequalities. Subset row (SR) inequalities were introduced by Jepsen et al. [2008] for the *vehicle routing problem with time windows*. They combine clique inequalities with odd-hole inequalities. Let $\mathscr{C} = \{C \subseteq V : |C| \geq 3\}$, and let η be an integer such that $\eta > 1$. Any feasible $RF1$ solution must satisfy the following SR inequality

$$\sum_{\ell \in \mathscr{R}} \varphi_\ell(C) x_\ell \leq \varphi_0(C), \quad C \in \mathscr{C},$$

where $\varphi_\ell(C) = \left\lfloor \frac{|R_\ell \cap C|}{\eta} \right\rfloor$, and $\varphi_0(C) = \left\lfloor \frac{|C|}{\eta} \right\rfloor$. We consider the following three special cases of the SR inequalities where $\eta = 2$.

1. **SR3 inequalities**. SR3 inequalities correspond to the subset $\mathscr{C}^3 \subseteq \mathscr{C}$ defined as $\mathscr{C}^3 = \{C \subseteq V : |C| = 3\}$. SR3 inequalities correspond to a subset of clique inequalities and impose that, at most, one of the routes visiting at least two of the customers in $C \in \mathscr{C}^3$ can be in the solution.

2. **SR5 inequalities**. SR5 inequalities correspond to the subset $\mathscr{C}^5 \subseteq \mathscr{C}$ defined as $\mathscr{C}^5 = \{C \subseteq V : |C| = 5\}$. SR5 inequalities correspond to odd-hole inequalities and impose that, at most, two of the routes visiting at least two of the customers in $C \in \mathscr{C}^5$ can be in the solution.

3. **WSR3 inequalities**. Weak subset row (WSR3) inequalities, introduced in §4.3.2, are downlifted SR3 inequalities. They are used as an alternative to the SR3 inequalities in the bounding procedures described in §5.5. Let $E(C)$, $C \in \mathscr{C}^3$, be the edges whose terminal vertices are both in C. The coefficients of the WSR3 inequalities, $\varphi_\ell(C)$, are computed as $\varphi_\ell(C) = 1$ if $|E(R_\ell) \cap E(C)| \geq 1$, and $\varphi_\ell(C) = 0$ if $|E(R_\ell) \cap E(C)| = 0$.

Hereafter, we use $\mathscr{C} = \mathscr{C}^3 \cup \mathscr{C}^5$ and $C \in \mathscr{C}$ to refer to both the index and the customer subset of an SR inequality. The SR3, SR5, and WSR3 inequalities are separated through complete enumeration.

Working Time Inequalities. These new inequalities strengthen inequality (5.18). They are given by the following proposition.

Proposition 3. Let t_{min} be a lower bound on the duration of any route $\ell \in \mathscr{R}$. If travel costs α_{ij} satisfy the triangle inequality, we can define $t_{min} = \min_{i \in V}\{2\tau_{0i}\}$. For a given time $t \in [t_{min}, T]$, let $\hat{t} = T mod(t) + 1$, and let $\mathscr{L} = \{L \subseteq \mathscr{R} : \tau_\ell \in [\hat{t}, t) \text{ for each } \ell \in L, \text{ and } R_{\ell'} \cap R_{\ell''} \neq \varnothing, \text{ for each } \ell', \ell'' \in L\}$. The following inequalities, called working time (WT) inequalities, are valid for $LRF1$

$$\sum_{\ell \in \mathscr{R} \setminus L} \left\lfloor \frac{\tau_\ell}{t} \right\rfloor x_\ell + \sum_{\ell \in L} x_\ell \leq \left\lfloor \frac{T}{t} \right\rfloor m, \quad t \in [t_{min}, T], \ L \in \mathscr{L}. \tag{5.21}$$

Proof. For a given $L \in \mathscr{L}$, inequalities (5.3) can be written as

$$\sum_{\ell \in \mathscr{R} \setminus L} \tau_\ell \xi_\ell^j \leq T - \sum_{\ell \in L} \tau_\ell \xi_\ell^j, \quad j \in M. \tag{5.22}$$

Inequalities (5.22) can be relaxed by replacing τ_ℓ with $\hat{t}$, for each route $\ell \in L$

$$\sum_{\ell \in \mathscr{R} \setminus L} \tau_\ell \xi_\ell^j \leq T - \hat{t} \sum_{\ell \in L} \xi_\ell^j, \quad j \in M. \tag{5.23}$$

By dividing inequalities (5.23) by t and rounding down the coefficients, we have

$$\sum_{\ell \in \mathscr{R} \setminus L} \left\lfloor \frac{\tau_\ell}{t} \right\rfloor \xi_\ell^j \leq \left\lfloor \frac{T - \hat{t} \sum_{\ell \in L} \xi_\ell^j}{t} \right\rfloor, \quad j \in M. \tag{5.24}$$

Notice that because of the definition of the set L, $\sum_{\ell \in L} \xi_\ell^j$ is equal to either 0 or 1 for any feasible MTVRP solution. If $\sum_{\ell \in L} \xi_\ell^j = 0$, then $\lfloor \frac{T - \hat{t} \sum_{\ell \in L} \xi_\ell^j}{t} \rfloor = \lfloor \frac{T}{t} \rfloor$. If $\sum_{\ell \in L} \xi_\ell^j = 1$, because of the definition of $\hat{t}$, we have $\lfloor \frac{T - \hat{t} \sum_{\ell \in L} \xi_\ell^j}{t} \rfloor = \lfloor \frac{T - \hat{t}}{t} \rfloor = \lfloor \frac{T}{t} \rfloor - 1$. Thus, we have

$$\left\lfloor \frac{T - \hat{t} \sum_{\ell \in L} \xi_\ell^j}{t} \right\rfloor = \left\lfloor \frac{T}{t} \right\rfloor - \sum_{\ell \in L} \xi_\ell^j, \quad j \in M. \tag{5.25}$$

From inequalities (5.24) and equations (5.25), we derive

$$\sum_{\ell \in \mathscr{R} \setminus L} \left\lfloor \frac{\tau_\ell}{t} \right\rfloor \xi_\ell^j + \sum_{\ell \in L} \xi_\ell^j \leq \left\lfloor \frac{T}{t} \right\rfloor, \quad j \in M. \tag{5.26}$$

By summing up inequalities (5.26) and replacing $\sum_{j \in M} \xi_\ell^j$ with $x_\ell \in \{0, 1\}$, we derive inequalities (5.21). $\square$

The separation of inequalities (5.21) is $\mathcal{NP}$-hard because it requires the computation of the set $\mathscr{L}$. Therefore, we use a subset of the WT inequalities (5.21) that are obtained by defining $L_{it} = \{\ell \in \mathscr{R}_i : \tau_\ell \in [\hat{t}, t)\}$, for each vertex $i \in V$, and by setting $\mathscr{L} = \{L_{it} : i \in V,\ t \in [t_{min}, T]\}$. Let $\mathscr{W} = \{(t, i) : t \in [t_{min}, T],\ i \in V\}$. These latter inequalities can be conveniently written as follows

$$\sum_{\ell \in \mathscr{R}} \vartheta_\ell(t, i) x_\ell \leq \vartheta_0(t), \quad (t, i) \in \mathscr{W}, \tag{5.27}$$

where, for a given $(t, i) \in \mathscr{W}$, $\vartheta_\ell(t, i) = \lfloor \frac{\tau_\ell}{t} \rfloor$, for each route $\ell \in \mathscr{R} \setminus L_{it}$, $\vartheta_\ell(t, i) = 1$, for each route $\ell \in L_{it}$, and $\vartheta_0(t) = \lfloor \frac{T}{t} \rfloor m$. Inequalities (5.27) can be separated by complete enumeration.

Example 2. Consider the MTVRP instance defined in Example 1 of §5.3.3. Problem $RF1$ involves the same set $\mathscr{R}$ of routes of formulation $F1$. An optimal solution of $RF1$ of cost $z(RF1) = z(LRF1) = 170$ is given by $x_1 = 1$, $x_4 = 1$, and $x_5 = 1$. Consider the WT inequality (5.27) for $t = 60$ and $i = 3$. As $\hat{t} = 41$, we have $L_{60,3} = \{5\}$. Thus, the coefficients of (5.27) are $\vartheta_1(60, 30) = 1$, $\vartheta_2(60, 30) = 0$, $\vartheta_3(60, 30) = 0$, $\vartheta_4(60, 30) = 1$, $\vartheta_5(60, 30) = 1$, and $\vartheta_0(60) = 2$. Inequality (5.27) corresponds to $x_1 + x_4 + x_5 \leq 2$, which is violated by the aforementioned solution of $RF1$. $\square$

Relaxation $\overline{LRF1}$ is the following problem

$$
\begin{aligned}
(\overline{LRF1}) \quad z(\overline{LRF1}) = \min & \sum_{\ell \in \mathscr{R}} d_\ell x_\ell && && (5.28)\\
s.t. & \sum_{\ell \in \mathscr{R}_i} x_\ell = 1, && i \in V, && (5.29)\\
& \sum_{\ell \in \mathscr{R}} \rho_\ell(S) x_\ell \geq 2k(S), && S \in \mathscr{S}, && (5.30)\\
& \sum_{\ell \in \mathscr{R}} \varphi_\ell(C) x_\ell \leq \varphi_0(C), && C \in \mathscr{C}, && (5.31)\\
& \sum_{\ell \in \mathscr{R}} \vartheta_\ell(t,i) x_\ell \leq \vartheta_0(t), && (t,i) \in \mathscr{W}, && (5.32)\\
& x_\ell \geq 0, && \ell \in \mathscr{R}. && (5.33)
\end{aligned}
$$

Let $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g}, \boldsymbol{h})$ be the dual variables of $\overline{LRF1}$, where $\boldsymbol{u} \in \mathbb{R}^n$ is associated with (5.29), $\boldsymbol{v} \in \mathbb{R}_+^{|\mathscr{S}|}$ with (5.30), $\boldsymbol{g} \in \mathbb{R}_-^{|\mathscr{C}|}$ with (5.31), and $\boldsymbol{h} \in \mathbb{R}_-^{|\mathscr{W}|}$ with (5.32).

In §5.5, we describe three bounding procedures to solve $\overline{LRF1}$, called H^1, H^2, and H^3, that are extensions of similar procedures proposed in Chapter 4 for the CVRP. The three procedures are executed in sequence, and the dual solution of procedure H^k is used to hot-start procedure H^{k+1}, $k = 1, 2$.

Procedures H^1 and H^2 are dual-ascent heuristics based on a *cut-and-column generation* (CCG) method that differs from standard CCG methods based on the simplex for the use of a dual-ascent heuristic to find a near-optimal dual solution of problem $\overline{LRF1}$. Procedure H^1 finds a dual solution $(\boldsymbol{u^1}, \boldsymbol{v^1}, \boldsymbol{g^1})$ of cost LB_1 of problem $\overline{LRF1}$ by adding SC and WSR3 inequalities, ignoring WT inequalities, and replacing the route set $\mathscr{R}$ with the set of all q-routes (see Christofides et al. [1981b]). Procedure H^2 differs from H^1 as it uses elementary routes instead of q-routes. Procedure H^2 achieves a dual solution $(\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2})$ of cost LB_2, such that $LB_1 \leq LB_2$. Procedure H^3 is a CCG method based on the simplex to solve problem $\overline{LRF1}$. Procedure H^3 inherits from H^2 the master and achieves an optimal dual solution $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3}, \boldsymbol{h^3})$ of cost LB_3.

5.4.2 Relaxation $\overline{LF2}$

Relaxation $\overline{LF2}$ is obtained by adding SC, SR3, and SR5 inequalities to $LF2$

$$
\begin{aligned}
(\overline{LF2}) \quad z(\overline{LF2}) = \min & \sum_{k\in\mathscr{H}} c_k y_k && && (5.34)\\
s.t. & \sum_{k\in\mathscr{H}\,:\,i\in V(\Omega_k)} y_k = 1, && i \in V, && (5.35)\\
& \sum_{k\in\mathscr{H}} y_k \le m, && && (5.36)\\
& \sum_{k\in\mathscr{H}} \rho_k(S) y_k \ge 2k(S), && S \in \mathscr{S}, && (5.37)\\
& \sum_{k\in\mathscr{H}} \varphi_k(C) y_k \le \varphi_0(C), && C \in \mathscr{C}, && (5.38)\\
& y_k \ge 0, && k \in \mathscr{H}, && (5.39)
\end{aligned}
$$

where $\rho_k(S)$ and $\varphi_k(C)$ of constraints (5.37) and (5.38) are computed as $\rho_k(S) = |E(\Omega_k)\cap\delta(S)|$ and $\varphi_k(C) = \lfloor\frac{|V(\Omega_k)\cap C|}{\eta}\rfloor$.

Let $(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{g})$ be the dual variables of $\overline{LF2}$, where $\boldsymbol{w} = (w_0, w_1, \dots, w_n)$ and $w_0 \in \mathbb{R}_-$ is associated with constraint (5.36), $(w_1, \dots, w_n) \in \mathbb{R}^n$ with constraints (5.35), $\boldsymbol{v} \in \mathbb{R}_+^{|\mathscr{S}|}$ with constraints (5.37), and $\boldsymbol{g} \in \mathbb{R}_-^{|\mathscr{C}|}$ with constraints (5.38).

In §5.6, we describe a CCG procedure, called H^4, to solve $\overline{LF2}$. We denote by $\boldsymbol{x^4}$ and $(\boldsymbol{w^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$ the optimal primal and dual solutions, respectively, of $\overline{LF2}$ of cost LB_4 achieved by H^4.

5.4.3 Using Multiple Dual Solutions to Improve Procedures H^2, H^3 and H^4

In this section, we describe two simple methods to improve the lower bounds and the computing times of procedures H^2, H^3 and H^4.

The method used in H^2 and H^3 is based on the following simple observation which applies to any problem with binary variables. Consider the following problem with n variables and m constraints

$$
\begin{aligned}
(F) \quad z(F) = \min\ & \boldsymbol{cx}\\
s.t.\ & \boldsymbol{Ax} = \boldsymbol{b},\\
& \boldsymbol{x} \in \{0,1\}^n.
\end{aligned}
$$

Let us denote by LF the LP-relaxation of problem F and by D the dual of LF, and let $z(F)$, $z(LF)$ and $z(D)$ be the optimal solution costs of F, LF and D, respectively. We assume to know a valid upper bound z_{UB} on $z(F)$.

Observation 1. Let $\boldsymbol{w'}$ be a feasible (nonnecessarily optimal) D solution of cost $z'(D)$. Because any optimal F solution $\boldsymbol{x^*}$ satisfies $z(F) = z'(D) + \sum_{j \in J} c'_j$, where $c'_j = c_j - \boldsymbol{w' a}_j$, $J = \{j : x^*_j = 1, 1 \le j \le n\}$ is the reduced cost of variable x_j and a_j is the jth column vector of matrix $\boldsymbol{A}$, then any variable x_j such that $z'(D)+c'_j > z_{UB}$ cannot be in any optimal solution of cost less than or equal to z_{UB} and can be removed from F. The resulting problem F' has the same optimal solutions of F. Let LF' be the LP-relaxation of F', and let $z(LF')$ be its optimal solution cost. It follows that $z(LF') \ge z(LF)$, and such inequality can be strict if $c'_j > z_{UB} - z(LF)$ for some variable x_j of the optimal basis of LF. In practice, it might happen that $z(LF) < z(LF') = z(F)$, as shown in the following example.

Example 3. Consider the following set partitioning problem F

$$\begin{array}{llllllll}
\min = & x_1 & +x_2 & +x_3 & +4x_4 & +3x_5 & +4.5x_6 & \\
s.t. & x_1 & & & +x_4 & +x_5 & +x_6 & = 1, \\
 & & x_2 & & +x_4 & +x_5 & +x_6 & = 1, \\
 & & & x_3 & +x_4 & & +x_6 & = 1, \\
 & x_1 & +x_2 & & & +x_5 & & = 1, \\
 & x_1 & & +x_3 & & & +x_6 & = 1, \\
 & & x_2 & +x_3 & & & & = 1, \\
 & & \multicolumn{5}{l}{x_j \in \{0,1\},\ j = 1, \ldots, 6.} &
\end{array}$$

A valid upper bound is $z_{UB} = 4.5$, whereas the optimal integer solution cost of F is $z(F) = 4$ corresponding to $x_3 = 1$, $x_5 = 1$, $x_j = 0, j \ne 3, 5$. The optimal LF solution cost is $z(LF) = 2$ corresponding to $\boldsymbol{x} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0)$. Consider the dual solution $\boldsymbol{w} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ of cost $z(D) = 2$; the vector $\boldsymbol{c'}$ of the reduced costs with respect to $\boldsymbol{w}$ is $\boldsymbol{c'} = (0, 0, 0, 3, 2, \frac{19}{6})$.

According to Observation 1, we can remove variables x_4 and x_6 from F as $z(D) + c'_4 > z_{UB}$ and $z(D) + c'_6 > z_{UB}$. Thus, we derive problem F' that involves variables x_1, x_2, x_3 and x_5 only.

An optimal solution of LF' is $\boldsymbol{x} = (0, 0, 1, 0, 1, 0)$ of cost $z(LF') = 4$, which is also an optimal integer solution of F; that is $z(LF) < z(LF') = z(F)$. $\square$

Observation 1 is used by procedures H^2 and H^3 for solving the pricing problem. Procedure H^k, $k = 2, 3$, uses the dual solution $(\boldsymbol{u^{k-1}}, \boldsymbol{v^{k-1}}, \boldsymbol{g^{k-1}})$ produced by H^{k-1} to reject any route of negative reduced cost with respect to the incumbent dual solution of problem $\overline{LRF1}$ but having reduced cost with respect to $(\boldsymbol{u^{k-1}}, \boldsymbol{v^{k-1}}, \boldsymbol{g^{k-1}})$ greater than $z_{UB} - LB_{k-1}$, where z_{UB} is a valid upper bound to the MTVRP that can be computed by running any of the heuristic algorithms available from the literature (hereafter, we assume to know such upper bound on the MTVRP).

In practice, this method improves both the final lower bound LB_k achieved by H^k and the computing time of H^k, avoiding the generation of routes of negative reduced cost that cannot be in any optimal integer solution.

To improve the lower bound value as well as the computing time of H^4, we use the following observation.

Observation 2. Let d^3_ℓ be the reduced cost of route $\ell \in \mathscr{R}$ with respect to the dual solution $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3}, \boldsymbol{h^3})$ of $\overline{LRF1}$ of cost LB_3 achieved by H^3. Let $z(F2)$ be the cost of an optimal $F2$ solution $\boldsymbol{y^*}$, and let $Y = \{k \in \mathscr{H} \,:\, y^*_k = 1\}$. The following inequality holds $z(F2) \geq LB_3 + \sum_{k \in Y} d^3(\Omega_k)$, where $d^3(\Omega_k) = \sum_{\ell \in \Omega_k} d^3_\ell$. Thus, any schedule $k \in \mathscr{H}$ such that $d^3(\Omega_k) > z_{UB} - LB_3$ cannot be in any optimal solution and can be removed from $F2$.

Observation 2 is used in procedure H^4 to reject any schedule Ω_k of negative reduced cost with respect to the dual solution of the master such that $d^3(\Omega_k) > z_{UB} - LB_3$.

5.5 Bounding Procedures H^1, H^2, and H^3 Based on Relaxation $\overline{LRF1}$

In this section, we describe three bounding procedures H^1, H^2 and H^3 to derive lower bounds LB_1, LB_2 and LB_3, respectively. Procedures H^1 and H^2 use a method, called CCG, proposed by Baldacci et al. [2008] for the CVRP and used in Chapter 4 for solving the VRPTW and the CVRP. Procedure H^3 is a CCG method, based on the simplex, that uses a new strategy to solve the pricing problem.

5.5.1 Algorithm CCG

This algorithm differs from standard CCG methods based on the simplex as it uses a dual-ascent heuristic to find a near-optimal dual solution of the master problem. The main advantages of this method are

1. it is faster than simplex based methods;
2. it is not affected by the typical degeneration of the simplex;
3. the dual solution produced does not correspond to a dual basic solution of the master, thus making the pricing problem easier to solve.

The Dual Heuristic to Solve the Master Problem. The master problem of $\overline{LRF1}$ with SC and WSR3 inequalities is defined by a subset $\bar{\mathscr{R}} \subseteq \mathscr{R}$ and the sets $\mathscr{S}$ and $\mathscr{C}$ of SC and WSR3 inequalities. The initializations of the sets $\bar{\mathscr{R}}$, $\mathscr{S}$ and $\mathscr{C}$ are described in §§5.5.2 and 5.5.4. The dual-ascent heuristic to compute a feasible dual solution $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ of the master problem corresponds to the method described in §4.5.1 and is based on the following theorem.

Theorem 5.1. *Let us associate penalties $\lambda_i \in \mathbb{R}$, $i \in V$, with constraints (5.29), $\mu_S \in \mathbb{R}_+$, $S \in \mathscr{S}$, with SC constraints (5.30), and $\omega_C \in \mathbb{R}_-$, $C \in \mathscr{C}$, with constraints (5.31) in the form of WSR3 inequalities. For each route R_ℓ ($\ell \in \mathscr{R}$), we indicate with $\bar{c}_\ell$ its reduced cost with respect to $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})$ (i.e., $\bar{c}_\ell = c_\ell - \sum_{i \in R_\ell} \lambda_i - \sum_{S \in \mathscr{S}} \rho_\ell(S)\mu_S - \sum_{C \in \mathscr{C}} \varphi_\ell(C)\omega_C$). For each $i \in V$, let us compute*

$$\beta_i = q_i \min_{\ell \in \widehat{\mathscr{R}}\,:\, i \in R_\ell} \left\{ \frac{\bar{c}_\ell}{\sum_{i \in R_\ell} q_i} \right\}.$$

A feasible dual solution $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g})$ of the master of cost $z(\overline{LRF1}(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}))$ is obtained by setting

$$u_i = \beta_i + \lambda_i,\ i \in V, \quad v_S = \mu_S,\ S \in \mathscr{S}, \quad \text{and} \quad g_C = \omega_C,\ C \in \mathscr{C}.$$

Proof. See §3.5.2. □

A near-optimal dual solution $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ of the current master problem is obtained by an iterative procedure that performs a predefined number $Maxit2$ of subgradient iterations to solve the problem

$$LCCG = \max_{(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})} \{ z(\overline{LRF1}(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})) \}.$$

We denote by $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ the near-optimal dual solution of $\overline{LRF1}$ achieved after $Maxit2$ iterations and by $\bar{\boldsymbol{x}}$ the corresponding nonnecessarily feasible solution of $\overline{LRF1}$.

Adding violated WSR3 inequalities and solving the pricing problem. After computing the master dual solution $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$, CCG adds, to the master, the largest subset $\mathscr{C}'$ of at most $\Delta(\mathscr{C})$ WSR3 inequalities most violated by $\bar{\boldsymbol{x}}$, where $\Delta(\mathscr{C})$ is a given parameter. In our computational results, we set $\Delta(\mathscr{C}) = 100$. The set $\mathscr{S}$ of SC inequalities is generated at the beginning of CCG, as described in Baldacci et al. [2008], and is not changed.

Moreover, CCG generates a subset $\mathscr{N} \subseteq \mathscr{R}$ of routes of negative reduced cost with respect to $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$. If $\mathscr{N} \neq \varnothing$, then CCG sets $\widehat{\mathscr{R}} = \widehat{\mathscr{R}} \cup \mathscr{N}$; otherwise, $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ is a feasible dual solution.

CCG terminates after $Maxit1$ macro iterations and provides an $\overline{LRF1}$ dual solution, $(\boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{g}^*)$, of cost $LCCG$ corresponding to the penalty vector $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\omega}^*)$.

5.5.2 Bounding Procedure H^1

Procedure H^1 enlarges the route set $\mathscr{R}$ with an extension of the q-routes introduced by Christofides et al. [1981b]. The initial route set $\bar{\mathscr{R}}$ of the master problem contains all single-customer routes $(0, i, 0)$, $i \in V$. We initialize $\mathscr{C} = \varnothing$ and $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}) = (\mathbf{0}, \mathbf{0}, \mathbf{0})$.

At a given macro iteration, the set $\mathscr{N}$ of routes of negative reduced cost with respect to the dual solution $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ is computed as follows. Define the modified arc cost

$$\bar{\alpha}_{ij} = \alpha_{ij} - \frac{1}{2}(\bar{u}_i + \bar{u}_j) - \sum_{S \in \mathscr{S} \,:\, \{i,j\} \in \delta(S)} \bar{v}_S, \quad \text{for each edge } \{i,j\} \in E,$$

with respect to $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$.

A (q, j, i)-path is a nonnecessarily elementary path that starts from the depot and visits a subset of customers (once or more) such that their total demand is q and the last two customers visited are j and i in this order. A (q, j, i)-path is defined for any load q $(q_j + q_i \leq q \leq Q)$, and for any pair of vertices $j, i \in V', j \neq i$. A $(q, j, 0)$-route (or simply a q-route) is a $(q, j, 0)$-path. Let $f(q, j, i)$ be the cost of a least-cost (q, j, i)-path using the modified arc costs $\bar{\alpha}_{ij}$, and let $\mathscr{C}_{ij} = \{C \in \mathscr{C} : \{i, j\} \in E(C)\}$. Functions $f(q, j, i)$ can be computed with the following dynamic programming (DP) recursion

$$f(q, j, i) = \min_{k \in V' \setminus \{i\}} \Big\{ f(q - q_i, k, j) + \bar{\alpha}_{ji} - \sum_{C \in \mathscr{C}_{ji} \setminus \mathscr{C}_{kj}} \bar{g}_C \Big\},$$

$$q = q_j + q_i, \ldots, Q, \quad j, i \in V', j \neq i, j \neq 0.$$

Functions $f(q, j, i)$ are initialized as $f(q_i, 0, i) = \bar{\alpha}_{0i}$, $i \in V$.

The set $\mathscr{N}$ contains any q-route of negative reduced cost corresponding to a least-cost q-route visiting customer $j \in V$ as the last customer before arriving at the depot.

At the end, procedure H^1 sets $(\boldsymbol{u}^{\mathbf{1}}, \boldsymbol{v}^{\mathbf{1}}, \boldsymbol{g}^{\mathbf{1}}) = (\boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{g}^*)$, $(\boldsymbol{\lambda}^{\mathbf{1}}, \boldsymbol{\mu}^{\mathbf{1}}, \boldsymbol{\omega}^{\mathbf{1}}) = (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\omega}^*)$, and $LB_1 = LCCG$.

5.5.3 Route Generation Algorithm GenR

In this section, we describe the two-phase algorithm GenR used by bounding procedure H^2 to initialize the master problem and to solve the pricing problem. Moreover, the second phase of GenR is used by H^3 to solve the pricing problem and by the exact method to generate all routes of reduced cost, with respect to $(\boldsymbol{u}^{\mathbf{3}}, \boldsymbol{v}^{\mathbf{3}}, \boldsymbol{g}^{\mathbf{3}}, \boldsymbol{h}^{\mathbf{3}})$, less than $z_{UB} - LB_3$. Algorithm GenR generates elementary routes and extends the method proposed by Baldacci et al. [2008] for the CVRP by introducing new bounding functions to reduce the state-space graph. GenR is based on the following observations.

Let $V(P) \subseteq V$ be the subset of customers visited by path P, and let $E(P)$ be the edges traversed by path P. Let $\mathscr{P}$ be the set of all elementary paths of minimum cost from the depot such that $q(P) \leq \frac{Q}{2} + q_{\sigma(P)}$, for each path $P \in \mathscr{P}$, where $q(P) = \sum_{i \in V(P)} q_i$, $\tau(P) = \sum_{\{i,j\} \in E(P)} \tau_{ij}$, and $\sigma(P)$ represent the load, the duration and the terminal customer of path P, respectively. We denote by $\pi(P)$ the immediate predecessor of $\sigma(P)$ in P.

Every route passing through customer $i \in V$ can be obtained by combining a pair of paths $P, \bar{P} \in \mathscr{P}$ such that

$$\sigma(P) = \sigma(\bar{P}) = i, \quad V(P) \cap V(\bar{P}) = \{i\}, \quad \tau(P) + \tau(\bar{P}) \leq T, \quad q(P) + q(\bar{P}) \leq Q + q_i. \tag{5.40}$$

Given a nonnecessarily feasible dual solution $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$ of $\overline{LRF1}$, a feasible dual solution $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$ of $\overline{LRF1}$ and four parameters $\Delta(\mathscr{P})$, $\Delta(\mathscr{R})$, γ and γ', `GenR` generates the largest subset $\mathscr{B}$ of routes such that $|\mathscr{B}| \leq \Delta(\mathscr{R})$, $\tilde{d}_\ell \leq \gamma$ and $d'_\ell \leq \gamma'$, for each route $\ell \in \mathscr{B}$, where $\tilde{d}_\ell$ and d'_ℓ are the reduced costs of route ℓ with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$ and $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'})$, respectively. In the first phase, `GenR` generates the path set $\mathscr{P}$ such that $|\mathscr{P}| \leq \Delta(\mathscr{P})$, where $\Delta(\mathscr{P})$ is a parameter that limits the cardinality of the set $\mathscr{P}$ and is imposed for memory limits. In the second phase, `GenR` combines the paths of $\mathscr{P}$ to provide the subset $\mathscr{B} \subseteq \mathscr{R}$. Parameters $\Delta(\mathscr{R})$, γ and γ' are defined according to the type of subset $\mathscr{B}$ that must be generated.

Define the modified path cost $\tilde{d}(P)$ with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ as follows

$$\tilde{d}(P) = \sum_{\{i,j\} \in E(P)} \tilde{\alpha}_{ij}, \tag{5.41}$$

where $\tilde{\alpha}_{ij} = \alpha_{ij} - \frac{1}{2}(\tilde{u}_i + \tilde{u}_j) - \sum_{S \in \mathscr{S} : \{i,j\} \in \delta(S)} \tilde{v}_S$. Let $lb(P)$ be a lower bound on the reduced cost with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$ of any route containing path P. For a given path $P \in \mathscr{P}$, let $\sigma(P) = i$. The lower bound $lb(P)$ is computed as

$$lb(P) = \tilde{d}(P) - \sum_{C \in \mathscr{C}} \left\lceil \frac{|E(P) \cap E(C)|}{2} \right\rceil \tilde{g}_C + \min_{j \in V \setminus V(P),\, q_i \leq q' \leq Q - q(P) + q_i} \left\{ f(q', j, i) + \tilde{g}_{\pi(P)ij} \right\},$$

where $\tilde{g}_{\pi(P)ij}$ is the dual of the WSR3 inequality $\{\pi(P), i, j\} \in \mathscr{C}$ (we assume $\tilde{g}_{\pi(P)ij} = 0$ if $\{\pi(P), i, j\} \notin \mathscr{C}$).

The two phases of `GenR` are shortly described as follows. For a more detailed description of `GenR` see Baldacci et al. [2008].

5.5.3.1 Phase 1 of `GenR`.

Phase 1 generates a sequence $\mathscr{P}$ of paths $(P^1, \ldots, P^k)$, with $k \leq \Delta(\mathscr{P})$, such that $lb(P^1) \leq \ldots \leq lb(P^k) \leq \gamma$. If Phase 1 terminates with $|\mathscr{P}| = \Delta(\mathscr{P})$, the entire algorithm terminates prematurely as there is no guarantee that all routes of reduced cost less than or equal to γ are generated by combining the pairs $P, \bar{P} \in \mathscr{P}$.

5.5.3.2 Phase 2 of `GenR`.

Phase 2 is executed only if $|\mathscr{P}| < \Delta(\mathscr{P})$. Consider a pair of paths $(P, \bar{P}) \in \mathscr{P}$ satisfying conditions (5.40). Let $\ell \in \mathscr{R}$ be the index of the route resulting from $(P, \bar{P})$ of reduced cost $\tilde{d}_\ell$ with respect to $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}})$. From (5.41), we have $\tilde{d}_\ell = \tilde{d}(P) + \tilde{d}(\bar{P}) - \sum_{C\in\mathscr{C}} \varphi_\ell(C)\tilde{g}_C$. As $\tilde{\boldsymbol{g}} \leq 0$, then $\tilde{d}(P) + \tilde{d}(\bar{P}) \leq \tilde{d}_\ell$, so only pairs of paths $(P, \bar{P})$ such that $\tilde{d}(P) + \tilde{d}(\bar{P}) \leq \gamma$ can generate routes of reduced cost less than or equal to γ.

Phase 2 dynamically generates a sequence of path pairs $(P^{r_1}, \bar{P}^{s_1})$, $\ldots$, $(P^{r_k}, \bar{P}^{s_k})$, $\ldots$, $(P^{r_h}, \bar{P}^{s_h})$, where each path pair satisfies conditions (5.40) and $\tilde{d}(P^{r_1}) + \tilde{d}(\bar{P}^{s_1}) \leq \ldots \leq \tilde{d}(P^{r_k}) + \tilde{d}(\bar{P}^{s_k}) \leq \ldots \leq \tilde{d}(P^{r_h}) + \tilde{d}(\bar{P}^{s_h}) \leq \gamma$. The pool $\mathscr{B}$ of routes generated contains any route $\ell \in \mathscr{R}$ resulting from the pairs of paths in the sequence such that $\tau_\ell \leq T$, $\tilde{d}_\ell \leq \gamma$, $d'_\ell \leq \gamma'$ and route ℓ is not dominated by any other route ℓ' previously generated (i.e., route ℓ' dominates route ℓ if $R_{\ell'} = R_\ell$ and $\tilde{d}_{\ell'} \leq \tilde{d}_\ell$).

5.5.4 Bounding Procedure H^2

Procedure H^2 is a straightforward implementation of CCG, where $\mathscr{R}$ is the set of elementary routes. The initial route set $\bar{\mathscr{R}}$ is obtained by executing `GenR` by setting $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'}) = (\boldsymbol{u^1}, \boldsymbol{v^1}, \boldsymbol{g^1})$, $\Delta(\mathscr{P}) = 10^7$, $\Delta(\mathscr{R}) = 10^4$, and $\gamma = \gamma' = z_{UB} - LB_1$, and adding all single-customer routes to $\bar{\mathscr{R}} = \mathscr{B}$. Procedure H^2 starts with the same sets $\mathscr{S}$ and $\mathscr{C}$ of SC and WSR3 inequalities resulting from H^1. We initialize $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}) = (\boldsymbol{\lambda^1}, \boldsymbol{\mu^1}, \boldsymbol{\omega^1})$.

At each macro iteration, a set $\mathscr{N}$ of routes of negative reduced cost with respect to $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ is given by $\mathscr{N} = \mathscr{B}$, where the set $\mathscr{B}$ is generated by `GenR` setting $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$, $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'}) = (\boldsymbol{u^1}, \boldsymbol{v^1}, \boldsymbol{g^1})$, $\Delta(\mathscr{P}) = 10^7$, $\Delta(\mathscr{R}) = 150$, $\gamma = 0$, and $\gamma' = z_{UB} - LB_1$.

At the end, H^2 sets $(\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2}) = (\boldsymbol{u^*}, \boldsymbol{v^*}, \boldsymbol{g^*})$, $(\boldsymbol{\lambda^2}, \boldsymbol{\mu^2}, \boldsymbol{\omega^2}) = (\boldsymbol{\lambda^*}, \boldsymbol{\mu^*}, \boldsymbol{\omega^*})$, and $LB_2 = LCCG$.

5.5.5 Generating the Path Set $\mathscr{P}^2$ and the Route Set $\mathscr{R}^2$

After the execution of H^2, we try to generate the route set $\mathscr{R}^2$ of all routes of reduced cost with respect to $(\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2})$ less than or equal to $z_{UB} - LB_2$. We call `GenR` by setting $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2})$, $(\boldsymbol{u'}, \boldsymbol{v'}, \boldsymbol{g'}) = (\boldsymbol{u^1}, \boldsymbol{v^1}, \boldsymbol{g^1})$, $\Delta(\mathscr{P}) = 10^7$, $\Delta(\mathscr{R}) = 5{\cdot}10^6$, $\gamma = z_{UB} - LB_2$, and $\gamma' = z_{UB} - LB_1$. We denote by $\mathscr{P}^2$ and $\mathscr{R}^2$ the path set $\mathscr{P}$ and the route set $\mathscr{B}$ generated by `GenR`, respectively. Moreover, we denote by $d^2(P)$

the modified cost of path $P \in \mathscr{P}^2$ with respect to $(\boldsymbol{u^2}, \boldsymbol{v^2})$ computed according to expression (5.41) by replacing $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ with $(\boldsymbol{u^2}, \boldsymbol{v^2})$.

We define $\mathscr{P}^2$ to be *optimal* if $|\mathscr{P}^2| < \Delta(\mathscr{P})$ because it contains any path generating any route of any optimal MTVRP solution. We define $\mathscr{R}^2$ to be *optimal* if $\mathscr{P}^2$ is optimal and $|\mathscr{R}^2| < \Delta(\mathscr{R})$.

Because H^3 and the exact algorithm require that $\mathscr{P}^2$ be optimal, whenever $\mathscr{P}^2$ is not optimal (i.e., $|\mathscr{P}^2| = \Delta(\mathscr{P})$), H^3 is not executed and the entire procedure terminates prematurely.

5.5.6 Bounding Procedure H^3

Procedure H^3 is a column-and-cut-generation procedure, based on the simplex algorithm, to solve $\overline{LRF1}$ with SC, SR3, SR5, and WT inequalities. H^3 differs from classical column-and-cut-generation algorithms because of the method used to solve the pricing problem.

H^3 uses sets $\mathscr{P}^2$ and $\mathscr{R}^2$, generated at the end of H^2, to generate routes of negative reduced cost. The route set $\bar{\mathscr{R}}$ of the master problem of H^3 is generated by extracting, from $\mathscr{R}^2$, the largest subset $\bar{\mathscr{R}}$ of routes of minimum reduced cost with respect to $(\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2})$ such that $|\widehat{\mathscr{R}}| \leq \Delta(\mathscr{B})$ and by adding all single-customer routes to $\bar{\mathscr{R}}$. The set $\mathscr{S}$ is set equal to the set of SC inequalities used in H^1 and H^2, and $\mathscr{W}$ and $\mathscr{C}$ are defined as $\mathscr{W} = \varnothing$, $\mathscr{C} = \varnothing$, where the set $\mathscr{C} = \mathscr{C}^3 \cup \mathscr{C}^5$ represents SR3 and SR5 inequalities.

Let $\bar{\boldsymbol{x}}$ and $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}}, \bar{\boldsymbol{h}})$ be, respectively, the primal and dual solutions of the master problem achieved at a given iteration. A set $\mathscr{N}$ of at most $\Delta(\mathscr{N})$ negative reduced cost routes with respect to $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}}, \bar{\boldsymbol{h}})$ is generated with the following algorithm.

1. Extract, from $\mathscr{R}^2$, the largest subset $\mathscr{N}$ of at most $\Delta(\mathscr{N})$ routes having the largest negative reduced cost, and add them to $\bar{\mathscr{R}}$. If $\mathscr{N} = \varnothing$ and $\mathscr{R}^2$ is not optimal, continue with Step 2; otherwise, Stop.

2. Compute the modified cost $\bar{d}(P)$, $P \in \mathscr{P}^2$, with respect to $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}})$ according to (5.41) by replacing $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ with $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}})$. Using the method described in §5.5.3.2, combine any pair of paths $P, P' \in \mathscr{P}^2$ such that $d^2(P) + d^2(P') \leq z_{UB} - LB_2$ and $\bar{d}(P) + \bar{d}(P') \leq 0$ to derive the set $\mathscr{N}$.

At each iteration, H^3 adds, to $\mathscr{C}$, a set of at most $\Delta(\mathscr{C})$ most violated SR3 and SR5 inequalities and, to $\mathscr{W}$, a set of at most $\Delta(\mathscr{W})$ WT inequalities most violated by $\bar{\boldsymbol{x}}$. Procedure H^3 ends whenever $\mathscr{N} = \varnothing$ and no cuts are added, and it achieves an $\overline{LRF1}$ dual solution $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3}, \boldsymbol{h^3})$ of cost LB_3.

5.5.7 Generating the Route Set $\mathscr{R}^3$ and Solving $F1$

After the execution of H^3, we try to generate the largest set $\mathscr{R}^3$ of routes such that $d^3_\ell \leq z_{UB} - LB_3$, $\ell \in \mathscr{R}^3$, and $|\mathscr{R}^3| \leq \Delta(\mathscr{R})$, where d^3_ℓ is the reduced cost of route ℓ with respect to $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3}, \boldsymbol{h^3})$. In generating $\mathscr{R}^3$ we have two cases

a. $\mathscr{R}^2$ is optimal. We set $\mathscr{R}^3 = \{\ell \in \mathscr{R}^2 : d^3_\ell \leq z_{UB} - LB_3\}$ and define $\mathscr{R}^3$ as *optimal.*

b. $\mathscr{R}^2$ is not optimal. Recall that in this case $\mathscr{P}^2$ is optimal, so $\mathscr{P}^2$ allows us to generate the routes of any optimal MTVRP solution. We compute the reduced cost $d^3(P)$ of each path $P \in \mathscr{P}^2$, with respect to $(\boldsymbol{u^3}, \boldsymbol{v^3})$, according to expression (5.41) by replacing $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ with $(\boldsymbol{u^3}, \boldsymbol{v^3})$. Then, we combine the path pairs of the set $\mathscr{P}^2$, by using the method of §5.5.3.2, to derive the set $\mathscr{R}^3$ of all routes such that $d^2_\ell \leq z_{UB} - LB_3$ and $d^3_\ell \leq z_{UB} - LB_3$, for each route $\ell \in \mathscr{R}^3$, and $|\mathscr{R}^3| \leq \Delta(\mathscr{R})$. If $|\mathscr{R}^3| < \Delta(\mathscr{R})$, then $\mathscr{R}^3$ is defined *optimal.*

If $|\mathscr{R}^3|$ is small (say, $|\mathscr{R}^3| \leq 5,000$), we find an optimal MTVRP solution by solving the reduced problem $F1$, obtained by replacing route set $\mathscr{R}$ with $\mathscr{R}^3$, through an IP solver. Otherwise, we execute bounding procedure H^4 (described in §5.6).

5.6 Bounding Procedure H^4 Based on Relaxation $\overline{LF2}$

Procedure H^4 is a column-and-cut generation method, based on the simplex, to solve relaxation $\overline{LF2}$.

The initial sets $\mathscr{S}$ and $\mathscr{C}$ of SC, SR3 and SR5 inequalities are those resulting at the end of H^3. The initial master problem, $\bar{\mathscr{H}} \subseteq \mathscr{H}$, is defined as follows. Execute procedure `GenSched` (described in §5.7) to derive the schedule set $\mathscr{D}$ containing at most $\Delta(\mathscr{D})$ (where $\Delta(\mathscr{D})$ is a given parameter) schedules such that $c^3_k \leq z_{UB} - LB_3$, $k \in \mathscr{D}$, where c^3_k is the reduced cost of schedule k with respect to $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3}, \boldsymbol{h^3})$ computed as $c^3_k = \sum_{\ell \in \Omega_k} d^3_\ell$. Set $\widehat{\mathscr{H}} = \mathscr{D}$, and add to $\widehat{\mathscr{H}}$ a schedule composed of a single-customer route for every customer i not covered by any schedule of $\mathscr{D}$.

At each iteration, `GenSched` generates a set $\mathscr{N}$ of at most $\Delta(\mathscr{D})$ schedules of negative reduced cost with respect to the dual solution $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{g}})$ of the master and such that $c^3_k \leq z_{UB} - LB_3$, for each schedule $k \in \mathscr{N}$. Moreover, the subset $\mathscr{C}'$ of the $\Delta(\mathscr{C})$ SR3 and SR5 inequalities most violated by the current master solution are added to $\mathscr{C}$. In our computational experiments, we used $\Delta(\mathscr{D}) = 10^4$ in generating $\mathscr{D}$, $\Delta(\mathscr{D}) = 150$ in generating $\mathscr{N}$, and $\Delta(\mathscr{C}) = 100$ in generating $\mathscr{C}'$.

Procedure H^4 terminates if $\mathscr{N} = \varnothing$ and $\mathscr{C}' = \varnothing$ and provides a dual solution $(\boldsymbol{w^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$ of cost LB_4.

5.7 Algorithm `GenSched` to Generate Vehicle Schedules

In this section, we describe procedure `GenSched` to generate the schedule sets $\bar{\mathscr{H}}$ and $\mathscr{N}$, which are required by bounding procedure H^4, and the schedule subset $\mathscr{H}^4 \subseteq \mathscr{H}$, which is required by the exact method described in §5.8.

Given a dual solution $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{g}})$ of $\overline{LF2}$ and two parameters $\Delta(\mathscr{D})$ and γ, algorithm `GenSched` combines the routes of set $\mathscr{R}^3$ to produce the largest subset $\mathscr{D}$ of schedules of minimum reduced costs with respect to $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{g}})$ such that $|\mathscr{D}| \leq \Delta(\mathscr{D})$, $c^3_k \leq z_{UB} - LB_3$, and $\hat{c}_k \leq \gamma$, for each schedule $k \in \mathscr{D}$, where $\hat{c}_k$ is the reduced cost of schedule k with respect to $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{g}})$.

To generate the schedule sets $\bar{\mathscr{H}}$, $\mathscr{N}$ and $\mathscr{H}^4$, `GenSched` requires the following parameter settings

- *The schedule set $\bar{\mathscr{H}}$ defining the initial master problem of H^4.* Define $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{g}}) = (\mathbf{0}, \mathbf{0}, \mathbf{0})$ and $\gamma = z_{UB}$.
- *The set $\mathscr{N}$ of schedules of negative reduced costs in H^4.* Define $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{g}})$ as the dual of the current master problem of H^4 and $\gamma = 0$.
- *The schedule subset $\mathscr{H}^4 \subseteq \mathscr{H}$ of all the schedules of any optimal MTVRP solution.* Define $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{g}}) = (\boldsymbol{w^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$ and $\gamma = z_{UB} - LB_4$.

Procedure `GenSched` is based on the following proposition.

Proposition 4. Let $\hat{d}_\ell = \sum_{\{i,j\}\in E(R_\ell)}(\alpha_{ij} - \frac{1}{2}(\hat{w}_i + \hat{w}_j) - \sum_{S\in\mathscr{S}\,:\,\{i,j\}\in\delta(S)} \hat{v}_S) - \sum_{C\in\mathscr{C}} \varphi_\ell(C)\hat{g}_C$, $\ell \in \mathscr{R}^3$, and let $\hat{d}(\Omega_k) = \sum_{\ell\in\Omega_k} \hat{d}_\ell$, $k \in \mathscr{H}$. The reduced cost $\hat{c}_k$ of schedule $k \in \mathscr{H}$ with respect to $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{g}})$ satisfies the inequality $\hat{c}_k \geq \hat{d}(\Omega_k)$.

Proof. This happens as $w_0 \leq 0$, $\hat{\boldsymbol{g}} \leq \mathbf{0}$, and $\varphi_k(C) \geq \sum_{\ell\in\Omega_k} \varphi_\ell(C)$, for any $C \in \mathscr{C}$ and any $k \in \mathscr{H}$.□

Algorithm `GenSched` is a two-phase procedure. In the first phase, it computes the reduced costs $\hat{d}_\ell$, $\ell \in \mathscr{R}^3$, and orders the routes of the set $\mathscr{R}^3$ so that $\hat{d}_{\ell_1} \leq \hat{d}_{\ell_2} \leq \ldots \leq \hat{d}_{|\mathscr{R}^3|}$. The second phase is an iterative procedure that combines the routes $\mathscr{R}^3$ in schedules. At iteration j, an attempt is made to add the jth route of the ordered set $\mathscr{R}^3$ to the schedules of a temporary schedule set POOL containing all the schedules composed of the first $j-1$ routes $\ell_1, \ell_2, \ldots, \ell_{j-1}$ of $\mathscr{R}^3$.

We assume that POOL is composed of *undominated* schedules, where schedule k' is *dominated* by schedule k'' if $V(\Omega_{k''}) = V(\Omega_{k'})$, $\hat{d}(\Omega_{k''}) \leq \hat{d}(\Omega_{k'})$, and $\tau(\Omega_{k''}) \leq \tau(\Omega_{k'})$. A schedule k' that is obtained by adding route ℓ_j to a schedule $k \in$ POOL becomes a new member of POOL only if $\hat{d}(\Omega_{k'}) \leq \gamma$, k' is not dominated by any other schedule $k'' \in$ POOL, and $c^3(\Omega_{k'}) = \sum_{\ell\in\Omega_{k'}} d^3_\ell \leq z_{UB} - LB_3$.

We denote by p the number of schedules in POOL. The second phase of `GenSched` can be described as follows.

1. Initialize POOL $= \varnothing$, $\mathscr{D} = \varnothing$ and $p = 0$.

2. For $\ell = 1, \ldots, |\mathscr{R}^3|$ repeat Step 3.

3. *Expansion of the temporary schedule set* POOL *with route* $\ell \in \mathscr{R}^3$. For each schedule $k \in$ POOL such that $V(\Omega_k) \cap R_\ell = \varnothing$, $\tau(\Omega_k) + \tau_\ell \leq T$, $c^3(\Omega_k) + d^3_\ell \leq z_{UB} - LB_3$, and $\hat{d}(\Omega_k) + \hat{d}_\ell \leq \gamma$, repeat the following steps

   - Define the schedule $\Omega' = \Omega_k \cup \{\ell\}$ of duration $\tau(\Omega') = \tau(\Omega_k) + \tau_\ell$, and set $\hat{d}(\Omega') = \hat{d}(\Omega) + \hat{d}_\ell$.
   - If Ω' is not dominated by any other schedule $\Omega'' \in$ POOL, insert $\Omega' \in$ POOL and set $p = p + 1$. Remove any schedule Ω'' dominated by Ω' from POOL, and update p, accordingly.
   - Compute $\hat{c}(\Omega')$. If $\hat{c}(\Omega') \leq \gamma$, add Ω' to $\mathscr{D}$. If $|\mathscr{D}| = p$, Stop.

5.8 Description of the Exact Method to Solve the MTVRP

The exact method we propose consists of seven main steps and can be described as follows.

1. **Solving relaxation $\overline{\mathbf{LRF1}}$ with procedures $\mathbf{H^1}$ and $\mathbf{H^2}$.** Solve relaxation $\overline{LRF1}$ with SC and WSR3 inequalities, executing, in sequence, procedures H^1 and H^2. Let LB_2 be the cost of the dual solution $(\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2})$ achieved by H^2 (see §§5.5.2 and 5.5.4).

2. **Generating path set $\mathscr{P}^2$ and route set $\mathscr{R}^2$.** Generate path set $\mathscr{P}^2$ and set $\mathscr{R}^2$ of the routes of reduced cost with respect to $(\boldsymbol{u^2}, \boldsymbol{v^2}, \boldsymbol{g^2})$ less than or equal to $z_{UB} - LB_2$ (see §5.5.5). If $\mathscr{P}^2$ is not optimal, Stop (the algorithm terminates without providing any optimal solution).

3. **Solving relaxation $\overline{\mathbf{LRF1}}$ with procedure $\mathbf{H^3}$.** Solve relaxation $\overline{LRF1}$ with SC, SR3, SR5, and WT inequalities with procedure H^3 (see §5.5.6). Let LB_3 be the lower bound corresponding to the dual solution $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3}, \boldsymbol{h^3})$ of $\overline{LRF1}$ achieved by H^3.

4. **Generating the route set $\mathscr{R}^3$.** Generate the largest set $\mathscr{R}^3$ of the routes of reduced cost with respect to $(\boldsymbol{u^3}, \boldsymbol{v^3}, \boldsymbol{g^3}, \boldsymbol{h^3})$ less than or equal to $z_{UB} - LB_3$ and such that $|\mathscr{R}^3| \leq \Delta(\mathscr{R})$ (see §5.5.7). If $|\mathscr{R}^3| < \Delta(\mathscr{R})$, $\mathscr{R}^3$ contains the routes of any optimal solution of cost less than or equal to z_{UB} and $\mathscr{R}^3$ is defined *optimal*. Otherwise, $\mathscr{R}^3$ is defined *not-optimal*.

5. **Solving the reduced problem F1.** If $m = 1$, or if $m > 1$, $\mathscr{R}^3$ is optimal, and $|\mathscr{R}^3| \leq 5,000$, then find an optimal MTVRP solution by solving, with an IP solver, problem $F1$ by replacing the route set $\mathscr{R}$ with the route set $\mathscr{R}^3$, and Stop.

6. **Solving relaxation $\overline{\textbf{LF2}}$ with procedure $\textbf{H}^4$.** If $m > 1$, and either $\mathscr{R}^3$ is not optimal or $|\mathscr{R}^3| > 5,000$, then solve $\overline{LF2}$ with SC, SR3, and SR5 inequalities with H^4 (see §5.6). Let LB_4 be the lower bound corresponding to an optimal dual solution $(\boldsymbol{w^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$ of $\overline{LF2}$ obtained by H^4. If the optimal primal solution $\boldsymbol{y}$ of $\overline{LF2}$ is integer, Stop.

7. **Solving the reduced problem F2.** Call `GenSched` to generate the largest set of schedules $\mathscr{H}^4 \subseteq \mathscr{H}$, by setting $(\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{g}}) = (\boldsymbol{w^4}, \boldsymbol{v^4}, \boldsymbol{g^4})$, $\gamma = z_{UB} - LB_4$, $\Delta(\mathscr{D}) = 10^6$.
If $|\mathscr{H}^4| \geq \Delta(\mathscr{D})$, as there is no guarantee that $\mathscr{H}^4$ contains all schedules of any optimal MTVRP solution, the algorithm terminates prematurely without providing any optimal solution.
Otherwise (i.e., if $|\mathscr{H}^4| < \Delta(\mathscr{D})$) compute an optimal MTVRP solution by solving, with an IP solver, the integer problem $F2$ obtained by replacing the schedule set $\mathscr{H}$ with the subset $\mathscr{H}^4$ and by adding all the valid inequalities saturated at the end of H^4.

5.9 Computational Results

This section reports on the computational results of the exact method described in §5.8. All algorithms were coded in C and compiled with Visual Studio 2008. CPLEX 12.1 was used as the LP solver in H^3 and H^4 and as the IP solver in Steps 5 and 7 of the exact method. All tests were performed on a Sony Vaio P8400 laptop (Intel Core 2 Duo@2.26 GHz with 4 GB of RAM).

The exact algorithm was tested on a subset of the benchmark instances proposed in Taillard et al. [1996], which were used to test all heuristic algorithms in the literature. These instances were generated by starting from the graphs, demands and vehicle capacities of five CVRP problems (namely, CMT-1, CMT-2, CMT-3, CMT-11, and CMT-12) proposed in Christofides et al. [1979] and by varying the number of vehicles, m. For each problem and each value of m, two instances with different values of T were generated: in the first instance, $T = [\frac{1.05 z_{RT}}{m}]$, and in the second one, $T = [\frac{1.10 z_{RT}}{m}]$, where $[x]$ is the integer value nearest to x and z_{RT} is the CVRP solution cost obtained by Rochat and Taillard [1995]. All instances feature $x - y$ coordinates. The travel costs coincide with the travel times and are real values computed as Euclidean distances between the vertices. For an accurate description of the test instances see Taillard et al. [1996].

Table 5.1: Upper Bounds on the Test Instances

Inst	m	T	z_{UB}	T_{UB}	*Inst*	m	T	z_{UB}	T_{UB}	*Inst*	m	T	z_{UB}	T_{UB}
CMT-1	1	551	524.61	27	CMT-3	1	867	826.14	106	CMT-12	1	861	819.56	93
	2	275	533.00	27		2	434	826.14	105		2	430	819.56	95
	1	577	524.61	27		3	289	826.14	107		3	287	819.56	95
	2	289	529.85	28		4	217	828.73	105		4	215	819.56	95
	3	192	552.68	30		5	173	851.47	107		5	172	845.37	95
	4	144	546.29	28		6	145	839.90	172		1	902	819.56	95
CMT-2	1	877	835.26	49		1	909	826.14	106		2	451	819.56	93
	2	439	835.26	49		2	454	826.14	106		3	301	819.56	93
	3	292	835.26	49		3	303	826.14	105		4	225	819.56	96
	4	219	835.26	50		4	227	826.14	107		5	180	824.78	96
	5	175	835.80	50		5	182	833.15	109		6	150	825.36	96
	6	146	857.00	111		6	151	842.21	109					
	1	919	835.26	49	CMT-11	1	1,094	1,042.11	156					
	2	459	835.26	49		2	547	1,042.11	161					
	3	306	835.26	50		3	365	1,042.11	160					
	4	230	835.26	50		4	274	1,085.92	210					
	5	184	835.26	51		5	219	1,042.11	161					
	6	153	839.22	51		1	1,146	1,042.11	160					
	7	131	872.64	51		2	573	1,042.11	159					
						3	382	1,042.11	160					
						4	287	1,042.11	161					
						5	229	1,042.11	164					

The exact method described in §5.8 requires an upper bound z_{UB} (see Steps 2, 4 and 7). In our experiments, such upper bound was set equal to the best of three upper bounds: (a) the upper bound computed by the heuristic of Taillard et al. [1996] (sent us by private communication Taillard [2009]), (b) the upper bound reported in Olivera and Viera [2007], and (c) the cost of the CVRP solution found by the heuristic of Mester and Bräysy [2005] if the routes of such solution can be arranged so as to obtain a feasible MTVRP solution.

For each instance, Table 5.1 reports the number of vehicles (m), the maximum driving time (T), the upper bound (z_{UB}), and the estimated computing time for computing z_{UB} (T_{UB}) on our machine. The estimated computing time T_{UB} is the sum of the computing times of the algorithms of Taillard et al. [1996], Olivera and Viera [2007], and Mester and Bräysy [2005] normalized to our machine by considering that, according to `http://www.cpubenchmark.net/cpu_list.php`, our machine is $\approx$ 100 times faster than the 100 MHz machine used by Taillard et al. [1996], 4 times faster than the 1.8 GHz AMD Athlon XP 2200+ used by Olivera and Viera [2007], and 6.3 times faster than the Pentium 4 2 GHz used by Mester and Bräysy [2005].

We did not test our algorithm on the four instances for which no feasible solution is known (namely, instances CMT-1 with $m = 3$ and $T = 184$, CMT-1 with $m = 4$ and $T = 138$, CMT-2 with $m = 7$ and $T = 125$, and CMT-12 with $m = 6$ and $T = 143$) because the exact algorithm described in §5.8 would run out of memory at Step 2 while generating the path set $\mathscr{P}^2$.

TABLE 5.2: Computational Results on CMT-1, CMT-2 and CMT-3 instances

					Formulation $F1$								Solving $F1$		Formulation $F2$				Solving $F2$	
Inst	m	T	z_{UB}	z^*	$\%LB_1$	$\%LB_2$	$\mathscr{P}^2$	$\mathscr{R}^2$	LB_3	$\%LB_3$	T_3	$\mathscr{R}^3$	T_{IP}	T_{TOT}	LB_4	$\%LB_4$	T_4	$\mathscr{H}^4$	T_{IP}	T_{TOT}
CMT-1	1	551	524.61	524.61	99.6	99.8	2,549	329	524.61	100.0	9.4	47	0.1	9.5						
n = 50	2	275	533.00	533.00	98.1	98.3	142,645	95,763	532.22	99.9	36.3	509	0.9	37.2						
	1	577	524.61	524.61	99.6	99.8	1,922	329	524.61	100.0	10.3	62	0.5	10.8						
	2	289	529.85	529.85	98.7	98.9	35,676	16,669	527.68	99.6	15.0	806	1.9	16.9						
	3	192	552.68		94.6	94.8	m.o.	-	-	-	34.8									
	4	144	546.29	546.29	95.7	96.0	2,365,662	3,941,493	543.50	99.5	588.8	1,268	15.5	603.3						
CMT-2	1	877	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	10.6	216.5						
n = 75	2	439	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	16.5	222.4						
	3	292	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	21.0	226.9						
	4	219	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	27.7	233.6						
	5	175	835.80	835.80	98.5	98.8	170,128	148,826	832.77	99.6	188.8	4,746	336.0	524.8						
	6	146	857.00		96.0	96.4	m.o.	-	-	-	48.3									
	1	919	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	10.6	216.5						
	2	459	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	16.5	222.4						
	3	306	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	21.4	227.3						
	4	230	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	28.6	234.5						
	5	184	835.26	835.26	98.5	98.9	138,112	114,078	832.78	99.7	205.9	3,117	54.2	260.1						
	6	153	839.22	839.22	98.1	98.4	472,756	551,114	832.79	99.2	221.4	49,511	$\Rightarrow$		834.12	99.4	374.2	343,626	4,209.1	4,583.3
	7	131	872.64		94.3	94.7	m.o.	-	-	-	47.6									
CMT-3	1	867	826.14	826.14	98.7	99.0	5,248,271	$> 5 \cdot 10^6$	826.14	100.0	2,766.7	1,471	0.1	2,766.8						
n = 100	2	434	826.14	826.14	98.7	99.0	5,248,271	$> 5 \cdot 10^6$	826.14	100.0	2,766.7	1,471	15.0	2,781.7						
	3	289	826.14	826.14	98.7	99.0	5,248,271	$> 5 \cdot 10^6$	826.14	100.0	2,766.7	1,471	27.2	2,793.9						
	4	217	828.73		98.4	98.7	m.o.	-	-	-	96.2									
	5	173	^{A}851.47		97.6	97.9	m.o.	-	-	-	95.7									
	6	145	^{B}839.90		97.5	97.9	m.o.	-	-	-	93.4									
	1	909	826.14	826.14	98.7	99.0	5,248,271	$> 5 \cdot 10^6$	826.14	100.0	2,766.7	1,471	0.1	2,766.8						
	2	454	826.14	826.14	98.7	99.0	5,248,271	$> 5 \cdot 10^6$	826.14	100.0	2,766.7	1,471	15.3	2,782.0						
	3	303	826.14	826.14	98.7	99.0	5,248,271	$> 5 \cdot 10^6$	826.14	100.0	2,766.7	1,471	28.5	2,795.2						
	4	227	826.14	826.14	98.7	99.0	5,248,271	$> 5 \cdot 10^6$	826.14	100.0	2,766.7	1,471	46.6	2,813.3						
	5	182	833.15		97.9	98.2	m.o.	-	-	-	94.1									
	6	151	^{C}842.21		97.5	97.8	m.o.	-	-	-	88.0									

^{A} Best known upper bound 836.01 from Alonso et al. [2008]

^{B} Best known upper bound 836.21 from Olivera [2005]

Table 5.3: Computational Results on CMT-11 and CMT-12 instances

					Formulation $F1$								Solving $F1$		Formulation $F2$				Solving $F2$	
Inst	m	T	z_{UB}	z^*	$\%LB_1$	$\%LB_2$	$\mathscr{P}^2$	$\mathscr{R}^2$	LB_3	$\%LB_3$	T_3	$\mathscr{R}^3$	T_{IP}	T_{TOT}	LB_4	$\%LB_4$	T_4	$\mathscr{H}^4$	T_{IP}	T_{TOT}
CMT-11	1	1,094	1,042.11	1,042.11	99.6	99.9	185,303	4,705	1,042.11	100.0	632.1	83	0.2	632.3						
n = 120	2	547	1,042.11	1,042.11	99.6	99.9	185,303	4,705	1,042.11	100.0	632.1	83	0.2	632.3						
	3	365	1,042.11	1,042.11	99.6	99.9	185,303	4,705	1,042.11	100.0	632.1	83	0.2	632.3						
	4	274	[A]1,085.92		96.3	96.6	m.o.	-	-	-	776.1									
	5	219	1,042.11	1,042.11	99.6	99.9	182,530	5,102	1,042.11	100.0	620.0	105	0.2	620.2						
	1	1,146	1,042.11	1,042.11	99.6	99.9	185,303	4,705	1,042.11	100.0	632.1	83	0.2	632.3						
	2	573	1,042.11	1,042.11	99.6	99.9	185,303	4,705	1,042.11	100.0	632.1	83	0.2	632.3						
	3	382	1,042.11	1,042.11	99.6	99.9	185,303	4,705	1,042.11	100.0	632.1	83	0.2	632.3						
	4	287	1,042.11	1,042.11	99.6	99.9	185,303	4,705	1,042.11	100.0	631.0	83	0.2	631.2						
	5	229	1,042.11	1,042.11	99.6	99.9	185,303	4,705	1,042.11	100.0	619.8	83	0.2	620.0						
CMT-12	1	861	819.56	819.56	99.7	100.0	49,368	71	819.56	100.0	39.9	24	0.3	40.2						
n = 100	2	430	819.56	819.56	99.7	100.0	46,578	60	819.56	100.0	44.7	21	0.3	45.0						
	3	287	819.56	819.56	99.7	100.0	43,219	95	819.56	100.0	43.0	23	0.2	43.2						
	4	215	819.56	819.56	99.7	100.0	46,578	60	819.56	100.0	45.0	21	0.2	45.2						
	5	172	845.37		96.6	96.9	m.o.	-	-	-	16.6									
	1	902	819.56	819.56	99.7	100.0	49,368	71	819.56	100.0	39.9	24	0.3	40.2						
	2	451	819.56	819.56	99.7	100.0	46,578	60	819.56	100.0	44.7	21	0.3	45.0						
	3	301	819.56	819.56	99.7	100.0	43,219	95	819.56	100.0	43.0	23	0.3	43.3						
	4	225	819.56	819.56	99.7	100.0	46,578	60	819.56	100.0	45.0	21	0.5	45.5						
	5	180	824.78	824.78	99.0	99.4	1,857,419	818,631	819.56	99.4	314.6	746,542	$\Rightarrow$		823.69	99.9	2,290.3	274,857	77.6	2,367.9
	6	150	825.36	823.14	99.2	99.6	2,724,943	1,717,379	819.56	99.6	402.5	1,520,016	$\Rightarrow$		823.14	100.0	1,058.2	3,819	0.2	1,058.4

[A] Best known upper bound $1,078.64$ from Olivera [2005]

Tables 5.2 and 5.3 report detailed computational results. For each instance, the columns report the following information: instance name, $Inst$; number of vehicles, m; maximum driving time, T; initial upper bound, z_{UB}; optimal solution cost, z^*; percentage lower bound, $\%LB_k$ ($k = 1, \ldots, 4$), achieved by bounding procedure H^k; lower bounds, LB_3 and LB_4, achieved by bounding procedures H^3 and H^4, respectively; cardinality of the sets $\mathscr{P}^2$, $\mathscr{R}^2$, $\mathscr{R}^3$, $\mathscr{H}^4$; global time spent, T_3, to execute H^1, H^2, and H^3; time spent, T_4, by H^4 (including T_3); time spent, T_{IP}, by CPLEX to solve $F1$ or $F2$; total computing time, T_{TOT}. In column $\mathscr{P}^2$, "m.o." means that the exact method terminated prematurely at Step 2 because $\mathscr{P}^2$ is not optimal (i.e., $|\mathscr{P}^2| > \Delta(\mathscr{P}) = 10^7$). In this case, T_3 indicates the time spent by H^1 and H^2 plus the computing time for generating $\mathscr{P}^2$.

Table 5.2 and 5.3 show that the proposed method could solve 42 out of 52 test instances. All but one of the instances were solved within one hour of computing time. Thirty-five of the 42 instances solved to optimality have the optimal solution cost equal to the cost of the corresponding CVRP instance. The other seven instances have optimal solution cost slightly greater than the cost of the corresponding CVRP instance.

The lower bounds proposed seem to be effective. The lower bounds achieved by H^1 and H^2 are, on average, 98.6% and 98.9% with respect to the best known solution cost, respectively. Bounding procedure H^3 closed the gap of 26 instances. Only three instances were solved by using formulation $F2$. In all such cases, lower bound LB_4 is strictly greater than LB_3, requiring the generation of few schedules (set $\mathscr{H}^4$) in Step 7 with respect to the routes generated at Step 4 (set $\mathscr{R}^3$).

For the sake of fairness, we have to say that the performance of the proposed algorithm strongly depends on the initial upper bound, z_{UB}, used at Steps 2, 4 and 7. For example, by increasing such upper bound by 1%, the exact method, as it is, could not solve any of the instances of the classes CMT-2, CMT-3 and CMT-11, and it could solve only 12 out of 17 instances of the classes CMT-1 and CMT-12. Anyway, as shown in Table 5.1, high-quality upper bounds can be computed in a few minutes by running the heuristic algorithms of Taillard et al. [1996], Olivera and Viera [2007], and Mester and Bräysy [2005].

5.10 Conclusions

In this chapter, we describe an exact method to solve the MTVRP based on two set partitioning-like formulations. We describe four different bounding procedures, based on the linear relaxations of both formulations, enforced by valid inequalities, that are embedded into an exact solution method. The computational results show that the proposed exact algorithm can solve, to optimality, 42 out of 52 benchmark instances, involving up to 120 customers, used in the literature by the heuristic algorithms.

Chapter 6

Two-Echelon Capacitated Vehicle Routing Problem

[1]

In the *two-echelon capacitated vehicle routing problem* (2E-CVRP), the delivery to customers from a *depot* uses intermediate depots, called *satellites*. The 2E-CVRP involves two levels of routing problems. The first level requires a design of the routes for a vehicle fleet located at the depot to transport the customer demands to a subset of the satellites. The second level concerns the routing of a vehicle fleet located at the satellites to supply all customers from the satellites supplied from the depot. The objective is to minimize the sum of routing and handling costs. This chapter describes a new mathematical formulation of the 2E-CVRP used to derive valid lower bounds and an exact method. Computational results on benchmark instances show that the new exact algorithm outperforms the state-of-the-art exact methods from the literature.

6.1 Introduction

The *two-echelon capacitated vehicle routing problem* (2E-CVRP) is a two-level distribution system where the deliveries to customers from a *depot* are managed through intermediate capacitated depots, called *satellites*. The first level consists of vehicle routes that start and end at the depot and deliver the customer demands to a subset of satellites. In the 2E-CVRP we consider, a satellite has a limited capacity and can be serviced by more than one 1^{st}-level route. The second level consists of vehicle routes that start and end at the same satellite and supply all customers. A homogeneous vehicle fleet is used at each level. The 1^{st}-level vehicles are located at the depot and supply the satellites only. The 2^{nd}-level vehicles have a capacity smaller than that

[1]This chapter is based on Baldacci et al. [2012b]

of the 1st-level vehicles and supply the customers from the satellites. The operations of unloading 1st-level vehicles and loading 2nd-level vehicles at the satellites imply a *handling cost*, which we assume to be proportional to the quantity loaded/unloaded.

The 2E-CVRP aims to find two sets of 1st and 2nd-level routes such that each customer is visited exactly once by a 2nd-level route and the total routing and handling cost is minimized.

6.2 Literature Review

The 2E-CVRP has become a relevant distribution system for supplying customers located in large cities. Because many municipalities impose legal restrictions to keep large vehicles out of city centers, distribution companies create suburban platforms (satellites) where they transport goods with large vehicles. Then, small vehicles service downtown customers from the satellites.

Nonetheless, only recently the 2E-CVRP has received some attention in the literature. Gonzales Feliu et al. [2007b] described a commodity flow formulation and an exact branch-and-cut algorithm that solved instances with up to 32 customers and 2 satellites. This algorithm was improved by Perboli et al. [2010] and Perboli et al. [2011] by adding valid inequalities. Perboli et al. [2011] reported optimal solutions for instances with up to 32 customers and 2 satellites, but their model has been shown by Jepsen et al. [2011] not to be correct on instances with 3 or more satellites. Jepsen et al. [2011] extended the problem considered by Gonzales Feliu et al. [2007b] and Perboli et al. [2011] by introducing fixed costs for the routes of both levels and satellite capacities; they described an exact branch-and-cut algorithm, based on the new formulation and new valid inequalities, that outperforms the method of Perboli et al. [2011]. Heuristic methods have been proposed by Crainic et al. [2008], Perboli et al. [2011], Crainic et al. [2011] and Hemmelmayr et al. [2011]. Variants of the 2E-CVRP were considered by Chao [2002], Tan et al. [2006] and Nguyen et al. [2010].

The 2E-CVRP considered in this paper corresponds to the 2E-CVRP considered by Jepsen et al. [2011] and generalizes the *capacitated location routing problem* (LRP). The LRP consists of opening one or more depots, on a given set of a-priori defined depot locations, and designing, for each open depot, a number of routes to supply customers. A fixed cost and a capacity are associated with each depot. The objective is to minimize the sum of the fixed costs for opening the depots and the routing cost. Exact algorithms for the LRP were presented by Laporte et al. [1986], Akca et al. [2009], Belenguer et al. [2011], and Baldacci et al. [2011c].

In this chapter, we introduce a new mathematical formulation of the 2E-CVRP that is used to derive both integer and continuous relaxations. We present a new bounding procedure based on dynamic programming (DP), a dual-ascent method, and an exact algorithm that decomposes the 2E-CVRP into a limited set of *multi-depot capacitated vehicle routing problems* (MDCVRP) with side constraints. Extensive computational results on instances from the literature and on new instances show that the proposed method outperforms previous exact algorithms, both for the quality of the lower bounds achieved and the number and the size of the instances solved.

6.3 Problem Description and Mathematical Formulation

An undirected graph $G = (N, E)$ is given, where the vertex set N is partitioned as $N = \{0\} \cup N_S \cup N_C$. Vertex 0 represents the depot, the set $N_S = \{1, 2, \ldots, n_s\}$ represents n_s satellites, and the set $N_C = \{n_s+1, \ldots, n_s+n_c\}$ represents n_c customers. The edge set E is defined as $E = \{\{0, j\} : j \in N_S\} \cup \{\{i, j\} : i, j \in N_S \cup N_C, i < j\}$. A travel cost d_{ij} is associated with each edge $\{i, j\} \in E$. We assume that matrix d_{ij} satisfies the triangle inequality. Each customer $i \in N_C$ requires q_i units of goods from depot 0. We denote with $q_{tot} = \sum_{i \in N_C} q_i$ the sum of the customer demands.

A fleet of m^1 identical vehicles of capacity Q_1 are located at depot 0 and are used to transport goods to satellites. If used, a 1^{st}-level vehicle incurs a fixed cost U_1 and performs a route passing through the depot 0 and a subset of satellites. The cost of a 1^{st}-level route is the sum of the costs of the traversed edges plus the fixed cost U_1. Each satellite $k \in N_S$ can be visited by more than one 1^{st}-level route and has a capacity B_k that limits the total customer demand that can be delivered to it by the 1^{st}-level routes. Moreover, a fleet of m_k identical vehicles of capacity $Q_2 < Q_1$ are available at satellite $k \in N_S$ for servicing the customers. Nevertheless, at most $m^2 \leq \sum_{k \in N_S} m_k$ 2^{nd}-level vehicles can be globally used. If used, a 2^{nd}-level vehicle incurs a fixed cost U_2 and performs a route, that is a simple cycle in G passing through a satellite and a subset of customers and such that the total demand of the visited customers does not exceed the vehicle capacity Q_2. The cost of a 2^{nd}-level route is the sum of the traversed edges plus the fixed cost U_2. The handling cost at satellite $k \in N_S$ is given by H_k times the quantity delivered to satellite k.

The problem asks to design the vehicle routes of both levels so that each customer is visited exactly once, the quantity delivered to customers from each satellite is equal to the quantity received from the depot, and the sum of the routing and handling costs is minimized.

The 2E-CVRP contains as a special case the LRP. The LRP is defined on an undirected graph $G' = (N', E')$, where N' is partitioned as $N' = L \cup V$, where L represents possible depot locations and V a set of customers. A travel cost d_{ij} is associated with each edge $\{i, j\} \in E'$. A fixed cost C_k and a capacity B_k are associated with each depot location $k \in L$. Each customer $i \in V$ has associated a nonnegative demand q_i. An unlimited fleet of identical vehicles of capacity Q are available at the depots to supply the customers. If used, a vehicle incurs a fixed cost U and performs a route passing through one of the depot locations and such that the total demand of the visited customers is at most Q. The cost of a route is the sum of the costs of the traversed edges plus the fixed cost U. The LRP consists of opening a set of depots and designing a set of routes for each open depot so that the total load of the routes operated from a depot $k \in L$ does not exceed its capacity B_k and each customer is visited by exactly one route. The objective is to minimize the sum of the cost of open depots and the costs of the routes.

Any LRP instance can be converted into an equivalent 2E-CVRP instance as follows.

(a) Define graph $G = (N, E)$ by setting $N_S = L$, $N_C = V$, and define the edge costs $d_{0k} = \frac{1}{2}C_k$, $k \in N_S$, and $d_{kj} = \infty$, $k, j \in N_S$, $k < j$.

(b) Define the 1st-level vehicle fleet by setting $m^1 = |N_S|$, $Q_1 = \infty$, and $U_1 = 0$;

(c) Define the 2nd-level vehicle fleet by setting $m^2 = |N_C|$, $U_2 = U$, $Q_2 = Q$, and $m_k = |N_C|$, $k \in N_S$.

Any optimal solution of the resulting 2E-CVRP instance is also an optimal solution of the original LRP instance. Because $d_{kj} = \infty$, for any $k, j \in N_S$, $k < j$, each 1st-level vehicle route can only be a single-satellite route $(0, k, 0)$ of cost C_k, $k \in N_S$. Because $Q_1 = \infty$, any optimal solution contains at most a single-satellite route for each $k \in N_S$ representing the opening of depot k.

As for the 2E-CVRP, we assume that the travel cost matrix $[d_{ij}]$ is modified as follows in order to consider vehicle fixed costs U_1 and U_2: (i) $d_{0k} = d_{0k} + \frac{1}{2}U_1$, $k \in N_S$, and (ii) $d_{ki} = d_{ki} + \frac{1}{2}U_2$, $k \in N_S$, $i \in N_C$.

Let $\mathscr{M}$ be the index set of all 1st-level routes, and let $\mathscr{M}_k \subseteq \mathscr{M}$ be the subset of 1st-level routes serving satellite $k \in N_S$. Let R_r and $E(R_r)$ be the subset of satellites visited and the subset of edges traversed by 1st-level route $r \in \mathscr{M}$, respectively. The cost g_r of route $r \in \mathscr{M}$ is $g_r = \sum_{\{i,j\} \in E(R_r)} d_{ij}$. We assume that the route set $\mathscr{M}$ contains $\lceil \min\{m_k Q_2,\ q_{tot}\}/Q_1 \rceil$ copies of the single-satellite route $(0, k, 0)$, for each satellite $k \in N_S$. Let w^{min} and w_r^{max} be the minimum and maximum loads of 1st-level route $r \in \mathscr{M}$, computed as $w^{min} = \max\{q_{tot} - (m^1 - 1)Q_1,\ 0\}$ and $w_r^{max} = \min\{Q_1,\ q_{tot},\ \sum_{k \in R_r} m_k Q_2\}$. Moreover, we denote by $W_r = \{w \in \mathbb{Z}_+ : w^{min} \le w \le w_r^{max}\}$ the set of possible loads of 1st-level route $r \in \mathscr{M}$. Because in real-world applications the number of satellites is small (say, $n_s \le 10$), in the following we assume that the set $\mathscr{M}$ is generated by pure enumeration.

Let $\mathscr{R}_k$ be the index set of the 2nd-level routes passing through satellite $k \in N_S$, and let $\mathscr{R}_{ik} \subseteq \mathscr{R}_k$ be the subset of routes passing through satellite $k \in N_S$ and customer $i \in N_C$. We indicate with $\mathscr{R} = \cup_{k \in N_S} \mathscr{R}_k$ the set of all 2nd-level routes and with π_ℓ the satellite visited by route $\ell \in \mathscr{R}$. Moreover, we indicate with $R_{k\ell}$ and $E(R_{k\ell})$ the subset of customers visited and the subset of edges traversed by route $\ell \in \mathscr{R}_k$, respectively. A load $w_{k\ell} = \sum_{i \in R_{k\ell}} q_i$ and a cost $c_{k\ell} = \sum_{\{i,j\} \in E(R_{k\ell})} d_{ij} + H_k w_{k\ell}$ are associated with route $\ell \in \mathscr{R}_k$.

Let y_r be a binary variable equal to 1 if and only if route $r \in \mathscr{M}$ is in solution, $x_{k\ell}$ a binary variable equal to 1 if and only if route $\ell \in \mathscr{R}_k$ of satellite $k \in N_S$ is in solution, and q_{kr} a nonnegative integer variable representing the quantity delivered by 1^{st}-level route $r \in \mathscr{M}$ to satellite $k \in R_r$ (we assume $q_{kr} = 0$, for each satellite $k \in N_S \setminus R_r$). The 2E-CVRP can be formulated as follows

$$(F) \quad z(F) = \min \sum_{k \in N_S} \sum_{\ell \in \mathscr{R}_k} c_{k\ell} x_{k\ell} + \sum_{r \in \mathscr{M}} g_r y_r \tag{6.1}$$

$$\text{s.t.} \sum_{k \in N_S} \sum_{\ell \in \mathscr{R}_{ik}} x_{k\ell} = 1, \qquad i \in N_C, \tag{6.2}$$

$$\sum_{\ell \in \mathscr{R}_k} x_{k\ell} \le m_k, \qquad k \in N_S, \tag{6.3}$$

$$\sum_{k \in N_S} \sum_{\ell \in \mathscr{R}_k} x_{k\ell} \le m^2, \tag{6.4}$$

$$\sum_{\ell \in \mathscr{R}_k} w_{k\ell} x_{k\ell} \le B_k, \qquad k \in N_S, \tag{6.5}$$

$$\sum_{r \in \mathscr{M}} y_r \le m^1, \tag{6.6}$$

$$\sum_{r \in \mathscr{M}_k} q_{kr} = \sum_{\ell \in \mathscr{R}_k} w_{k\ell} x_{k\ell}, \qquad k \in N_S, \tag{6.7}$$

$$\sum_{k \in R_r} q_{kr} \le Q_1 y_r, \qquad r \in \mathscr{M}, \tag{6.8}$$

$$x_{k\ell} \in \{0,1\}, \qquad k \in N_S, \ell \in \mathscr{R}_k, \tag{6.9}$$

$$y_r \in \{0,1\}, \qquad r \in \mathscr{M}, \tag{6.10}$$

$$q_{kr} \in \mathbb{Z}_+, \qquad k \in R_r, r \in \mathscr{M}. \tag{6.11}$$

The objective function (6.1) states to minimize the total cost. Constraints (6.2) specify that each customer $i \in N_C$ must be visited by exactly one 2^{nd}-level route. Constraints (6.3), (6.4), and (6.6) impose the upper bounds on the number of 1^{st} and 2^{nd}-level routes in solution. Constraints (6.5) impose the satellite capacities. The balance between the quantity delivered by 1^{st}-level routes to a satellite and the customer demands supplied from the satellite is imposed by constraints (6.7). Finally, constraints (6.8) impose that the vehicle capacity of the 1^{st}-level vehicles is not exceeded.

6.4 Relaxations of Formulation F

Let LF be the LP-relaxation of formulation F, and let $z(LF)$ be its optimal solution cost. Notice that, in any optimal LF solution, y_r is equal to $(\sum_{k \in R_r} q_{kr})/Q_1$, for each

1^{st}-level route $r \in \mathscr{M}$. Thus, the higher the 1^{st}-level routing cost, the worse the lower bound $z(LF)$. In the following, we describe an integer relaxation of the 2E-CVRP, called RF, that can provide a lower bound better than $z(LF)$.

6.4.1 Relaxation RF

This relaxation derives from problem F by relaxing, in a Lagrangean fashion, constraints (6.2), (6.3), and (6.4) with penalties $\lambda_i \in \mathbb{R}$, $i \in N_C$, $\mu_k \in \mathbb{R}_-$, $k \in N_S$, and $\mu_0 \in \mathbb{R}_-$, respectively, and by defining the *marginal routing costs* β_{ik} for servicing customer $i \in N_C$ from depot $k \in N_S$ as a feasible solution of the following inequalities

$$\sum_{i \in N_C} a_{ik\ell}\beta_{ik} \leq c_{k\ell} - \sum_{i \in N_C} a_{ik\ell}\lambda_i - \mu_k - \mu_0, \quad \ell \in \mathscr{R}_k,\ k \in N_S, \tag{6.12}$$

where $a_{ik\ell}$ is the number of times customer $i \in N_C$ is visited by route $\ell \in \mathscr{R}_k$ of satellite $k \in N_S$.

Problem RF involves binary variables ξ_{ik} equal to 1 if customer $i \in N_C$ is supplied from satellite $k \in N_S$ (0 otherwise) and variables y_r and q_{kr}, as defined for problem F. Relaxation RF is

$$\begin{aligned}
(RF) \quad z(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu})) = \min \ & \sum_{k \in N_S}\sum_{i \in N_C} \beta_{ik}\xi_{ik} + \sum_{r \in \mathscr{M}} g_r y_r + \\
& \sum_{i \in N_C} \lambda_i + \sum_{k \in N_S} m_k\mu_k + m^2\mu_0 && (6.13)\\
s.t. \ & \sum_{k \in N_S} \xi_{ik} = 1, \qquad i \in N_C, && (6.14)\\
& \sum_{r \in \mathscr{M}_k} q_{kr} = \sum_{i \in N_C} q_i\xi_{ik}, \qquad k \in N_S, && (6.15)\\
& \sum_{i \in N_C} q_i\xi_{ik} \leq B_k, \qquad k \in N_S, && (6.16)\\
& (6.6), (6.8), (6.10) \text{ and } (6.11), && (6.17)\\
& \xi_{ik} \in \{0,1\}, \qquad i \in N_C, k \in N_S. && (6.18)
\end{aligned}$$

Theorem 6.1. *Value $z(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}))$ is a valid lower bound on the 2E-CVRP for any solution $\boldsymbol{\beta}$ of inequalities (6.12) and any pair of vectors $\boldsymbol{\lambda} \in \mathbb{R}^{N_C}$ and $\boldsymbol{\mu} \in \mathbb{R}_-^{N_S+1}$.*

Proof. Consider an F solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{q}})$ of cost $\bar{z}(F)$. Let $J_k = \{\ell \in \mathscr{R}_k : \bar{x}_{k\ell} = 1\}$, $k \in N_S$, be the set of 2^{nd}-level routes and $L = \{r \in \mathscr{M} : \bar{y}_r = 1\}$ the set of 1^{st}-level routes in solution. We denote by $\bar{V}_k = \{i \in R_{k\ell} : \ell \in J_k\}$ the subset of customers serviced from satellite $k \in N_S$ and by $\bar{N}_S = \{k \in R_r : r \in L\}$ the subset of satellites used in solution.

Let $z(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$ be the cost of an optimal RF solution for a given set of values β_{ik}, satisfying inequalities (6.12), and penalty vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. Define the reduced cost $\tilde{c}_{k\ell}$ of route $\ell \in \mathscr{R}_k$, $k \in N_S$, with respect to $\boldsymbol{\beta}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\mu}$ as $\tilde{c}_{k\ell} = c_{k\ell} - \sum_{i \in N_C} a_{ik\ell}(\beta_{ik} + \lambda_i) - \mu_k - \mu_0$. Note that $\tilde{c}_{k\ell} \geq 0$, $\ell \in \mathscr{R}_k$, $k \in N_S$, from the definition of values β_{ik} (see inequalities (6.12)). From the definition of $\tilde{c}_{k\ell}$, we derive

$$\sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} \tilde{c}_{k\ell} = \sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} c_{k\ell} - \sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} \sum_{i \in N_C} a_{ik\ell}(\beta_{ik} + \lambda_i) - \sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} (\mu_k + \mu_0). \quad (6.19)$$

Because any F solution imposes that each customer is visited exactly once, we have

$$\sum_{\ell \in J_k} \sum_{i \in N_C} a_{ik\ell}(\beta_{ik} + \lambda_i) = \sum_{i \in \bar{V}_k} (\beta_{ik} + \lambda_i), \quad k \in \bar{N}_S. \quad (6.20)$$

Because $|J_k| \leq m_k$, $\mu_k \leq 0$, $k \in \bar{N}_S$, $\sum_{k \in L} |J_k| \leq m^2$, and $\mu_0 \leq 0$, we have

$$\sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} (\mu_k + \mu_0) = \sum_{k \in \bar{N}_S} |J_k| \mu_k + \sum_{k \in \bar{N}_S} |J_k| \mu_0 \geq \sum_{k \in \bar{N}_S} m_k \mu_k + m^2 \mu_0. \quad (6.21)$$

By adding and subtracting the term $\sum_{k \in L} g_r$ to the right-hand side of expression (6.19) and by using expressions (6.20) and (6.21), we obtain the following inequality

$$\sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} \tilde{c}_{k\ell} \leq \sum_{k \in L} g_r + \sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} c_{k\ell} - \sum_{k \in L} g_r - \sum_{k \in \bar{N}_S} \sum_{i \in \bar{V}_k} (\beta_{ik} + \lambda_i) - \sum_{k \in \bar{N}_S} m_k \mu_k - m^2 \mu_0. \quad (6.22)$$

Note that $\sum_{k \in L} g_r + \sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} c_{k\ell} = \bar{z}(F)$, so from expression (6.22) we derive

$$\sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} \tilde{c}_{k\ell} \leq \bar{z}(F) - \Big(\sum_{k \in L} g_r + \sum_{k \in \bar{N}_S} \sum_{i \in \bar{V}_k} (\beta_{ik} + \lambda_i) + \sum_{k \in \bar{N}_S} m_k \mu_k + m^2 \mu_0 \Big). \quad (6.23)$$

The second term of expression (6.23) is the cost, say $\tilde{z}(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$, of a feasible but nonnecessarily optimal RF solution $(\tilde{\boldsymbol{\xi}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{q}})$ that is derived from the F solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{q}})$ by setting

1. $\tilde{\boldsymbol{y}} = \bar{\boldsymbol{y}}$;
2. $\tilde{\xi}_{ik} = 1$, $i \in \bar{V}_k$, $k \in \bar{N}_S$;
3. $\tilde{\xi}_{ik} = 0$, $i \in N_C \setminus \bar{V}_k$, $k \in \bar{N}_S$, and $\tilde{\xi}_{ik} = 0$, $i \in N_C$, $k \in N_S \setminus \bar{N}_S$;
4. $\tilde{\boldsymbol{q}} = \bar{\boldsymbol{q}}$.

Because $\tilde{z}(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$ is greater than or equal to the optimal solution cost $z(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$ of problem RF, from inequality (6.23) we derive

$$\sum_{k \in \bar{N}_S} \sum_{\ell \in J_k} \tilde{c}_{k\ell} \leq \bar{z}(F) - z(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})). \quad (6.24)$$

Because $\tilde{c}_{k\ell} \geq 0$, $\ell \in \mathscr{R}_k$, $k \in N_S$, from inequality (6.24) we have

$$z(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})) \leq \bar{z}(F). \square$$

From Theorem 6.1, the following corollary follows.

Corollary 1. Let z_{UB} be a valid upper bound on the 2E-CVRP. For a given pair of vectors $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and any solution $\boldsymbol{\beta}$ of inequalities (6.12), let $\tilde{c}_{k\ell} = c_{k\ell} - \sum_{i \in N_C} a_{ik\ell}(\beta_{ik} + \lambda_i) - \mu_k - \mu_0$ be the reduced cost of 2^{nd}-level route $\ell \in \mathscr{R}_k$, $k \in N_S$. Any optimal 2E-CVRP solution of cost smaller than z_{UB} cannot contain any 2^{nd}-level route $\ell \in \mathscr{R}_k$, $k \in N_S$, of reduced cost $\tilde{c}_{k\ell} \geq z_{UB} - z(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$.

Proof. The corollary directly follows from inequality (6.24) because $\tilde{c}_{k\ell} \geq 0$, $\ell \in \mathscr{R}_k$, $k \in N_S$. $\square$

Theorem 6.2. *The relation* $\max_{\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}}\{z(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))\} \geq z(LF)$ *holds, and this inequality can be strict.*

Proof. We show that, from any optimal LF dual solution of cost $z(LF)$, two penalty vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ and a solution $\boldsymbol{\beta}$ of inequalities (6.12) such that $z(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})) \geq z(LF)$ can be derived.

Let $\boldsymbol{u} \in \mathbb{R}^{|N_C|}$, $\boldsymbol{v} \in \mathbb{R}_-^{|N_S|}$, $v_0 \in \mathbb{R}_-$, $\boldsymbol{\sigma} \in \mathbb{R}_-^{|N_S|}$, $\alpha_0 \leq 0$, $\boldsymbol{\alpha} \in \mathbb{R}^{|N_S|}$, $\boldsymbol{\omega} \in \mathbb{R}_-^{|\mathscr{M}|}$, and $\boldsymbol{\vartheta} \in \mathbb{R}_-^{|\mathscr{M}|}$ be the dual variables associated with constraints (6.2)-(6.8) and with the continuous relaxation of constraints (6.10) (i.e., $0 \leq y_r \leq 1$, $r \in \mathscr{M}$). The dual of problem LF is as follows.

$$\begin{aligned}
z(LF) = \max \; & \sum_{i \in N_C} u_i + \sum_{k \in N_S} m_k v_k + m^2 v_0 + \sum_{k \in N_S} B_k \sigma_k \\
& + m^1 \alpha_0 + \sum_{r \in \mathscr{M}} \vartheta_r && && (6.25)\\
s.t. \; & \sum_{i \in R_{k\ell}} u_i + v_k + v_0 + w_{k\ell}\sigma_k - w_{k\ell}\alpha_k \leq c_{k\ell}, && \ell \in \mathscr{R}_k,\ k \in N_S, && (6.26)\\
& \alpha_0 - Q_1 \omega_r + \vartheta_r \leq g_r, && r \in \mathscr{M}, && (6.27)\\
& \alpha_k + \omega_r \leq 0, && k \in N_S,\ r \in \mathscr{M} && (6.28)\\
& u_i \in \mathbb{R}, && i \in N_C, && (6.29)\\
& v_k \leq 0,\ \alpha_k \in \mathbb{R},\ \sigma_k \leq 0, && k \in N_S, && (6.30)\\
& v_0 \leq 0,\ \alpha_0 \leq 0, && && (6.31)\\
& \omega_r \leq 0,\ \vartheta_r \leq 0, && r \in \mathscr{M}. && (6.32)
\end{aligned}$$

Let $\boldsymbol{u}^* = (u_1^*, \ldots, u_{n_c}^*)$, $\boldsymbol{v}^* = (v_0^*, v_1^*, \ldots, v_{n_s}^*)$, $\boldsymbol{\sigma}^* = (\sigma_1^*, \ldots, \sigma_{n_s}^*)$, $\boldsymbol{\alpha}^* = (\alpha_0^*, \alpha_1^*, \ldots, \alpha_{n_s}^*)$, $\boldsymbol{\omega}^* = (\omega_1^*, \ldots, \omega_{|\mathscr{M}|}^*)$, and $\boldsymbol{\vartheta}^* = (\vartheta_1^*, \ldots, \vartheta_{|\mathscr{M}|}^*)$ be an optimal dual solution of LF.

Define the vectors $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ with respect to $\boldsymbol{u}^*$, $\boldsymbol{v}^*$, $\boldsymbol{\sigma}^*$ and $\boldsymbol{\alpha}^*$ by setting $\lambda_i = 0$, $i \in N_C$, $\mu_k = v_k^*$, $k \in N_S$, $\mu_0 = v_0^*$, and $\beta_{ik} = u_i^* + q_i(\sigma_k^* - \alpha_k^*)$, $i \in N_C$, $k \in N_S$.

First, we show that vectors $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ as defined above provide a feasible solution of inequalities (6.12). Because $\sum_{i\in N_C} a_{ik\ell}q_i = w_{k\ell}$, we have

$$\sum_{i\in N_C} a_{ik\ell}\beta_{ik} = \sum_{i\in N_C} a_{ik\ell}u_i^* + w_{k\ell}\sigma_k^* - w_{k\ell}\alpha_k^*, \quad \ell \in \mathscr{R}_k,\ k\in N_S. \tag{6.33}$$

Because of inequalities (6.26), from (6.33) we derive

$$\sum_{i\in N_C} a_{ik\ell}\beta_{ik} \le c_{k\ell} - v_k^* - v_0^*, \quad \ell \in \mathscr{R}_k,\ k\in N_S. \tag{6.34}$$

Inequalities (6.34) correspond to inequalities (6.12) because $\mu_k = v_k^*$, $k\in N_S$, $\mu_0 = v_0^*$, and $\lambda_i = 0$, $i\in N_C$.

Let $(\boldsymbol{\xi}^*, \boldsymbol{y}^*, \boldsymbol{q}^*)$ be an optimal RF solution using $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ as defined above. We have

$$\begin{aligned} z(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu})) &= \sum_{k\in N_S}\sum_{i\in N_C}(u_i^* + q_i(\sigma_k^* - \alpha_k^*))\xi_{ik}^* + \sum_{r\in\mathscr{M}} g_r y_r^* + \sum_{k\in N_S} m_k v_k^* + m^2 v_0^* \\ &= \sum_{i\in N_C} u_i^* \sum_{k\in N_S}\xi_{ik}^* + \sum_{k\in N_S}\sigma_k^*\sum_{i\in N_C} q_i\xi_{ik}^* - \sum_{k\in N_S}\alpha_k^*\sum_{i\in N_C} q_i\xi_{ik}^* \\ &\qquad + \sum_{r\in\mathscr{M}} g_r y_r^* + \sum_{k\in N_S} m_k v_k^* + m^2 v_0^*. \end{aligned} \tag{6.35}$$

From constraints (6.14), we have

$$\sum_{i\in N_C} u_i^* \sum_{k\in N_S}\xi_{ik}^* = \sum_{i\in N_C} u_i^*. \tag{6.36}$$

From constraints (6.27), we derive

$$\sum_{r\in\mathscr{M}} g_r y_r^* \ge -\sum_{r\in\mathscr{M}} Q_1\omega_r^* y_r^* + \alpha_0^*\sum_{r\in\mathscr{M}} y_r^* + \sum_{r\in\mathscr{M}}\vartheta_r^* y_r^*. \tag{6.37}$$

Because $\alpha_0^* \le 0$, from inequality (6.6) we obtain

$$\alpha_0^*\sum_{r\in\mathscr{M}} y_r^* \ge m^1\alpha_0^* \tag{6.38}$$

and, because $\vartheta_r^* \le 0$, we have

$$\sum_{r\in\mathscr{M}}\vartheta_r^* y_r^* \ge \sum_{r\in\mathscr{M}}\vartheta_r^*. \tag{6.39}$$

Thus, from expressions (6.35), (6.36), (6.37), (6.38) and (6.39), we derive

$$
\begin{aligned}
z(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu})) \geq \sum_{i\in N_C} u_i^* + \sum_{k\in N_S} \sigma_k^* \sum_{i\in N_C} q_i\xi_{ik}^* - \sum_{k\in N_S} \alpha_k^* \sum_{i\in N_C} q_i\xi_{ik}^* - \sum_{r\in\mathscr{M}} Q_1\omega_r^* y_r^* \\
+ m^1\alpha_0^* + \sum_{r\in\mathscr{M}} \vartheta_r^* + \sum_{k\in N_S} m_k v_k^* + m^2 v_0^*.
\end{aligned}
\tag{6.40}
$$

From inequalities (6.8) and because $\omega_r^* \leq 0$, we obtain

$$
-\sum_{r\in\mathscr{M}} Q_1\omega_r^* y_r^* \geq -\sum_{r\in\mathscr{M}} \omega_r^* \sum_{k\in R_r} q_{kr}^*. \tag{6.41}
$$

Because $\sum_{i\in N_C} q_i\xi_{ik}^* = \sum_{r\in\mathscr{M}} q_{kr}^*$, $k \in N_S$, from (6.41) we have

$$
\begin{aligned}
\sum_{k\in N_S} \sigma_k^* \sum_{i\in N_C} q_i\xi_{ik}^* - \sum_{k\in N_S} \alpha_k^* \sum_{i\in N_C} q_i\xi_{ik}^* - \sum_{r\in\mathscr{M}} Q_1\omega_r^* y_r^* \geq \\
\sum_{k\in N_S} (\sigma_k^* - \alpha_k^*) \sum_{r\in\mathscr{M}} q_{kr}^* - \sum_{r\in\mathscr{M}} \omega_r^* \sum_{k\in R_r} q_{kr}^*.
\end{aligned}
\tag{6.42}
$$

Notice that $\sum_{k\in N_S} \alpha_k^* \sum_{r\in\mathscr{M}} q_{kr}^* = \sum_{r\in\mathscr{M}} \sum_{k\in R_r} \alpha_k^* q_{kr}^*$. Because $\sigma_k^* \leq 0$ and $\sum_{r\in\mathscr{M}} q_{kr}^* \leq B_k$ (see constraints (6.5)), we have $\sigma_k^* \sum_{r\in\mathscr{M}} q_{kr}^* \geq B_k\sigma_k^*$, $k \in N_S$. Thus, from (6.42), we derive

$$
\begin{aligned}
\sum_{k\in N_S} (\sigma_k^* - \alpha_k^*) \sum_{r\in\mathscr{M}} q_{kr}^* - \sum_{r\in\mathscr{M}} \omega_r^* \sum_{k\in R_r} q_{kr}^* \geq \\
\sum_{k\in N_S} B_k\sigma_k^* - \sum_{r\in\mathscr{M}} \sum_{k\in R_r} \alpha_k^* q_{kr}^* - \sum_{r\in\mathscr{M}} \omega_r^* \sum_{k\in R_r} q_{kr}^*.
\end{aligned}
\tag{6.43}
$$

Because of constraints (6.28), we have

$$
-\sum_{r\in\mathscr{M}} \sum_{k\in R_r} \alpha_k^* q_{kr}^* - \sum_{r\in\mathscr{M}} \omega_r^* \sum_{k\in R_r} q_{kr}^* = -\sum_{r\in\mathscr{M}} \sum_{k\in R_r} (\alpha_k^* + \omega_r^*) q_{kr}^* \geq 0. \tag{6.44}
$$

From inequalities (6.40), (6.43) and (6.44) we obtain

$$
z(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu})) \geq \sum_{i\in N_C} u_i^* + \sum_{k\in N_S} m_k v_k^* + m^2 v_0^* + \sum_{k\in N_S} B_k\sigma_k^* + m^1\alpha_0^* + \sum_{r\in\mathscr{M}} \vartheta_r^* = z(LF).
$$

Finally, notice that $z(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu})) > z(LF)$ whenever the optimal dual solution $(\boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{\sigma}^*, \boldsymbol{\alpha}^*, \boldsymbol{\omega}^*, \boldsymbol{\vartheta}^*)$ does not saturate constraint (6.27) for some route $r \in \mathscr{M}$ that is in the optimal RF solution (i.e., $y_r^* = 1$) or/and constraint (6.28) for some satellite $k \in N_S$ and route $r \in \mathscr{M}$ such that $q_{kr}^* > 0$. In these cases, inequality (6.37) or/and inequality (6.43) are strict and, consequently, $z(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu})) > z(LF)$. □

6.4.2 Relaxation $\overline{RF}$

Solving RF to optimality can be time-consuming, so we describe a further relaxation of RF, called $\overline{RF}$, that can be efficiently solved.

Let ϕ_{rw} be a lower bound on the cost for delivering a demand $w \in W_r$ to customers from the subsets of satellites R_r visited by 1^{st}-level route $r \in \mathscr{M}$. For a given vector $\boldsymbol{\beta}$ satisfying inequalities (6.12), the value ϕ_{rw} is defined as the optimal solution cost of the following continuous knapsack problem, called $KP(r,w)$

$$\begin{aligned}(KP(r,w)) \quad \phi_{rw} = \min & \sum_{i \in N_C} \min_{k \in R_r}\{\beta_{ik}\} z_i \\ s.t. & \sum_{i \in N_C} q_i z_i = w, \\ & 0 \le z_i \le 1, \qquad i \in N_C.\end{aligned}$$

For each pair (r,w), we denote by $z_i^*(r,w)$, $i \in N_C$, the optimal solution of problem $KP(r,w)$, and we define $V(r,w) = \{i \in N_C : z_i^*(r,w) > 0\}$. Let ζ_{rw}, $r \in \mathscr{M}$, $w \in W_r$, be a binary variable equal to 1 if and only if 1^{st}-level route r delivering w units of goods is in solution. Problem $\overline{RF}$ is defined as

$$\begin{aligned}(\overline{RF}) \quad z(\overline{RF}(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu})) = \min & \sum_{r \in \mathscr{M}} \sum_{w \in W_r} (g_r + \phi_{rw})\zeta_{rw} + \\ & \sum_{i \in N_C} \lambda_i + \sum_{k \in N_S} m_k \mu_k + m^2 \mu_0 && (6.45) \\ s.t. & \sum_{r \in \mathscr{M}} \sum_{w \in W_r} w \zeta_{rw} = q_{tot}, && (6.46) \\ & \sum_{w \in W_r} \zeta_{rw} \le 1, \qquad r \in \mathscr{M}, && (6.47) \\ & \zeta_{rw} \in \{0,1\}, \qquad r \in \mathscr{M},\ w \in W_r. && (6.48)\end{aligned}$$

Problem $\overline{RF}$ is an integer problem that can be conveniently solved by DP. The following theorem shows that $\overline{RF}$ is a relaxation of problem RF.

Theorem 6.3. *The relation $z(\overline{RF}(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu})) \le z(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}))$ holds for any solution $\boldsymbol{\beta}$ of inequalities (6.12) and for any penalty vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$.*

Proof. Let $(\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{q}})$ be an RF solution of cost $\bar{z}(RF(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}))$. This solution provides a feasible $\overline{RF}$ solution $\bar{\boldsymbol{\zeta}}$ of cost $\bar{z}(\overline{RF}(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}))$ as follows. Let $\bar{\mathscr{M}} = \{r \in \mathscr{M} : \bar{y}_r = 1\}$, and let $\bar{V}_k = \{i \in N_C : \bar{\xi}_{ik} = 1\}$, $k \in N_S$. Define $\bar{\zeta}_{rw} = 0$, $r \in \mathscr{M} \setminus \bar{\mathscr{M}}$ and $w \in W_r$. For any $r \in \mathscr{M}$, define $\bar{w}_r = \sum_{k \in R_r} \bar{q}_{rk}$. Set $\bar{\zeta}_{r\bar{w}_r} = 1$, $r \in \bar{\mathscr{M}}$, and $\bar{\zeta}_{rw} = 0$, $r \in \bar{\mathscr{M}}$, $w \ne \bar{w}_r$.

It is quite obvious to observe that $\bar{\boldsymbol{\zeta}}$ satisfies constraints (6.47). To show that $\bar{\boldsymbol{\zeta}}$ also satisfies constraints (6.46), we observe that, by adding up constraints 6.15 of problem

RF, we obtain

$$\sum_{k\in N_S}\sum_{r\in \mathscr{M}_k} \bar{q}_{rk} = \sum_{i\in N_C} q_i \sum_{k\in N_S} \bar{\xi}_{ik}. \tag{6.49}$$

Because $\bar{q}_{rk} = 0$, $r \in \mathscr{M} \setminus \bar{\mathscr{M}}$, $k \in N_S$, and $r \in \bar{\mathscr{M}}$, $k \in N_S \setminus R_r$, we have

$$\sum_{k\in N_S}\sum_{r\in \mathscr{M}_k} \bar{q}_{rk} = \sum_{r\in \bar{\mathscr{M}}}\sum_{k\in R_r} \bar{q}_{rk}. \tag{6.50}$$

Because of equations (6.14), from (6.49) and (6.50) we derive

$$\sum_{r\in \bar{\mathscr{M}}}\sum_{k\in R_r} \bar{q}_{rk} = \sum_{i\in N_C} q_i. \tag{6.51}$$

From the definition of $\bar{w}_r$, $r \in \mathscr{M}$, and of $\bar{\zeta}_{rw}$, $r \in \mathscr{M}$, $w \in W_r$, we have

$$\sum_{r\in \bar{\mathscr{M}}}\sum_{k\in R_r} \bar{q}_{rk} = \sum_{r\in \bar{\mathscr{M}}}\sum_{w\in W_r} w\bar{\zeta}_{rw},$$

which shows, together with equality (6.51), that $\bar{\boldsymbol{\zeta}}$ also satisfies constraint (6.46).

For each $r \in \bar{\mathscr{M}}$, define θ_{ikr}, $i \in N_C$, $k \in N_S$, by setting

$$\theta_{ikr} = \begin{cases} 0, & \text{if } \bar{\xi}_{ik} = 0 \\ \frac{\bar{q}_{rk}}{\sum_{r\in \bar{\mathscr{M}}_k} \bar{q}_{rk}}, & \text{if } \bar{\xi}_{ik} = 1 \end{cases} \tag{6.52}$$

where $\bar{\mathscr{M}}_k = \mathscr{M}_k \cap \bar{\mathscr{M}}$.

In the following, we show that the values θ_{ikr}, defined above, provide a feasible solution to problems $KP(r, \bar{w}_r)$, $r \in \bar{\mathscr{M}}$. For each $r \in \bar{\mathscr{M}}$, define $\bar{z}_i(r, \bar{w}_r) = \sum_{k\in R_r} \theta_{ikr}$, $i \in N_C$. Since customer i is assigned exactly to a satellite in any feasible RF solution, from the definition of θ_{ikr}, given by expression (6.52), we have $0 \le \bar{z}_i(r, \bar{w}_r) \le 1$, $i \in N_C$. Moreover, we have

$$\sum_{i\in N_C} q_i \bar{z}_i(r, \bar{w}_r) = \sum_{i\in N_C} q_i \sum_{k\in R_r} \theta_{ikr}, \quad r \in \bar{\mathscr{M}}. \tag{6.53}$$

Because $\theta_{ikr} = 0$, $i \in N_C$, $k \in N_S \setminus R_r$, we have

$$\sum_{i\in N_C} q_i \sum_{k\in R_r} \theta_{ikr} = \sum_{k\in R_r} \Big(\sum_{i\in \bar{V}_k} q_i\Big)\theta_{ikr} = \sum_{k\in R_r} \Big(\sum_{i\in \bar{V}_k} q_i\Big) \frac{\bar{q}_{rk}}{\sum_{r\in \bar{\mathscr{M}}_k} \bar{q}_{rk}}, \quad r \in \bar{\mathscr{M}}. \tag{6.54}$$

Since $\sum_{r\in \bar{\mathscr{M}}_k} \bar{q}_{rk} = \sum_{i\in \bar{V}_k} q_i$, from (6.53), (6.54) and the definition of $\bar{w}_r$, we obtain

$$\sum_{i\in N_C} q_i \bar{z}_i(r, \bar{w}_r) = \sum_{k\in R_r} \bar{q}_{rk} = \bar{w}_r, \quad r \in \bar{\mathscr{M}}.$$

The cost $\bar{\phi}_{r\bar{w}_r}$ of solution of problem $KP(r, \bar{w}_r)$ is

$$\bar{\phi}_{r\bar{w}_r} = \sum_{i \in N_C} \min_{k \in R_r} \{\beta_{ik}\} \bar{z}_i(r, \bar{w}_r), \quad r \in \bar{\mathscr{M}}.$$

Thus, the cost $\bar{z}(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$ of the $\overline{RF}$ solution $\bar{\boldsymbol{\zeta}}$ obtained from the RF solution $(\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{q}})$ is

$$\bar{z}(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})) = \sum_{r \in \bar{\mathscr{M}}} \bar{\phi}_{r\bar{w}_r} + \sum_{r \in \bar{\mathscr{M}}} g_r + \sum_{i \in N_C} \lambda_i + \sum_{k \in N_S} m_k \mu_k + m^2 \mu_0.$$

The cost $\bar{z}(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$ of the RF solution $(\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{q}})$ is

$$\bar{z}(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})) = \sum_{i \in N_C} \sum_{k \in N_S} \beta_{ik} \bar{\xi}_{ik} + \sum_{r \in \bar{\mathscr{M}}} g_r + \sum_{i \in N_C} \lambda_i + \sum_{k \in N_S} m_k \mu_k + m^2 \mu_0. \tag{6.55}$$

From the definition of θ_{ikr} (see expression (6.52)), we have

$$\bar{\xi}_{ik} = \sum_{r \in \bar{\mathscr{M}}_k} \theta_{ikr}. \tag{6.56}$$

From (6.55) and (6.56) we obtain

$$\bar{z}(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})) = \sum_{r \in \bar{\mathscr{M}}} \Big(\sum_{i \in N_C} \sum_{k \in R_r} \beta_{ik} \theta_{ikr} + g_r \Big) + \sum_{i \in N_C} \lambda_i + \sum_{k \in N_S} m_k \mu_k + m^2 \mu_0. \tag{6.57}$$

It is quite clear that the following inequalities hold

$$\sum_{i \in N_C} \sum_{k \in R_r} \beta_{ik} \theta_{ikr} \geq \sum_{i \in N_C} \min_{k \in R_r} \{\beta_{ik}\} \sum_{k \in R_r} \theta_{ikr}, \quad r \in \bar{\mathscr{M}}. \tag{6.58}$$

Thus, using the definition of $\bar{z}_i(r, \bar{w}_r)$, inequalities (6.58) become

$$\sum_{i \in N_C} \sum_{k \in R_r} \beta_{ik} \theta_{ikr} \geq \sum_{i \in N_C} \min_{k \in R_r} \{\beta_{ik}\} \bar{z}_i(r, \bar{w}_r) = \bar{\phi}_{r\bar{w}_r}, \quad r \in \bar{\mathscr{M}}. \tag{6.59}$$

Finally, from expression (6.57) and inequalities (6.59), we obtain

$$\bar{z}(RF(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})) \geq \sum_{r \in \bar{\mathscr{M}}} (\bar{\phi}_{r\bar{w}_r} + g_r) + \sum_{i \in N_C} \lambda_i + \sum_{k \in N_S} m_k \mu_k + m^2 \mu_0 = \bar{z}(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})). \square$$

Because of Theorem 6.3 and Corollary 1, any optimal 2E-CVRP solution of cost smaller than a known upper bound z_{UB} cannot contain any 2^{nd}-level route $\ell \in \mathscr{R}_k$, $k \in N_S$, of reduced cost $\tilde{c}_{k\ell} \geq z_{UB} - z(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$, where $\tilde{c}_{k\ell}$ is defined as above.

A valid lower bound $LD1$ on the 2E-CVRP can be computed as the cost of a near-optimal solution of problem

$$LD1 = \max_{\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}} \{z(\overline{RF}(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}))\}. \tag{6.60}$$

6.5 Lower bound $LD1$ and bounding procedure DP^1

In this section, we describe a bounding procedure, called DP^1, and a heuristic algorithm based on relaxation $\overline{RF}$ to compute lower and upper bounds $LD1$ and $UB1$, respectively. Procedure DP^1 finds a near-optimal solution of problem (6.60) and uses a DP algorithm to solve problem $\overline{RF}$.

Bounding procedure DP^1 is based on a relaxation of the 2E-CVRP, where the 2^{nd}-level route sets $\mathscr{R}_k$, $k \in N_S$, are enlarged to also contain nonnecessarily elementary routes. The method used by DP^1 to find a feasible solution of inequalities (6.12) is based on the following theorem.

Theorem 6.4. *Let us associate penalties $\lambda_i \in \mathbb{R}$, $i \in N_C$, with constraints (6.2), $\mu_k \in \mathbb{R}_-$, $k \in N_S$, with constraints (6.3), and $\mu_0 \in \mathbb{R}_-$ with constraint (6.4). Let $\widehat{\mathscr{R}}_k \supseteq \mathscr{R}_k$ be the index set of nonnecessarily elementary routes for satellite k. A feasible solution β_{ik} of inequalities (6.12) is given by*

$$\beta_{ik} = q_i \min_{\ell \in \widehat{\mathscr{R}}_{ik}} \left\{ \frac{c_{k\ell} - \sum_{i \in N_C} a_{ik\ell}\lambda_i - \mu_k - \mu_0}{\sum_{i \in N_C} a_{ik\ell} q_i} \right\}, \quad i \in N_C,\ k \in N_S. \tag{6.61}$$

Proof. Consider route $\ell \in \widehat{\mathscr{R}}_k$ of a given satellite $k \in N_S$. From expressions (6.61), we derive

$$\beta_{ik} \leq \frac{q_i(c_{k\ell} - \sum_{i \in N_C} a_{ik\ell}\lambda_i - \mu_k - \mu_0)}{\sum_{i \in N_C} a_{ik\ell} q_i}, \quad i \in R_{k\ell}. \tag{6.62}$$

By summing up inequalities (6.62) for all $i \in R_{k\ell}$, we obtain

$$\begin{aligned} \sum_{i \in R_{k\ell}} a_{ik\ell}\beta_{ik} &\leq \sum_{i \in R_{k\ell}} \frac{a_{ik\ell} q_i(c_{k\ell} - \sum_{i \in N_C} a_{ik\ell}\lambda_i - \mu_k - \mu_0)}{\sum_{i \in N_C} a_{ik\ell} q_i} \\ &= c_{k\ell} - \sum_{i \in N_C} a_{ik\ell}\lambda_i - \mu_k - \mu_0, \quad \ell \in \widehat{\mathscr{R}}_k,\ k \in N_S. \square \end{aligned}$$

In procedure DP^1, the route set $\widehat{\mathscr{R}}_k$ is defined as the set of ng-routes introduced in §4.3.3 that are shortly described below.

Let $N_i \subseteq N_C$, $i \in N_C$, be a set of selected customers for customer i (according to some criterion), such that $N_i \ni i$ and $|N_i| \leq \Delta(N_i)$, where $\Delta(N_i)$ is a parameter. The sets N_i allow us to associate with each path $P = (k, i_1, \ldots, i_t)$ that starts from satellite $k \in N_S$, visits vertices $i_1, \dots, i_t \in N_C$, and ends at vertex i_t, the subset $\Pi(P)$ containing i_t and every customer i_s, $s = 1, \dots, t-1$, of P that belongs to all sets $N_{s+1}, \dots, N_{i_t}$ associated with the customers $i_{s+1}, \dots, i_t$ visited after i_s. The set $\Pi(P)$ is defined as

$$\Pi(P) = \Big\{ i_s \ : \ i_s \in \bigcap_{j=s+1}^{t} N_{i_j}, \ s = 1, \dots, t-1 \Big\} \bigcup \{i_t\}.$$

A forward ng-path (NG, k, q, i) is a nonnecessarily elementary path $P = (k, i_1, \dots, i_{t-1}, i_t = i)$ that starts from satellite $k \in N_S$, ends at customer i, visits a subset of customers of total demand equal to q, and such that $NG = \Pi(P)$ and $i \notin \Pi(P')$, where $P' = (k, i_1, \dots, i_{t-1})$. An (NG, k, q, i)-route (or simply ng-route) is obtained by adding edge $\{i, k\}$ to an ng-path (NG, k, q, i).

Algorithm DP^1 uses column generation to solve equations (6.61) and subgradient optimization to solve problem (6.60).

6.5.1 Description of Procedure DP^1

To solve equations (6.61), procedure DP^1 uses a limited set $\bar{\mathscr{R}}_k \subseteq \widehat{\mathscr{R}}_k$, $k \in N_S$, of ng-routes. Procedure DP^1 initializes each set $\bar{\mathscr{R}}_k$ with all single-customer routes (k, i, k), $i \in N_C$, and sets $\boldsymbol{\lambda} = \mathbf{0}$, $\boldsymbol{\mu} = \mathbf{0}$, $LD1 = 0$ and $UB1 = \infty$. Procedure DP^1 executes an a-priori defined number ($Maxit1$) of macro iterations where, at each macro iteration, the following steps are performed.

(1) Initialize $z^* = 0$, and perform $Maxit2$ iterations of the following steps.

 (i) Compute values β_{ik}, $i \in N_C$, $k \in N_S$, through expression (6.61), where each set $\widehat{\mathscr{R}}_k$ is replaced with set $\bar{\mathscr{R}}_k$, $k \in N_S$.

 (ii) Solve $\overline{RF}$ using values β_{ik} as described in §6.5.2. If $z(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})) > z^*$, then update $z^* = z(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$, $\boldsymbol{\beta^*} = \boldsymbol{\beta}$, $\boldsymbol{\lambda^*} = \boldsymbol{\lambda}$ and $\boldsymbol{\mu^*} = \boldsymbol{\mu}$.

 (iii) Update penalty vector $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ as described in §6.5.3.

(2) Generate a set of ng-routes $\mathscr{N}_k \subseteq \widehat{\mathscr{R}}_k \setminus \bar{\mathscr{R}}_k$, $k \in N_S$, for which inequalities (6.12) are violated by $\boldsymbol{\beta^*}$, $\boldsymbol{\lambda^*}$ and $\boldsymbol{\mu^*}$ as described in §6.5.4. There are two cases

 (i) If $\mathscr{N}_k = \varnothing$, for each satellite $k \in N_S$. If $LD1 < z^*$, then update $LD1 = z^*$, $\boldsymbol{\beta^1} = \boldsymbol{\beta^*}$, $\boldsymbol{\lambda^1} = \boldsymbol{\lambda^*}$, $\boldsymbol{\mu^1} = \boldsymbol{\mu^*}$, and execute the heuristic algorithm described in §6.5.5 producing upper bound z_{UB}. Update $UB1 = \min\{UB1, z_{UB}\}$.

 (ii) If $\mathscr{N}_k \neq \varnothing$, for some satellite $k \in N_S$, then update $\bar{\mathscr{R}}_k = \bar{\mathscr{R}}_k \cup \mathscr{N}_k$.

Notice that $\boldsymbol{\beta^1}$, $\boldsymbol{\lambda^1}$ and $\boldsymbol{\mu^1}$ are the vectors producing lower bound $LD1$ in problem (6.60).

6.5.2 Solving Problem $\overline{RF}$

Problem $\overline{RF}$ can be solved by DP as follows. Let $h(r, w)$ be the optimal solution cost of $\overline{RF}$ obtained by using the 1st-level routes $1, \ldots, r$, $0 \leq r \leq |\mathscr{M}|$, and replacing q_{tot} in equation (6.46) with $w \in \mathbb{Z}_+$, $w^{min} \leq w \leq q_{tot}$. The DP recursion for computing functions $h(r, w)$, $r = 1, \ldots, |\mathscr{M}|$, $w^{min} \leq w \leq q_{tot}$, is

$$h(r,w) = \min\{h(r-1,w), \min_{w^{min} \leq w' \leq \min\{w,\, w_r^{max}\}} \{h(r-1, w-w') + g_r + \phi_{rw'}\}\}. \quad (6.63)$$

The recursion is initialized by setting $h(r, 0) = 0$, $r = 0, \ldots, |\mathscr{M}|$, and $h(0, w) = \infty$, $w = 1, \ldots, q_{tot}$. The $\overline{RF}$ optimal solution cost is $z(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})) = h(|\mathscr{M}|, q_{tot})$.

Let $\bar{z}$ be an upper bound on $z(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$. The number of states (r, w) to generate in order to compute $z(\overline{RF}(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}))$ can be reduced by using bounding functions $lb(r, w)$, described below, to eliminate any state (r, w) that cannot lead to any $\overline{RF}$ solution of cost smaller than $\bar{z}$.

We denote by $lb(r, w)$ a lower bound on the optimal solution cost of problem $\overline{RF}$ where the set $\mathscr{M}$ is replaced with the subset $\{r, r+1, \ldots, |\mathscr{M}|\}$ and q_{tot} with w. Let $\alpha_r = \min_{w^{min} \leq w \leq w_r^{max}} \{(g_r + \phi_{rw})/w\}$, $r \in \mathscr{M}$. By assuming that the routes in the set $\mathscr{M}$ are indexed so that $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_{|\mathscr{M}|}$, functions $lb(r, w)$ can be computed using the following backward recursion.

Initialize (i) $lb(r, 0) = 0$, $r = 1, \ldots, |\mathscr{M}|$, (ii) $lb(r, w) = \infty$, $0 < w < w^{min}$, $r = 1, \ldots, |\mathscr{M}|$, (iii) $lb(|\mathscr{M}|, w) = w\alpha_{|\mathscr{M}|}$, $w \in W_{|\mathscr{M}|}$, and (iv) $lb(|\mathscr{M}|, w) = \infty$, $w_r^{max} < w \leq q_{tot}$.

For each 1st-level route $r = |\mathscr{M}| - 1, |\mathscr{M}| - 2, \ldots, 1$ and each load $w^{min} \leq w \leq q_{tot}$ compute

$$lb(r,w) = \begin{cases} w\alpha_r & \text{if } w \leq w_r^{max} \\ w_r^{max}\alpha_r + lb(r+1, w - w_r^{max}) & \text{if } w_r^{max} + 1 \leq w \leq q_{tot}. \end{cases}$$

Thus, a state (r, w), $r < |\mathscr{M}|$, is fathomed if $h(r, w) + lb(r+1, q_{tot} - w) \geq \bar{z}$.

In performing recursion (6.63), the upper bound $\bar{z}$ is initialized as $\bar{z} = UB1$ and dynamically updated, at the end of stage r, as $\bar{z} = \min\{\bar{z}, h(r, q_{tot})\}$.

6.5.3 Computing a subgradient

Usual backtracking can be used to derive the $\overline{RF}$ solution $\boldsymbol{\zeta}$ of cost $h(|\mathscr{M}|, q_{tot})$. Given $\boldsymbol{\zeta}$ and the sets $V(r, w)$, as defined in §6.4.2, associated to ϕ_{rw}, we derive the index sets $\widetilde{\mathscr{R}}_k \subseteq \bar{\mathscr{R}}_k$, $k \in N_S$, of the 2nd-level routes in solution and the index $\ell(i, k)$ of the route in $\widetilde{\mathscr{R}}_k$ associated with β_{ik} as follows.

(i) Initialize $\widetilde{\mathscr{R}}_k = \varnothing$, $k \in N_S$, and $\ell(i,k) = 0$, $i \in N_C$, $k \in N_S$;

(ii) Repeat the following steps for each 1^{st}-level route $r \in \mathscr{M}$ such that $\zeta_{rw} = 1$ for some load $w \in W_r$

  (a) Compute $\bar{k}(i) = \text{argmin}_{k \in R_r}\{\beta_{ik}\}$, $i \in V(r,w)$. Let $\ell(i,\bar{k}(i))$, $i \in V(r,w)$, be the index of the route in $\widehat{\mathscr{R}}_k$ associated with $\beta_{i\,\bar{k}(i)}$ in expressions (6.61);

  (b) For each $i \in V(r,w)$, set $\widetilde{\mathscr{R}}_{\bar{k}(i)} = \widetilde{\mathscr{R}}_{\bar{k}(i)} \cup \{\ell(i,\bar{k}(i))\}$.

A subgradient to function $z(\overline{RF}(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}))$, at point $(\boldsymbol{\lambda},\boldsymbol{\mu})$, can be computed as follows. Let $\tilde{\boldsymbol{x}}$ be a vector whose components are computed as

$$\tilde{x}_{k\ell} = \sum_{i \in N_C\,:\,\ell(i,k)=\ell} \frac{a_{ik\ell}q_i}{\sum_{i \in N_C} a_{ik\ell}q_i}, \quad \ell \in \widetilde{\mathscr{R}}_k,\ k \in N_S,$$

and let

$$\begin{cases} \alpha_i = \sum_{k \in N_S} \sum_{\ell \in \widetilde{\mathscr{R}}_k} a_{ik\ell}\tilde{x}_{k\ell}, & i \in N_C, \\ \delta_k = \sum_{\ell \in \widetilde{\mathscr{R}}_k} \tilde{x}_{k\ell}, & k \in N_S, \\ \delta_0 = \sum_{k \in N_S} \delta_k. \end{cases}$$

Then, penalty vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are modified as

$$\begin{cases} \lambda_i = \lambda_i - \epsilon\gamma(\alpha_i - 1), & i \in N_C, \\ \mu_k = \min\{0, \mu_k - \epsilon\gamma(\delta_k - m_k)\}, & k \in N_S, \\ \mu_0 = \min\{0, \mu_0 - \epsilon\gamma(\delta_0 - m^2)\}, \end{cases}$$

where ϵ is a positive constant and γ is defined as

$$\gamma = \frac{0.2z(\overline{RF}(\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}))}{\sum_{i \in N_C}(\alpha_i - 1)^2 + \sum_{k \in N_S}(\delta_k - m_k)^2 + (\delta_0 - m^2)^2}.$$

6.5.4 Generating the ng-route set $\mathscr{N}_k$ for a given satellite k

We describe the procedure to generate, for a given satellite $k \in N_S$, the set of ng-routes $\mathscr{N}_k \subseteq \widehat{\mathscr{R}}_k$ that violate inequalities (6.12) for given vectors $\boldsymbol{\beta}^*$, $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$, when the route set $\mathscr{R}_k$ is replaced by the route set $\widehat{\mathscr{R}}_k$.

Define the modified edge costs $\bar{d}_{ij} = d_{ij} - \frac{1}{2}(\beta^*_{ik} + \lambda^*_i + H_k q_i) - \frac{1}{2}(\beta^*_{jk} + \lambda^*_j + H_k q_j)$ and the sets $N_i \subseteq N_C$, $i \in N_C$, to contain the $\Delta(N_i)$ nearest customers to i according to d_{ij}.

Let $f(NG,k,q,i)$ be the cost of a least-cost ng-path (NG,k,q,i) using the modified edge cost $\bar{d}_{ij}$. Functions $f(NG,k,q,i)$ are computed using the DP recursions described in §4.3.3 on the state-space graph $\mathscr{H} = (\mathscr{V},\Psi)$, defined for a given satellite $k \in N_S$, as

$$\mathscr{V} = \{(NG, k, q, i) \,:\, q_i \le q \le Q_2,\, \forall NG \subseteq N_i \text{ s.t. } NG \ni i \text{ and } \sum_{j \in NG} q_j \le q,\, \forall i \in N_C\},$$

$$\Psi = \{((NG', k, q', j), (NG, k, q, i)) \,:\, \forall (NG', k, q', j) \in \Psi^{-1}(NG, k, q, i),\, \forall (NG, k, q, i) \in \mathscr{E}\},$$

where $\Psi^{-1}(NG, k, q, i) = \{(NG', k, q - q_i, j) \,:\, \forall NG' \subseteq N_j \text{ s.t. } NG' \ni j \text{ and } NG' \cap N_i = NG \setminus \{i\},\ j \in N_C \text{ s.t. } \{i, j\} \in E \text{ if } i < j \text{ or } \{j, i\} \in E \text{ if } j < i\}$.

Let $r(i, k) = \min\limits_{(NG,k,q,i) \in \mathscr{V}} \{f(NG, k, q, i) - \mu^*_k - \mu^*_0 + \bar{d}_{ik}\}$ be the cost of a least-cost *ng*-route visiting customer $i \in N_C$ immediately before arriving at satellite k. The route set $\mathscr{N}_k$ contains the *ng*-routes corresponding to $r(i, k) < 0$, for each customer $i \in N_C$.

6.5.5 A Lagrangean heuristic

Procedure DP^1 is interwoven with a heuristic algorithm that produces a feasible 2E-CVRP solution of cost z_{UB} using the 2^{nd}-level route sets $\widetilde{\mathscr{R}}_k$, $k \in N_S$, and vector $\tilde{\boldsymbol{x}}$ associated with an $\overline{RF}$ solution (see §6.5.3). First, the routes in $\widetilde{\mathscr{R}}_k$, $k \in N_S$, are modified with the objective of obtaining a solution vector $\boldsymbol{x}$ satisfying constraints (6.2)-(6.5). Then, the solution vector $\boldsymbol{x}$ is used to derive solution vectors $\boldsymbol{y}$ and $\boldsymbol{q}$ such that $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{q})$ represents a feasible 2E-CVRP solution.

A step-by-step description of the heuristic algorithm is the following.

1) *Initialization.* Let $\widetilde{\mathscr{R}} = \cup_{k \in N_S} \widetilde{\mathscr{R}}_k$. Initialize $SOL = \varnothing$ and $\delta(i) = 0$, $i \in N_C$.

2) *Extract a subset of routes* $SOL \subseteq \widetilde{\mathscr{R}}$. Let ℓ^* be the route of $\widetilde{\mathscr{R}}$ where $\tilde{x}_{\pi_{\ell^*}\ell^*} = \max\{\tilde{x}_{\pi_\ell \ell} \,:\, \ell \in \widetilde{\mathscr{R}}\}$. Remove ℓ^* from $\widetilde{\mathscr{R}}$. If $\delta(i) = 0$, for some $i \in R_{\pi_{\ell^*}\ell^*}$, then update $SOL = SOL \cup \{\ell^*\}$ and $\delta(i) = \delta(i) + 1$, $i \in R_{\pi_{\ell^*}\ell^*}$. Repeat Step 1 until $\widetilde{\mathscr{R}} = \varnothing$.

3) *Modify the route set SOL.* Remove from SOL any route $\ell \in SOL$ such that $\delta(i) > 1$, $i \in R_{\pi_\ell \ell}$, and update $\delta(i) = \delta(i) - 1$. For each $\ell \in SOL$, compute the savings that can be achieved by removing from route ℓ every customer $i \in R_{\pi_\ell \ell}$ having $\delta(i) > 1$. Let $\ell^* \in SOL$ be the route of maximum saving. Remove from route ℓ^* every customer $i \in R_{\pi_{\ell^*}\ell^*}$ with $\delta(i) > 1$, and update $\delta(i) = \delta(i) - 1$. Repeat Step 3 until $\delta(i) \le 1$, for each $i \in N_C$.

4) *Insert unrouted customers.* For each unrouted customer i (i.e., $\delta(i) = 0$) perform the following operations. Compute the minimum extra-mileage $exm(i, \ell)$ for inserting i in route $\ell \in SOL$. We set $exm(i, \ell) = \infty$ if the total load of the resulting route ℓ exceeds the vehicle capacity Q_2. Let ℓ^* be such that $exm(i, \ell^*) = \min_{\ell \in SOL}[exm(i, \ell)]$. If $exm(i, \ell^*) = \infty$, then set $z_{UB} = \infty$ and stop; otherwise, insert customer i in route ℓ^* in the position of cost $exm(i, \ell^*)$, and set $\delta(i) = 1$.

5) *Define the F solution $\boldsymbol{x}$.* Define $x_{\pi_\ell \ell} = 1$, for each route $\ell \in SOL$, and $x_{\pi_\ell \ell} = 0$, for each route $\ell \in \mathscr{R} \setminus SOL$. If $\boldsymbol{x}$ does not satisfy constraints (6.3)-(6.5), then set $z_{UB} = \infty$ and the algorithm terminates.

6) *Improve the cost of the routes in SOL.* The post-optimization procedure for improving the total cost of the routes in SOL applies the following procedure in the order specified below.

   (a) *Exchange of one customer between two routes of SOL.* For each customer $i \in N_C$, compute the saving $move(i, \ell)$ achieved by removing i from its current route ℓ_i and inserting i in the least-cost position of route $\ell \in SOL$. Set $move(i, \ell_i) = 0$ and $move(i, \ell) = -\infty$ if the load of the resulting route ℓ violates constraint (6.5) for satellite π_ℓ or if customer i cannot be inserted in route ℓ without violating the vehicle capacity Q_2. Let i^* and ℓ^* be determined such that $move(i^*, \ell^*) = \max[move(i, \ell) : i \in N_C, \ \ell \in SOL]$. If $move(i^*, \ell^*) > 0$, then remove customer i^* from its current route and insert it in the best position of route ℓ^*. This procedure is repeated until $move(i^*, \ell^*) \leq 0$.

   (b) *Exchange of two customers between two routes of SOL.* For all pairs of routes $\ell, \ell' \in SOL$ and for each pair of customers $i \in R_{\pi_\ell \ell}$ and $j \in R_{\pi_{\ell'} \ell'}$, compute the saving $sav(i, j)$ obtained by moving customer i from route ℓ to route ℓ' and customer j from route ℓ' to route ℓ. We set $sav(i, j) = -\infty$ if the exchange violates constraint (6.5) for one of the two satellites $\pi_\ell, \pi_{\ell'}$ or if the total load of one of the two routes exceeds the vehicle capacity Q_2. The two customers, i^* and j^*, producing the maximum saving are then exchanged if $sav(i^*, j^*) > 0$. This procedure is repeated until $sav(i^*, j^*) \leq 0$. Whenever this procedure improves the solution, then the post-optimization routing is restarted from the beginning.

   (c) Optimize each route $\ell \in SOL$ using a 3-optimal method.

7) *Constructing a feasible 2E-CVRP solution.* Let $\omega_k = \sum_{\ell \in \mathscr{R}_k} w_{k\ell} x_{k\ell}$, $k \in N_S$, be the total demand associated with satellite k by the solution vector $\boldsymbol{x}$ defined above. We solve the following integer problem to optimality

$$
\begin{aligned}
(F(\boldsymbol{x})) \quad z(F(\boldsymbol{x})) = \min \ & \sum_{r \in \mathscr{M}} g_r y_r \\
s.t. \ & \sum_{r \in \mathscr{M}} y_r \leq m^1, \\
& \sum_{r \in \mathscr{M}_k} q_{kr} = \omega_k, && k \in N_S, \\
& \sum_{k \in R_r} q_{kr} \leq Q_1 y_r, && r \in \mathscr{M}, \\
& y_r \in \{0, 1\}, && r \in \mathscr{M}, \\
& q_{kr} \geq 0, && k \in R_r, \, r \in \mathscr{M}.
\end{aligned}
$$

Problem $F(\boldsymbol{x})$ is solved to optimality with an integer programming solver. Let $(\boldsymbol{y}, \boldsymbol{q})$ be the optimal $F(\boldsymbol{x})$ solution (we assume $z(F(\boldsymbol{x})) = \infty$ if problem $F(\boldsymbol{x})$ does not admit a feasible solution). If problem $F(\boldsymbol{x})$ admits a feasible solution, then the vectors $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{q})$ represents a feasible 2E-CVRP solution of cost $z_{UB} = z(F(\boldsymbol{x})) + \sum_{k \in N_S} \sum_{\ell \in \mathscr{R}_k} c_{k\ell} x_{k\ell}$.

6.6 An exact method for solving the 2E-CVRP

The method for solving the 2E-CVRP is based on the following re-formulation of problem F.

Let $\mathscr{P} = \{M \subseteq \mathscr{M} \,:\, |M|Q_1 \geq q_{tot},\, |M| \leq m^1\}$. We call *configuration* each element M of the set $\mathscr{P}$. For each configuration $M \in \mathscr{P}$, let $N_S(M) = \bigcup_{r \in M} R_r$, $M_k = M \cap \mathscr{M}_k$, $k \in N_S$, and $U(M) = \sum_{r \in M} g_r$. An optimal 2E-CVRP solution can be computed as

$$z(F) = \min_{M \in \mathscr{P}} \left\{ U(M) + z(F(M)) \right\},$$

where $z(F(M))$ is the optimal solution cost of the following problem $F(M)$

$$
\begin{aligned}
(F(M)) \quad z(F(M)) = \min \; & \sum_{k \in N_S(M)} \sum_{\ell \in \mathscr{R}_k} c_{k\ell} x_{k\ell} \\
s.t. \; & \sum_{k \in N_S(M)} \sum_{\ell \in \mathscr{R}_{ik}} x_{k\ell} = 1, && i \in N_C, \\
& \sum_{\ell \in \mathscr{R}_k} x_{k\ell} \leq m_k, && k \in N_S(M), \\
& \sum_{k \in N_S(M)} \sum_{\ell \in \mathscr{R}_k} x_{k\ell} \leq m^2, \\
& \sum_{\ell \in \mathscr{R}_k} w_{k\ell} x_{k\ell} \leq B_k, && k \in N_S(M), \\
& \sum_{r \in M_k} q_{kr} = \sum_{\ell \in \mathscr{R}_k} w_{k\ell} x_{k\ell}, && k \in N_S(M), \\
& \sum_{k \in R_r} q_{kr} \leq Q_1, && r \in M, \\
& x_{k\ell} \in \{0, 1\}, && k \in N_S(M),\ \ell \in \mathscr{R}_k, \\
& q_{kr} \geq 0, && k \in R_r,\ r \in M.
\end{aligned}
$$

We assume $z(F(M)) = \infty$ if $F(M)$ has no feasible solution for configuration $M \in \mathscr{P}$.

Problem $F(M)$ is an extension of the *multi-depot capacitated vehicle routing problem* (MDCVRP) considered by Baldacci and Mingozzi [2009].

The exact method we propose (i) generates the set $\mathscr{M}$ of 1^{st}-level routes and executes bounding procedure DP^1 to compute lower bound $LD1$ corresponding to the optimal cost of problem $\overline{RF}(\boldsymbol{\beta^1}, \boldsymbol{\lambda^1}, \boldsymbol{\mu^1})$ and upper bound $UB1$; (ii) then, generates the set $\mathscr{P}$ of configurations and limits its size by using $LD1$, $UB1$ and vectors $(\boldsymbol{\beta^1}, \boldsymbol{\lambda^1}, \boldsymbol{\mu^1})$; (iii) finally, for each configuration $M \in \mathscr{P}$, solves the associated problem $F(M)$.

The last two steps of the exact method are described in detail in the following.

6.6.1 Generating the Set of Configurations $\mathscr{P}$

The generation of the set $\mathscr{P}$ of configurations is based on the following propositions.

Let LB_R be a lower bound on the 2^{nd}-level routing cost of any optimal 2E-CVRP solution computed as $LB_R = \sum_{i \in N_C} \min_{k \in N_S}\{\beta^1_{ik}\} + \sum_{i \in N_C} \lambda^1_i + \sum_{k \in N_S} m_k \mu^1_k + m^2 \mu^1_0$, and let $LBW(M)$ be a lower bound on $z(F(M))$ computed as $LBW(M) = \sum_{i \in N_C} \min_{k \in N_S(M)}\{\beta^1_{ik}\} + \sum_{i \in N_C} \lambda^1_i + \sum_{k \in N_S(M)} m_k \mu^1_k + m^2 \mu^1_0$.

Proposition 1. Let z_{UB} be a valid upper bound on the 2E-CVRP. A configuration $M \in \mathscr{P}$ can belong to an optimal 2E-CVRP solution if and only if it satisfies the following conditions

$$\left.\begin{array}{ll}
|R_r \cap R_{r'}| \le 1,\ r, r' \in M, r \neq r', & (a)\\
\sum_{r \in M} \min\{Q_1, \sum_{k \in R_r} m_k Q_2\} \ge q_{tot}, & (b)\\
\sum_{r \in M} \sum_{k \in R_r} m_k \ge \lceil q_{tot}/Q_2 \rceil, & (c)\\
U(M) < z_{UB} - LB_R, & (d)\\
U(M) < z_{UB} - LBW(M). & (e)
\end{array}\right\} \qquad (6.64)$$

Condition (a) is a property of the feasible solutions of the *split delivery vehicle routing problem* (see Dror and Trudeau [1990]). Conditions (b) and (c) are feasibility conditions. Conditions (d) and (e) follow from the properties of any optimal 2E-CVRP solution of cost less than z_{UB}.

Proposition 2. For a given configuration $M \in \mathscr{P}$, let $\theta(k)$, $k \in N_S(M)$, be a lower bound on the quantity that must be supplied to satellite k in any feasible $F(M)$ solution by the 1^{st}-level routes $M_k \subseteq M$ passing through satellite k. Problem $F(M)$ has no feasible solution if either $\sum_{k \in N_S(M)} \lceil \theta(k)/Q_2 \rceil > m^2$ or $\lceil \theta(k)/Q_2 \rceil > m_k$, for some satellite $k \in N_S$. In this case, configuration M can be removed from $\mathscr{P}$.

Lower bound $\theta(k)$ can be computed as the optimal solution cost of the following problem

$$\theta(k) = \min \sum_{r \in M_k} q_{kr} \tag{6.65}$$

$$s.t. \sum_{h \in R_r} q_{hr} \leq Q_1, \qquad r \in M, \tag{6.66}$$

$$\sum_{r \in M} \sum_{h \in R_r} q_{hr} = q_{tot}, \tag{6.67}$$

$$\sum_{r \in M_h} q_{hr} \geq q^{min}, \qquad h \in N_S(M), \tag{6.68}$$

$$\sum_{r \in M_h} q_{hr} \leq m_k Q_2, \qquad h \in N_S(M), \tag{6.69}$$

$$q_{hr} \geq 1, \qquad h \in R_r,\ r \in M, \tag{6.70}$$

where $q^{min} = \max\{\min_{i \in N_C}\{q_i\}, q_{tot} - (m^2 - 1)Q_2\}$. We assume $\theta(k) = \infty$ if problem (6.65)-(6.70) has no feasible solution.

The set $\mathscr{P}$ is generated by pure enumeration by using Propositions 1 and 2 to eliminate any configuration M that cannot lead to an optimal 2E-CVRP solution.

6.6.2 Solving Problem $F(M)$

Problem $F(M)$ is solved with the following three-phase method. In the *first phase*, bounding procedure DP^1 is used to compute lower bound $LD1(M)$ on $z(F(M))$ by replacing $\mathscr{M}$ with M. In the *second phase*, a near-optimal dual solution of the LP-relaxation of $F(M)$ strengthened by valid inequalities, called problem $\bar{F}(M)$, is computed. In the *third phase*, the $\bar{F}(M)$ dual solution is used to generate the subsets $\mathscr{R}'_k \subseteq \mathscr{R}_k$, $k \in N_S(M)$, of all 2nd-level routes of any $F(M)$ optimal solution. An $F(M)$ optimal solution is obtained by replacing, in $F(M)$, each set $\mathscr{R}_k$ with $\mathscr{R}'_k$, $k \in N_S(M)$, and solving the resulting problem, called $F'(M)$, with an integer programming solver.

6.6.2.1 Phase 1: computing lower bound $LD1(M)$

We execute bounding procedure DP^1, by replacing the set $\mathscr{M}$ with M, to compute lower bound $LD1(M)$ and upper bound $UB1(M)$ on $F(M)$. If $LD1(M)$ is greater than a known upper bound on the 2E-CVRP, Phases 2 and 3 are skipped.

6.6.2.2 Phase 2: solving $\bar{F}(M)$

Problem $\bar{F}(M)$ corresponds to the LP-relaxation of problem $F(M)$ strengthened with the following valid inequalities

a) *Capacity constraints.* Let $\mathscr{S} = \{H : H \subseteq N_C, |H| \geq 2\}$. The capacity constraints are

$$\sum_{k \in N_S(M)} \sum_{\ell \in \mathscr{R}_k : R_{k\ell} \cap H \neq \varnothing} x_{k\ell} \geq \left\lceil \frac{\sum_{i \in H} q_i}{Q_2} \right\rceil, \quad H \in \mathscr{S}. \tag{6.71}$$

b) *Clique inequalities.* Let $\mathscr{R}(M) = \bigcup_{k \in N_S(M)} \mathscr{R}_k$, and let $\mathscr{G} = (\mathscr{R}(M), \mathscr{E})$ be the conflict graph associated with the route set $\mathscr{R}(M)$, where the edge set $\mathscr{E}$ contains every edge $\{\ell, \ell'\}$, $\ell, \ell' \in \mathscr{R}(M)$, such that $\ell < \ell'$ and $R_{\pi_\ell \ell} \cap R_{\pi_{\ell'} \ell'} \neq \varnothing$. Let $\mathcal{C}$ be the set of all cliques of graph $\mathscr{G}$. The clique inequalities are

$$\sum_{\ell \in C} x_{\pi_\ell \ell} \leq 1, \quad C \in \mathscr{C}. \tag{6.72}$$

Problem $\bar{F}(M)$ is solved with a CCG procedure that starts by setting $\mathscr{S} = \varnothing$, $\mathscr{C} = \varnothing$. The master problem is initialized with a set of elementary routes obtained from the final set of ng-routes generated in Phase 1 for computing lower bound $LD1(M)$ by removing, from each ng-route, the customers visited more than once. At each iteration, a set of negative reduced cost routes are generated and a set of violated inequalities (6.71) and (6.72) are added as described in Baldacci and Mingozzi [2009]. The procedure ends when no negative reduced cost routes exist and no inequalities (6.71) and (6.72) are violated and provides an $\bar{F}(M)$ dual solution of cost $z(\bar{F}(M))$.

6.6.2.3 Phase 3: solving $F(M)$ to optimality

In Phase 3, two steps are performed.

(1) Define the reduced problem $F'(M)$ resulting from $F(M)$ by

  (i) Replacing the route set $\mathscr{R}_k$, $k \in N_S(M)$, with the largest subset $\mathscr{R}'_k \subseteq \mathscr{R}_k$ of routes such that $c'_{k\ell} < z_{UB} - (U(M) + z(\bar{F}(M)))$, $\ell \in \mathscr{R}'_k$, $k \in N_S(M)$, where c'_ℓ is the reduced cost of route $\ell \in \mathscr{R}'_k$ with respect to the $\bar{F}(M)$ dual solution achieved at Phase 2 and z_{UB} is the current best upper bound on the 2E-CVRP.

  (ii) Adding all constraints (6.71) and (6.72) saturated by the final $\bar{F}(M)$ solution.

(2) Solve problem $F'(M)$ with a general purpose integer programming solver.

6.6.3 Description of the Exact Method

The exact method we propose for solving the 2E-CVRP can be described as follows.

(1) *Generate the set $\mathscr{M}$ and compute a lower bound on the 2E-CVRP.*

- Generate the set $\mathscr{M}$ of 1st-level routes by pure enumeration.
- Execute bounding procedure DP^1 to produce lower and upper bounds $LD1$ and $UB1$.

(2) *Generate the set $\mathscr{P}$ of configurations as described in §6.6.1.*

(3) *Solve the 2E-CVRP.*

(a) Initialize $z(F) = UB1$, $LB = UB1$, $z_{UB} = UB1$, $\bar{\mathscr{P}} = \varnothing$ and $r^{max} = 0$.

(b) If $\mathscr{P} = \varnothing$, then Stop.
Let $M = \text{argmin}_{M' \in \mathscr{P}}\{LBW(M')\}$. Remove M from $\mathscr{P}$. If $LBW(M) \geq z(F)$ then Stop ($z(F)$ is the optimal 2E-CVRP solution cost).

(c) *Solve problem $F(M)$.*

(i) Execute Phase 1 (see §6.6.2.1) to compute lower bound $LD1(M)$ and upper bound $UB1(M)$ on $F(M)$. Update $z_{UB} = \min\{z_{UB},\ UB1(M)\}$, $z(F) = \min\{z(F),\ UB1(M)\}$, and $LB = \min\{LB,\ z(F)\}$.
If $LD1(M) \geq z(F)$, go to Step 3.b.

(ii) Execute Phase 2 (see §6.6.2.2) to compute lower bound $z(\bar{F}(M))$, and update $LB = \min\{LB,\ \max\{LD1(M),\ U(M) + z(\bar{F}(M))\}\}$.
If $U(M) + z(\bar{F}(M)) \geq z(F)$, go to Step 3.b.
If the $\bar{F}(M)$ solution of cost $z(\bar{F}(M))$ is integer, update $z(F) = U(M) + z(\bar{F}(M))$ and go to Step 3.b.

(iii) Execute Phase 3 (see §6.6.2.3) to solve problem $F(M)$. Let $z(F(M))$ be the optimal solution cost of $F'(M)$. Update $z(F) = \min\{z(F),\ U(M) + z(F(M))\}$, $\bar{\mathscr{P}} = \bar{\mathscr{P}} \cup \{M\}$ and $r^{max} = \max\{r^{max},\ \sum_{k \in N_S(M)} |\mathscr{R}'_k|\}$. Go to Step 3.b.

Notice that Step 3.c-(iii) is executed for any configuration $M \in \mathscr{P}$ such that $U(M) + z(\bar{F}(M)) < z(F)$ and the $\bar{F}(M)$ solution is not integer. Thus, if Step 3.c-(iii) is never executed, the algorithm terminates with $\bar{\mathscr{P}} = \varnothing$, implying that $LB = z(F)$ and the optimal 2E-CVRP solution corresponds to either the initial upper bound $UB1(M)$ computed at Step 3.c-(i) or to the integer $\bar{F}(M)$ solution achieved at Step 3.c-(ii) for some configuration $M \in \mathscr{P}$.

At the end of the exact method, LB represents a valid lower bound on the 2E-CVRP because it corresponds to $LB = \min_{M \in \mathscr{P}}\{\max\{LD1(M),\ U(M) + z(\bar{F}(M))\}\}$. Value r^{max} is the maximum number of 2nd-level routes generated, and set $\bar{\mathscr{P}}$ contains the configurations for which the corresponding problem $F(M)$ was solved to optimality at Step 3.c-(iii). Upper bound z_{UB} is the cost of the best upper bound computed at Step 1 or at Step 3.c-(i). Finally, because we impose a limit Δ^{max} on the maximum number of 2nd-level routes, $\bigcup_{k \in N_S(M)} \mathscr{R}'_k$, to generate at Step 3.c-(iii), whenever such limit is reached for some configuration $M \in \bar{\mathscr{P}}$, at the end of the algorithm, the value $z(F)$ is an upper bound on the 2E-CVRP but is not necessarily the optimal solution cost.

Table 6.1: Satellite coordinates of Set 6

Name	n_S		$k=1$		$k=2$		$k=3$		$k=4$		$k=5$		$k=6$
A-n51-4	4	7	(21.0,47.0)	11	(51.0,21.0)	17	(52.0,41.0)	42	(10.0,17.0)				
A-n51-5	5	5	(20.0,26.0)	7	(21.0,47.0)	16	(36.0,16.0)	23	(42.0,57.0)	39	(45.0,35.0)		
A-n51-6	6	3	(49.0,49.0)	5	(20.0,26.0)	15	(12.0,42.0)	16	(36.0,16.0)	49	(25.0,55.0)	50	(48.0,28.0)
A-n76-4	4	30	(52.0,26.0)	36	(55.0,50.0)	45	(21.0,48.0)	74	(27.0,24.0)				
A-n76-5	5	33	(22.0,53.0)	49	(48.0,21.0)	53	(54.0,38.0)	54	(55.0,57.0)	74	(27.0,24.0)		
A-n76-6	6	6	(55.0,20.0)	13	(35.0,51.0)	17	(21.0,36.0)	53	(54.0,38.0)	54	(55.0,57.0)	63	(30.0,20.0)
A-n101-4	4	8	(20.0,50.0)	22	(45.0,20.0)	51	(47.0,47.0)	96	(25.0,24.0)				
A-n101-5	5	3	(35.0,17.0)	51	(47.0,47.0)	55	(57.0,29.0)	83	(15.0,47.0)	90	(26.0,35.0)		
A-n101-6	6	5	(55.0,20.0)	19	(20.0,40.0)	52	(49.0,58.0)	62	(12.0,24.0)	69	(56.0,39.0)	98	(25.0,21.0)
B-n51-4	4	7	(21.0,47.0)	11	(51.0,21.0)	17	(52.0,41.0)	42	(10.0,17.0)				
B-n51-5	5	5	(20.0,26.0)	7	(21.0,47.0)	16	(36.0,16.0)	23	(42.0,57.0)	39	(45.0,35.0)		
B-n51-6	6	3	(49.0,49.0)	5	(20.0,26.0)	15	(12.0,42.0)	16	(36.0,16.0)	49	(25.0,55.0)	50	(48.0,28.0)
B-n76-4	4	30	(52.0,26.0)	36	(55.0,50.0)	45	(21.0,48.0)	74	(27.0,24.0)				
B-n76-5	5	33	(22.0,53.0)	49	(48.0,21.0)	53	(54.0,38.0)	54	(55.0,57.0)	74	(27.0,24.0)		
B-n76-6	6	6	(55.0,20.0)	13	(35.0,51.0)	17	(21.0,36.0)	53	(54.0,38.0)	54	(55.0,57.0)	63	(30.0,20.0)
B-n101-4	4	8	(20.0,50.0)	22	(45.0,20.0)	51	(47.0,47.0)	96	(25.0,24.0)				
B-n101-5	5	3	(35.0,17.0)	51	(47.0,47.0)	55	(57.0,29.0)	83	(15.0,47.0)	90	(26.0,35.0)		
B-n101-6	6	5	(55.0,20.0)	19	(20.0,40.0)	52	(49.0,58.0)	62	(12.0,24.0)	69	(56.0,39.0)	98	(25.0,21.0)
C-n51-4	4	22	(62.0,42.0)	26	(7.0,38.0)	32	(37.0,69.0)	46	(39.0,10.0)				
C-n51-5	5	26	(7.0,38.0)	32	(37.0,69.0)	37	(63.0,69.0)	40	(59.0,15.0)	44	(5.0,64.0)		
C-n51-6	6	22	(62.0,42.0)	26	(7.0,38.0)	32	(37.0,69.0)	34	(46.0,10.0)	37	(63.0,69.0)	44	(5.0,64.0)
C-n76-4	4	11	(40.0,66.0)	25	(7.0,43.0)	55	(67.0,41.0)	62	(36.0, 6.0)				
C-n76-5	5	55	(67.0,41.0)	56	(10.0,70.0)	57	(6.0,25.0)	61	(64.0, 4.0)	67	(57.0,72.0)		
C-n76-6	6	19	(9.0,56.0)	32	(31.0,76.0)	57	(6.0,25.0)	58	(65.0,27.0)	60	(70.0,64.0)	62	(36.0, 6.0)
C-n101-4	4	30	(64.0,42.0)	33	(35.0,69.0)	42	(42.0, 7.0)	46	(6.0,38.0)				
C-n101-5	5	30	(64.0,42.0)	42	(42.0, 7.0)	46	(6.0,38.0)	65	(15.0,77.0)	66	(62.0,77.0)		
C-n101-6	6	16	(30.0, 5.0)	30	(64.0,42.0)	46	(6.0,38.0)	65	(15.0,77.0)	66	(62.0,77.0)	68	(67.0, 5.0)

6.7 Computational results

We report on the computational results of the exact method (hereafter BMRW) described in §6.6.3 and its comparison with the methods of Perboli et al. [2011] (hereafter PTV) and Jepsen et al. [2011] (hereafter JSR). BMRW was coded in Fortran 77. CPLEX 12.1 was used as the linear programming and integer programming solver. All tests were run on an IBM Intel Xeon X7350 Server (2.93 GHz - 16 GB of RAM).

We considered four sets of instances from the literature: Set 2 and 3 from by Gonzales Feliu et al. [2007a], Set 4 from Crainic et al. [2010], and Set 5 from Hemmelmayr et al. [2011]. Sets 2 to 4 are available at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/vrp2einfo.html`, whereas Set 5 was kindly provided by the authors. On all instances, travel costs are computed as real Euclidean distances. The 39 instances of Sets 2 and 3 feature 21, 32 or 50 customers and 2 or 4 satellites; in Set 2, satellites are randomly spread on the plane, whereas, in Set 3, satellites are located peripherally. Set 4 consists of 54 randomly generated instances with 50 customers and 2, 3 or 5 satellites; different criteria were adopted to spread satellites and customers on the plane. Set 5 is made up of 18 instances obtained by adapting LRP benchmark

TABLE 6.2: Computational results on Set 2 instances

Name	n_s	$z(F)$	$\%LD1$	t_{LD1}	$\lvert\mathscr{P}\rvert$	$\%UB$	$\%LB$	t_{LB}	$\lvert\bar{\mathscr{P}}\rvert$	r^{max}	t_{tot}
E-n22-k4-s6-17	2	417.07	99.9	0.4	1	100.0	100.0	0.5	0	0	0.5
E-n22-k4-s8-14	2	384.96	99.5	0.4	1	100.0	100.0	0.7	0	0	0.7
E-n22-k4-s9-19	2	470.60	95.4	0.5	1	100.0	100.0	1.2	0	0	1.2
E-n22-k4-s10-14	2	371.50	99.6	0.5	1	100.0	100.0	0.5	0	0	0.5
E-n22-k4-s11-12	2	427.22	96.5	0.4	2	100.5	100.0	1.3	0	0	1.3
E-n22-k4-s12-16	2	392.78	96.7	0.5	2	100.0	100.0	1.1	0	0	1.1
E-n33-k4-s1-9	2	730.16	97.9	25.1	1	100.0	100.0	37.6	0	0	37.6
E-n33-k4-s2-13	2	714.63	97.8	27.8	2	100.0	100.0	34.9	0	0	34.9
E-n33-k4-s3-17	2	707.48	95.0	28.9	3	105.8	100.0	48.1	0	0	48.1
E-n33-k4-s4-5	2	778.74	94.1	23.1	4	100.9	100.0	72.5	0	0	72.5
E-n33-k4-s7-25	2	756.85	96.8	27.4	3	101.0	100.0	47.1	0	0	47.1
E-n33-k4-s14-22	2	779.05	98.7	26.0	3	100.0	100.0	31.7	0	0	31.7
E-n51-k5-s3-18	2	597.49	93.5	3.0	5	100.0	99.8	23.7	1	33,547	25.8
E-n51-k5-s5-47	2	530.76	98.1	3.1	4	101.6	99.8	25.9	1	34,110	27.5
E-n51-k5-s7-13	2	554.81	94.6	3.3	6	100.2	98.9	37.3	2	42,075	55.1
E-n51-k5-s12-20	2	581.64	95.5	3.1	3	100.5	99.3	27.1	1	39,033	44.3
E-n51-k5-s28-48	2	538.22	95.8	3.2	6	100.0	99.7	40.1	2	34,797	44.0
E-n51-k5-s33-38	2	552.28	95.4	3.8	3	100.0	100.0	13.6	0	0	13.6
E-n51-k5-s3-5-18-47	4	530.76	96.6	6.6	55	100.0	99.9	259.2	1	62,913	260.8
E-n51-k5-s7-13-33-38	4	531.92	94.7	7.6	68	100.0	99.4	263.6	1	68,796	266.6
E-n51-k5-s12-20-28-48	4	527.63	95.6	9.0	24	100.0	99.6	71.8	1	67,620	74.2

instances; among them, we only considered the 6 instances with 5 satellites and 100 customers.

We generated another set of 27 instances (called Set 6) by starting from the CVRP instances E-n51-k5, E-n76-k10 and E-n101-k14 (available at http://branchandcut.org/VRP/data). For each CVRP instance, we generated nine 2E-CVRP instances by maintaining customer locations and demands and vehicle capacity (Q_2) of the original CVRP instance and by setting $Q_1 = 4Q_2$. The 27 instances are divided in 3 classes (A, B and C), where, in classes A and C, the depot is on the bottom-left-hand corner at coordinates $(1, 1)$ and, in class B, the depot location coincides with that of the original CVRP instance. The number of satellites is 4, 5 or 6, and their locations coincide with those of some customers. In class C, satellites are peripheral to customers, whereas, in classes A and B, they are positioned in the middle of the customers.

Table 6.1 reports the details of the satellite coordinates of the 27 instances of the Set 6. The table reports, for each instance, the name of the instance (*Name*), the number of satellites (n_S) and the list of the satellite coordinates. For each satellite $k \in N_S$, the index of the corresponding customer in the original CVRP instance and the corresponding X and Y coordinates are reported.

In all instances, the handling costs are 0, and satellite capacities are unlimited (i.e., $H_k = 0$ and $W_k = \infty$, $k \in N_S$). Furthermore, in Sets 2, 3 and 5 the maximum number of vehicles per satellite is unlimited (i.e., $m_k = \infty$, $k \in N_S$). Set 4 was treated differently by Jepsen et al. [2011] who considered the given upper bounds on

TABLE 6.3: Computational results on Set 3 instances

Name	n_s	$z(F)$	$\%LD1$	t_{LD1}	$\lvert\mathscr{P}\rvert$	$\%UB$	$\%LB$	t_{LB}	$\lvert\bar{\mathscr{P}}\rvert$	r^{max}	t_{tot}
E-n22-k4-s13-14	2	526.15	96.4	0.4	4	100.0	100.0	2.1	0	0	2.1
E-n22-k4-s14-19	2	498.80	93.2	0.5	6	100.0	100.0	2.4	0	0	2.4
E-n22-k4-s13-16	2	521.09	94.9	0.4	4	100.0	100.0	2.8	0	0	2.8
E-n22-k4-s17-19	2	512.80	95.5	0.5	4	100.0	100.0	2.6	0	0	2.6
E-n22-k4-s13-17	2	496.38	96.8	0.5	1	100.0	100.0	1.2	0	0	1.2
E-n22-k4-s19-21	2	520.42	94.9	0.5	5	100.0	100.0	3.8	0	0	3.8
E-n33-k4-s22-26	2	680.37	94.7	28.2	3	100.1	99.8	71.8	1	23,690	73.3
E-n33-k4-s16-22	2	672.17	92.0	32.6	5	102.0	99.4	115.4	1	26,474	127.5
E-n33-k4-s16-24	2	666.02	94.6	38.0	5	100.1	99.9	125.2	1	23,040	128.4
E-n33-k4-s24-28	2	670.43	95.6	31.0	3	100.0	100.0	73.1	1	24,266	78.8
E-n33-k4-s19-26	2	680.37	94.0	27.3	3	100.1	99.6	70.8	2	22,549	72.8
E-n33-k4-s25-28	2	650.58	95.7	30.7	3	100.3	100.0	56.0	0	0	56.0
E-n51-k5-s13-19	2	560.73	95.6	3.3	5	100.0	99.6	43.8	2	34,800	48.0
E-n51-k5-s13-42	2	564.45	97.8	3.6	1	100.3	99.1	18.3	1	44,083	50.1
E-n51-k5-s13-44	2	564.45	96.8	3.2	3	101.4	99.0	29.7	1	50,444	73.0
E-n51-k5-s40-42	2	746.31	91.2	3.6	5	102.6	99.0	34.8	1	53,016	107.2
E-n51-k5-s41-42	2	771.56	97.7	5.7	2	100.1	98.9	36.9	1	315,861	2,078.6
E-n51-k5-s41-44	2	802.91	91.8	4.1	4	101.2	99.6	39.8	1	39,979	59.4

the maximum number of vehicles per satellite, m_k, and Perboli et al. [2011] who ignored such values. To compare BMRW with both JSR and PTV, we considered two versions of Set 4, namely Set 4A and Set4B, where Set 4A corresponds to the original Set 4 whereas, in Set 4B, m_k is unbounded (i.e., $m_k = \infty$, $k \in N_S$).

Perboli et al. [2011] considered and solved to optimality 66 instances with 12 customers and 2 satellites (therein Set 1), as well. We do not report the results of BMRW on such instances because they were easily solved in a few seconds.

According to SPEC (`http://www.spec.org/benchmarks.html`), our machine is 10% faster than the Intel(R) Xeon X5550 2.67 GHz with 24 GB of memory and 8 cores used by JSR and twice as fast as the 3 GHz Pentium PC with 1 GB of Ram used by PTV. A time limit of 10,000 seconds was imposed on PTV and JSR.

In testing BMRW, we used the following parameter settings. In procedure DP^1, we set $\Delta(N_i) = 12$, and we set $Maxit1 = 25$, $\epsilon = 1.0$, $Maxit2 = 200$, at Step 1, and $Maxit1 = 10$, $\epsilon = 0.5$, $Maxit2 = 100$ at Step 3.c-(i). Moreover, we set $\Delta^{max} = 10^6$ and imposed a time limit of 5,000 seconds to solve problem $F'(M)$.

Tables 6.2-6.7 report the results obtained by BMRW on the 6 sets of instances. The tables report the instance name, the number n_s of satellites, the cost $z(F)$ of the best solution found, the percentage ratio $\%LD1$ of lower bound $LD1$ over $z(F)$ (i.e., $\%LD1 = 100\, LD1/z(F)$), the time t_{LD1} in seconds for computing $LD1$, the cardinality $|\mathscr{P}|$ of the set $\mathscr{P}$ at the beginning of Step 3, the percentage ratio $\%UB$ of upper bound z_{UB} over $z(F)$, the percentage ratio $\%LB$ of lower bound LB over $z(F)$, the total time t_{LB} in seconds for computing $LD1$ and LB, the cardinality $|\bar{\mathscr{P}}|$ of the set $\bar{\mathscr{P}}$, the value of r^{max}, and finally the total computing time t_{tot} in seconds. Whenever $|\bar{\mathscr{P}}| > 0$, the difference $t_{tot} - t_{LB}$ is the time spent by CPLEX for solving the problems $F'(M)$, for all configurations $M \in \bar{\mathscr{P}}$.

Tables 6.8-6.10 compare BMRW with PTV and JSR. Under the headings "*PTV*" and "*JSR*", we report the percentage ratio, over $z(F)$, of the upper bound ($\%UB$) and of the lower bound ($\%LB$) achieved at the root node, the percentage gap ($\%gap$) between the best lower and upper bound computed, and the total computing time (t_{tot}) in seconds. The values in columns $z(F)$ are in bold whenever the instances were open before BMRW. The last lines of the tables reports, for each method, the number of instances solved to optimality (in columns $\%gap$), and, for JSR and BMRW, the average percentage deviation of the upper and lower bounds (in columns $\%UB$ and $\%LB$) and the average computing time (in columns t_{tot}), computed over all instances solved by JSR, that are a subset of the instances solved by BMRW.

Tables 6.2-6.7 show that BMRW solved to optimality 166 out of 180 instances. Columns $|\mathscr{P}|$ and $|\bar{\mathscr{P}}|$ show the effectiveness of both the procedure applied at Step 2 for generating the set $\mathscr{P}$ and the procedures applied at Steps 3.c-(i) and 3.c-(ii) for computing valid lower bounds on $z(F(M))$. Notice that few problems $F(M)$ required solving by CPLEX (see columns $|\bar{\mathscr{P}}|$). BMRW was able to solve 25 out of 33 instances of the Sets 5 and 6 (see Tables 6.6-6.7). On the other 8 instances, BMRW could not generate all 2^{nd}-level routes required to solve some problems $F(M)$ to optimality because of the gap between the computed lower and upper bounds.

On the Sets 2 and 3 (see Tables 6.8 and 6.9), BMRW solved to optimality all 39 instances whereas PTV and JSR solved to optimality 13 and 32 instances, respectively. Of the 54 instances of the Set 4A, BMRW and JSR solved 50 and 15 of them, respectively. None of the 18 instances considered by PTV of Set 4B were solved to optimality whereas 52 instances out of 54 were solved to optimality by BMRW. Tables 6.8-6.10 show that BMRW outperforms both PTV and JSR.

Finally, Tables 6.8 and 6.9 indicate that the heuristic algorithm described in §6.5.5 provided better solutions, on average, than the heuristic algorithms of PTV and JSR.

6.8 Conclusions

In this chapter, we proposed a new exact method for solving the two-echelon capacitated vehicle routing problem (2E-CVRP). We described a bounding procedure that is used by the exact algorithm to decompose the 2E-CVRP into a limited set of multi-depot capacitated vehicle routing problems (MDCVRP) with side constraints. The optimal 2E-CVRP solution is obtained by solving the set of MDCVRPs generated.

The proposed method was tested on 180 instances, both taken from the literature and newly generated, with up to 100 customers and 6 satellites. The new exact algorithm

Table 6.4: Computational results on Set 4A instances

$Name$	n_s	$z(F)$	$\%LD1$	t_{LD1}	$\|\mathscr{P}\|$	$\%UB$	$\%LB$	t_{LB}	$\|\bar{\mathscr{P}}\|$	r^{max}	t_{tot}
Instance50-1	2	1,569.42	97.1	23.3	2	100.1	99.9	72.5	2	30,903	75.5
Instance50-2	2	1,438.33	95.8	14.6	3	100.9	99.6	110.7	2	44,411	161.9
Instance50-3	2	1,570.43	97.1	23.3	2	102.6	99.9	67.8	1	31,377	70.6
Instance50-4	2	1,424.04	96.4	20.2	2	101.7	99.4	59.6	1	44,211	101.8
Instance50-5	2	2,193.52	98.3	25.1	5	100.2	99.6	286.2	5	67,491	663.7
Instance50-6	2	1,279.87	95.2	18.5	2	100.0	100.0	42.7	0	0	42.7
Instance50-7	2	1,458.63	98.0	29.7	2	104.7	99.8	92.7	2	30,995	100.4
Instance50-8	2	1,363.74	95.5	20.8	3	100.1	99.5	199.2	2	404,659	2,261.9
Instance50-9	2	1,450.27	98.0	28.5	2	104.5	99.9	82.6	1	29,683	84.6
Instance50-10	2	1,407.65	92.9	22.9	2	100.3	99.6	71.9	1	52,513	112.9
Instance50-11	2	2,047.46	99.0	35.7	5	100.8	99.5	225.5	5	88,929	339.1
Instance50-12	2	1,209.42	93.1	25.0	2	100.1	100.0	69.4	0	0	69.4
Instance50-13	2	1,481.83	95.5	24.5	2	102.4	99.9	86.1	2	30,277	92.1
Instance50-14	2	1,393.61	93.6	21.8	3	100.9	99.4	126.2	2	181,889	1,188.3
Instance50-15	2	1,489.94	95.5	25.1	2	102.4	99.8	66.6	1	30,130	71.5
Instance50-16	2	1,389.17	95.0	16.1	2	101.1	99.8	56.0	1	36,097	62.6
Instance50-17	2	2,088.49	97.3	28.8	5	100.7	99.8	253.0	2	40,721	305.3
Instance50-18	2	1,227.61	93.1	16.9	2	100.0	99.3	60.5	1	49,390	117.2
Instance50-19	3	1,564.66	92.5	72.7	8	100.0	99.3	179.3	2	53,950	234.3
Instance50-20	3	1,272.97	93.7	24.5	8	101.3	99.1	59.4	1	84,784	140.1
Instance50-21	3	1,577.82	96.0	62.1	4	100.1	99.2	139.8	2	58,059	218.9
Instance50-22	3	1,281.83	95.1	33.9	8	101.4	100.0	76.2	1	50,177	79.2
Instance50-23	3	1,807.35	89.3	52.1	11	100.0	98.7	310.9	3	190,099	1,510.9
Instance50-24	3	1,282.68	95.1	28.6	14	100.0	100.0	80.0	0	0	80.0
Instance50-25	3	1,522.42	91.3	63.0	8	102.0	99.2	221.7	2	67,095	335.9
Instance50-26	3	1,167.46	97.2	27.1	1	100.2	99.9	51.9	1	49,094	54.0
Instance50-27	3	1,481.57	93.9	72.2	4	102.0	99.3	196.0	2	67,175	355.9
Instance50-28	3	1,210.44	93.2	38.3	10	100.0	100.0	279.5	1	55,285	295.6
Instance50-29	3	1,722.04	89.9	67.7	12	102.5	98.8	461.4	3	Δ^{max}	9,092.9
Instance50-30	3	1,211.59	93.5	32.8	13	100.5	100.0	243.1	0	0	243.1
Instance50-31	3	1,490.34	91.8	65.1	8	102.2	98.1	325.9	4	Δ^{max}	11,561.3
Instance50-32	3	1,199.00	94.1	25.9	7	100.1	98.7	262.7	1	619,322	4,009.4
Instance50-33	3	1,508.30	93.4	64.5	6	101.2	98.0	234.6	2	Δ^{max}	12,922.3
Instance50-34	3	1,233.92	93.2	30.7	10	100.0	99.0	130.8	1	85,850	207.0
Instance50-35	3	1,718.41	87.6	63.9	12	100.1	98.3	619.9	5	Δ^{max}	20,377.6
Instance50-36	3	1,228.89	93.3	28.6	14	100.0	99.3	121.6	1	59,745	154.1
Instance50-37	5	1,528.73	94.5	162.7	116	100.7	99.5	778.3	3	82,225	807.8
Instance50-38	5	1,169.20	93.9	52.5	134	100.9	99.0	429.3	1	253,662	1,648.2
Instance50-39	5	1,520.92	94.6	168.0	63	100.6	99.8	688.0	2	78,320	695.0
Instance50-40	5	1,199.42	90.3	55.4	66	101.6	99.6	986.8	2	99,764	996.4
Instance50-41	5	1,667.96	95.3	195.4	64	100.3	99.6	1,302.9	4	79,889	1,344.7
Instance50-42	5	1,194.54	95.2	50.5	61	101.6	99.4	177.3	1	94,931	223.2
Instance50-43	5	1,439.67	95.4	175.5	56	101.3	99.5	1,032.2	3	87,237	1,095.7
Instance50-44	5	1,045.13	95.6	84.4	100	100.2	99.8	424.1	1	90,132	435.8
Instance50-45	5	1,450.96	94.9	160.9	34	101.7	99.2	577.7	2	135,411	774.0
Instance50-46	5	1,088.77	91.8	68.0	62	100.2	99.3	1,150.5	5	131,810	1,345.4
Instance50-47	5	1,587.29	96.2	205.5	62	102.1	99.4	1,470.7	2	98,905	1,566.3
Instance50-48	5	1,082.20	96.7	56.7	6	101.1	100.0	91.0	0	0	91.0
Instance50-49	5	1,434.88	95.0	164.3	74	102.3	100.0	714.8	0	0	714.8
Instance50-50	5	1,083.12	93.2	55.1	134	100.5	99.1	869.1	1	239,534	1,337.0
Instance50-51	5	1,398.05	94.6	179.6	64	101.0	100.0	744.0	1	73,279	748.4
Instance50-52	5	1,125.67	90.3	52.0	65	101.5	99.0	1,231.9	7	167,299	1,533.7
Instance50-53	5	1,567.77	95.0	211.4	63	100.1	98.7	1,712.0	2	268,139	4,223.3
Instance50-54	5	1,127.61	94.1	48.3	58	100.2	98.9	343.4	1	414,080	1,041.6

TABLE 6.5: Computational results on Set 4B instances

Name	n_s	$z(F)$	$\%LD1$	t_{LD1}	$\|\mathscr{P}\|$	$\%UB$	$\%LB$	t_{LB}	$\|\bar{\mathscr{P}}\|$	r^{max}	t_{tot}
Instance50-1	2	1,569.42	95.0	39.2	3	101.5	100.0	113.3	2	31,196	117.0
Instance50-2	2	1,438.33	95.8	14.7	5	101.0	99.7	129.8	2	42,949	188.5
Instance50-3	2	1,570.43	95.1	38.9	3	101.5	100.0	95.1	1	31,017	97.8
Instance50-4	2	1,424.04	96.3	21.9	3	101.2	99.3	69.3	1	48,595	115.6
Instance50-5	2	2,193.52	98.3	40.8	7	100.2	99.6	326.6	5	62,190	631.5
Instance50-6	2	1,279.87	95.1	17.4	3	102.9	99.9	49.1	1	34,244	52.6
Instance50-7	2	1,408.57	98.5	41.5	3	101.6	99.9	73.0	1	31,607	76.5
Instance50-8	2	1,360.32	95.7	17.3	5	100.2	99.6	222.8	3	604,910	3,293.6
Instance50-9	2	1,403.53	98.7	40.0	3	103.3	99.9	80.3	1	35,515	81.7
Instance50-10	2	1,360.56	96.1	20.8	3	100.0	100.0	46.3	0	0	46.3
Instance50-11	2	2,047.46	99.0	50.6	7	100.8	99.5	296.8	6	88,960	450.8
Instance50-12	2	1,209.42	93.0	22.1	3	100.0	99.9	106.9	1	48,537	111.9
Instance50-13	2	1,450.93	96.1	44.1	3	102.3	100.0	94.1	0	0	94.1
Instance50-14	2	1,393.61	93.6	20.6	5	101.1	99.4	147.4	2	175,756	1,069.7
Instance50-15	2	1,466.83	95.5	39.7	3	101.1	99.9	103.4	1	29,552	106.0
Instance50-16	2	1,387.83	95.1	16.4	3	100.1	99.8	76.7	2	43,428	99.0
Instance50-17	2	2,088.49	97.3	44.3	7	100.6	99.8	306.0	2	38,365	358.3
Instance50-18	2	1,227.61	93.1	17.0	3	100.0	99.2	69.7	1	51,842	127.8
Instance50-19	3	1,546.28	93.7	86.3	16	101.2	99.2	262.5	1	60,815	293.4
Instance50-20	3	1,272.97	93.8	24.3	9	101.3	99.0	59.7	1	107,430	170.9
Instance50-21	3	1,577.82	96.0	85.6	12	100.8	99.2	199.8	2	58,718	250.9
Instance50-22	3	1,281.83	95.2	31.2	9	103.0	100.0	68.9	0	0	68.9
Instance50-23	3	1,652.98	96.6	83.6	7	102.0	100.0	193.7	0	0	193.7
Instance50-24	3	1,282.68	95.2	29.1	16	100.0	100.0	79.7	0	0	79.7
Instance50-25	3	1,408.57	98.2	87.2	7	101.9	99.9	150.9	1	45,580	155.0
Instance50-26	3	1,167.46	97.2	27.1	2	100.2	99.9	52.6	1	48,772	55.9
Instance50-27	3	1,444.51	96.5	93.1	8	102.3	99.9	194.2	1	44,411	198.1
Instance50-28	3	1,210.44	92.9	34.9	11	100.9	100.0	249.6	0	0	249.6
Instance50-29	3	1,552.66	96.7	104.4	7	103.0	100.0	257.3	1	48,508	258.4
Instance50-30	3	1,211.59	93.2	38.6	16	100.5	99.9	239.9	1	61,768	245.9
Instance50-31	3	1,440.86	94.8	87.5	13	101.2	100.0	262.2	0	0	262.2
Instance50-32	3	1,199.00	94.1	31.1	8	100.1	98.7	116.9	1	579,861	3,812.3
Instance50-33	3	1,478.86	95.4	86.0	11	101.5	99.0	239.6	1	121,197	467.9
Instance50-34	3	1,233.92	93.1	27.6	11	101.1	99.1	166.6	1	131,443	287.7
Instance50-35	3	1,570.72	94.7	95.5	7	102.4	98.9	332.0	3	221,318	1,299.7
Instance50-36	3	1,228.89	93.1	26.8	16	100.0	99.2	126.2	1	72,167	180.1
Instance50-37	5	1,528.73	93.7	206.7	223	102.0	97.9	1,520.3	10	Δ^{max}	14,522.3
Instance50-38	5	1,163.07	94.6	78.8	153	100.0	99.1	485.7	3	229,650	1,163.0
Instance50-39	5	1,520.92	93.1	212.4	112	100.3	98.8	1,097.6	7	126,528	1,791.0
Instance50-40	5	1,163.04	93.0	59.2	82	101.3	99.6	312.8	1	96,251	348.0
Instance50-41	5	1,652.98	95.3	266.4	117	101.0	99.6	2,229.6	14	153,550	2,370.6
Instance50-42	5	1,190.17	95.5	55.5	75	101.9	99.2	238.4	2	132,823	432.4
Instance50-43	5	1,406.11	95.2	210.3	79	101.3	99.6	1,065.4	1	94,023	1,098.4
Instance50-44	5	1,035.03	96.4	67.0	93	100.9	100.0	382.9	1	84,919	387.2
Instance50-45	5	1,401.87	95.3	187.3	55	102.7	99.6	464.7	1	82,077	484.7
Instance50-46	5	1,058.11	94.7	83.5	65	100.4	100.0	426.9	0	0	426.9
Instance50-47	5	1,552.66	95.8	260.8	103	103.0	100.0	1,220.1	2	146,044	1,227.0
Instance50-48	5	1,074.50	97.3	60.0	6	100.5	99.9	121.4	1	74,056	125.5
Instance50-49	5	1,434.88	94.4	217.4	142	101.1	98.1	1,498.7	8	Δ^{max}	13,940.3
Instance50-50	5	1,065.25	94.8	89.2	126	100.0	99.9	500.7	1	83,400	508.0
Instance50-51	5	1,387.51	93.9	225.0	92	102.2	98.9	845.9	2	113,910	1,299.1
Instance50-52	5	1,103.42	92.0	56.0	81	100.1	99.4	788.8	1	109,254	846.0
Instance50-53	5	1,545.73	95.2	283.1	97	101.6	99.0	2,113.7	3	232,398	2,395.8
Instance50-54	5	1,113.62	95.2	51.4	36	100.8	99.0	232.0	2	222,523	1,027.9

TABLE 6.6: Computational results on Set 5 instances

Name	n_s	$z(F)$	$\%LD1$	t_{LD1}	$\lvert\mathscr{P}\rvert$	$\%UB$	$\%LB$	t_{LB}	$\lvert\bar{\mathscr{P}}\rvert$	r^{max}	t_{tot}
2eVRP_100-5-1	5	1,564.46	97.9	16.5	44	102.1	99.3	381.8	5	147,996	9,359.6
2eVRP_100-5-1b	5	1,142.53	93.8	36.5	60	100.0	94.6	1,339.2	10	Δ^{max}	24,028.9
2eVRP_100-5-2	5	1,016.32	95.6	15.2	197	100.6	99.1	928.8	19	156,424	10,517.6
2eVRP_100-5-2b	5	796.53	95.2	29.9	100	100.0	96.6	1,750.8	10	Δ^{max}	26,099.7
2eVRP_100-5-3	5	1,045.29	97.5	16.3	50	100.3	99.2	262.5	13	137,379	2,930.2
2eVRP_100-5-3b	5	833.94	95.7	39.7	95	100.0	97.6	1,475.1	8	Δ^{max}	32,693.8

TABLE 6.7: Computational results on Set 6 instances

Name	n_s	$z(F)$	$\%LD1$	t_{LD1}	$\lvert\mathscr{P}\rvert$	$\%UB$	$\%LB$	t_{LB}	$\lvert\bar{\mathscr{P}}\rvert$	r^{max}	t_{tot}
A-n51-4	4	652.00	94.6	6.7	16	100.3	99.7	108.0	4	55,040	119.1
A-n51-5	5	663.41	95.7	10.4	81	100.3	99.7	149.8	2	64,146	154.8
A-n51-6	6	662.51	94.9	15.9	246	100.6	100.0	262.8	0	0	263.1
A-n76-4	4	985.95	95.9	16.1	71	101.9	99.4	267.6	2	78,465	343.7
A-n76-5	5	979.15	95.8	27.6	281	101.8	99.6	818.9	4	91,512	857.2
A-n76-6	6	970.20	95.8	46.0	1,391	101.8	99.5	3,215.2	8	108,558	3,327.4
A-n101-4	4	1,194.17	95.9	52.7	120	101.0	99.2	1,709.6	11	461,313	5,971.3
A-n101-5	5	1,211.38	96.6	43.7	647	102.5	99.4	3,540.3	4	234,285	4,823.3
A-n101-6	6	1,158.98	95.8	91.9	4,814	101.7	98.7	24,851.2	51	Δ^{max}	118,077.4
B-n51-4	4	563.98	96.0	6.6	10	101.1	98.8	30.3	1	87,968	56.8
B-n51-5	5	549.23	94.8	9.7	64	101.0	99.0	105.8	1	100,364	130.9
B-n51-6	6	556.32	94.3	15.5	106	100.0	100.0	125.5	0	0	125.6
B-n76-4	4	792.73	94.5	12.4	23	102.1	99.3	257.9	2	119,142	333.7
B-n76-5	5	783.93	94.0	21.3	167	101.5	99.3	572.2	1	144,513	610.8
B-n76-6	6	774.17	94.4	32.6	877	102.2	99.6	2,023.2	2	164,489	2,075.0
B-n101-4	4	939.21	97.3	42.1	10	102.0	98.9	195.7	1	485,177	2,512.5
B-n101-5	5	967.82	94.8	37.3	587	102.1	99.1	4,772.3	7	331,423	7,058.8
B-n101-6	6	960.29	96.2	76.9	456	103.0	99.1	1,970.4	4	273,755	3,772.4
C-n51-4	4	689.18	95.5	6.7	26	100.3	99.4	54.2	1	51,909	78.3
C-n51-5	5	723.12	93.6	10.3	64	100.3	98.9	81.4	1	125,120	431.6
C-n51-6	6	697.00	94.4	16.3	126	101.1	99.4	145.8	1	76,426	166.0
C-n76-4	4	1,054.89	94.9	15.5	66	102.9	99.1	290.7	4	108,471	454.3
C-n76-5	5	1,115.32	92.5	21.6	303	103.0	99.2	1,089.2	6	136,054	1,817.0
C-n76-6	6	1,064.72	92.2	31.7	1,449	101.7	97.8	4,786.7	20	Δ^{max}	47,840.9
C-n101-4	4	1,305.68	95.0	49.3	116	101.7	98.4	1,392.8	10	Δ^{max}	29,626.4
C-n101-5	5	1,309.42	96.4	84.4	206	101.6	98.6	1,203.2	2	Δ^{max}	10,865.1
C-n101-6	6	1,284.48	96.2	94.6	1,373	103.3	98.7	6,804.4	8	Δ^{max}	27,969.4

solved to optimality 144 out of the 153 instances from literature and closed 97 of them for the first time. The comparison with the state-of-the-art exact methods from the literature show that new exact method outperforms the other exact methods in terms of size, number of problems solved to optimality, and computing time.

Table 6.8: Comparison with the exact methods PTV and JSR on Set 2 instances

		PTV		*JSR*				*BMRW*				
Name	$z(F)$	$\%UB$	$\%gap$	$\%UB$	$\%LB$	$\%gap$	t_{tot}	$\%UB$	$\%LB$	t_{LB}	$\%gap$	t_{tot}
E-n22-k4-s6-17	417.07	100.0	0.0	100.0	96.7	0.0	0.2	100.0	100.0	0.5	0.0	0.5
E-n22-k4-s8-14	384.96	106.0	0.0	100.0	98.0	0.0	1.0	100.0	100.0	0.7	0.0	0.7
E-n22-k4-s9-19	470.60	100.0	0.0	113.1	90.4	0.0	12.4	100.0	100.0	1.2	0.0	1.2
E-n22-k4-s10-14	371.50	117.3	0.0	100.0	96.8	0.0	1.2	100.0	100.0	0.5	0.0	0.5
E-n22-k4-s11-12	427.22	100.0	0.0	104.1	94.7	0.0	3.2	100.5	100.0	1.3	0.0	1.3
E-n22-k4-s12-16	392.78	108.4	0.0	100.0	95.9	0.0	2.0	100.0	100.0	1.0	0.0	1.1
E-n33-k4-s1-9	730.16	100.9	0.0	100.0	87.3	0.0	49.4	100.0	100.0	37.6	0.0	37.6
E-n33-k4-s2-13	714.63	103.0	1.5	100.0	89.4	0.0	34.2	100.0	100.0	34.9	0.0	34.9
E-n33-k4-s3-17	707.48	104.5	1.7	113.2	91.0	0.0	1,126.8	105.8	100.0	48.1	0.0	48.1
E-n33-k4-s4-5	778.74	104.9	1.5	100.0	87.6	0.0	54.9	100.9	100.0	72.5	0.0	72.5
E-n33-k4-s7-25	756.85	100.0	1.6	100.0	86.0	0.0	87.5	101.0	100.0	47.1	0.0	47.1
E-n33-k4-s14-22	779.05	100.0	1.6	105.9	88.0	0.0	2.4	100.0	100.0	31.7	0.0	31.7
E-n51-k5-s3-18	**597.49**	100.0	2.6	100.0	92.6	4.5	-	100.0	99.8	23.7	0.0	25.8
E-n51-k5-s5-47	530.76	102.3	1.8	102.4	97.0	0.0	13.3	101.6	99.8	25.9	0.0	27.5
E-n51-k5-s7-13	**554.81**	100.0	4.1	100.0	94.4	1.6	-	100.2	98.9	37.3	0.0	55.1
E-n51-k5-s12-20	581.64	100.4	3.7	104.2	94.2	0.0	213.6	100.5	99.3	27.1	0.0	44.3
E-n51-k5-s28-48	**538.22**	100.0	2.0	100.0	95.5	0.8	-	100.0	99.7	40.1	0.0	44.0
E-n51-k5-s33-38	552.28	104.7	0.7	100.0	95.8	0.0	2,114.0	100.0	100.0	13.6	0.0	13.6
E-n51-k5-s3-5-18-47	530.76	102.2	2.8	103.3	94.4	0.0	84.0	100.0	99.9	259.2	0.0	260.8
E-n51-k5-s7-13-33-38	531.92	107.5	3.6	102.7	94.6	0.0	3,642.8	100.0	99.4	263.6	0.0	266.6
E-n51-k5-s12-20-28-48	527.63	113.8	1.5	109.4	95.5	0.0	798.7	100.0	99.6	71.8	0.0	74.2
Avg./Solved		103.6	7	102.8	93.1	18	457.9	100.5	99.8		21	53.6

Table 6.9: Comparison with the exact methods PTV and JSR on Set 3 instances

		PTV		*JSR*				*BMRW*				
Name	$z(F)$	$\%UB$	$\%gap$	$\%UB$	$\%LB$	$\%gap$	t_{tot}	$\%UB$	$\%LB$	t_{LB}	$\%gap$	t_{tot}
E-n22-k4-s13-14	526.15	100.1	0.0	102.2	98.2	0.0	3.2	100.0	100.0	2.1	0.0	2.1
E-n22-k4-s14-19	498.80	105.0	0.0	105.0	91.1	0.0	61.2	100.0	100.0	2.4	0.0	2.4
E-n22-k4-s13-16	521.09	100.0	0.0	101.0	98.2	0.0	2.3	100.0	100.0	2.8	0.0	2.8
E-n22-k4-s17-19	512.80	100.0	0.0	104.8	93.8	0.0	8.0	100.0	100.0	2.6	0.0	2.6
E-n22-k4-s13-17	496.38	100.0	0.0	100.0	93.4	0.0	1.1	100.0	100.0	1.2	0.0	1.2
E-n22-k4-s19-21	520.42	101.4	0.0	101.4	95.4	0.0	5.5	100.0	100.0	3.8	0.0	3.8
E-n33-k4-s22-26	680.37	100.0	4.2	101.5	91.0	0.0	6.3	100.1	99.8	71.8	0.0	73.3
E-n33-k4-s16-22	**672.17**	100.0	5.7	113.2	93.0	2.1	-	102.0	99.4	115.4	0.0	127.5
E-n33-k4-s16-24	666.02	100.4	6.0	100.0	96.4	0.0	747.4	100.1	99.9	125.2	0.0	128.4
E-n33-k4-s24-28	670.43	103.3	5.6	100.0	94.8	0.0	17.6	100.0	100.0	73.1	0.0	78.8
E-n33-k4-s19-26	680.37	100.0	4.7	109.2	89.4	0.0	26.4	100.1	99.6	70.8	0.0	72.8
E-n33-k4-s25-28	650.58	100.0	5.3	100.0	92.6	0.0	158.2	100.3	100.0	56.0	0.0	56.0
E-n51-k5-s13-19	560.73			100.0	95.5	0.0	1,007.9	100.0	99.6	43.8	0.0	48.0
E-n51-k5-s13-42	564.45			106.1	96.6	0.0	208.3	100.3	99.1	18.3	0.0	50.1
E-n51-k5-s13-44	564.45			107.6	96.8	0.0	288.5	101.4	99.0	29.7	0.0	73.0
E-n51-k5-s40-42	**746.31**			100.9	88.9	7.5	-	102.6	99.0	34.8	0.0	107.2
E-n51-k5-s41-42	**771.56**			100.5	95.1	0.6	-	100.1	98.9	36.9	0.0	2,078.6
E-n51-k5-s41-44	**802.91**			100.0	89.7	7.0	-	101.2	99.6	39.8	0.0	59.4
Avg./Solved		100.8	6	103.0	93.9	14	181.6	100.5	99.7		18	42.5

Table 6.10: Comparison with the exact method JSR on Set 4A instances

			JSR				*BMRW*				
Name	n_s	$z(F)$	$\%UB$	$\%LB$	$\%gap$	t_{tot}	$\%UB$	$\%LB$	t_{LB}	$\%gap$	t_{tot}
Instance50-1	2	**1,569.42**	112.9	91.2	1.7	-	100.1	99.9	72.5	0.0	75.5
Instance50-2	2	1,438.33	100.0	91.5	0.0	1,146.7	100.9	99.6	110.7	0.0	161.9
Instance50-3	2	**1,570.43**	112.7	88.7	1.6	-	102.6	99.9	67.8	0.0	70.6
Instance50-4	2	**1,424.04**	100.0	90.1	0.9	-	101.7	99.4	59.6	0.0	101.8
Instance50-5	2	**2,193.52**	100.3	85.6	0.4	-	100.2	99.6	286.2	0.0	663.7
Instance50-6	2	1,279.87	100.0	97.2	0.0	4,463.4	100.0	100.0	42.6	0.0	42.7
Instance50-7	2	**1,458.63**	114.2	89.9	1.5	-	104.7	99.8	92.7	0.0	100.4
Instance50-8	2	1,363.74	100.0	91.8	0.0	1,164.5	100.1	99.5	199.2	0.0	2,261.9
Instance50-9	2	**1,450.27**	113.9	89.8	1.3	-	104.5	99.9	82.6	0.0	84.6
Instance50-10	2	1,407.65	101.0	98.4	0.0	3,933.1	100.3	99.6	71.9	0.0	112.9
Instance50-11	2	**2,047.46**	100.9	89.7	0.6	-	100.8	99.5	225.5	0.0	339.1
Instance50-12	2	1,209.42	100.0	95.7	0.0	22.3	100.1	100.0	69.4	0.0	69.4
Instance50-13	2	**1,481.83**	111.9	91.7	1.2	-	102.4	99.9	86.1	0.0	92.1
Instance50-14	2	**1,393.61**	101.6	91.2	0.1	-	100.9	99.4	126.2	0.0	1,188.3
Instance50-15	2	**1,489.94**	111.8	91.7	1.1	-	102.4	99.8	66.6	0.0	71.5
Instance50-16	2	1,389.17	100.0	91.0	0.0	1,045.1	101.1	99.8	56.0	0.0	62.6
Instance50-17	2	**2,088.49**	100.4	85.7	0.1	-	100.7	99.8	253.0	0.0	305.3
Instance50-18	2	1,227.61	100.0	96.5	0.0	8,130.1	100.0	99.3	60.5	0.0	117.2
Instance50-19	3	**1,564.66**	109.8	82.2	0.5	-	100.0	99.3	179.3	0.0	234.3
Instance50-20	3	**1,272.97**	116.6	93.2	0.6	-	101.3	99.1	59.4	0.0	140.1
Instance50-21	3	**1,577.82**	106.8	88.5	0.4	-	100.1	99.2	139.8	0.0	218.9
Instance50-22	3	1,281.83	105.1	86.2	0.0	8,636.7	101.4	100.0	76.2	0.0	79.2
Instance50-23	3	**1,807.35**	100.0	83.2	10.0	-	100.0	98.7	310.9	0.0	1,510.9
Instance50-24	3	1,282.68	101.8	93.9	0.0	6,559.9	100.0	100.0	79.9	0.0	80.0
Instance50-25	3	**1,522.42**	100.8	83.7	1.2	-	102.0	99.2	221.7	0.0	335.9
Instance50-26	3	1,167.46	113.4	95.1	0.0	66.4	100.2	99.9	51.9	0.0	54.0
Instance50-27	3	**1,481.57**	107.0	86.4	1.0	-	102.0	99.3	196.0	0.0	355.9
Instance50-28	3	1,210.44	104.1	86.7	0.0	2,046.0	100.0	100.0	279.5	0.0	295.6
Instance50-29	3	1,722.04	100.9	80.5	0.8	-	102.5	98.8	461.4	1.2	9,092.9
Instance50-30	3	1,211.59	101.8	92.1	0.0	17.4	100.5	100.0	243.1	0.0	243.1
Instance50-31	3	1,490.34	109.7	90.3	1.5	-	102.2	98.1	325.9	1.9	11,561.3
Instance50-32	3	**1,199.00**	104.8	86.9	0.5	-	100.1	98.7	262.7	0.0	4,009.4
Instance50-33	3	1,508.30	105.3	82.7	1.3	-	101.2	98.0	234.6	2.0	12,922.3
Instance50-34	3	**1,233.92**	102.5	85.6	0.1	-	100.0	99.0	130.8	0.0	207.0
Instance50-35	3	1,718.41	100.3	79.3	1.2	-	100.1	98.3	619.9	1.7	20,377.6
Instance50-36	3	1,228.89	100.1	85.8	0.0	2,038.2	100.0	99.3	121.6	0.0	154.1
Instance50-37	5	**1,528.73**	108.7	82.6	2.9	-	100.7	99.5	778.3	0.0	807.8
Instance50-38	5	**1,169.20**	108.2	82.6	0.2	-	100.9	99.0	429.3	0.0	1,648.2
Instance50-39	5	**1,520.92**	106.4	84.6	0.4	-	100.6	99.8	688.0	0.0	695.0
Instance50-40	5	**1,199.42**	101.0	81.5	2.6	-	101.6	99.6	986.8	0.0	996.4
Instance50-41	5	**1,667.96**	108.1	86.6	1.4	-	100.3	99.6	1,302.9	0.0	1,344.7
Instance50-42	5	**1,194.54**	112.8	83.3	1.2	-	101.6	99.4	177.3	0.0	223.2
Instance50-43	5	**1,439.67**	113.3	87.4	1.7	-	101.3	99.5	1,032.2	0.0	1,095.7
Instance50-44	5	1,045.13	109.5	80.4	0.0	144.0	100.2	99.8	424.1	0.0	435.8
Instance50-45	5	**1,450.96**	108.5	82.1	1.0	-	101.7	99.2	577.7	0.0	774.0
Instance50-46	5	**1,088.77**	101.7	77.4	1.0	-	100.2	99.3	1,150.5	0.0	1,345.4
Instance50-47	5	**1,587.29**	109.7	83.5	1.0	-	102.1	99.4	1,470.7	0.0	1,566.3
Instance50-48	5	1,082.20	115.8	86.1	0.0	133.4	101.1	100.0	91.0	0.0	91.0
Instance50-49	5	**1,434.88**	108.4	84.3	2.1	-	102.3	100.0	714.7	0.0	714.8
Instance50-50	5	**1,083.12**	105.4	77.9	1.7	-	100.5	99.1	869.1	0.0	1,337.0
Instance50-51	5	**1,398.05**	106.6	82.3	4.6	-	101.0	100.0	744.0	0.0	748.4
Instance50-52	5	**1,125.67**	100.2	81.0	1.1	-	101.5	99.0	1,231.9	0.0	1,533.7
Instance50-53	5	**1,567.77**	109.4	83.8	1.3	-	100.1	98.7	1,712.0	0.0	4,223.3
Instance50-54	5	**1,127.61**	110.6	88.2	0.9	-	100.2	98.9	343.4	0.0	1,041.6
Avg./Solved			105.9	87.1	15	2,599.1	101.0	99.4		50	276.0

Bibliography

Z. Akca, R. T. Berger, and T. K. Ralphs. A branch-and-price algorithm for combined location and routing problems under capacity restrictions. In J. W. Chinneck, B. Kristjansson, M. J. Saltzman, R. Sharda, S. Voß, and R. Sharda, editors, *Operations Research and Cyber-Infrastructure*, volume 47 of *Operations Research/Computer Science Interfaces Series*, pages 309–330. Springer US, 2009.

F. Alonso, M. J. Alvarez, and J. E. Beasley. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59(7):963–976, 2008.

D. Applegate, R. E. Bixby, V. Chvátal, and W. Cook. Concorde - a code for solving traveling salesman problems. Release, 12 1999. http://www.tsp.gatech.edu/concorde/downloads/downloads.htm.

D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The traveling salesman problem: a computational study*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, USA, 2007.

L. Art and H. Mauch. *Dynamic Programming A Computational Tool*. Studies in Computational Intelligence. Springer, Berlin / Heidelberg, 2007.

N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, 36(2):69–79, 2000.

N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming, Series A*, 90(3):475–506, 2001.

P. Augerat. *Approche polyédrale du problème de tournées de véhicules*. PhD thesis, Institut National Polytechnique de Grenoble, 1995.

P. J. Augerat, M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report 1RR949-M, ARTEMIS-IMAG, Grenoble, France, 1995.

N. Azi, M. Gendreau, and J.-Y. Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763, 2010.

E. Baker. An exact algorithm for the time-constrained traveling salesman problem. *Operations Research*, 31(5):938–945, 1983.

E. Balas. The asymmetric assignment problem and some new facets of the traveling salesman polytope on a directed graph. *SIAM Journal on Discrete Mathematics*, 2 (4):425–451, 1989.

E. Balas. Personal communication, 2000.

E. Balas and N. Christofides. A restricted Lagrangean approach to the traveling salesman problem. *Mathematical Programming, Series A*, 21(1):19–46, 1981.

E. Balas and N. Simonetti. Linear time dynamic-programming algorithms for new classes of restricted TSPs: a computational study. *INFORMS Journal on Computing*, 13(1):56–75, 2001.

E. Balas and P. Toth. Branch and bound methods. In E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 361–401. Wiley, Chichester, 1985.

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

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(5):723–738, 2004.

R. Baldacci, L. D. Bodin, and A. Mingozzi. The multiple disposal facilities and multiple inventory locations rollon-rolloff vehicle routing problem. *Computers & Operations Research*, 33(9):2667–2702, 2006.

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, Series A*, 115(2):351–385, 2008.

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(3):229–268, 2010.

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

R. Baldacci, A. Mingozzi, and R. Roberti. New state-space relaxation for solving the TSPTW. *INFORMS Journal on Computing*, 2011b.

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

R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012a.

R. Baldacci, A. Mingozzi, R. Roberti, and R. Wolfler Calvo. An exact algorithm for the two-echelon capacitated vehicle routing problem. Submitted for publication, 2012b.

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

R. E. Bellman. *Dynamic Programming*. Princeton University Press, U.S.A., 1957.

M. Bellmore and J. C. Malone. Pathology of traveling-salesman subtour-elimination algorithms. *Operations Research*, 19(2):278–307, 1971.

D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1 of *Athena Scientific optimization and computation series*. Athena Scientific, 1995.

D. P. Bertsekas. *Dynamic programming and optimal control*, volume 2 of *Athena Scientific optimization and computation series*. Athena Scientific, 2000.

R. G. Bland and D. F. Shallcross. Large traveling salesman problems arising experiments in x-ray crystallography: A preliminary report on computation. *Operations Research Letters*, 8(3):125–128, 1989.

M. A. Boschetti, A. Mingozzi, and S. Ricciardelli. A dual ascent procedure for the set partitioning problem. *Discrete Optimization*, 5(4):735–747, 2008.

J. C. S. Brandão and A. Mercer. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100(1): 180–191, 1997.

J. C. S. Brandão and A. Mercer. The multi-trip vehicle routing problem. *Journal of the Operational Research Society*, 49(8):799–805, 1998.

G. Carpaneto and P. Toth. Some new branching and bounding criteria for the asymmetric travelling salesman problem. *Management Science*, 26(7):736–743, 1980.

G. Carpaneto and P. Toth. Primal-dual algorithms for the assignment problem. *Discrete Applied Mathematics*, 18(2):137–153, 1987.

G. Carpaneto, M. Dell'Amico, and P. Toth. Exact solution of large-scale asymmetric traveling salesman problems. *ACM Transactions on Mathematical Software*, 21(4): 394–409, 1995.

A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006.

I.-M. Chao. A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51, 2002.

N. Christofides. *The traveling salesman problem: a guided tour of combinatorial optimization*, chapter Vehicle Routing, pages 431–448. Wiley, Chichester, UK, 1985.

N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. John Wiley & Sons, Chichester, UK, 1979.

N. Christofides, A. Mingozzi, and P. Toth. An algorithm for the time constrained travelling salesman problem. Technical Report IC OR 8125, Imperial College, 1981a.

N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxation. *Mathematical Programming, Series A*, 20(1):255–280, 1981b.

N. Christofides, A. Mingozzi, and P. Toth. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2):145–164, 1981c.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

A. Claus. A new formulation for the travelling salesman problem. *SIAM Journal on Algebraic and Discrete Methods*, 5(1):21–25, 1984.

T. G. Crainic and G. Laporte. *Fleet management and logistics.* Center for Research on Transportation 25th anniversary series, 1971-1996. Kluwer, 1998.

T. G. Crainic, S. Mancini, G. Perboli, and R. Tadei. Clustering-based heuristics for the two-echelon vehicle routing problem. Technical Report CIRRELT-2008-46, CIRRELT, Montreal, Canada, November 2008.

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

T. G. Crainic, S. Mancini, G. Perboli, and R. Tadei. Multi-start heuristics for the two-echelon vehicle routing problem. In P. Merz and J.-K. Hao, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 6622 of *Lecture Notes in Computer Science*, pages 179–190. Springer Berlin / Heidelberg, 2011.

R. F. da Silva and S. Urrutia. A general VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, 7(4):203–211, 2010.

C. D'Ambrosio, A. Lodi, and S. Martello. Combinatorial traveling salesman problem algorithms. In *Wiley Encyclopedia of Operations Research and Management Science*, volume 1, pages 738–747. Wiley, 2010.

E. Danna and C. Pape. Branch-and-price heuristics: a case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 99–129. Springer US, springer edition, 2005.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solutions of a large-scale traveling-salesman problem. *Operations Research*, 2(4):393–410, 1954.

S. Dash, O. Günlük, A. Lodi, and A. Tramontani. A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(1):132–147, 2012.

M. S. Daskin. *Network and Discrete Location: Models Algorithms and Applications.* John Wiley & Sons, New York, 1995.

G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation.* Business and Economics. Springer US, 2005.

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(3):387–404, 2008.

M. Desrochers and G. Laporte. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1990.

M. Desrochers, J. Desrosiers, and M. M. Solomon. A new optimization algorithm for the vehicle-routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

Z. Drezner. *Facility location: a survey of applications and methods.* Springer series in operations research. Springer, 1995.

M. Dror and P. Trudeau. Split delivery routing. *Naval Research Logistics*, 37:383–402, 1990.

Y. Dumas, J. Desrosiers, E. Gélinas, and M. M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371, 1995.

J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards, Sec. B*, 71B(4):233–240, 1967.

M. A. Efroymson and T. L. Ray. A branch-bound algorithm for plant location. *Operations Research*, 14(3):361–368, 1966.

S. Eilon and N. Christofides. *Distribution Management.* Hafner Press, 1971.

H. A. Eiselt and G. Laporte. A combinatorial optimization problem arising in dartboard design. *Journal of the Operational Research Society*, 42(2):113–118, 1991.

H. A. Eiselt and V. Marianov. *Foundations of Location Analysis.* International Series in Operations Research & Management Science. Springer, 2011.

Eurostat. *Panorama of Transport.* Publications Office of the European Union, 2009.

Eurostat. *Energy, transport and environment indicators.* Pocketbooks. Publications Office of the European Union, 2011.

R. Z. Farahani and M. Hekmatfar. *Facility Location: Concepts, Models, Algorithms and Case Studies.* Contributions to Management Science. Physica-Verlag, 2009.

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(3):216–229, 2004.

G. Finke, A. Claus, and E. Gunn. A two-commodity network flow approach to the traveling salesman problem. *Congressus Numerantium*, 41:167–178, 1984.

M. L. Fischer. Vehicle routing. In *Network Routing*, volume 8 of *Handbook in Operations Research and Management Science*, pages 1–33. North-Holland, Amsterdan, 1995.

M. Fischetti. Facets of the asymmetric traveling salesman polytope. *Mathematics of Operations Research*, 16(1):42–56, 1991.

M. Fischetti and E. Balas. A lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets. *Mathematical Programming, Series A*, 58 (1-3):325–352, 1993.

M. Fischetti and P. Toth. An additive bounding procedure for combinatorial optimization problems. *Operations Research*, 37(2):319–328, 1989.

M. Fischetti and P. Toth. An additive bounding procedure for the asymmetric travelling salesman problem. *Mathematical Programming, Series A*, 53(1-3):173–197, 1992.

M. Fischetti and P. Toth. An efficient algorithm for the min-sum arborescence problem. *ORSA Journal on Computing*, 5:426–434, 1993.

M. Fischetti and P. Toth. A polyhedral approach to the asymmetric traveling salesman problem. *Management Science*, 43(11):1520–1536, 1997.

M. Fischetti, A. Lodi, S. Martello, and P. Toth. A polyhedral approach to simplified crew scheduling and vehicle scheduling problems. *Management Science*, 47(6):833–850, 2001.

M. Fischetti, A. Lodi, and P. Toth. Solving real-world ATSP instances by branch-and-cut. In M. Jünger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization - Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 64–77. Springer Berlin / Heidelberg, Berlin, 2003.

M. Fischetti, A. Lodi, and P. Toth. Exact methods for the asymmetric traveling salesman problem. In G. Gutin, A. Punnen, D.-Z. Du, and P. M. Pardalos, editors, *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*, pages 169–205. Springer US, 2004.

B. Fleischmann. The vehicle routing problem with multiple use of vehicles. Facbereich Wirtschaftswissenschafte Universitat Hamburg, Hamburg, Germany, 1990.

F. Focacci, A. Lodi, and M. Milano. A hybrid exact algorithm for the TSPTW. *INFORMS Journal on Computing*, 14(4):403–417, 2002.

K. R. Fox, B. Gavish, and S. C. Graves. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, 28(4):1018–1021, 1980.

R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi 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(3):491–511, 2006.

M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP completeness.* W. H. Freeman & Co., San Francisco, CA, 1979.

R. S. Garfinkel. On partitioning the feasible set in a branch-and-bound algorithm for the asymmetric traveling-salesman problem. *Operations Research*, 21(1):340–343, 1973.

R. S. Garfinkel. Minimizing wallpaper waste, part 1: A class of traveling salesman problems. *Operations Research*, 25(5):741–751, 1977.

B. Gavish and S. C. Graves. The travelling salesman problem and related problems. Working Paper GR-078-78, Operations Research Center, Massachusetts Institute of Technology, 1978.

M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

M. Gendreau, A. Hertz, G. Laporte, and M. Stan. A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research*, 46(3): 330–335, 1998.

M. T. Godinho, L. Gouveia, and P. Pesneau. On a time-dependent flow-based formulation and an updated classification of formulations for the travelling salesman problem. In *Progress in Combinatorial Optimization*, chapter 7, pages 223–254. Wiley, 2011.

B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges.* Operations research/computer science interfaces series. Springer US, 2008.

B. L. Golden and A. A. Assad. *Vehicle routing: methods and studies.* Studies in management science and systems. North-Holland, 1988.

B. L. Golden and E. A. Wasil. Computerized vehicle routing in the soft drink industry. *Operations Research*, 35(1):6–17, 1987.

J. Gonzales Feliu, G. Perboli, R. Tadei, and D. Vigo. The two-echelon capacitated vehicle routing problem. Technical Report DEIS OR.INGCE 2007/2(R), DEIS, Bologna, Italy, 2007a.

J. Gonzales Feliu, G. Perboli, R. Tadei, and D. Vigo. The two-echelon capacitated vehicle routing problem. In *Proceedings of the 22th European Conference on Operational Research*, 2007b.

L. Gouveia. A result on projection for the vehicle routing problem. *European Journal of Operational Research*, 85(3):610–624, 1995.

L. Gouveia and P. Pesneau. On extended formulations for the precedence constrained asymmetric traveling salesman problem. *Networks*, 48(2):77–89, 2006.

L. Gouveia and J. M. Pires. The asymmetric travelling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints. *European Journal of Operational Research*, 112(1):134–146, 1999.

L. Gouveia and J. M. Pires. The asymmetric travelling salesman problem: on generalizations of disaggregated Miller-Tucker-Zemlin constraints. *Discrete Applied Mathematics*, 112(1-3):129–145, 2001.

L. Gouveia and S. Voß. A classification of formulations for the (time-dependent) travelling salesman problem. *European Journal of Operational Research*, 83(1):69–82, 1995.

I. Gribkovskaia, B. O. Gullberg, K. J. Hovden, and S. W. Wallace. Optimization model for a livestock collection problem. *International Journal of Physical Distribution & Logistics Management*, 36(2):136–152, 2006.

M. Grötschel and M.W. Padberg. Polyhedral theory. In E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 251–305. Wiley, Chichester, 1985.

G. Gutin and A. P. Punnen. *The traveling salesman problem and its variations.* Combinatorial Optimization. Kluwer Academic Publishers, 2002.

M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming, Series A*, 1(1):6–25, 1971.

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, CIRRELT, Montreal, Canada, 2011.

Institute of Logistics and Distribution Management. The 1985 survey of distribution costs, 1985.

S. Irnich and D. Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18(3):391–406, 2006.

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

M. Jepsen, S. Spoorendonk, and S. Røpke. A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science*, 2011. Forthcoming.

R. Jonker and T. Volgenant. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2(4):161–163, 1983.

M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M. Ball, T. L. Magnanti, C. L. Monma, and G. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 255–330. North Holland, Amsterdam, 1995.

R. Karp. A patching algorithm for the nonsymmetric traveling-salesman problem. *SIAM Journal on Computing*, 8(4):561–573, 1979.

R. M. Karp. Reducibility among combinatorial optimization problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

N. Kohl and O. B. G. Madsen. An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. *Operations Research*, 45(3):395–406, 1997.

N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1): 101–116, 1999.

T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76, 1957.

A. A. Kuehn and M. J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9(4):643–666, 1963.

B. J. LaLonde and P. H. Zinszer. *Customer service: meaning and measurement.* National Council of Physical Distribution Management, Chicago, 1976.

A. Langevin. *Planification des tournées de véhicules.* PhD thesis, École Polytechnique de Montréal, 1988.

A. Langevin, F. Soumis, and J. Desrosiers. Classification of travelling salesman problem formulations. *Operations Research Letters*, 9(2):127–132, 1990.

A. Langevin, M. Desrochers, J. Desrosiers, S. Gélinas, and F. Soumis. A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. *Networks*, 23(7):631–640, 1993.

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

E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*, volume 40. Holt, Rinehart and Winston, 1976.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The traveling salesman problem: a guided tour of combinatorial optimization.* Wiley, Chichester, UK, 1985.

J. K. Lenstra and A. H. G. Rinnooy Kan. Some simple applications of the travelling salesman problem. *Operational Research Quarterly*, 26(4):717–733, 1975.

J.-Q. Li. A computational study of bi-directional dynamic programming for the traveling salesman problem with time windows. Working paper, 2009.

J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Operations Research*, 11(6):972–989, 1963.

L. López Ibáñez and C. Blum. Beam-ACO for the travelling salesman problem with time windows. *Computers & Operations Research*, 37(9):1570–1583, 2010.

R. J. Loulou. On commodity flow formulations for the TSP. Working paper, McGill University, Montréal, 1988.

R. F. Love, J. G. Morris, and G. O. Wesolowsky. *Facilities location: models & methods.* Publications in operations research series. North-Holland, 1988.

J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming, Series A*, 100 (2):423–445, 2004.

A. S. Manne. Plant location under economies-of-scale-decentralization and computation. *Management Science*, 11(2):213–235, 1964.

K. Menger. Das botenproblem. *Ergebnisse Eines Mathematischen Kolloquiums*, 2: 11–12, 1932.

D. Mester and O. Bräysy. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, 32(6): 1593–1614, 2005.

C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7(4):326–329, 1960.

D. L. Miller and J. F. Pekny. Results from a parallel branch and bound algorithm for the asymmetric traveling salesman problem. *Operations Research Letters*, 8(3): 129–135, 1989.

A. Mingozzi, N. Christofides, and E. Hadjiconstantinou. An exact algorithm for the vehicle routing problem based on the set partitioning formulation. Technical report, University of Bologna, Bologna, Italy, 1994.

A. Mingozzi, L. Bianco, and S. Ricciardelli. Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints. *Operations Research*, 45(3):365–377, 1997.

A. Mingozzi, R. Roberti, and P. Toth. An exact algorithm for the multi-trip vehicle routing problem. *INFORMS Journal on Computing*, 2012.

P. B. Mirchandani and R. L. Francis. *Discrete location theory.* Interscience series in discrete mathematics and optimization. Wiley, 1990.

V.-P. Nguyen, C. Prins, and C. Prodhon. A multi-start evolutionary local search for the two-echelon location routing problem. In M. Blesa, C. Blum, G. Raidl, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 6373 of *Lecture Notes in Computer Science*, pages 88–102. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010.

J. W. Ohlmann and B. W. Thomas. A compressed-annealing heuristic for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 19(1):80–90, 2007.

A. Olivera. *Memorias adaptativas para el problema de ruteo de vehículos con múltiples viajes*. PhD thesis, Instituto de Computación - Facultad de Ingeniería - Universidad de la República - Montevideo, Uruguay, 2005.

A. Olivera and O. Viera. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34(1):28–47, 2007.

T. Öncan, I. Kuban Altinel, and G. Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654, 2009.

M. Padberg and T. Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming, Series A*, 52(1-3):315–357, 1991.

M. W. Padberg and G. Rinaldi. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming, Series A*, 47(1-3):19–36, 1990a.

M. W. Padberg and G. Rinaldi. Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming, Series A*, 47(1-3):219–257, 1990b.

J. F. Pekny and D. L. Miller. A parallel branch and bound algorithm for solving large asymmetric traveling salesman problems. *Mathematical Programming, Series A*, 55 (1-3):17–33, 1992.

G. Perboli, R. Tadei, and F. Masoero. New families of valid inequalities for the two-echelon vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 36(2): 639–646, 2010.

G. Perboli, R. Tadei, and D. Vigo. The two-echelon capacitated vehicle routing problem: models and math-based heuristics. *Transportation Science*, 45(3):364–380, 2011.

G. Pesant, M. Gendreau, J.-Y. Potvin, and J.-M. Rousseau. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32(1):12–29, 1998.

R. J. Petch and S. Salhi. A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133(1-3):69–92, 2004.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

J.-Y. Potvin and S. Bengio. The vehicle routing problem with time windows part II: Genetic search. *INFORMS Journal on Computing*, 8(2):165–172, 1996.

G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

G. Reinelt. *The traveling salesman problem: computational solutions for TSP applications.* Lecture Notes in Computer Science. Springer - Verlag, Berlin, 1994.

G. Righini and M. Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006.

G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008.

R. Roberti and P. Toth. Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison. Submitted for publication, 2012.

J. B. Robinson. On the hamiltonian game (a traveling salesman problem). Technical Report RM-303, RAND Research Memorandum, 1949.

Y. Rochat and É. D. Taillard. Probabilistic intensification and diversification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, 1995.

S. Ropke. *Heuristic and exact algorithms for vehicle routing problems.* PhD thesis, Computer Science Department, University of Copenhagen (DIKU), Copenhagen, 2005.

S. Ropke. Personal communication, 2010. January 29.

S. Salhi and R. J. Petch. A GA based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms*, 6(4):591–613, 2007.

S. C. Sarin, H. D. Sherali, and A. Bhootra. New tighter polynomial length formulations for the asymmetric travelling salesman problem with and without precedence constraints. *Operations Research Letters*, 33(1):62–70, 2005.

M. W. P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1):285–305, 1985.

H. D. Sherali and P. J. Driscoll. On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems. *Operations Research*, 50 (4):656–669, 2002.

H. D. Sherali, S. C. Sarin, and P.-F. Tsai. A class of lifted path and flow-based formulations for the asymmetric travelling salesman problem with and without precedence constraints. *Discrete Optimization*, 3(1):20–32, 2006.

D. Simchi-Levi, X. Chen, and J. Bramel. *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics and Supply Chain Management.* Springer Series in Operations Research and Financial Engineering. Springer New York, 2005.

T. H. C. Smith, V. Srinivasan, and G. L. Thompson. Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems. In B. H. Korte P. L. Hammer, E. L. Johnson and G. L. Nemhauser, editors, *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*, pages 495–506. Elsevier, 1977.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):254–265, 1987.

É. D. Taillard. Personal communication, 2009. June 16.

É. D. Taillard, G. Laporte, and M. Gendreau. Vehicle routeing with multiple use of vehicles. *Journal of the Operational Research Society*, 47(8):1065–1070, 1996.

K. C. Tan, Y. H. Chew, and L. H. Lee. A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, 172(3):855–885, 2006.

E. Taniguchi. *City logistics: network modelling and intelligent transport systems*. Pergamon, 2001.

E. Taniguchi and R. G. Thompson. *Innovations in city logistics*. Nova Science Publishers, 2008.

R. E. Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.

P. Toth and D. Vigo. *The vehicle routing problem*. SIAM monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics, 2002.

R. Wolfler Calvo. A new heuristic for the traveling salesman problem with time windows. *Transportation Science*, 34(1):113–124, 2000.

R. T. Wong. Integer programming formulations of the traveling salesman problem. In *Proceedings of the IEEE international conference of circuits and computers*, 1980.