Rafael Martinelli Pinto

Exact Algorithms for Arc and Node Routing Problems

TESE DE DOUTORADO

DEPARTAMENTO DE INFORMÁTICA

Programa de Pós–Graduação em Informática

Rio de Janeiro
August 2012

Pontifícia Universidade Católica
do Rio de Janeiro

Rafael Martinelli Pinto

Exact Algorithms for Arc and Node Routing Problems

TESE DE DOUTORADO

Thesis presented to the Programa de Pós–Graduação em Informática of the Departamento de Informática – PUC-Rio as partial fulfillment of the requirements for the degree of Doutor.

Advisor: Prof. Marcus Vinicius Soledade Poggi de Aragão

Rio de Janeiro
August 2012

Pontifícia Universidade Católica
do Rio de Janeiro

Rafael Martinelli Pinto

Exact Algorithms for Arc and Node Routing Problems

Thesis presented to the Programa de Pós–Graduação em Informática of the Departamento de Informática of Centro Técnico Científico – PUC-Rio as partial fulfillment of the requirements for the degree of Doutor. Approved by the examining committee undersigned:

Prof. Marcus Vinicius Soledade Poggi de Aragão
Advisor
Departamento de Informática – PUC-Rio

Prof. Eduardo Sany Laber
Departamento de Informática – PUC-Rio

Prof. Alexandre Street de Aguiar
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Abilio Pereira de Lucena Filho
UFRJ

Prof. Alysson Machado Costa
USP

Prof. Artur Alves Pessoa
UFF

Prof. José Eugenio Leal
Coordinator of the Centro Técnico Científico – PUC-Rio

Rio de Janeiro, August 28, 2012

Rafael Martinelli Pinto

Rafael Martinelli Pinto obtained a MSc in Computer Science from Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) and a BSc in Computer Science from Universidade do Estado do Rio de Janeiro (UERJ). During the doctoral he held a scholarship from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), maintaining an excellent academic performance. He has experience as a software developer of decision making systems using mathematical optimization techniques, as a teaching assistant in graduate courses at PUC-Rio and as a temporary teacher at UERJ.

Bibliographic Data

Martinelli, Rafael

Exact Algorithms for Arc and Node Routing Problems / Rafael Martinelli Pinto; Advisor: Marcus Vinicius Soledade Poggi de Aragão. – 2012.

106 f: il. (color.) ; 30 cm

Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2012.

Inclui bibliografia

1. Informática – Teses. 2. Problemas de Roteamento. 3. Geração de Colunas. 4. Separação de Cortes. 5. Branch-Cut-and-Price. 6. Programação Inteira. I. Poggi, Marcus. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my mother and my son.

Acknowledgments

To my mother Véra, to whom I owe everything I have achieved in life.

To my son Luca, for bringing me happiness in a way I had never experienced before.

To my family, who gave me many valuable lessons throughout my life.

To my advisor Marcus Poggi, who always supported and encouraged me, even in the most difficult times, since the beginning of my MSc more than seven years ago.

To my friend and doctoral colleague Diego Pecin, which assiduously participated in most of the development of this work.

To my friends and research colleagues Anand Subramanian, Eduardo Uchoa and Artur Pessoa, who always were willing to have long conversations about our research, even after midnight.

To all who have shared moments with me in recent years. My friends from Méier, CEFET, WhileTrue, PUC, among several others. It is impossible to list them all.

To Pontifícia Universidade Católica do Rio de Janeiro, which provided all the necessary infrastructure for the proper development of my work.

To the Brazilians who pay taxes and to the CNPq, for the financial support received over the years.

Resumo

Martinelli, Rafael; Poggi, Marcus. **Algoritmos Exatos para Problemas de Roteamento em Arcos e em Vértices**. Rio de Janeiro, 2012. 106p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Os problemas de roteamento estão entre os problemas combinatórios mais difíceis de encontrar limites melhores do que os existentes ou de provar novas soluções ótimas. Nesta tese, são abordados o *Capacitated Arc Routing Problem* (CARP) e o *Generalized Vehicle Routing Problem* (GVRP). Em ambos os problemas, existe um conjunto de clientes os quais estão espalhados por um grafo dado, onde cada cliente possui uma demanda que deve ser atendida por exatamente um veículo de um conjunto de veículos idênticos. Os custos de travessia e o vértice de depósito são dados. O objetivo é encontrar rotas que coletam todas as demandas com custo mínimo, sem exceder a capacidade de nenhum veículo. No CARP, os clientes são um subconjunto de arestas, chamadas de arestas *requireds*, e para o GVRP, cada cliente é um subconjunto de vértices, chamado de grupo, onde cada grupo deve ser atendido visitando-se exatamente um vértice deste grupo. Além disto, vale notar que quando cada grupo possui apenas um vértice, o problema passa a ser o *Capacitated Vehicle Routing Problem* (CVRP). Primeiramente, são investigados métodos para melhorar os limites inferiores de instâncias de grande porte. É proposta a exploração da velocidade de uma heurística *dual ascent* para gerar cortes de capacidade. Em seguida, é apresentado um algoritmo de geração de colunas com um *pricing* eficiente para um tipo especial de rota não-elementar. O *pricing* proposto combina a técnica *Decremental State-Space Relaxation* (DSSR) com limites de complemento. Estas técnicas permitem o fortalecimento da regra de dominância entre as rotas, reduzindo drasticamente o número total de rótulos utilizados pela programação dinâmica. Finalmente, um algoritmo de *branch-cut-and-price* é criado o qual usa a geração de colunas e a separação de cortes previamente apresentadas. Além disto, este *branch-cut-and-price* é implementado usando *strong branching* e fixação por custo reduzido. Ao fim de cada parte, são apresentados resultados computacionais os quais avaliam a qualidade dos algoritmos propostos, os quais obtém novos limites inferiores para um grande número de instâncias do CARP e do GVRP.

Palavras–chave

Problemas de Roteamento; Geração de Colunas; Separação de Cortes; Branch-Cut-and-Price; Programação Inteira.

Abstract

Martinelli, Rafael; Poggi, Marcus. **Exact Algorithms for Arc and Node Routing Problems**. Rio de Janeiro, 2012. 106p. DSc Thesis – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Routing problems stand among the hardest combinatorial problems to find high quality bounds or to prove new optimal solutions. In this thesis, we tackle the Capacitated Arc Routing Problem (CARP) and the Generalized Vehicle Routing Problem (GVRP). For both problems, there are a set of customers spread over a given graph, where each customer has a demand which must be serviced by exactly one vehicle from a set of identical vehicles. The traversal costs and a depot vertex are given. The objective is to find routes that collect all the demands, without exceeding the capacity of any vehicle, at minimum cost. For the CARP, the customers are a subset of edges, called the required edges, and for the GVRP, each customer is a subset of vertices, called clusters, where each cluster must be serviced by visiting exactly one vertex of it. Furthermore, it is noteworthy that when every cluster contains just a single vertex, the problem is the Capacitated Vehicle Routing Problem (CVRP). Firstly, we investigate methods to improve lower bounds for large scale instances. We propose to explore the speed of a new dual ascent heuristic to generate capacity cuts. The quality of the cuts found is next improved with a new exact separation which is used in the linear program resolution that follows the dual heuristic. Following, we present a column generation algorithm with an efficient pricing for a special kind of non-elementary routes. The proposed pricing algorithm combines Decremental State-Space Relaxation (DSSR) technique with completion bounds. These techniques allow the strengthening of the domination rule between routes, drastically reducing the total number of labels used during the dynamic programming. Finally, we devise a branch-cut-and-price algorithm which uses the previously presented column generation and cut separation. Moreover, this branch-cut-and-price is implemented using strong branching and reduced cost fixing. At the end of each part, we present computational experiments which evaluate the quality of the proposed algorithms and show new best lower bounds for a large number of CARP and GVRP instances.

Keywords

Routing Problems; Column Generation; Cut Separation; Branch-Cut-and-Price; Integer Programming.

Contents

List of Figures

List of Tables

I
Introduction

Among the combinatorial optimization problems studied in the literature over the last decades, Vehicle Routing Problems (VRPs) play an important role for several reasons. Since the seminal work of Dantzig and Ramser in 1959 [21], where the authors introduced a problem of oil distribution from supply stations, VRPs have become widely investigated. This interest arises from the existence of many practical applications and from the difficulty of solving them.

A large number of problems can be modeled as a vehicle routing problem, specially problems in transportation, distribution and logistics. It is remarked in Toth and Vigo [75] that a high percentage of the value added to commercial goods reflects distribution costs, therefore the use of computational methods for planning and operation is justified, as these often result in significant savings in the overall costs.

VRPs consist in finding a set of routes for a fleet of vehicles. These vehicles are located at one or more special places, called the depots. Spread over a network, there is a set of customers, each one requiring exactly one vehicle to service its demands. Notice that customers' demands may be the collection of goods, the delivery of goods or in some cases they may even be both. Typically, the solution of this problem imposes the routes to begin and end in the same depot, while meeting all customers' demands and minimizing the overall transportation cost. Besides, some operational constraints may be imposed. For instance, there may be a limitation on the capacity of each vehicle or periods of the day on which each customer can be visited.

It is important to notice that part of the difficulty of solving any vehicle routing problem comes from the fact that it generalizes the Traveling Salesman Problem (TSP), which is one of the most famous combinatorial problems and it is known to be a strongly $\mathcal{NP}$-hard problem [32]. Furthermore, depending on the operational constraints imposed, other hard problems may come into play, and the VRP will only admit solutions in the intersection of the feasible solution space of those problems. For instance, if the fleet of vehicles contains more than one vehicle, the operation of selecting which vehicle will service which customer can be considered as solving a Bin Packing Problem, which is

also known to be an $\mathcal{NP}$-hard problem [32].

We can cite some important vehicle routing problems which have been extensively studied in the literature recently. The Capacitated Vehicle Routing Problem (CVRP) is the most well-known. In this problem, the customers are located at the vertices of the network and there is a special vertex representing the single depot. The fleet of vehicles is limited, homogeneous and the vehicles have a maximum capacity, which cannot be exceeded. The Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) is a variation of the CVRP and defines time windows for each customer to be serviced. Another variation of the CVRP is the Multi Depot Capacitated Vehicle Routing Problem (MDCVRP), where more than one depot is considered. Furthermore, another vehicle routing problem, which is not a variation of the CVRP, is the Capacitated Arc Routing Problem (CARP). The main difference from the CVRP is that the customers' demands are spread along a subset of the edges of the network. These edges are called the required edges and each one must be serviced by a single vehicle. Finally, we can cite the Generalized Vehicle Routing Problem (GVRP), which is a generalization of the CVRP and the TSP. In this problem, the customers are sets of vertices, called clusters, and each one has to be serviced by a single vehicle by visiting just one vertex of the cluster.

During the last years, various types of computational approaches have been used with the intent of solving such problems. Among others, exact approaches based on solving mixed-integer linear programming formulations is one of the most successful nowadays. The best results are obtained by solving the problems using a formulation with an exponential number of columns (variables). In order to deal with this large number of columns, a technique called column generation is used. This technique was introduced in 1958 by Ford and Fulkerson [30] and it was later extended to mixed-integer programming by Gilmore and Gomory in 1961 [35] to solve the Cutting Stock Problem. It starts with a small number of columns and, at each iteration, it selects (prices) some columns to be inserted in the formulation which improve the solution. This operation is done by solving an auxiliary subproblem, called the pricing subproblem, which should be an easy problem to solve. Usually it is not true for VRPs, as the pricing subproblem can also be an $\mathcal{NP}$-hard problem. The immediate drawback of this approach is that it allows for solving only the linear relaxation of the mixed-integer formulation, i.e., all variables are considered as continuous variables.

Analyzing the structure of each problem, some valid inequalities can be devised and inserted in the formulation in order to improve the solution of

the linear relaxation given by the column generation algorithm. The search for such inequalities is called cut separation and the algorithm which solves a formulation with cut separation is known as a cutting plane algorithm. Notice that this approach need not necessarily be used within a column generation.

Although there are some cuts which enforce the integrality of the continuous variables, they are not widely used within column generations yet. For this purpose, there is a well-known technique, called Branch-and-Bound, which enumerates all possible integer solutions through a search tree, discarding the branches with solutions greater than a known upper bound. This technique, when used together with column generation and cut separation is called *Branch-Cut-and-Price* (BCP).

The main objective of this thesis is to devise new exact algorithms for some VRPs. The focus is primarily on the CARP, however we also devise some algorithms for the GVRP, which allow us to solve other VRPs. In this work, we solve CVRP instances as GVRP instances.

I.1 Contributions

The main contributions of this work are described as follows.

- Proposes a new exact cut separation for the CARP, which outperforms the existing separation, as shown in the experimental results.
- Develops a dual ascent heuristic for the CARP, which improves the exact cut separation, being able to be applied on large scale instances.
- Describes an efficient implementation for the state-of-the-art pricing algorithm for both CARP and GVRP, which is used with some cut separations and is able to improve several lower bounds for the GVRP, also obtaining some optimal solutions.
- Assembles all the algorithms presented in a new Branch-Cut-and-Price algorithm for the CARP, which is able to obtain new lower bounds and optimal solutions for some competitive instances.

The research presented in this thesis has also generated some published works, as the column generation with elementary and non-elementary routes for the CARP [58], the Branch-Cut-and-Price with non-elementary pricing for the CARP [59] and the dual ascent with exact cut separation for the CARP [60]. Furthermore, there is a working paper on the efficient implementation for the state-of-the-art pricing algorithm for both CARP and GVRP which is currently in its concluding stage.

I.2 Thesis Outline

This work is organized as follows.

- Chapter II describes the problems discussed in this work, as well as their literature reviews.
- Chapter III shows some of the known formulations for each problem.
- Chapter IV introduces an exact cut separation and a dual ascent heuristic. The dual ascent heuristic is used for the CARP as hot start to a cutting plane algorithm. This cutting plane algorithm uses an existing exact cut separation and the presented new one.
- Chapter V presents a column generation algorithm which uses an efficient restricted non-elementary route pricing.
- Chapter VI presents a branch-cut-and-price for the CARP which put together all the algorithms described in the preceding chapters.
- Chapter VII contains the conclusions of the work.

II
A Review of the Problems

This chapter details and reviews the problems tackled in this work, particularly the Capacitated Arc Routing Problem, the Capacitated Vehicle Routing Problem and the Generalized Vehicle Routing Problem. We also explain how other routing problems can be solved using the GVRP. Furthermore, some detailed and comprehensive surveys on routing problems can be found in [36], [75], [20], [39].

II.1 The Capacitated Arc Routing Problem

The Capacitated Arc Routing Problem (CARP) can be defined as follows. Consider a connected undirected graph $G = (V, E)$, with vertex set V and edge set E, costs $c : E \rightarrow \mathbb{Z}_0^+$, demands $d : E \rightarrow \mathbb{Z}_0^+$, a set of identical vehicles $\mathcal{K}$ with capacity Q and a distinguished depot vertex labeled 0. Define $E_R = \{e \in E | d_e > 0\}$ as the set of required edges. Let $\mathscr{R}$ be a set of closed routes which start and end at the depot, where the edges in a route can be either *serviced* or *deadheaded*. An edge is serviced when a vehicle traverses it collecting all its demand and is deadheaded when a vehicle traverses it without collecting any demand. The set $\mathscr{R}$ is a feasible CARP solution if: (i) each required edge is serviced by exactly one route in $\mathscr{R}$; (ii) The sum of demands of the serviced edges in each route in $\mathscr{R}$ does not exceed the vehicle capacity Q and (iii) $|\mathscr{R}| = |\mathcal{K}|$. We want to find a solution minimizing the sum of the costs of the routes. Notice that it corresponds to minimize the sum of the deadheaded edges' cost in the routes.

The CARP has several applications in real life. One can think of this problem as a garbage collection problem. Suppose a company is responsible for collecting the garbage of a neighborhood. This company has a set of identical vehicles available in its base and knows *a priori* the amount of garbage generated on each street. As a company, its objective is to minimize the operational costs. So, the company needs to define routes for the vehicles which minimize the total distance traveled. These routes must begin and end at the company's base, none of the vehicles can have its capacity violated, the

garbage of a given street may be collected only by a single vehicle and all the garbage must be collected. Other possible applications for this problem are street sweeping, winter gritting, electric meter reading and airline scheduling, as described in the work of Wøhlk [77].

This problem is strongly $\mathcal{NP}$-hard as shown by Golden and Wong in 1981 [38], where it was first proposed. Since then, several solution algorithms were suggested for it, following different approaches. We can cite some specific primal heuristics like Augment-Merge [38, 37], Path-Scanning [37], Parallel-Insert [18], Construct-Strike [64] and Augment-Insert [65]. There are several heuristics for obtaining lower bounds, like Matching [38], Node Scanning [3], Matching-Node Scanning [66], Node Duplication [43], Hierarchical Relaxations [2] and Multiple Cuts Node Duplication [78].

Recently, most of the non-exact algorithms suggested for the CARP are based on metaheuristics. All kinds of metaheuristics have already been proposed for the CARP, some of them are Simulated Annealing [26], the CARPET Tabu Search [42], Genetic Algorithm [48], Memetic Algorithm [49], Ant Colony [23], Guided Local Search [13], Deterministic Tabu Search [17], Improved Tabu Search [61], Improved Memetic Algorithm [62], Improved Ant Colony [73] and Improved Local Search through a transformation for the Capacitated Vehicle Routing Problem [60].

Being a hard problem, it is very difficult to solve the CARP to optimality. The works which tried to achieve this usually use integer programming. The first integer programming formulation was proposed by Golden and Wong [38] and since then some other formulations were proposed by Belenguer and Benavent [9] and Letchford [52], who also proposed valid inequalities for the problem. These integer formulations were solved using techniques such as Branch-and-Bound [44], Cutting Planes [10, 1], Column Generation [40, 53, 58], Branch-and-Cut [50], Branch-Cut-and-Price [59], Cut-First Branch-and-Price-Second [16] and Cut-and-Column Generation based technique combined with a Set Partitioning Approach [7].

II.2 The Capacitated Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) is defined over a graph $G = (V, E)$, where V is the vertex set and $E = \{(i, j)|i, j \in V, i < j\}$ is the edge set. From the definition of the edge set, the graph G is a complete graph. The vertex 0 is the depot and the others are the customers. The depot is the location of a fleet of vehicles containing a set $\mathcal{K}$ of identical vehicles of capacity Q. A cost function $c : E \to \mathbb{Z}_0^+$ is associated with the edges of G while a demand function $d : V \to \mathbb{Z}_0^+$ is associated with its vertices (the depot has

demand $d_0 = 0$). The objective is to find a set of closed routes with minimum total cost such that: (i) each customer must be visited exactly once and by a single vehicle which must service all of its demand; (ii) all of the $|\mathcal{K}|$ vehicles must be used and each vehicle must be associated with only one route; (iii) Vehicle routes must start and end at the depot, servicing at least one vertex; (iv) The total serviced demand by each vehicle must not exceed its capacity Q.

This problem was proposed on the work of Dantzig and Ramser [21] and is strongly $\mathcal{NP}$-hard, since it is a generalization of the TSP, which, as mentioned before, is known to be strongly $\mathcal{NP}$-hard [32].

Since it was proposed, several works have been published for the CVRP. Since it is not our intention to do a complete bibliographic review of the problem, we refer the reader the following recent CVRP surveys: the annotated bibliography on metaheuristics done by Gendreau et al. in 2008 [33], the survey on exact and heuristic algorithms done by Laporte in 2009 [51] and the survey on the latest exact approaches done by Baldacci et al. in 2010 [6].

Nevertheless, these surveys do not cite the most recent exact approaches for the CVRP. Recently, Baldacci et al. [5] introduced the state-of-the-art exact solution algorithm for vehicle routing problems, which is further improved in the current work. The approach from Baldacci et al. [5] is used to improve the solutions obtained by the best algorithms for the CVRP which before were the branch-and-cut from Lysgaard et al. [57], the robust branch-cut-and-price of Fukasawa et al. [31], the set partitioning with additional cuts based algorithm of Baldacci et al. [4] and the extended formulation with capacity-indexed variables of Pessoa et al. [69].

II.3 The Generalized Vehicle Routing Problem

The Generalized Vehicle Routing Problem (GVRP) can be defined as follows. Let $G = (V, E)$ be a graph with vertex set V and edge set E. There is a special vertex 0 called the depot. The vertices are partitioned into disjoint sets, called clusters, $C = \{C_0, C_1, \ldots, C_t\}$, where $C_0 = \{0\}$ contains only the depot. Given the cluster index set $M = \{0, 1, \ldots, t\}$, let $\mu(i) \in M$ be defined, for each vertex $i \in V$, as the index of the cluster which contains i. There exists a demand function $d : M \to \mathbb{Z}^+$ associated with all clusters, in which the depot has demand $d_0 = 0$. These demands are to be serviced by a set $\mathcal{K}$ of identical vehicles with capacity Q, located at the depot. The edge set $E = \{\{i, j\} | i, j \in V, \mu(i) \neq \mu(j)\}$ contains the edges between all pairs of vertices from different clusters. Associated with these edges, there exists

a traversal cost function $c : E \to \mathbb{Z}_0^+$. Let $\mathscr{R}$ be the set of all possible closed routes starting and ending at the depot. The objective of the GVRP is to select a subset of k routes from $\mathscr{R}$ which: (i) minimizes the total traversal cost; (ii) the demand from every cluster is serviced by a single vehicle on exactly one vertex from each cluster; (iii) the total demand serviced by each route does not exceed the vehicle capacity Q.

The GVRP is a generalization of the Capacitated Vehicle Routing Problem (CVRP) and the Generalized Traveling Salesman Problem (GTSP). When all the clusters contain only one vertex, it is simply the CVRP. Similarly, when there is only one vehicle, it is simply the GTSP. It is clear that, when both conditions are true, it is simply the Traveling Salesman Problem (TSP). In the view of this, it is easy to see that any solution algorithm given to the GVRP can be directly used to solve these problems. In the chapters that follow, any algorithm designed for the GVRP will also be applied to the CVRP in this way.

This problem is strongly $\mathcal{NP}$-hard and has gained attention in the literature in recent years. As far as we know, the first published work to deal with this problem is Ghiani and Improta [34], where a transformation to the Capacitated Arc Routing Problem (CARP) is presented in order to use the existing algorithms for the CARP. One instance was proposed and solved using this approach.

Since then, few works have been published on the GVRP. Recently, the work of Bektaş et al. [8] has proposed four formulations for the GVRP. After extensive experiments with these formulations, a branch-and-cut algorithm was devised using one of them, an undirected formulation with an exponential number of constraints. The reader may consult this paper for further details on these formulations.

Furthermore, the work of Bektaş et al. [8] also points out some applications for the GVRP. We can cite some of them: routing vessels in maritime transportation, health-care logistics, urban waste collection and survivable telecommunication design.

III
Mathematical Formulations

This chapter presents the formulations used in this work. Firstly, we show the set partitioning approach, which can be applied to most of known routing problems. Next, for each problem, we show some specific existing formulations and how to transform them to obtain the set partitioning formulation.

III.1 The Set Partitioning Approach

The Set Partitioning Approach is a simple way to formulate VRPs. In this formulation, a binary variable λ_r is created for each route. As mentioned before, a route is a walk which starts at a depot, services a subset of the customers and returns to the same depot. Given the set of all possible routes, called Ω, the objective is to select exactly $|\mathcal{K}|$ routes, where all customers are serviced and total cost is minimized. This formulation is presented below.

$$\text{Minimize} \sum_{r\in\Omega} c_r\lambda_r \tag{III.1}$$

$$\text{s.t.} \quad \sum_{r\in\Omega} \lambda_r = |\mathcal{K}| \tag{III.2}$$

$$\sum_{r\in\Omega} a_r^i\lambda_r = 1 \quad \forall i \in \mathcal{C} \tag{III.3}$$

$$\lambda_r \in \{0,1\} \quad \forall r \in \Omega\ . \tag{III.4}$$

The objective function (III.1) minimizes the overall cost of the selected routes, where c_r is the cost of route r. Constraints (III.2) impose the number $|\mathcal{K}|$ of vehicles to be used. Given the binary constant a_r^i, which indicates whether customer i is serviced by route r, constraints (III.3) guarantee that each customer is serviced by exactly one route. Finally, constraints (III.4) force the variables to be binary.

It is easy to see that Ω contains an exponential number of routes, thus evolving at an exponential number of variables. This is the drawback of this formulation, as it is necessary to use column generation in order to deal with the large number of variables involved. Anyway, the quality of the solutions

obtained using this approach is very good, what can be evidenced by the existence of several works using this formulation to solve VRPs.

Ideally, it is preferable that the set Ω contains only feasible routes, called elementary routes. However, as we will show later on, this restriction turns the pricing subproblem into a problem that is hard to solve. A way to handle this difficulty is to relax the elementarity of the routes, allowing them to service same customer more than once. This kind of non-elementary route is also known as the *q-routes*. Since pricing non-elementary routes is less difficult than pricing elementary routes, it is expected as a side effect a reduction in the value of the bounds obtained. For this reason, recent works try to find a compromise between elementary and non-elementary routes, restricting the non-elementarity in different ways.

III.2 Formulations for the CARP

(a) The Two-Index Formulation

An intuitive formulation for the CARP is the *Two-Index Formulation*. According to Letchford and Oukil [53], which was also the first work to use this name to the formulation, this approach was first suggested by Welz [76] in 1994, but it was only explored in depth in the work of Belenguer and Benavent [11] in 1998. In this formulation, there is a binary variable x_e^k, for each required edge and each vehicle, which is 1 if a vehicle k services the required edge e and 0 otherwise. Furthermore, there is an integer variable z_e^k, for every edge and each vehicle, which represents the number of times vehicle k deadheads edge e. These variables are used as follows.

$$\text{Minimize} \sum_{k\in\mathcal{K}} \left(\sum_{e\in E_R} c_e x_e^k + \sum_{e\in E} c_e z_e^k \right) \tag{III.5}$$

subject to

$$\sum_{k\in\mathcal{K}} x_e^k = 1 \qquad \forall e \in E_R \tag{III.6}$$

$$\sum_{e\in E_R} d_e x_e^k \leq Q \qquad \forall k \in \mathcal{K} \tag{III.7}$$

$$\sum_{e\in\delta_R(S)} x_e^k + \sum_{e\in\delta(S)} z_e^k \geq 2x_f^k \qquad \forall S \subseteq V\backslash\{0\}, f \in E_R(S), k \in \mathcal{K} \tag{III.8}$$

$$\sum_{e\in\delta_R(S)} x_e^k + \sum_{e\in\delta(S)} z_e^k \equiv 0 \bmod (2) \qquad \forall S \subseteq V\backslash\{0\}, k \in \mathcal{K} \tag{III.9}$$

$$x_e^k \in \{0,1\} \qquad \forall e \in E_R, k \in \mathcal{K} \tag{III.10}$$

$$z_e^k \in \mathbb{Z}_0^+ \qquad \forall e \in E, k \in \mathcal{K} \; . \tag{III.11}$$

The objective function (III.5) minimizes the overall cost of the edges used. Constraints (III.6) assure that all required edges are serviced. Constraints (III.7) limit the total demand serviced by each vehicle to the capacity Q. Given a vertex set S, let $\delta(S) = \{(i,j) \in E : i \in S \wedge j \notin S\}$ be the set of edges which have one endpoint inside S and the other outside S and let $\delta_R(S) = \{(i,j) \in E_R : i \in S \wedge j \notin S\}$ be the set of *required* edges which have one endpoint inside S and the other outside S. Analogously, define $E(S) = \{(i,j) \in E : i \in S \wedge j \in S\}$ and $E_R(S) = \{(i,j) \in E_R : i \in S \wedge j \in S\}$ as the sets of edges with both endpoints inside S. Constraints (III.8) assure that every route is connected and constraints (III.9) force every route in the solution to induce an Eulerian graph. Notice that these latter constraints can be easily modeled as MIP constraints by introducing an integer variable which should appear on the right-hand side with a coefficient of 2.

The drawback of this formulation comes from its large number of variables and its symmetry. Letchford and Oukil [53] notice that there are at least $|\mathcal{K}|!$ optimal solutions. These turn this formulation prohibitive to be solved for instances with a large number of vehicles. The work of Ghiani et al. [50] shows how this symmetry can be tackled in order to use the formulation in a branch-and-cut algorithm.

(b) The One-Index Formulation

In their work of 2003, Belenguer and Benavent [10] developed another CARP formulation, usually referred as the *One-Index Formulation* [53]. In contrast to other approaches, this formulation only makes use of variables representing the deadheading of an edge. In addition, all vehicles are aggregated. Due to these simplifications, this formulation is a relaxation, i.e., it may result in an infeasible solution for the original problem. Moreover, it is strongly $\mathcal{NP}$-hard to decide whether a solution given by this formulation is feasible or not. Nevertheless, these issues do not prevent such formulation from giving very good lower bounds in practice.

For each deadheaded edge e, there is an integer variable z_e representing the number of times edge e is deadheaded by *any* vehicle. Given a vertex set S, with $|\delta_R(S)|$ odd, it is easy to conclude that at least one edge in $\delta(S)$ must be deadheaded because each vehicle entering the set S must leave and return to the depot. This is the definition of the following *odd-edge cutset inequalities*.

$$\sum_{e \in \delta(S)} z_e \geq 1 \quad \forall S \subseteq V \backslash \{0\}, |\delta_R(S)| \text{ odd} \tag{III.12}$$

Furthermore, given a lower bound on the number of vehicles needed to meet the demands in $\delta_R(S) \cup E_R(S)$, called $k(S)$, we can state that at least $k(S)$ vehicles must enter and leave the set S. Thus, at least $2k(S) - |\delta_R(S)|$ times an edge in $\delta(S)$ will be deadheaded. If this value is positive, we can define the following *capacity cut*.

$$\sum_{e \in \delta(S)} z_e \geq 2k(S) - |\delta_R(S)| \quad \forall S \subseteq V \backslash \{0\} \tag{III.13}$$

The best value for $k(S)$ is obtained by solving the Bin Packing problem for each vertex set S, but as shown by Garey and Johnson [32], this problem is an $\mathcal{NP}$-hard problem. Due to this fact, we prefer to use the approximation $k(S) = \lceil \sum_{e \in \delta_R(S) \cup E_R(S)} d_e / Q \rceil$.

Since the left-hand side of both (III.12) and (III.13) are the same, they can be represented in the formulation by using just a single constraint. This can be done by introducing $\alpha(S)$, which is defined as follows.

$$\alpha(S) = \begin{cases} \max\{2k(S) - |\delta_R(S)|, 1\}, \text{ if } |\delta_R(S)| \text{ is odd,} \\ \max\{2k(S) - |\delta_R(S)|, 0\}, \text{ if } |\delta_R(S)| \text{ is even.} \end{cases} \tag{III.14}$$

The odd-edge cutset inequalities and the capacity cuts are the only constraints of the one-index formulation, besides the limits on the z_e variables. Therefore, the formulation can be defined as follows.

$$\text{Minimize} \sum_{e \in E} c_e z_e \tag{III.15}$$

subject to

$$\sum_{e \in \delta(S)} z_e \geq \alpha(S) \quad \forall S \subseteq V \backslash \{0\} \tag{III.16}$$

$$z_e \in \mathbb{Z}_0^+ \qquad \forall e \in E \tag{III.17}$$

The objective function (III.15) minimizes the cost of the deadheaded edges. Constraints (III.16) combine cuts (III.12) and (III.13). It is important to notice that in order to obtain the overall cost of a solution, it is necessary to add the costs of the required edges ($\sum_{e \in E_R} c_e$) to the resulting cost.

Notice that, by the definition of the set S, it is easy to see that there is an exponential number of odd-edge cutset inequalities and capacity cuts. For this reason, it is not an option to generate all these cuts *a priori*. These must be generated in a cutting plane approach using a separation routine, as the

one to be presented in Chapter IV.

(c) The Set Partitioning Approach

The set partitioning formulation for the CARP can be obtained from the two-index formulation with the introduction of the binary variable λ_r for every possible CARP route. A CARP route is a walk which starts at the depot, traverses a set of required edges – servicing their demands – and then returns to the depot without exceeding the capacity of the vehicle. Between a pair of serviced edges, the route may traverses a set of deadheaded edges, which contribute for the total cost of the route, but not for the total capacity.

The new variables λ_r have a natural association with the x_e^k and z_e^k variables from the two-index formulation. Given the binary constant a_r^e, which is 1 if and only if route r services the required edge e and the integer constant b_r^e, which represents the number of times edge e is deadheaded in route r, the two-index formulation can be rewritten as follows.

$$\text{Minimize} \sum_{k\in\mathcal{K}} \left(\sum_{e\in E_R} c_e x_e^k + \sum_{e\in E} c_e z_e^k \right) \tag{III.18}$$

subject to

$$\sum_{r\in\Omega} a_r^e \lambda_r = \sum_{k\in\mathcal{K}} x_e^k \quad \forall e \in E_R \tag{III.19}$$

$$\sum_{r\in\Omega} b_r^e \lambda_r = \sum_{k\in\mathcal{K}} z_e^k \quad \forall e \in E \tag{III.20}$$

$$\sum_{r\in\Omega} \lambda_r = |\mathcal{K}| \tag{III.21}$$

$$\sum_{k\in\mathcal{K}} x_e^k = 1 \qquad \forall e \in E_R \tag{III.22}$$

$$\lambda_r \in \{0,1\} \quad \forall \lambda \in \Omega \tag{III.23}$$

$$x_e^k \in \{0,1\} \quad \forall e \in E_R, k \in \mathcal{K} \tag{III.24}$$

$$z_e^k \in \mathbb{Z}_0^+ \qquad \forall e \in E, k \in \mathcal{K}\ . \tag{III.25}$$

Constraints (III.19) and (III.20) create the relation between variables from the two-index formulation and the λ_r variables. Furthermore, due to the definition of a feasible CARP route and the introduction of the new variables, constraints (III.7), (III.8) and (III.9) are no longer required. Besides that, it is important to notice that due to the aggregation of the vehicles, constraint (III.21) is introduced in order to force the given number of vehicles to be used.

Variables x_e^k and z_e^k can be completely replaced using the relation constraints (III.19) and (III.20). This variable replacement and the relaxation

of the integrality constraints (III.23) generate the following Dantzig-Wolfe relaxation.

$$\text{Minimize} \sum_{r\in\Omega} c_r\lambda_r \tag{III.26}$$

subject to

$$\sum_{r\in\Omega} \lambda_r = |\mathcal{K}| \tag{III.27}$$

$$\sum_{r\in\Omega} a_r^e\lambda_r = 1 \qquad \forall e \in E_R \tag{III.28}$$

$$\lambda_r \in [0,1] \quad \forall\lambda \in \Omega \tag{III.29}$$

$$\tag{III.30}$$

The objective function (III.26) minimizes the total cost of the used routes, where the cost of routes is given by $c_r = \sum_{e\in E_R} c_e a_r^e + \sum_{e\in E} c_e b_r^e$. Constraints (III.27) limit the number of routes used to the number of available vehicles and constraints (III.28) assure each required edge is serviced by exactly one route.

Furthermore, in order to improve the bounds of this linear relaxation, the odd-edge cutset inequalities and the capacity cuts can be also rewritten using the λ_r variable and included in this formulation. The resulting constraint from this modification applied on (III.16) is as follows.

$$\sum_{r\in\Omega}\sum_{e\in\delta(S)} b_r^e\lambda_r \geq \alpha(S) \quad \forall S \subseteq V\backslash\{0\} \tag{III.31}$$

III.3 Formulations for the GVRP

(a) The Undirected Formulation with an Exponential Number of Constraints

The *Undirected Formulation with an Exponential Number of Constraints* was proposed by Bektaş et al. [8] and uses one type of variable, called z_e, which represents the number of times any vehicle traverses edge e. In general, these variables are binary variables, i.e., they can hold only values 0 or 1. The only exception occurs when the edge e is adjacent to the deposit, in this case they can hold a third value. When a vehicle leaves the depot, services just one cluster and then returns to the depot using the same edge, the variable receives the value 2.

$$\text{Minimize} \sum_{e \in E} c_e z_e \tag{III.32}$$

subject to

$$\sum_{e \in \delta(C_0)} z_e = 2|\mathcal{K}| \tag{III.33}$$

$$\sum_{e \in \delta(C_m)} z_e = 2 \qquad \forall m \in M \backslash \{0\} \tag{III.34}$$

$$\sum_{e \in \delta(S)} z_e + 2 \sum_{(i,j) \in L : i \notin S} z(\{i\} : C_j) \leq 2 \qquad \forall m \in M \backslash \{0\}, \forall S \subseteq C_m, \forall L \in \bar{L}_m \tag{III.35}$$

$$\sum_{e \in \delta(S)} z_e \geq 2k(S) \qquad S \subseteq C \backslash \{C_0\} \tag{III.36}$$

$$z_e \in \{0, 1, 2\} \quad \forall e \in \delta(0) \tag{III.37}$$

$$z_e \in \{0, 1\} \qquad \forall e \in E \backslash \delta(0) \tag{III.38}$$

The objective function (III.32) minimizes the total traversal cost. Constraints (III.33) ensure that the degree of the depot cluster is $2|\mathcal{K}|$, forcing the number of vehicles to be equal to $|\mathcal{K}|$. Constraints (III.34) ensure that the degree of any cluster $m \neq 0$ is 2, forcing all these clusters to be serviced by only one vehicle.

Constraints (III.35) are the *same-vertex inequalities*. Given $\bar{L}_m = \{L : L \subseteq \bigcup_{i \in C_m} L_i, |L \cap L_i| = 1, \forall i \in C_m\}$ where $L_i = \{i\} \times (M \backslash \{0, \mu(i)\})$ for all $i \in V \backslash \{0\}$, they ensure that if a vehicle enters in a cluster using a vertex v, it must leave the cluster using the same vertex. We refer the reader to the work of Bektaş et al. [8], where an extensive explanation of the same-vertex inequalities can be found, together with a graphic example, the proof of its correctness and a separation routine.

Constraints (III.36) are the *capacity constraints*, as they ensure that in the solution there will be no vehicle violating the capacity Q and there will be no subtours, i.e., every route must be connected with the depot. Analogously to the CARP, given a cluster set S, $k(S)$ represents a lower bound on the number of vehicles needed to service the cluster set S. As mentioned before, this can be calculated by solving the Bin Packing problem, which is $\mathcal{NP}$-hard [32]. However, in the GVRP case the approximation used is $k(S) = \lceil \sum_{m \in S} d_m / Q \rceil$.

Notice that, as well as the CARP capacity cuts, there is an exponential number of GVRP capacity cuts and therefore *a priori* generation may not be possible. So, a cutting plane algorithm during the resolution of the problem is required.

(b) Set Partitioning Approach

The undirected formulation with an exponential number of constraints can be naturally rewritten using route variables. A GVRP route is a walk that starts at the depot, traverses a set of clusters without violating the vehicle capacity Q, and returns to the depot. Given the set of all GVRP routes, called Ω. We define a binary variable $\lambda_r, \forall r \in \Omega$, which is 1 when the route r is used by a vehicle in the solution, and 0 otherwise. Let b_r^e be a constant value indicating how many times route r traverses edge e. Therefore, the new formulation is as follows.

$$\text{Minimize} \sum_{e \in E} c_e z_e \tag{III.39}$$

subject to

$$\sum_{r \in \Omega} b_r^e \lambda_r = z_e \qquad \forall e \in E \tag{III.40}$$

$$\sum_{e \in \delta(C_0)} z_e = 2|\mathcal{K}| \tag{III.41}$$

$$\sum_{e \in \delta(C_m)} z_e = 2 \qquad \forall m \in M \backslash \{0\} \tag{III.42}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in \Omega \tag{III.43}$$

$$z_e \in \{0, 1, 2\} \quad \forall e \in \delta(0) \tag{III.44}$$

$$z_e \in \{0, 1\} \qquad \forall e \in E \backslash \delta(0) \tag{III.45}$$

Constraints (III.40) define the relation between variables z_e and λ_r. Notice that constraints (III.35) and (III.36) are no longer required because of the definition of the GVRP routes together with constraints (III.40).

From now on, we can replace the z_e variables using its relation with the λ_r variables and relax the integrality constraints (III.43) to obtain the Dantzig-Wolfe relaxation as follows.

$$\text{Minimize} \sum_{r \in \Omega} \left(\sum_{e \in E} c_e b_r^e \right) \lambda_r \tag{III.46}$$

subject to

$$\sum_{r \in \Omega} \left(\sum_{e \in \delta(C_0)} b_r^e \right) \lambda_r = 2|\mathcal{K}| \tag{III.47}$$

$$\sum_{r \in \Omega} \left(\sum_{e \in \delta(C_m)} b_r^e \right) \lambda_r = 2 \qquad \forall m \in M \backslash \{0\} \tag{III.48}$$

$$\lambda_r \in [0,1] \quad \forall r \in \Omega \tag{III.49}$$

(III.50)

There is a simpler way to write the above formulation. Knowing that every route must start and end at the depot, we can state that $\sum_{e\in\delta(C_0)} b_r^e = 2$, $\forall r \in \Omega$. Given $a_r^m \in \{0,1\}$, the number of times route r traverses cluster m, by the definition of b_r^e, we can also state that $\sum_{e\in\delta(C_m)} b_r^e = 2a_r^m$, $\forall m \in M\backslash\{0\}$ and $\forall r \in \Omega$. Furthermore, we can define the total cost of a route r as $c_r = \sum_{e\in E} c_e b_r^e$, $\forall r \in \Omega$. Therefore, the final formulation follows:

$$\text{Minimize} \sum_{r\in\Omega} c_r \lambda_r \tag{III.51}$$

subject to

$$\sum_{r\in\Omega} \lambda_r = |\mathcal{K}| \tag{III.52}$$

$$\sum_{r\in\Omega} a_r^m \lambda_r = 1 \qquad \forall m \in M\backslash\{0\} \tag{III.53}$$

$$\lambda_r \in [0,1] \quad \forall r \in \Omega \tag{III.54}$$

Notice that the simplicity of this formulation makes it easy to understand that constraints (III.52) ensure that $|\mathcal{K}|$ vehicles must be used and constraints (III.53) ensure that every cluster must be visited only once.

In order to improve the solution obtained by this relaxation, we can define some cuts to be used within the formulation. Using the equation (III.40), any cut which is valid for the undirected formulation can be translated into a valid cut for the set partitioning formulation. Thus, we can rewrite the capacity cuts (III.36) as follows.

$$\sum_{r\in\Omega} \sum_{e\in\delta(S)} b_r^e \lambda_r \geq 2k(S) \quad S \subseteq C\backslash\{C_0\} \tag{III.55}$$

IV
Fast Lower Bounds with a Dual Ascent Heuristic and Exact Cut Separations

This chapter investigates methods to improve on lower bounds for the CARP instances, including instances on graphs with over two hundred vertices and three hundred edges, dimensions that, today, can be considered of large scale. We propose to explore the speed of a dual ascent heuristic to generate capacity cuts. The quality of the cuts is next improved with a new exact separation which is used in the linear program resolution that follows the dual heuristic. Computational experiments were carried out on the regular size instances sets and also on the set of large scale instances generated by Brandão and Eglese [17]. The experiments on the former allow for evaluating the quality of the proposed solution approaches, while those on the latter present improved lower bounds for all instances of the corresponding set.

IV.1 Existing Exact Separations

(a) Exact Odd-Degree Cutset Inequalities Separation

The exact separation of the odd-degree cutset inequalities (III.12) can be done in polynomial time using the *Odd Minimum Cutset Algorithm* of Padberg and Rao [63]. We believe that the application of the algorithm is not immediate and therefore we decided to provide a brief description of the separation routine, which is as follows.

The odd minimum cutset algorithm creates a *Gomory-Hu Tree* [41] using just the vertices with odd $|\delta_R(\{v\})|$, called *terminals*. The algorithm builds this tree recursively and, at each step, it finds the maximum flow between any pair of terminals and split the graph in two by the minimum cut. Then it creates one edge on the resulting Gomory-Hu Tree between the two new graphs and put as its value the same found by the maximum flow algorithm. The same procedure is then called for both new graphs.

The Gomory-Hu Tree represents a *maximum flow tree*, i.e., the maximum

flow for any pair of vertices is represented on this tree. In order to obtain the maximum flow between a pair of vertices, one only needs to find the least cost edge on the unique path between these two vertices. This edge also represents the minimum cut between them. Hence, to determine a violated odd-degree cutset inequality, one needs to find any edge with a value less than one. This can be done during the execution of the algorithm, but we prefer to run it until the end to find as many violated cuts as possible.

This whole operation can be done running at most $|V| - 1$ times any maximum flow algorithm. In this work we use the *Edmonds-Karp Algorithm* [25], which takes $\mathcal{O}(|V|.|E|^2)$, resulting in a total complexity of $\mathcal{O}(|V|^2.|E|^2)$.

(b) Ahr's Exact Capacity Cut Separation

The only exact separation routine for the capacity cuts available in the CARP literature was proposed by Ahr [1] in 2004. This algorithm runs a mixed-integer formulation several times, one for each possible number of vehicles. This approach was inspired on the exact separation of the capacity cuts for the CVRP proposed by Fukasawa et al. [31]. In Ahr's work, this separation was used to solve the two-index formulation. As we only wish to separate the cuts, we changed the objective function of the mixed-integer formulation to use it with the one-index formulation.

The formulation is composed by three types of variables. The first one is the binary variable h_e, $\forall e \in E$, which is 1 when exactly one endpoint of e is inside the cut (what we call *cut edge*) and 0 otherwise. The second variable is the binary variable f_e, $\forall e \in E$, which is 1 when both endpoints of e are inside the cut (called *inner edge*) and 0 otherwise. The last variable is the binary variable s_i, $\forall i \in V$, which is 1 if vertex i is inside the cut and 0 otherwise. These variables are sufficient to describe a capacity cut. Thus, the following formulation is created for $k = 0...\lceil \sum_{e \in E_R} d_e/Q \rceil - 1$:

$$\text{Minimize} \sum_{e \in E} \tilde{z}_e h_e + \sum_{e \in E_R} h_e \tag{IV.1}$$

subject to

$$h_e - s_i + s_j \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.2}$$

$$h_e + s_i - s_j \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.3}$$

$$-h_e + s_i + s_j \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.4}$$

$$s_i - f_e \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.5}$$

$$s_j - f_e \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.6}$$

$$s_i + s_j - f_e \leq 1 \qquad \forall e = \{i, j\} \in E \tag{IV.7}$$

$$\sum_{e \in \delta(\{i\})} (h_e + f_e) - s_i \geq 0 \qquad \forall i \in V \tag{IV.8}$$

$$h_e + f_e \leq 1 \qquad \forall e \in E \tag{IV.9}$$

$$\sum_{e \in E_R} d_e \, (h_e + f_e) \geq kQ + 1 \tag{IV.10}$$

$$s_0 = 0 \tag{IV.11}$$

$$h_e, f_e \in \{0, 1\} \qquad \forall e \in E \tag{IV.12}$$

$$s_i \in [0, 1] \qquad \forall i \in V \backslash \{0\} \tag{IV.13}$$

The objective function (IV.1) uses a solution of the one-index formulation $\tilde{z}_e$ and minimizes the total value of the cut edges plus the number of cut edges that are required. Constraints (IV.2), (IV.3) and (IV.4) bind the variables s_i and h_e. Analogously, constraints (IV.5), (IV.6) and (IV.7) bind the variables s_i and f_e. The constraints (IV.8) assure that if a vertex i is inside the cut, at least one edge adjacent to i is a cut edge or an inner edge. Constraints (IV.9) assure that an edge e cannot be a cut edge and an inner edge at the same time. Constraints (IV.10) ensure that the total demand of the cut found is at least $kQ+1$, which forces the solution to use more than k vehicles. Constraint (IV.11) forbids the inclusion of the depot in a cut. Notice that due to the association of s_i with h_e and f_e, the variables s_i need not to be integral.

Given the value of the objective function Z^* associated to a solution in a given iteration k, the cut which can be generated using the s_i variables is a violated capacity cut if $Z^* < 2(k+1)$. Therefore, the problem needs to be solved to optimality only when we aim at finding the most violated capacity cut.

This separation routine has the disadvantage of running several MIPs, one for every possible number of vehicles. Depending on the instance, this number may be up to 42. Nevertheless, in his work, Ahr could not manage to run this separation for all CARP instances due to memory limitations.

IV.2 A New Exact Capacity Cut Separation

The exact separation suggested by Ahr requires solving several MIPs because it is not possible to build a mixed-integer formulation that directly represents the *ceiling function* ($\lceil \cdot \rceil$) of the capacity cut. In order to deal with this issue, we developed a new formulation which is capable of separating a capacity cut in an exact fashion considering any possible number of vehicles. Our approach was inspired by the exact separation of the *Chvátal-Gomory cuts* proposed by Fischetti and Lodi in 2007 [28].

Our mixed-integer formulation uses the same three variables presented in Ahr's formulation, h_e, f_e and s_i. In addition, we also consider an integer variable κ indicating the value of $k(S)$ in the formulation and a continuous slack variable γ representing the fractional difference of applying the ceiling function to obtain κ. This difference must be within the range $[0, 1)$.

Furthermore, we use constraints (IV.2-IV.9) and (IV.11) from Ahr's formulation. These constraints are required to depict a capacity cut. We write our complete formulation as follows:

$$\text{Maximize } 2\kappa - \sum_{e \in E_R} h_e - \sum_{e \in E} \tilde{z}_e h_e \tag{IV.14}$$

subject to

$$h_e - s_i + s_j \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.15}$$

$$h_e + s_i - s_j \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.16}$$

$$-h_e + s_i + s_j \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.17}$$

$$s_i - f_e \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.18}$$

$$s_j - f_e \geq 0 \qquad \forall e = \{i, j\} \in E \tag{IV.19}$$

$$s_i + s_j - f_e \leq 1 \qquad \forall e = \{i, j\} \in E \tag{IV.20}$$

$$\sum_{e \in \delta(\{i\})} (h_e + f_e) - s_i \geq 0 \qquad \forall i \in V \tag{IV.21}$$

$$h_e + f_e \leq 1 \qquad \forall e \in E \tag{IV.22}$$

$$\sum_{e \in E} \frac{d_e \, (h_e + f_e)}{Q} + \gamma = \kappa \tag{IV.23}$$

$$s_0 = 0 \tag{IV.24}$$

$$h_e, f_e \in \{0, 1\} \qquad \forall e \in E \tag{IV.25}$$

$$s_i \in [0, 1] \qquad \forall i \in V \backslash \{0\} \tag{IV.26}$$

$$\kappa \in \mathbb{Z}_0^+ \tag{IV.27}$$

$$\gamma \in [0, 1) \tag{IV.28}$$

The objective function (IV.14) maximizes the violation of the capacity cut, while constraint (IV.23) limits the difference between κ and the fractional value using the slack variable γ. As mentioned, constraints (IV.15-IV.22) and (IV.24) are from Ahr's formulation. Of the way the objective function depicts the violation of the cut, when its value is greater than zero, a violated cut can be built in the same manner as in the preceding formulation. We will further show in the computational experiments that this formulation can perform better in practice than Ahr's formulation.

IV.3 A Dual Ascent Heuristic

Even with the improvement on the exact separation of the capacity cuts, the separation routine still takes a long time when applied to large instances. However, if we use a heuristic approach to generate valid cuts to be used as a hot-start for the separation algorithm, the number of iterations of the separation routine could reduce drastically. In view of this, we propose a dual ascent heuristic.

A dual ascent heuristic is usually devised to obtain good lower bounds for a problem. It was introduced in 1967 in a technical report of Bilde and Krarup [14]. However, as this work is written in Danish, it attracted limited attention outside Scandinavia. The same authors republished it in English in 1977 [15], but the approach began to gain attention only after the publication of the work done by Erlenkotter in 1978 [27]. A recent example of the application of this approach can be found in the work of Poggi et al. [70] on the Steiner Tree Problem.

We will show next that when this heuristic is applied over the one-index formulation for the CARP, it can generate several cuts on each iteration. If good cuts are found during these iterations, they can be very helpful to hot-start the exact separation.

(a) Main Algorithm

The main algorithm of the dual ascent heuristic works on the dual of the linear relaxation of the one-index formulation:

$$\text{Maximize} \sum_{S \subseteq V \setminus \{0\}} \alpha(S)\pi_S \tag{IV.29}$$

subject to

$$\sum_{S \subseteq V \setminus \{0\} : e \in \delta(S)} \pi_S \leq c_e \quad \forall e \in E \tag{IV.30}$$

$$\pi_S \in \mathbb{R}_0^+ \quad \forall S \subseteq V \setminus \{0\} \tag{IV.31}$$

In this formulation, the variables π_S are associated with constraints (III.16) and constraints (IV.30) are associated with z_e variables. These last constraints impose a limit on the dual variables. The sum of the dual variables associated with the cuts which contain an edge $e \in \delta(S)$ must not exceed the cost of this edge e. This is the base of our dual ascent heuristic, which is shown in Algorithm IV.1.

As already mentioned, the objective of our dual ascent heuristic is to find

Algorithm IV.1 The Dual Ascent Heuristic

```
1: procedure DUALASCENT(G)
2:     input: graph G
3:     output: a lower bound LB

4:     LB ← 0
5:     for all e ∈ E_R do
6:         LB ← LB + c_e
7:     while |V| > 1 do
8:         S* ← selectBestCut(G)
9:         π_S* ← min_{e∈δ(S*)}(c_e)
10:        for e ∈ δ(S*) do
11:            c_e ← c_e − π_S*
12:            if c_e = 0 then
13:                contract edge e
14:        LB ← LB + α(S*).π_S*
15:    return LB
```

a lower bound for the CARP. Therefore, it starts with the trivial lower bound, which is $LB = \sum_{e \in E_R} c_e$ for the one-index formulation, as stated in Chapter III. At each iteration, several cuts are generated using a strategy that will be further discussed. Among these cuts, only one is chosen using an arbitrary criterion. A good cut is one with a large $\alpha(S)$ or, in case of a tie, one with a large contribution to the objective function. The contribution of a cut can be calculated as presented in (IV.32).

$$\sigma(S) = \alpha(S).min\{c_e : e \in \delta(S)\} \tag{IV.32}$$

Given the selected cut S^*, the heuristic updates its lower bound ($LB = LB + \sigma(S^*)$) and it also changes the dual formulation to reflect the use of this cut. Knowing the value of the variable $\pi_{S^*} = min\{c_e : e \in \delta(S^*)\}$ associated with the cut, each constraint of the dual formulation where $e \in \delta(S^*)$ must have its right-hand side modified to $c'_e = c_e - \pi_{S^*}$. As a result, the variable π_{S^*} is removed from the formulation.

This last operation has a direct effect on the graph G. The update of the right-hand side of the constraints (IV.30) is the same of reducing the costs of the edges $e \in \delta(S)$. When an edge $e = (i, j)$ is saturated, i.e., the edge has its cost reduced to 0, the heuristic contracts the vertices i and j as shown in Fig. IV.1. This contraction guarantees that no saturated edges appear as cut edges on future iterations of the heuristic.

The next iteration of the heuristic is then applied over the new graph. When the graph has only one vertex (the depot), the heuristic stops. Notice that at each iteration, at least one edge is saturated. Due to this fact, the heuristic performs at most $|V| - 1$ iterations.

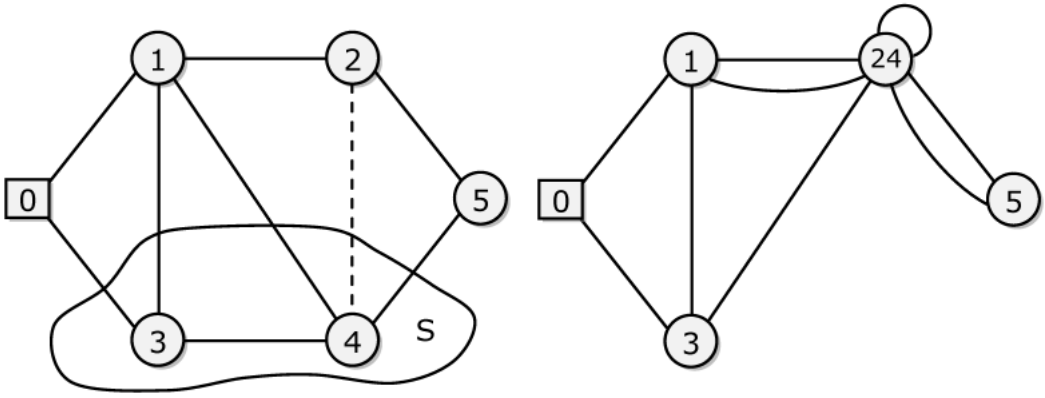


Figure IV.1: Example of vertex contraction: vertices 2 and 4 are contracted, becoming one vertex.

(b) Cut Generation

As pointed before, the dual ascent heuristic can only give good lower bounds if good cuts are chosen. Therefore, the cut generation strategies are the most important part of the heuristic. Any strategy can be used within our heuristic. After some preliminary experiments, we decided to turn attention to four different strategies. When a previously generated cut or a cut S with $\alpha(S) = 0$ is obtained by any of the strategies, this new cut is discarded.

Simple Cut

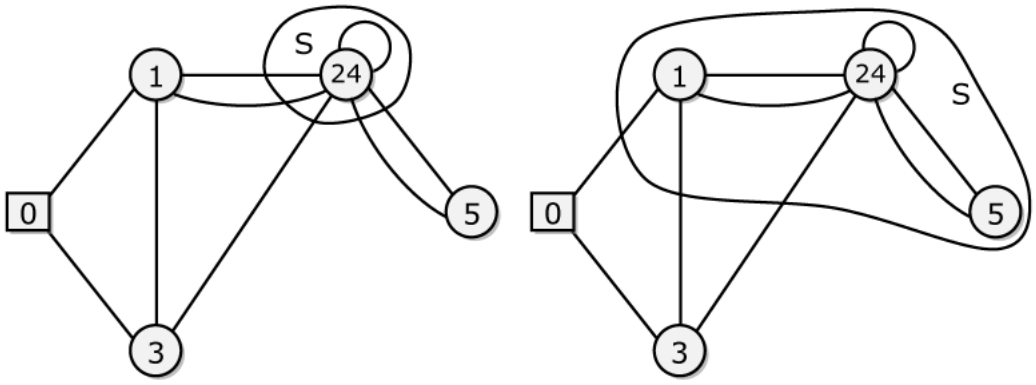
In the *simple cut* strategy, we create a set of cuts $S = \{v\}$, $\forall v \in V \backslash \{0\}$, containing only one vertex. Such vertex cannot be the depot. It is noteworthy to mention that as the graph is modified during the iterations of the heuristic, a vertex at some iteration might not be a single vertex on the original graph. An example of this strategy is shown in Fig. IV.2(a). This strategy takes time $\mathcal{O}(|V|)$ to build the cuts and generates at most $|V| - 1$ cuts.

Complete Cut

In the *complete cut* strategy, we create a set of cuts which, for each vertex $v \in V$ (including the depot), contains all the vertices of the graph except v and the depot. Analogously to the previous strategy, the vertex left out of the cut might not be a single vertex at a given iteration of the heuristic. An example of this strategy is shown in Fig. IV.2(b). This strategy takes time $\mathcal{O}(|V|^2)$ to build the cuts and generates at most $|V|$ cuts.

Connected Cut

The *connected cut* strategy inserts vertices in the cut using a *breadth-first search* approach. Firstly, it chooses a random size for the cut between 2 and



IV.2(a): Simple Cuts strategy. IV.2(b): Complete Cut strategy.

Figure IV.2: Examples of Simple Cuts and Complete Cuts Strategies.

$|V\backslash\{0\}|-2$, as all the cuts of size 1 and $|V\backslash\{0\}|-1$ are generated in the first two strategies. Secondly, it chooses a random vertex (excluding the depot) to start the search. Each time the breadth-first search visits a new vertex, this vertex is added to the cut. The search stops when the size of the cut is equal to the desired size. This operation is repeated $|E|$ times. The whole operation takes time $\mathcal{O}(|E|(|V|+|E|))$ and generates at most $|E|$ cuts.

MST Cut

The *MST cut* strategy starts by generating the *Minimum Spanning Tree* (MST) of the graph. Each edge of the MST defines two vertex set on the graph. Those which do not contain the depot are then generated as cuts (see Fig. IV.3). Using the *Kruskal's Algorithm* [47] for MST, along with any search algorithm, this strategy takes time $\mathcal{O}(|E|log|V|)$ and generates at most $|V|-1$ cuts.

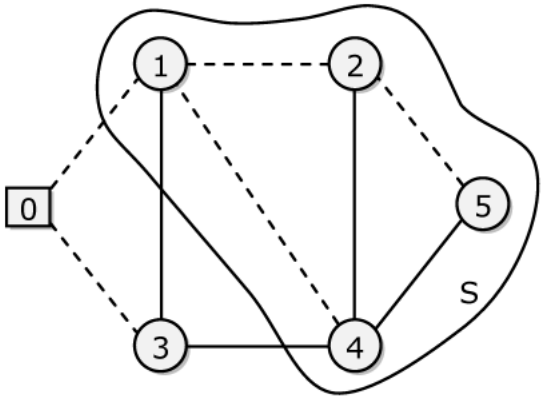


Figure IV.3: Example of a MST cut defined by edge $(0,1)$. The minimum spanning tree is shown by dashed edges.

IV.4 Experimental Results

For the sake of comparison, we applied our algorithms to all well-known CARP instance datasets, namely: *kshs* [46], *gdb* [22, 37], *bccm* (`[1-10][A-D]`) [12], *eglese* (`e-[1-4]-[A-C]` and `s-[1-4]-[A-C]`) [54, 55], *beullens* (`[C-F][01-25]`) [13] and *egl-large* (`g-[1-4]-[A-C]`) [17]. The first four are known as the classical CARP instance datasets and have been widely used in the literature over the past 20 years. The last two were created more recently and only some recent works have attempted to solve them.

The datasets *kshs*, *gdb* and *bccm* were artificially generated and have no non-required edges. On the other hand, the *eglese* and *egl-large* datasets were constructed using as underlying graph regions of the road network of the county of Lancashire (UK). Analogously, the *beullens* dataset was constructed based on the intercity road network in Flanders (Belgium). The instances belonging to these last three datasets have costs and demands proportional to the length of the edges and most of them have non-required edges.

As mentioned before, one of the objectives of this chapter is focused on solving the large scale CARP instances. The instances we consider as large scale are those of the *egl-large* dataset. These instances have 255 vertices and up to 375 required edges. As far as we know, only metaheuristics were used to solve these instances, which explains our lack of knowledge of lower bounds for them.

(a) Exact Separation

The exact separation algorithms were implemented in C++, using Windows Vista 32-bits, Visual C++ 2010 Express Edition and IBM Cplex 12.4. Tests were conducted on an Intel Core 2 Duo 2.8 GHz, with 4 GB of RAM and using only one core (IBM Cplex 12.4 uses both cores when running the branch-and-cut for the mixed-integer program). We compare the execution of the presented exact separation algorithms, the one from Section IV.1(b) proposed by Ahr [1] and our new algorithm from Section IV.2, both executed together with the exact separation of the odd-degree cutset inequalities from Section IV.1(a).

For both algorithms, we first apply the separation on the linear relaxation of the one-index formulation. Once the linear optimum is found, the z_e variables are then shifted to integer and the separation continues until the integer optimum is obtained. For our new exact separation, in order to model the γ limits on equation (IV.28), we use a constant $\delta = 0.001$ and set $\gamma \in [0, 1 - \delta]$. Results are shown in Table IV.1.

Columns `Ins`, $|V|$, $|E_R|$, $|E|$ and $|\mathcal{K}|$ show the name, number of vertices, required edges, total edges and number of vehicles of each instance, respectively. The column `LB` displays the best known lower bounds, with optimal values underlined when it is the case. For each following column `X`, $\texttt{X}_1$ shows the results obtained at the end of the first part of the experiment, when just the linear relaxation of the one-index formulation is used. Furthermore, $\texttt{X}_2$ shows the results of the complete experiment, i.e., after the solution of the integer one-index formulation. Column `Cost` shows the lower bounds obtained after separating (III.12) and (III.13), which is the same for all algorithms. For each algorithm, columns `Cap`, `Odd` and `Time` show the total number of capacity cuts, the total number of odd-degree cutset inequalities and the total time in seconds. Values equal to the best known are highlighted in boldface.

From Table IV.1, it can be observed that our algorithm performs better in nearly every instance tested. Comparing the times obtained after the first part of the experiment, our algorithm was faster for all datasets, on average: 60.82% for *kshs*, 83.32% for *gdb*, 69.96% for *bccm*, 52.01% for *eglese*, 54.01% for *C*, 44.00% for *D*, 65.14% for *E* and 44.92% for *F*, in a total of 59.41% improvement overall.

Notice that the algorithms were not tested on the large scale instance dataset because the complete separation of the capacity cuts does not run in reasonable time without some hot-start technique, as shown next.

(b) Dual Ascent Heuristic

The dual ascent heuristic was implemented using the same configuration of the exact separation algorithms. In order to show the benefit of each strategy, we tested each one separately. In addition, a complete test was also performed as follows. At each iteration of the dual ascent heuristic, we generate a cut pool using the strategies in the following order: complete cuts, single cuts, connected cuts and MST cuts. Next, the best cut is chosen from this pool, the graph is updated as described in Section IV.3(a) and all cuts found in this iteration are added to another pool of cuts, the resulting pool. At the end of the dual ascent, we take all the cuts from the resulting pool, add them into an one-index formulation and use CPLEX 12.4 to solve it to optimality. The results of these tests are shown in Table IV.2.

As in the previous experiments, columns `Ins` and `LB` show the name and the best known lower bound for each instance, respectively. The next 9 columns show the results regarding the full execution of the dual ascent. Columns `Cost` and `Time` show the solution cost and time before calling the one-index formulation. Columns `Cuts`, `SGL`, `CMP`, `CON` and `MST` show the total

number of distinct cuts found overall and the total number of cuts found by each strategy, respectively. Columns `Int` and `Time` show the solution cost and time after calling the one-index formulation. Next, we show the results for each of the four strategies. Columns `Cost`, `Time`, `Cuts` show the cost, the time and the total number of cuts found.

With the view of comparing the different strategies used, we underline the best cost found among them. Moreover, the total of best costs for each strategy is shown in the last row of each table, called `best`. In addition, when a value from any `Cost` column is equal to the best known, it is highlighted in boldface. Notice that the dual ascent heuristic is capable of finding good bounds quite fast, thus generating a large number of cuts.

In Table IV.3, we show the reduction on the number of cuts and time obtained using the cuts of the dual ascent heuristic in our exact separation. In addition to the lower running time, one can notice a decrease in the separation of the cuts, more prominent in the almost total absence of separation of odd-degree cutset inequalities.

As pointed before, with the use of the dual ascent heuristic, we were capable of running our exact separation for the *egl-large* instance dataset, proposed by Brandão and Eglese in 2008 [17]. The results are shown in Table IV.4. As shown in previous tables, columns `Ins`, $|V|$, $|E_R|$, $|E|$ and $|\mathcal{K}|$ show the name, number of vertices, required edges, total edges and number of vehicles of each instance, respectively. The next three columns, `Cost`, `Cuts` and `Time`, show the cost, the number of cuts and the total time in seconds of the dual ascent heuristic, *without* calling the one-index formulation. The last four columns, `Cost`, `Cap`, `Odd` and `Time`, show the cost, the number of capacity cuts, the number of odd-edge cutset inequalities and the total time of our exact separation using the dual ascent heuristic as hot-start. Furthermore, in contrast to what was done for the other datasets, we only performed the separation on the linear relaxation of the one-index formulation, interrupting the execution when the linear optimum was achieved. The continuous values were rounded up to the next integer.

Table IV.1: Exact separation results

Ins	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|\mathcal{K}\|$	LB	$\texttt{Cost}_1$	$\texttt{Cost}_2$	Ahr's Exact Sep						Our Exact Sep					
								$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$	$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$
kshs1	8	15	15	4	14661	**14661**	**14661**	2	5	0.091	2	5	0.136	2	5	0.046	2	5	0.072
kshs2	10	15	15	4	9863	**9863**	**9863**	4	6	0.124	4	6	0.162	4	6	0.075	4	6	0.099
kshs3	6	15	15	4	9320	**9320**	**9320**	1	6	0.196	1	6	0.292	1	6	0.038	1	6	0.070
kshs4	8	15	15	4	11498	11098	11098	3	6	0.163	3	6	0.230	3	6	0.048	3	6	0.066
kshs5	8	15	15	3	10957	**10957**	**10957**	1	8	0.182	1	8	0.261	1	8	0.043	1	8	0.060
kshs6	9	15	15	3	10197	**10197**	**10197**	0	17	0.031	0	17	0.061	0	17	0.016	0	17	0.031
gdb1	12	22	22	5	316	**316**	**316**	2	17	0.228	2	17	0.353	2	17	0.058	2	17	0.085
gdb2	12	26	26	6	339	**339**	**339**	1	7	0.233	1	7	0.372	1	7	0.031	1	7	0.052
gdb3	12	22	22	5	275	**275**	**275**	2	14	0.311	2	14	0.420	2	14	0.077	2	14	0.103
gdb4	11	19	19	4	287	**287**	**287**	4	8	0.244	4	8	0.346	4	8	0.065	4	8	0.083
gdb5	13	26	26	6	377	**377**	**377**	3	15	0.289	3	15	0.428	3	15	0.076	3	15	0.100
gdb6	12	22	22	5	298	**298**	**298**	2	13	0.308	2	13	0.415	2	13	0.062	2	13	0.093
gdb7	12	22	22	5	325	**325**	**325**	2	23	0.218	2	23	0.317	2	23	0.049	2	23	0.078
gdb8	27	46	46	10	348	344	344	14	33	1.400	14	33	1.804	21	33	0.760	21	33	0.825
gdb9	27	51	51	10	303	**303**	**303**	14	28	1.690	14	28	2.192	11	28	0.316	11	28	0.438
gdb10	12	25	25	4	275	**275**	**275**	0	7	0.451	0	7	0.898	0	7	0.049	0	7	0.096
gdb11	22	45	45	5	395	**395**	**395**	1	21	0.453	1	21	0.692	1	21	0.066	1	21	0.116
gdb12	13	23	23	7	458	450	450	5	11	0.328	5	11	0.472	3	11	0.054	3	11	0.082
gdb13	10	28	28	6	536	**536**	**536**	1	11	1.717	1	11	2.349	1	11	0.089	1	11	0.132
gdb14	7	21	21	5	100	**100**	**100**	1	0	0.687	1	0	1.103	1	0	0.020	1	0	0.032
gdb15	7	21	21	4	58	**58**	**58**	1	0	0.444	1	0	0.689	1	0	0.024	1	0	0.039
gdb16	8	28	28	5	127	**127**	**127**	1	7	1.459	1	7	2.077	1	7	0.045	1	7	0.081
gdb17	8	28	28	5	91	**91**	**91**	0	8	0.790	0	8	1.581	0	8	0.016	0	8	0.032
gdb18	9	36	36	5	164	**164**	**164**	1	0	1.542	1	0	2.462	1	0	0.056	1	0	0.101
gdb19	8	11	11	3	55	**55**	**55**	0	10	0.023	0	10	0.046	0	10	0.016	0	10	0.032
gdb20	11	22	22	4	121	**121**	**121**	0	14	0.198	0	14	0.391	0	14	0.019	0	14	0.057

Continued on next page

Table IV.1: *Continued from previous page*

Ins	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|\mathcal{K}\|$	LB	$\texttt{Cost}_1$	$\texttt{Cost}_2$	Ahr's Exact Sep						Our Exact Sep					
								$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$	$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$
gdb21	11	33	33	6	156	**156**	**156**	1	7	1.972	1	7	2.529	1	7	0.079	1	7	0.114
gdb22	11	44	44	8	200	**200**	**200**	1	33	4.657	1	33	6.538	1	33	0.115	1	33	0.176
gdb23	11	55	55	10	233	**233**	**233**	2	0	4.569	2	0	6.824	2	0	0.086	2	0	0.138
1A	24	39	39	2	173	**173**	**173**	0	48	0.115	0	48	0.216	0	48	0.057	0	48	0.106
1B	24	39	39	3	173	**173**	**173**	0	48	0.117	0	48	0.226	0	48	0.051	0	48	0.093
1C	24	39	39	8	245	235	235	23	48	1.458	23	48	1.772	13	48	0.277	13	48	0.325
2A	24	34	34	2	227	**227**	**227**	3	36	0.280	3	36	0.399	3	36	0.132	3	36	0.169
2B	24	34	34	3	259	257	257	6	44	0.436	6	44	0.552	6	44	0.209	6	44	0.250
2C	24	34	34	8	457	455	455	24	31	1.446	24	31	1.701	20	31	0.477	20	31	0.519
3A	24	35	35	2	81	**81**	**81**	2	26	0.198	2	26	0.264	2	26	0.106	2	26	0.146
3B	24	35	35	3	87	**87**	**87**	9	26	0.580	9	26	0.724	8	25	0.221	8	25	0.262
3C	24	35	35	7	138	135	135	22	23	1.481	22	23	1.731	18	23	0.365	18	23	0.405
4A	41	69	69	3	400	**400**	**400**	5	32	0.918	5	32	1.327	4	34	0.307	4	34	0.403
4B	41	69	69	4	412	**412**	**412**	5	32	1.097	5	32	1.517	4	34	0.272	4	34	0.362
4C	41	69	69	5	428	**428**	**428**	9	34	2.173	9	34	2.666	9	34	0.463	9	34	0.554
4D	41	69	69	9	530	519.5	521	39	63	7.764	39	63	8.489	31	62	2.022	31	63	2.140
5A	34	65	65	3	423	**423**	**423**	2	57	0.905	2	57	1.182	2	57	0.225	2	57	0.305
5B	34	65	65	4	446	443	443	4	63	1.114	4	63	1.466	4	63	0.303	4	63	0.384
5C	34	65	65	5	474	467	467	6	77	1.825	6	77	2.463	7	77	0.348	7	77	0.430
5D	34	65	65	9	577	571	571	17	56	3.208	17	56	3.971	15	56	0.693	15	56	0.775
6A	31	50	50	3	223	**223**	**223**	2	36	0.281	2	36	0.423	2	36	0.115	2	36	0.166
6B	31	50	50	4	233	229	229	3	38	0.659	3	38	0.846	3	38	0.217	3	38	0.269
6C	31	50	50	10	317	307	307	30	36	2.690	30	36	3.207	22	36	0.950	22	36	1.044
7A	40	66	66	3	279	**279**	**279**	0	30	0.132	0	30	0.261	0	30	0.077	0	30	0.151
7B	40	66	66	4	283	**283**	**283**	1	30	0.377	1	30	0.551	1	30	0.098	1	30	0.176
7C	40	66	66	9	334	327	327	16	112	2.300	16	112	2.881	11	106	0.666	11	106	0.775
8A	30	63	63	3	386	**386**	**386**	1	31	0.603	1	31	0.813	1	31	0.165	1	31	0.231

Continued on next page

Table IV.1: *Continued from previous page*

Ins	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|\mathcal{K}\|$	LB	$\texttt{Cost}_1$	$\texttt{Cost}_2$	Ahr's Exact Sep						Our Exact Sep					
								$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$	$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$
8B	30	63	63	4	395	**395**	**395**	4	31	0.917	4	31	1.228	4	31	0.228	4	31	0.303
8C	30	63	63	9	521	509	509	25	64	3.844	25	64	4.570	19	65	0.659	19	65	0.728
9A	50	92	92	3	323	**323**	**323**	0	112	0.367	0	112	0.690	0	112	0.154	0	112	0.266
9B	50	92	92	4	326	**326**	**326**	1	122	1.671	1	122	2.158	1	122	0.320	1	122	0.427
9C	50	92	92	5	332	**332**	**332**	2	126	2.026	2	126	2.619	2	126	0.339	2	126	0.459
9D	50	92	92	10	391	378	378	23	112	5.241	23	112	6.175	15	112	1.213	15	112	1.344
10A	50	97	97	3	428	**428**	**428**	1	66	0.830	1	66	1.245	1	66	0.203	1	66	0.335
10B	50	97	97	4	436	**436**	**436**	2	68	2.400	2	68	3.249	2	68	0.346	2	68	0.464
10C	50	97	97	5	446	**446**	**446**	6	69	3.508	6	69	4.453	7	70	0.652	7	70	0.778
10D	50	97	97	10	526	521.5	522	47	73	15.851	48	73	17.376	28	74	2.592	28	74	2.719
C01	69	98	79	9	4105	4070	4075	99	80	23.401	99	80	24.693	95	92	12.037	96	92	12.482
C02	48	66	53	7	3135	**3135**	**3135**	65	69	7.108	65	69	7.717	47	48	3.125	47	48	3.215
C03	46	64	51	6	2575	2525	2525	51	66	5.306	51	66	5.734	66	66	4.506	66	66	4.607
C04	60	84	72	8	3478	3455	3455	67	54	9.970	67	54	10.650	44	54	3.224	44	54	3.407
C05	56	79	65	10	5365	5305	5305	154	46	27.472	154	46	28.515	80	49	6.516	80	49	6.683
C06	38	55	51	6	2535	2495	2495	16	40	1.385	16	40	1.727	12	40	0.450	12	40	0.522
C07	54	70	52	8	4075	4015	4015	119	54	14.855	119	54	15.531	86	54	6.004	86	54	6.178
C08	66	88	63	8	4090	4000	4000	123	27	24.239	123	27	25.164	86	27	8.249	86	27	8.518
C09	76	117	97	12	5233	5215	5215	131	215	42.724	131	215	44.829	56	189	7.966	56	189	8.238
C10	60	82	55	9	4700	4597.5	4620	131	78	19.331	134	80	21.279	141	73	17.857	148	73	19.215
C11	83	118	94	10	4583	4550	4550	200	234	60.606	200	234	62.023	79	248	12.758	79	248	13.079
C12	62	88	72	9	4209	4140	4140	209	66	43.984	209	66	45.038	111	84	15.603	111	84	15.801
C13	40	60	52	7	2955	2895	2895	23	28	2.216	23	28	2.650	26	28	1.363	26	28	1.472
C14	58	79	57	8	4030	3970	3970	73	80	10.182	73	80	11.040	54	80	3.719	54	80	3.911
C15	97	140	107	11	4912	4845	4845	144	110	55.379	144	110	58.131	123	110	41.717	123	110	42.664
C16	32	42	32	3	1475	1470	1470	26	21	1.476	26	21	1.643	31	21	1.076	31	21	1.130
C17	43	56	42	7	3555	3535	3535	71	40	6.818	71	40	7.317	91	40	5.941	91	40	6.044

Continued on next page

Table IV.1: *Continued from previous page*

Ins	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|\mathcal{K}\|$	LB	$\texttt{Cost}_1$	$\texttt{Cost}_2$	Ahr's Exact Sep						Our Exact Sep					
								$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$	$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$
C18	93	133	121	11	5577	5550	5550	148	79	59.036	148	79	61.354	104	81	21.444	104	81	22.346
C19	62	84	61	6	3096	3065	3065	99	78	12.728	99	78	13.376	60	75	5.052	60	75	5.218
C20	45	64	53	5	2120	**2120**	**2120**	24	55	2.702	24	55	3.010	11	55	0.480	11	55	0.562
C21	60	84	76	8	3960	3950	3950	65	38	9.727	65	38	10.418	50	38	3.714	50	38	3.884
C22	56	76	43	4	2245	**2245**	**2245**	35	51	3.518	35	51	3.845	35	51	1.848	35	51	1.921
C23	78	109	92	8	4032	4012.5	**4040**	149	169	41.487	155	169	45.886	99	193	16.280	102	193	17.604
C24	77	115	84	7	3384	3370	3370	118	97	29.205	118	97	30.421	73	97	10.006	73	97	10.241
C25	37	50	38	5	2310	**2310**	**2310**	48	106	3.420	48	106	3.648	33	110	1.620	33	110	1.700
D01	69	98	79	5	3215	**3215**	**3215**	13	57	2.367	13	57	2.972	11	57	0.877	11	57	1.023
D02	48	66	53	4	2520	**2520**	**2520**	22	30	1.414	22	30	1.632	17	30	0.823	17	30	0.896
D03	46	64	51	3	2065	**2065**	**2065**	7	73	0.943	7	73	1.196	6	73	0.380	6	73	0.462
D04	60	84	72	4	2785	**2785**	**2785**	28	71	3.961	28	71	4.458	19	65	1.470	19	65	1.596
D05	56	79	65	5	3935	**3935**	**3935**	30	47	2.980	30	47	3.295	25	42	1.026	25	42	1.111
D06	38	55	51	3	2125	**2125**	**2125**	1	40	0.311	1	40	0.478	1	40	0.094	1	40	0.162
D07	54	70	52	4	3115	3015	3015	9	50	1.159	9	50	1.575	9	50	0.512	9	50	0.611
D08	66	88	63	4	2995	2975	2975	22	27	3.113	22	27	3.629	36	27	3.994	36	27	4.139
D09	76	117	97	6	4120	**4120**	**4120**	21	59	6.020	21	59	6.960	13	59	1.411	13	59	1.569
D10	60	82	55	5	3340	3330	3330	8	53	1.346	8	53	1.777	11	53	0.974	11	53	1.072
D11	83	118	94	5	3745	**3745**	**3745**	18	281	6.068	18	281	6.880	18	277	2.026	18	277	2.187
D12	62	88	72	5	3310	**3310**	**3310**	50	64	7.498	50	64	8.004	39	64	2.111	39	64	2.230
D13	40	60	52	4	2535	**2535**	**2535**	5	54	0.877	5	54	1.107	3	54	0.202	3	54	0.273
D14	58	79	57	4	3272	3270	3270	38	81	4.502	38	81	4.837	42	81	3.998	42	81	4.093
D15	97	140	107	6	3990	**3990**	**3990**	11	110	3.647	11	110	4.777	18	110	1.611	18	110	1.833
D16	32	42	32	2	1060	**1060**	**1060**	3	20	0.287	3	20	0.405	5	20	0.234	5	20	0.278
D17	43	56	42	4	2620	**2620**	**2620**	23	48	1.426	23	48	1.603	14	44	0.668	14	44	0.737
D18	93	133	121	6	4165	**4165**	**4165**	40	87	10.832	40	87	11.905	39	86	2.972	39	86	3.173
D19	62	84	61	3	2393	2370	2370	18	66	3.018	18	66	3.394	29	63	2.609	29	63	2.743

Continued on next page

Table IV.1: *Continued from previous page*

Ins	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|\mathcal{K}\|$	LB	$\texttt{Cost}_1$	$\texttt{Cost}_2$	Ahr's Exact Sep						Our Exact Sep					
								$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$	$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$
D20	45	64	53	3	1870	**1870**	**1870**	1	55	0.475	1	55	0.636	1	55	0.191	1	55	0.272
D21	60	84	76	4	2985	2940	2940	18	38	1.951	18	38	2.397	20	38	1.435	20	38	1.650
D22	56	76	43	2	1865	**1865**	**1865**	15	51	0.660	15	51	0.802	19	51	0.820	19	51	0.929
D23	78	109	92	4	3114	3110	3110	10	94	3.221	10	94	3.931	12	94	1.186	12	94	1.364
D24	77	115	84	4	2676	2660	2660	25	95	4.754	25	95	5.223	19	95	1.814	19	95	1.973
D25	37	50	38	3	1815	**1815**	**1815**	13	75	0.859	13	75	0.992	17	75	1.045	17	75	1.098
E01	73	105	85	10	4885	4830	4830	104	81	28.878	104	81	30.342	57	81	5.990	57	81	6.245
E02	58	81	58	8	3990	3960	3960	94	161	12.502	94	161	13.160	32	161	2.578	32	161	2.679
E03	46	61	47	5	2015	**2015**	**2015**	26	56	2.063	26	56	2.382	25	56	0.927	25	56	1.009
E04	70	99	77	9	4155	4125	4125	98	81	18.284	98	81	19.375	55	81	6.819	55	81	7.111
E05	68	94	61	9	4585	4555	4555	80	79	16.531	80	79	17.533	38	79	3.136	38	79	3.291
E06	49	66	43	5	2055	**2055**	**2055**	28	39	2.853	28	39	3.188	22	39	1.233	22	39	1.312
E07	73	94	50	8	4155	4035	4035	121	126	20.459	121	126	21.300	58	126	5.066	58	126	5.254
E08	74	98	59	9	4710	4640	4640	260	169	56.820	260	169	57.811	129	169	20.040	129	169	20.270
E09	93	141	103	12	5780	5745	5745	236	237	105.287	236	237	108.340	116	237	29.640	116	237	30.296
E10	56	76	49	7	3605	**3605**	**3605**	87	90	10.077	87	90	10.702	48	90	3.195	48	90	3.315
E11	80	113	94	10	4637	4620	4630	216	247	65.394	216	249	67.176	103	273	20.861	103	273	21.130
E12	74	103	67	9	4180	4065	4065	177	348	42.218	177	348	43.436	50	348	5.548	50	348	5.790
E13	49	73	52	7	3345	3305	3320	126	59	17.960	126	59	18.578	70	52	6.109	70	54	6.280
E14	53	72	55	8	4115	4085	4085	57	92	6.859	57	92	7.551	27	92	1.379	27	92	1.504
E15	85	126	107	9	4189	4170	4170	64	496	17.723	64	496	19.276	84	496	14.160	84	496	14.391
E16	60	80	54	7	3755	3735	3735	104	44	13.644	104	44	14.278	100	42	8.183	100	42	8.325
E17	38	50	36	5	2740	**2740**	**2740**	82	40	6.688	82	40	6.994	55	43	2.490	55	43	2.557
E18	78	110	88	8	3825	**3825**	**3825**	111	343	26.712	111	343	27.614	58	343	5.014	58	343	5.165
E19	77	103	66	6	3222	3192.5	3200	148	97	28.123	150	99	31.857	84	87	7.646	86	87	8.363
E20	56	80	63	7	2802	2785	2785	44	232	6.091	44	232	6.668	35	232	2.189	35	232	2.300
E21	57	82	72	7	3728	3725	3725	38	56	5.614	38	56	6.216	39	61	2.077	39	61	2.191

Continued on next page

Table IV.1: *Continued from previous page*

Ins	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|\mathcal{K}\|$	LB	$\texttt{Cost}_1$	$\texttt{Cost}_2$	Ahr's Exact Sep						Our Exact Sep					
								$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$	$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$
E22	54	73	44	5	2470	2440	2440	62	160	6.352	62	160	6.648	50	160	3.873	50	160	3.954
E23	93	130	89	8	3686	3675	3675	190	246	58.855	190	246	60.477	151	209	32.832	151	209	33.221
E24	97	142	86	8	4001	3930	3930	154	261	66.954	154	261	68.266	80	261	12.624	80	261	12.972
E25	26	35	28	4	1615	**1615**	**1615**	13	60	0.652	13	60	0.789	9	60	0.229	9	60	0.268
F01	73	105	85	5	4040	**4040**	**4040**	20	81	5.465	20	81	6.343	15	81	1.995	15	81	2.145
F02	58	81	58	4	3300	**3300**	**3300**	14	163	2.577	14	163	2.994	12	165	1.780	12	165	1.866
F03	46	61	47	3	1665	**1665**	**1665**	10	56	0.468	10	56	0.570	12	56	0.355	12	56	0.419
F04	70	99	77	5	3476	3475	3475	34	88	6.511	34	88	7.048	23	88	2.042	23	88	2.151
F05	68	94	61	5	3605	**3605**	**3605**	22	79	4.196	22	79	4.683	13	79	0.636	13	79	0.738
F06	49	66	43	3	1875	**1875**	**1875**	15	40	1.081	15	40	1.232	7	40	0.502	7	40	0.571
F07	73	94	50	4	3335	**3335**	**3335**	38	126	6.297	38	126	6.795	29	126	3.780	29	126	3.901
F08	74	98	59	5	3690	**3690**	**3695**	66	183	11.802	66	183	12.232	50	202	3.534	50	202	3.701
F09	93	141	103	6	4730	**4730**	**4730**	22	235	7.745	22	235	9.033	17	235	3.154	17	235	3.427
F10	56	76	49	4	2925	**2925**	**2925**	22	109	2.080	22	109	2.364	30	116	2.097	30	116	2.175
F11	80	113	94	5	3835	**3835**	**3835**	12	327	5.069	12	327	5.905	13	300	1.606	13	300	1.756
F12	74	103	67	5	3390	3385	3385	8	348	2.040	8	348	2.689	4	348	0.437	4	348	0.586
F13	49	73	52	4	2855	**2855**	**2855**	6	49	0.894	6	49	1.132	6	49	0.240	6	49	0.332
F14	53	72	55	4	3330	**3330**	**3330**	20	92	1.982	20	92	2.316	9	92	0.566	9	92	0.653
F15	85	126	107	5	3560	**3560**	**3560**	6	494	2.265	6	494	2.908	6	494	0.778	6	494	0.932
F16	60	80	54	4	2725	**2725**	**2725**	17	42	0.941	17	42	1.192	17	42	0.582	17	42	0.674
F17	38	50	36	3	2055	**2055**	**2055**	15	29	0.897	15	29	1.040	12	29	0.495	12	29	0.582
F18	78	110	88	4	3063	3060	3060	12	343	2.044	12	343	2.533	14	343	1.373	14	343	1.518
F19	77	103	66	3	2500	2485	2485	35	64	5.859	35	64	6.274	52	64	7.246	52	64	7.406
F20	56	80	63	4	2445	**2445**	**2445**	5	232	0.763	5	232	1.037	5	232	0.391	5	232	0.475
F21	57	82	72	4	2930	**2930**	**2930**	33	54	2.926	33	54	3.132	48	54	3.510	48	54	3.606
F22	54	73	44	3	2075	**2075**	**2075**	27	160	1.807	27	160	1.964	20	160	0.927	20	160	1.000
F23	93	130	89	4	2994	2985	2985	28	108	9.256	28	108	10.096	19	102	3.168	19	102	3.330

Continued on next page

Table IV.1: *Continued from previous page*

Ins	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|\mathcal{K}\|$	LB	$\texttt{Cost}_1$	$\texttt{Cost}_2$	Ahr's Exact Sep						Our Exact Sep					
								$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$	$\texttt{Cap}_1$	$\texttt{Odd}_1$	$\texttt{Time}_1$	$\texttt{Cap}_2$	$\texttt{Odd}_2$	$\texttt{Time}_2$
F24	97	142	86	4	3210	**3210**	**3210**	18	267	5.954	18	267	6.592	23	267	3.350	23	267	3.560
F25	26	35	28	2	1390	**1390**	**1390**	5	60	0.179	5	60	0.227	5	60	0.192	5	60	0.235
e1-A	77	98	51	5	3548	3527	3527	167	81	25.487	167	81	26.017	123	81	9.698	123	81	9.850
e1-B	77	98	51	7	4498	4463.7	4468	274	82	44.833	274	82	45.568	229	82	25.598	229	82	25.781
e1-C	77	98	51	10	5595	5513	5513	280	81	50.179	280	81	51.190	208	81	23.685	208	81	23.927
e2-A	77	98	72	7	5018	4995	4995	106	101	16.478	106	101	17.192	105	101	7.826	105	101	7.966
e2-B	77	98	72	10	6305	6271	6273	168	101	26.946	169	101	28.087	140	101	14.638	140	101	14.804
e2-C	77	98	72	14	8335	8160.5	8165	248	101	44.159	250	101	46.252	194	101	25.035	194	101	25.353
e3-A	77	98	87	8	5898	5893.8	**5898**	111	209	20.363	111	209	21.169	87	213	9.230	87	213	9.377
e3-B	77	98	87	12	7729	7648.7	7649	149	161	33.012	149	161	34.498	110	175	13.252	110	175	13.561
e3-C	77	98	87	17	10244	10124.5	10138	170	138	37.693	171	139	41.034	144	139	15.081	148	141	16.074
e4-A	77	98	98	9	6408	6378	6378	75	304	15.082	75	304	16.157	48	298	3.501	48	298	3.685
e4-B	77	98	98	14	8935	8838	8838	126	280	24.165	126	280	25.891	104	310	9.656	104	310	10.201
e4-C	77	98	98	19	11493	11376	11383	176	270	35.119	176	270	37.468	127	279	13.885	127	279	14.300
s1-A	140	190	75	7	5018	5010	5010	571	215	265.508	571	215	267.202	410	215	123.690	410	215	124.410
s1-B	140	190	75	10	6388	6368	6368	865	215	461.099	865	215	463.965	507	215	140.192	507	215	141.378
s1-C	140	190	75	14	8518	8404	8404	801	215	394.296	801	215	398.753	533	215	152.893	533	215	154.139
s2-A	140	190	147	14	9825	9737	9737	240	234	182.092	240	234	187.452	164	315	72.018	164	315	76.212
s2-B	140	190	147	20	13017	12901	12901	357	171	240.034	357	171	247.175	215	171	68.024	215	171	71.968
s2-C	140	190	147	27	16425	16247.3	16274	525	171	617.949	526	171	632.157	330	171	426.016	347	171	452.645
s3-A	140	190	159	15	10146	10082.5	10083	263	545	186.441	263	545	191.255	210	370	144.411	210	370	145.777
s3-B	140	190	159	22	13648	13568	13568	399	240	276.988	399	240	284.552	269	240	165.409	269	240	168.352
s3-C	140	190	159	29	17188	17006.4	17019	467	240	716.157	469	240	738.009	322	240	612.498	328	240	637.911
s4-A	140	190	190	19	12144	12026	12026	181	139	114.597	181	139	120.558	136	139	31.896	136	139	33.040
s4-B	140	190	190	27	16103	15984	16001	396	139	322.022	399	139	337.803	232	139	178.946	239	139	186.743
s4-C	140	190	190	35	20430	20235.3	20256	462	139	368.004	466	139	387.719	278	139	235.180	294	139	246.744

Table IV.2: Dual Ascent results

Ins	LB	Dual Ascent									Single Cuts			Complete Cuts			Connected Cuts			MST Cuts		
		Cost	Time	Cuts	SGL	CMP	CON	MST	Int	Time	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts
kshs1	14661	**14661**	<0.01	42	11	19	19	5	**14661**	<0.01	11277	<0.01	10	**14661**	<0.01	20	10542	<0.01	17	**14661**	<0.01	8
kshs2	9863	**9863**	<0.01	70	25	34	34	3	**9863**	<0.01	8099	<0.01	12	9325	<0.01	33	8160	<0.01	40	9275	<0.01	7
kshs3	9320	**9320**	<0.01	21	11	9	5	0	**9320**	<0.01	9045	<0.01	8	8813	<0.01	7	8114	<0.01	5	9045	<0.01	5
kshs4	11498	11098	<0.01	46	9	23	15	6	11098	<0.01	8680	<0.01	5	11098	<0.01	22	8998	<0.01	16	10774	<0.01	6
kshs5	10957	**10957**	<0.01	43	12	16	22	4	**10957**	<0.01	10353	<0.01	8	10921	<0.01	17	9934	<0.01	18	**10957**	<0.01	9
kshs6	10197	**10197**	<0.01	59	9	22	28	6	**10197**	<0.01	**10197**	<0.01	11	**10197**	<0.01	28	9932	<0.01	33	10192	<0.01	12
best											2			4			0			3		
gdb1	316	311	<0.01	109	32	43	57	20	**316**	0.01	**316**	<0.01	18	299	<0.01	36	280	<0.01	46	308	<0.01	22
gdb2	339	**339**	<0.01	102	18	40	59	11	**339**	<0.01	315	<0.01	9	332	<0.01	36	310	<0.01	45	**339**	<0.01	11
gdb3	275	**275**	<0.01	84	21	35	51	9	**275**	<0.01	259	<0.01	13	272	<0.01	35	258	<0.01	55	267	<0.01	10
gdb4	287	**287**	<0.01	82	24	30	41	4	**287**	<0.01	266	<0.01	11	283	<0.01	29	265	<0.01	50	282	<0.01	13
gdb5	377	371	<0.01	150	21	48	81	20	**377**	<0.01	346	<0.01	17	369	<0.01	42	339	<0.01	49	346	<0.01	20
gdb6	298	**298**	<0.01	66	7	32	36	5	**298**	<0.01	279	<0.01	6	**298**	<0.01	44	279	<0.01	29	**298**	<0.01	11
gdb7	325	**325**	<0.01	105	27	38	62	13	**325**	<0.01	304	<0.01	17	317	<0.01	32	291	<0.01	52	309	<0.01	22
gdb8	348	329	<0.01	485	119	196	240	94	344	0.02	275	<0.01	36	323	<0.01	179	335	<0.01	183	324	<0.01	57
gdb9	303	**303**	0.01	538	111	182	333	81	**303**	0.02	240	<0.01	22	289	<0.01	159	289	<0.01	273	286	<0.01	64
gdb10	275	**275**	<0.01	87	18	30	43	11	**275**	<0.01	**275**	<0.01	7	266	<0.01	28	273	<0.01	43	**275**	<0.01	12
gdb11	395	**395**	<0.01	305	84	86	188	26	**395**	0.01	387	<0.01	21	381	<0.01	65	380	<0.01	147	387	<0.01	27
gdb12	458	450	<0.01	146	32	52	79	8	450	0.01	384	<0.01	11	446	<0.01	58	406	<0.01	72	423	<0.01	24
gdb13	536	**536**	<0.01	66	12	21	60	5	**536**	<0.01	520	<0.01	6	531	<0.01	25	520	<0.01	31	532	<0.01	5
gdb14	100	**100**	<0.01	1	0	1	0	0	**100**	<0.01	96	<0.01	0	**100**	<0.01	1	96	<0.01	0	96	<0.01	0
gdb15	58	**58**	<0.01	1	0	1	0	0	**58**	<0.01	56	<0.01	0	**58**	<0.01	1	56	<0.01	0	56	<0.01	0
gdb16	127	**127**	<0.01	27	15	5	12	2	**127**	<0.01	125	<0.01	8	125	<0.01	3	121	<0.01	13	125	<0.01	12
gdb17	91	87	<0.01	16	7	1	8	0	**91**	<0.01	**91**	<0.01	7	87	<0.01	1	85	<0.01	9	**91**	<0.01	7
gdb18	164	**164**	<0.01	2	0	1	0	1	**164**	<0.01	158	<0.01	0	**164**	<0.01	1	158	<0.01	0	**164**	<0.01	1
gdb19	55	**55**	<0.01	35	11	18	13	0	**55**	<0.01	**55**	<0.01	6	**55**	<0.01	15	**55**	<0.01	11	**55**	<0.01	6
gdb20	121	**121**	<0.01	63	16	25	30	4	**121**	<0.01	**121**	<0.01	10	117	<0.01	18	116	<0.01	21	121	<0.01	13

Continued on next page

Table IV.2: *Continued from previous page*

Ins	LB	Dual Ascent									Single Cuts			Complete Cuts			Connected Cuts			MST Cuts		
		Cost	Time	Cuts	SGL	CMP	CON	MST	Int	Time	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts
gdb21	156	**156**	<0.01	51	6	17	34	2	**156**	<0.01	154	<0.01	5	**156**	<0.01	20	153	<0.01	21	154	<0.01	5
gdb22	200	199	<0.01	42	7	11	26	3	**200**	<0.01	196	<0.01	8	198	<0.01	10	193	<0.01	26	196	<0.01	8
gdb23	233	**233**	<0.01	11	0	11	0	0	**233**	<0.01	223	<0.01	0	**233**	<0.01	11	223	<0.01	0	223	<0.01	0
best											7			5			3			10		
1A	173	170	<0.01	270	64	71	130	46	**173**	0.02	**173**	<0.01	44	162	<0.01	62	166	<0.01	92	171	<0.01	52
1B	173	171	<0.01	278	75	82	150	40	**173**	0.01	**173**	<0.01	44	162	<0.01	62	167	<0.01	116	171	<0.01	52
1C	245	231	<0.01	341	88	131	184	56	232	0.01	177	<0.01	44	216	<0.01	113	216	<0.01	215	226	<0.01	66
2A	227	**227**	<0.01	211	74	78	117	36	**227**	0.01	217	<0.01	27	221	<0.01	83	207	<0.01	116	225	<0.01	23
2B	259	257	<0.01	291	86	122	156	52	257	0.01	217	<0.01	27	249	<0.01	106	219	<0.01	177	235	<0.01	22
2C	457	449	<0.01	541	114	227	305	93	455	0.01	282	<0.01	39	445	<0.01	208	360	<0.01	273	427	<0.01	45
3A	81	77	<0.01	192	54	59	94	42	**81**	0.01	77	<0.01	13	78	<0.01	71	74	<0.01	89	80	<0.01	57
3B	87	85	<0.01	265	64	82	150	49	**87**	0.01	77	<0.01	13	84	<0.01	78	78	<0.01	120	84	<0.01	57
3C	138	135	<0.01	361	75	144	207	78	135	0.01	79	<0.01	14	134	<0.01	141	106	<0.01	141	123	<0.01	48
4A	400	395	0.01	874	205	223	573	141	396	0.03	385	<0.01	39	382	<0.01	216	379	0.01	494	395	<0.01	66
4B	412	405	0.01	973	188	350	553	142	**412**	0.03	385	<0.01	39	396	<0.01	265	388	0.01	422	407	<0.01	82
4C	428	419	0.01	877	212	317	518	148	424	0.03	385	<0.01	39	413	<0.01	346	407	0.01	460	419	<0.01	82
4D	530	511	0.01	1200	229	458	704	189	515	0.03	385	<0.01	39	489	<0.01	418	494	0.01	583	472	<0.01	94
5A	423	420	0.01	670	117	191	394	79	**423**	0.02	410	<0.01	62	408	<0.01	202	410	<0.01	402	**423**	<0.01	85
5B	446	440	0.01	678	123	200	416	92	441	0.02	412	<0.01	62	426	<0.01	193	404	<0.01	317	441	<0.01	85
5C	474	459	0.01	774	144	247	460	109	467	0.02	416	<0.01	62	450	<0.01	224	420	0.01	407	451	<0.01	76
5D	577	569	0.01	887	163	340	506	105	569	0.02	430	<0.01	62	558	<0.01	304	503	0.01	479	528	<0.01	90
6A	223	222	0.01	474	97	166	249	72	**223**	0.01	220	<0.01	51	208	<0.01	122	217	<0.01	290	**223**	<0.01	53
6B	233	228	0.01	483	107	151	310	74	229	0.01	220	<0.01	51	214	<0.01	134	221	<0.01	279	223	<0.01	53
6C	317	296	0.01	548	129	230	335	106	300	0.01	220	<0.01	51	279	<0.01	216	278	<0.01	265	248	<0.01	54
7A	279	278	0.01	485	129	171	268	64	**279**	0.02	**279**	<0.01	36	264	<0.01	130	276	<0.01	287	278	<0.01	38
7B	283	282	0.01	527	145	180	304	76	**283**	0.02	279	<0.01	36	264	<0.01	130	273	<0.01	214	282	<0.01	39
7C	334	323	0.01	528	172	245	252	121	323	0.02	279	<0.01	36	301	<0.01	236	314	0.01	357	317	<0.01	44
8A	386	385	0.01	536	126	122	331	72	**386**	0.02	383	<0.01	45	375	<0.01	144	367	<0.01	276	383	<0.01	45

Continued on next page

Table IV.2: *Continued from previous page*

Ins	LB	Dual Ascent									Single Cuts			Complete Cuts			Connected Cuts			MST Cuts		
		Cost	Time	Cuts	SGL	CMP	CON	MST	Int	Time	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts
8B	395	**395**	0.01	600	136	148	379	64	**395**	0.02	383	<0.01	45	386	<0.01	156	382	<0.01	297	385	<0.01	41
8C	521	503	0.01	660	130	222	423	57	508	0.02	383	<0.01	45	499	<0.01	242	468	<0.01	375	458	<0.01	61
9A	323	319	0.02	1172	299	320	732	187	**323**	0.04	321	<0.01	81	299	<0.01	178	303	0.01	526	320	<0.01	147
9B	326	320	0.02	1091	285	296	671	171	**326**	0.04	321	<0.01	81	305	<0.01	182	303	0.01	580	321	<0.01	159
9C	332	325	0.02	1068	242	313	638	158	**332**	0.04	321	<0.01	81	311	<0.01	200	302	0.01	543	321	<0.01	159
9D	391	374	0.02	993	267	335	567	148	377	0.04	325	<0.01	83	359	<0.01	261	351	0.01	555	337	<0.01	163
10A	428	418	0.02	1096	249	250	727	150	**428**	0.04	420	<0.01	73	400	<0.01	218	399	0.01	542	424	<0.01	149
10B	436	429	0.02	1240	286	321	786	172	**436**	0.05	420	<0.01	73	406	<0.01	247	406	0.01	686	431	<0.01	149
10C	446	437	0.02	1294	288	354	817	188	444	0.05	420	<0.01	73	415	<0.01	264	416	0.01	675	439	<0.01	149
10D	526	509	0.02	1487	314	475	931	201	517	0.05	420	<0.01	73	490	<0.01	388	470	0.01	788	491	<0.01	149
best											8			9			1			20		
C01	4105	3760	0.04	2447	695	1003	1370	381	3865	0.09	2965	<0.01	157	3625	0.01	910	3700	0.03	1644	3750	0.01	266
C02	3135	3090	0.02	1019	283	487	506	76	3095	0.03	2340	<0.01	43	2830	<0.01	377	2935	0.01	534	2850	<0.01	85
C03	2575	2490	0.01	1033	329	394	581	150	2525	0.02	1985	<0.01	59	2265	<0.01	352	2340	0.01	557	2455	<0.01	96
C04	3478	3335	0.03	1600	494	660	852	247	3410	0.04	2645	<0.01	155	3025	0.01	595	3210	0.02	1239	3305	<0.01	205
C05	5365	5015	0.02	1671	331	691	872	292	5225	0.04	3885	<0.01	66	4825	0.01	655	4945	0.01	973	4850	<0.01	252
C06	2535	2445	0.01	828	283	311	457	148	2485	0.02	2155	<0.01	46	2275	<0.01	322	2240	0.01	480	2440	<0.01	86
C07	4075	3815	0.02	1885	425	721	937	277	3915	0.04	2945	<0.01	118	3485	0.01	687	3630	0.01	980	3575	<0.01	194
C08	4090	3885	0.04	2320	360	882	1219	252	3955	0.08	2675	<0.01	72	3635	0.01	890	3715	0.02	1264	3795	<0.01	142
C09	5233	5105	0.06	3182	565	1228	1820	385	5165	0.11	3845	<0.01	145	4740	0.02	1019	4925	0.03	1771	4775	0.01	222
C10	4700	4285	0.03	2446	343	921	1252	321	4515	0.08	3060	<0.01	138	3860	0.01	763	3975	0.02	1215	4050	<0.01	262
C11	4583	4345	0.07	3661	869	1280	2175	420	4420	0.24	3465	<0.01	161	4015	0.02	1285	4220	0.04	1832	4005	0.01	361
C12	4209	4000	0.03	2142	418	869	1180	412	4070	0.08	3060	<0.01	125	3830	0.01	777	3830	0.02	1436	3655	<0.01	232
C13	2955	2795	0.01	1051	254	406	555	202	2875	0.02	2320	<0.01	43	2590	<0.01	395	2645	0.01	528	2615	<0.01	126
C14	4030	3890	0.04	2483	538	1008	1345	377	3950	0.07	2990	<0.01	140	3515	0.01	831	3735	0.02	1288	3750	<0.01	171
C15	4912	4675	0.12	4736	1138	1866	2613	698	4810	0.19	3920	<0.01	153	4020	0.03	1624	4480	0.06	2275	4425	0.01	430
C16	1475	1410	0.01	654	176	235	341	93	1470	0.02	1020	<0.01	22	1220	<0.01	201	1320	<0.01	336	1270	<0.01	56
C17	3555	3340	0.02	1436	321	520	825	239	3535	0.07	2380	<0.01	55	2975	<0.01	422	3210	0.01	861	3175	<0.01	145

Continued on next page

Table IV.2: *Continued from previous page*

Ins	LB	Dual Ascent									Single Cuts			Complete Cuts			Connected Cuts			MST Cuts		
		Cost	Time	Cuts	SGL	CMP	CON	MST	Int	Time	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts
C18	5577	5415	0.09	4557	1266	1693	2447	745	5485	0.14	3730	<0.01	213	5045	0.02	1407	5160	0.05	2410	5005	0.01	460
C19	3096	2875	0.04	2494	704	1026	1336	493	2960	0.18	2275	<0.01	76	2600	0.01	854	2795	0.02	1296	2835	0.01	322
C20	2120	2020	0.02	1350	370	476	709	201	2035	0.03	1860	<0.01	62	1810	<0.01	397	1980	0.01	534	1960	<0.01	165
C21	3960	3670	0.03	2015	423	801	1104	191	3710	0.05	2640	<0.01	59	3435	0.01	726	3520	0.02	1210	3785	<0.01	160
C22	2245	2225	0.03	1651	342	565	898	316	2235	0.04	1885	<0.01	111	1825	0.01	541	2105	0.01	895	2100	<0.01	187
C23	4032	3645	0.06	3501	797	1190	1909	577	3895	0.11	3115	<0.01	174	3220	0.02	1008	3515	0.04	1874	3595	0.01	397
C24	3384	3240	0.05	2547	528	750	1448	374	3350	0.09	2435	<0.01	136	2910	0.01	851	3185	0.04	1674	3265	0.01	337
C25	2310	2160	0.01	912	170	382	477	174	2290	0.02	1740	<0.01	77	2075	<0.01	361	2095	<0.01	422	2095	<0.01	122
best											0			1			14			12		
D01	3215	3115	0.04	1757	540	604	1006	196	3145	0.08	2965	<0.01	157	2925	0.01	635	3095	0.03	1284	3120	0.01	221
D02	2520	**2520**	0.01	873	233	375	457	60	**2520**	0.03	2340	<0.01	43	2365	<0.01	323	2395	0.01	389	2400	<0.01	27
D03	2065	2045	0.01	793	268	283	446	94	**2065**	0.02	1985	<0.01	59	1875	<0.01	244	1965	0.01	493	2045	<0.01	66
D04	2785	2695	0.03	1875	507	650	970	276	2740	0.06	2645	<0.01	155	2470	0.01	466	2555	0.02	883	2720	<0.01	241
D05	3935	3815	0.03	1801	374	584	1054	271	3855	0.05	3455	<0.01	58	3495	0.01	533	3660	0.01	840	3735	<0.01	168
D06	2125	2115	0.01	820	194	257	456	119	**2125**	0.02	2075	<0.01	46	1905	<0.01	178	1915	0.01	369	2120	<0.01	101
D07	3115	3015	0.02	1397	402	490	755	169	3015	0.04	2945	<0.01	118	2685	0.01	498	2855	0.01	756	2975	<0.01	155
D08	2995	2895	0.02	1269	223	587	603	114	2975	0.04	2675	<0.01	72	2795	0.01	648	2825	0.02	808	2895	<0.01	102
D09	4120	4095	0.05	2484	502	881	1497	254	4100	0.11	3845	<0.01	145	3880	0.01	744	3980	0.03	1342	3970	0.01	162
D10	3340	3280	0.03	1752	226	718	902	238	3330	0.05	2950	<0.01	138	3060	0.01	618	3110	0.02	961	3050	<0.01	174
D11	3745	3620	0.07	3323	700	1077	1843	447	3710	0.15	3465	<0.01	161	3275	0.02	901	3525	0.04	1611	3585	0.01	438
D12	3310	3210	0.03	1681	363	691	806	341	3260	0.06	3060	<0.01	125	3020	0.01	610	3110	0.02	1178	3135	<0.01	176
D13	2535	2470	0.01	982	260	313	529	158	**2535**	0.03	2320	<0.01	43	2240	<0.01	280	2290	0.01	448	2405	<0.01	147
D14	3272	3170	0.03	1973	561	698	933	290	3220	0.05	2980	<0.01	137	2795	0.01	647	3030	0.02	1023	3130	<0.01	217
D15	3990	3935	0.11	4120	1019	1374	2407	562	3970	0.17	3865	<0.01	151	3475	0.02	1076	3790	0.06	2140	3930	0.01	336
D16	1060	1050	<0.01	390	101	142	200	51	1050	0.04	1020	<0.01	22	1050	<0.01	132	**1060**	<0.01	175	1050	<0.01	39
D17	2620	2600	0.01	845	247	302	431	111	**2620**	0.02	2370	<0.01	55	2395	<0.01	342	2540	0.01	451	2470	<0.01	94
D18	4165	4115	0.09	4008	1229	1470	2172	649	**4165**	0.15	3730	<0.01	213	3940	0.02	1125	3855	0.04	1684	3930	0.01	498
D19	2393	2370	0.03	2079	585	671	1106	365	2370	0.06	2215	<0.01	82	2010	0.01	538	2175	0.02	1025	2360	0.01	282

Continued on next page

Table IV.2: *Continued from previous page*

Ins	LB	Dual Ascent									Single Cuts			Complete Cuts			Connected Cuts			MST Cuts		
		Cost	Time	Cuts	SGL	CMP	CON	MST	Int	Time	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts
D20	1870	1860	0.01	993	282	298	574	129	**1870**	0.02	1860	<0.01	62	1660	<0.01	301	1725	0.01	521	1860	<0.01	115
D21	2985	2930	0.03	1443	379	442	856	185	2940	0.04	2640	<0.01	59	2665	0.01	449	2655	0.01	697	2935	<0.01	129
D22	1865	1835	0.02	1422	304	476	766	258	1845	0.04	1725	<0.01	107	1545	0.01	383	1760	0.01	765	**1865**	<0.01	220
D23	3114	3005	0.06	2559	633	913	1393	326	3080	0.09	2955	<0.01	173	2640	0.01	642	2830	0.03	1423	3015	0.01	282
D24	2676	2605	0.05	2291	461	669	1327	282	2660	0.09	2435	<0.01	136	2460	0.01	610	2510	0.03	1238	2610	0.01	213
D25	1815	**1815**	0.01	640	166	239	329	112	**1815**	0.02	1740	<0.01	77	1680	<0.01	232	1655	<0.01	268	**1815**	<0.01	106
best											1			1			4			20		
E01	4885	4660	0.06	3464	987	1222	1994	480	4785	0.14	3800	<0.01	239	4320	0.02	1089	4435	0.03	1865	4440	0.01	424
E02	3990	3885	0.03	1790	419	661	890	299	3935	0.05	3120	<0.01	152	3500	0.01	556	3805	0.02	1297	3495	<0.01	177
E03	2015	2005	0.01	902	301	386	441	146	**2015**	0.02	1585	<0.01	79	1735	<0.01	322	1860	0.01	461	1945	<0.01	90
E04	4155	4015	0.05	2746	874	1025	1529	445	4105	0.08	3270	<0.01	203	3680	0.01	855	3805	0.03	1727	3840	0.01	334
E05	4585	4395	0.04	2229	466	843	1174	372	4535	0.07	3410	<0.01	190	4175	0.01	807	4410	0.02	1363	4180	<0.01	253
E06	2055	2045	0.02	1055	228	433	528	140	**2055**	0.03	1720	<0.01	46	1840	<0.01	413	1965	0.01	597	1980	<0.01	103
E07	4155	3925	0.06	3602	668	1244	1854	621	4005	0.11	3095	<0.01	177	3450	0.01	1022	3815	0.03	1826	3645	0.01	325
E08	4710	4335	0.06	3587	651	1192	1892	589	4555	0.13	3290	<0.01	156	3940	0.01	1053	4265	0.03	1913	4145	0.01	325
E09	5780	5495	0.11	5175	1108	1692	3127	845	5695	0.23	4530	0.01	399	4935	0.03	1489	5240	0.07	3021	5000	0.01	417
E10	3605	3450	0.03	2005	559	788	985	379	3515	0.05	2630	<0.01	163	3015	0.01	629	3245	0.01	941	3325	<0.01	238
E11	4637	4375	0.07	3561	1018	1330	1893	607	4525	0.13	3575	<0.01	270	4010	0.02	1182	4100	0.03	1547	4105	0.01	425
E12	4180	4005	0.06	3321	546	1079	1856	497	4065	0.10	3220	<0.01	203	3630	0.01	1033	3900	0.03	1911	3755	0.01	367
E13	3345	3230	0.02	2011	397	633	1170	292	3260	0.09	2695	<0.01	79	2875	0.01	626	2930	0.01	831	2830	<0.01	193
E14	4115	3940	0.03	2247	682	809	1222	431	3990	0.05	3135	<0.01	150	3465	0.01	716	3690	0.01	1065	3655	<0.01	210
E15	4189	4095	0.08	3645	1197	1398	1910	791	4155	0.12	3470	<0.01	225	3275	0.02	1162	3975	0.05	2191	3985	0.01	519
E16	3755	3445	0.04	2661	672	1029	1460	539	3635	0.10	2485	<0.01	106	3115	0.01	955	3365	0.02	1350	3205	<0.01	212
E17	2740	2450	0.02	928	239	324	460	192	2595	0.08	1890	<0.01	57	2180	<0.01	352	2375	0.01	593	2635	<0.01	186
E18	3825	3720	0.06	3256	905	1147	1678	577	3785	0.09	3165	<0.01	162	3315	0.01	995	3525	0.03	1624	3505	0.01	391
E19	3222	2895	0.06	3734	761	1359	1985	610	3030	0.11	2340	<0.01	118	2725	0.02	1122	2915	0.04	2116	3000	0.01	439
E20	2802	2640	0.03	1857	463	675	1030	285	2735	0.05	2395	<0.01	146	2395	0.01	641	2620	0.02	1089	2480	<0.01	172
E21	3728	3430	0.03	1954	553	720	1116	207	3500	0.06	2585	<0.01	124	3070	0.01	651	3375	0.02	1077	3440	<0.01	227

Continued on next page

Table IV.2: *Continued from previous page*

Ins	LB	Dual Ascent									Single Cuts			Complete Cuts			Connected Cuts			MST Cuts		
		Cost	Time	Cuts	SGL	CMP	CON	MST	Int	Time	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts
E22	2470	2345	0.02	1509	325	499	859	299	2390	0.04	2090	<0.01	149	1995	0.01	509	2285	0.01	979	2260	<0.01	204
E23	3686	3455	0.10	4463	832	1571	2623	560	3560	0.18	2860	<0.01	255	3095	0.03	1383	3295	0.05	2074	3275	0.01	308
E24	4001	3810	0.12	5034	1127	1866	2666	822	3930	0.22	2955	<0.01	293	3295	0.03	1427	3510	0.07	2741	3730	0.01	496
E25	1615	1610	<0.01	520	168	188	253	96	**1615**	0.01	1380	<0.01	59	1420	<0.01	147	1495	<0.01	198	1600	<0.01	84
best											0			0			13			12		
F01	4040	3875	0.06	3198	1007	1021	1771	534	4015	0.12	3800	<0.01	239	3450	0.01	719	3700	0.04	1816	3825	0.01	396
F02	3300	3235	0.03	2220	519	723	1109	477	3290	0.06	3120	<0.01	152	2820	0.01	417	3160	0.02	1083	3290	<0.01	301
F03	1665	**1665**	0.01	803	273	309	406	118	**1665**	0.02	1585	<0.01	79	1555	<0.01	262	1575	0.01	512	1655	<0.01	88
F04	3476	3415	0.05	2810	874	850	1648	482	3475	0.09	3270	<0.01	203	3045	0.01	657	3150	0.03	1516	3345	0.01	320
F05	3605	3525	0.04	2426	557	803	1285	410	**3605**	0.10	3280	<0.01	190	3065	0.01	606	3385	0.03	1479	3230	<0.01	197
F06	1875	1830	0.01	735	198	242	384	133	**1875**	0.03	1720	<0.01	46	1650	<0.01	322	1725	0.01	558	1830	<0.01	99
F07	3335	3205	0.05	3048	630	1093	1556	547	3315	0.11	3095	<0.01	177	2665	0.01	868	3000	0.03	1577	3135	0.01	362
F08	3690	3635	0.06	3203	691	949	1665	599	3685	0.11	3270	<0.01	156	3190	0.01	868	3430	0.03	1577	3405	0.01	348
F09	4730	4555	0.12	4848	1100	1526	2713	780	**4730**	0.25	4480	0.01	399	4070	0.02	1080	4385	0.07	2784	4480	0.01	665
F10	2925	2860	0.03	1743	456	596	966	296	**2925**	0.05	2600	<0.01	163	2465	0.01	532	2610	0.02	1174	2690	<0.01	233
F11	3835	3780	0.07	3360	941	1076	1824	503	**3835**	0.13	3575	<0.01	270	3290	0.01	857	3535	0.04	1954	3755	0.01	397
F12	3390	3375	0.06	3051	565	863	1754	512	3385	0.11	3220	<0.01	203	2880	0.01	743	3150	0.03	1547	3345	0.01	507
F13	2855	2720	0.02	1363	314	379	770	197	2845	0.04	2695	<0.01	79	2425	<0.01	410	2600	0.01	677	2770	<0.01	265
F14	3330	3160	0.03	1959	615	644	1005	355	3320	0.04	3125	<0.01	150	2740	0.01	592	3045	0.01	924	3160	<0.01	241
F15	3560	3495	0.08	3668	974	1106	2062	641	**3560**	0.15	3445	<0.01	223	2895	0.02	780	3240	0.04	1834	3530	0.01	496
F16	2725	2655	0.03	1918	520	736	992	279	**2725**	0.07	2485	<0.01	106	2425	0.01	565	2545	0.02	934	**2725**	<0.01	151
F17	2055	2030	0.01	669	223	243	329	90	**2055**	0.02	1890	<0.01	57	1825	<0.01	263	1975	0.01	465	2005	<0.01	145
F18	3063	3035	0.06	2849	851	878	1546	523	3060	0.11	2925	<0.01	159	2660	0.01	712	2815	0.03	1367	3025	0.01	520
F19	2500	2455	0.05	2627	676	830	1362	423	2485	0.09	2310	<0.01	118	2120	0.01	743	2325	0.03	1280	2470	0.01	370
F20	2445	2395	0.03	1593	485	535	829	261	**2445**	0.05	2385	<0.01	146	2070	0.01	443	2270	0.02	965	2380	<0.01	262
F21	2930	2865	0.03	1951	574	618	1092	259	**2930**	0.05	2585	<0.01	124	2440	0.01	450	2540	0.02	870	2885	<0.01	267
F22	2075	1990	0.02	1486	310	497	817	290	2060	0.04	1940	<0.01	153	1705	<0.01	366	1900	0.01	871	1985	<0.01	215
F23	2994	2860	0.08	3437	693	1147	1905	552	2945	0.15	2860	<0.01	255	2565	0.02	906	2795	0.05	1884	2890	0.01	342

Continued on next page

Table IV.2: *Continued from previous page*

Ins	LB	Dual Ascent									Single Cuts			Complete Cuts			Connected Cuts			MST Cuts		
		Cost	Time	Cuts	SGL	CMP	CON	MST	Int	Time	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts	Cost	Time	Cuts
F24	3210	3115	0.10	3876	818	1109	2342	642	3205	0.20	2955	<0.01	297	2680	0.02	925	2930	0.08	2708	3110	0.01	439
F25	1390	1380	<0.01	339	127	103	177	69	**1390**	0.01	1340	<0.01	56	1200	<0.01	117	1275	<0.01	143	**1390**	<0.01	84
best											3			0			2			21		
e1-A	3548	3468	0.08	4368	836	1637	2381	593	3527	0.12	2089	<0.01	175	3005	0.02	1416	3386	0.05	2758	3442	0.01	412
e1-B	4498	4294	0.09	5093	857	1957	2919	792	4372	0.13	2097	<0.01	166	3831	0.02	1707	4225	0.05	3079	4272	0.01	559
e1-C	5595	5345	0.08	4643	921	1917	2678	807	5459	0.11	4363	<0.01	192	4912	0.03	2048	5277	0.05	3039	5089	0.01	444
e2-A	5018	4834	0.09	4996	1471	2087	2533	888	4898	0.12	2702	<0.01	280	4201	0.02	1702	4561	0.05	2915	4748	0.01	523
e2-B	6305	6165	0.08	4716	1475	2060	2468	953	6192	0.12	2931	<0.01	266	5457	0.03	1926	5686	0.06	2765	5797	0.01	462
e2-C	8335	7752	0.09	5370	1534	2269	3027	1061	7936	0.14	3252	<0.01	258	7309	0.03	2220	7580	0.04	2430	7394	0.01	461
e3-A	5898	5715	0.09	5163	1879	2019	2856	908	5783	0.12	3150	<0.01	292	5012	0.03	1976	5403	0.05	2462	5499	0.01	671
e3-B	7729	7412	0.08	4599	2085	2181	2590	1019	7478	0.12	3260	<0.01	319	6739	0.03	2085	6974	0.05	3149	6914	0.01	585
e3-C	10244	9769	0.08	4719	2019	2233	2803	962	9955	0.13	7131	<0.01	239	9071	0.03	2318	9487	0.04	2738	9042	0.01	464
e4-A	6408	6237	0.08	4419	1659	1726	2593	862	6242	0.11	3322	<0.01	245	5611	0.03	1953	5861	0.04	2456	5820	0.01	469
e4-B	8935	8681	0.09	5079	1643	2154	2886	1024	8763	0.13	3612	<0.01	248	7878	0.03	2041	7852	0.04	2692	8009	0.01	522
e4-C	11493	10940	0.08	5139	1666	2293	3052	1041	11243	0.13	8091	<0.01	262	10476	0.03	2428	10177	0.05	3282	10220	0.01	463
s1-A	5018	4693	0.48	14843	1784	5979	8413	2659	4841	0.90	2476	0.01	752	4189	0.28	7996	4740	0.23	7882	4305	0.03	1257
s1-B	6388	5850	0.63	15994	1748	6634	9831	3007	6109	1.10	2759	0.01	766	5565	0.29	8102	5918	0.25	9222	5342	0.03	905
s1-C	8518	7983	0.64	20068	1876	7269	11834	3660	8230	1.38	4864	0.01	708	7699	0.29	8266	8113	0.25	9428	7282	0.03	1025
s2-A	9825	9411	0.56	16026	5956	7240	8914	3045	9605	0.97	4971	0.01	807	8404	0.29	8114	9077	0.27	9277	8606	0.03	1528
s2-B	13017	12431	0.60	17613	6113	7997	11002	3367	12745	2.05	4844	0.01	864	11699	0.29	8242	12306	0.26	9886	10631	0.03	1224
s2-C	16425	15715	0.61	18153	5975	7897	11637	3821	16059	1.74	5543	0.01	630	15110	0.28	8055	15517	0.28	10828	13190	0.03	970
s3-A	10146	9608	0.51	14609	5117	6753	8274	2557	9801	0.81	4730	0.01	755	8628	0.25	7089	9363	0.27	9639	8370	0.03	1256
s3-B	13648	13190	0.58	16767	5490	7609	10479	3051	13391	1.62	5067	0.01	704	12270	0.26	7300	12922	0.26	9673	10994	0.03	1287
s3-C	17188	16491	0.73	18648	5531	7683	10903	3508	16766	1.75	5285	0.01	633	15843	0.28	7865	16332	0.28	10923	14613	0.03	1279
s4-A	12144	11721	0.56	14912	5251	7109	8580	3000	11881	0.99	5209	0.01	781	10759	0.26	7539	11357	0.26	9092	10549	0.03	1141
s4-B	16103	15557	0.63	16854	5292	7628	10643	3331	15800	1.37	5246	0.01	751	14729	0.29	8117	15106	0.27	10388	13570	0.03	1195
s4-C	20430	19767	0.60	17697	5450	7964	11296	4153	20064	1.53	5660	0.01	755	19127	0.29	7843	19598	0.28	10895	17023	0.03	1444
best											0			1			17			6		

Table IV.3: Improvement of the exact separation using the dual ascent heuristic as hot-start

Dataset	Cap	Odd	Time
kshs	100.00%	100.00%	34.03%
gdb	100.00%	100.00%	36.20%
bccm	91.03%	99.50%	48.79%
C	76.30%	95.00%	66.76%
D	83.07%	99.68%	64.94%
E	77.16%	97.11%	68.71%
F	91.86%	99.22%	69.76%
eglese	69.62%	97.73%	32.94%

Table IV.4: Results for *egl-large* dataset

Ins	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|\mathcal{K}\|$	Dual Ascent			DA + Our			
					Cost	Cuts	Time	Cost	Cap	Odd	Time
g1-a	255	375	347	20	927232	54246	4.201	970495	351	196	2091.639
g1-b	255	375	347	25	1044780	58934	4.542	1085096	323	106	2149.614
g1-c	255	375	347	30	1153372	59753	4.605	1201028	475	147	5394.857
g1-d	255	375	347	35	1263641	69159	5.336	1325317	557	256	6509.326
g1-e	255	375	347	40	1384581	73761	5.699	1461469	610	266	7456.939
g2-a	255	375	375	22	1020539	54511	4.298	1061103	278	240	1965.443
g2-b	255	375	375	27	1129794	57237	4.440	1173286	379	254	3181.108
g2-c	255	375	375	32	1252044	62286	4.701	1295036	416	89	3868.572
g2-d	255	375	375	37	1360453	67949	5.267	1430267	571	46	5748.443
g2-e	255	375	375	42	1479110	73621	5.725	1557159	574	101	7919.063

V
Efficient Restricted Non-Elementary Route Pricing

The scope of this chapter is to present an efficient method which prices a special kind of non-elementary routes, called the ng-routes. Such non-elementary routes are built with respect to what we call the ng-sets. These ng-sets are defined for each customer and they act like a memory of each customer. For instance, an elementary route must remember every traversed customer in order to prohibit any repetition. The ng-routes have a weaker memory. Each time it arrives at a new customer, the ng-route can only remember any other traversed customer if it belongs to the memory of the current customer, otherwise the customer is forgotten. For this reason, the larger is the ng-sets, the best are the routes obtained.

The interest on the ng-routes arose from the need of obtaining better bounds than the existing relaxations available in the literature, but without dealing with the high costs incurred by pricing elementary routes.

The proposed algorithm starts by combining Righini and Salani's [72] Decremental State Space Relaxation (DSSR) technique with completion bounds. The DSSR approach allows controlling the number of labels in the dynamic programming matrix by relaxing the state-space defined by the ng-sets associated with each customer, leading to a global computation time much smaller than solving once for the original ng-sets space. The completion bounds are used to estimate a lower bound on the reduced cost of the best route a label can lead to. These bounds may prevent the algorithm to unnecessarily generate routes with non-negative reduced costs.

The algorithm is then adapted to both CARP and GVRP. We report experiments showing that the algorithm is capable of pricing ng-routes with ng-set sizes up to sixty-four for hard instances of CARP, GVRP and CVRP, a number about three times greater than what has been published so far. As mentioned earlier, the CVRP instances are solved considering them as GVRP instances. The results of the column generation algorithm also provide a clear idea of the improvements in terms of lower bounds when the ng-set

size parameter increases, as well as the time required for computing it. In addition, several new best lower bounds are found for the GVRP, specially for large instances. Moreover, impressive lower bounds for open CARP and CVRP instances are obtained, keeping in mind that they were computed using only column generation and some classical cuts.

V.1 Column Generation

Column generation is a technique commonly used to solve a linear program with a huge number of variables (columns). The algorithm starts with a small subset of columns, which gives rise to a small linear program (LP). This LP, generally called the restricted master, can be easily solved. Next, the algorithm tries to price a new column with a reduced cost suitable to improve the current solution. This iterative process continues until no improving column exists.

As previously mentioned, the column generation presented in this chapter uses the set partitioning formulations described in Section III.1. As a minimization problem, each time one wants to find a new column to improve the current solution, this column must have necessarily a negative reduced cost. This is what the pricing subproblem must search for: columns representing routes with negative reduced costs regarding the current solution of the restricted master problem.

Given the dual variables γ and β_i, associated with the constraints (III.2) and (III.3), we can define the reduced cost of a route r as follows.

$$\bar{c}_r = c_r - \gamma - \sum_{i \in \mathcal{C}} a_r^c \beta_i. \tag{V.1}$$

Therefore, the value of the reduced cost for a route is its actual cost minus the dual variable γ and the dual variable β for each customer it visits. This equation by itself is not very interesting for the pricing algorithm, since one needs to know the reduced cost of each edge. With this in mind, this equation must be rewritten as a function of the edges. There are some different ways to handle this issue and our choice is described below.

In the case of the CARP, where each customer is an edge, the dual variable γ is associated with the constraint (III.27) and the dual variables β_e are associated with the constraints (III.28). The reduced cost of an edge e is as follows.

$$\bar{c}_e = \begin{cases} c_e - \beta_e & \text{if } e \notin \delta(\{0\}) \\ c_e - \left(\beta_e + \frac{\gamma}{2}\right) & \text{if } e \in \delta(\{0\}) \end{cases}. \tag{V.2}$$

Moreover, if each customer is a vertex (CVRP) or a cluster (GVRP), the dual variables γ and β_i are associated to constraints (III.52) and (III.53), respectively. In both cases, we can define the reduced cost of an edge $e = (i, j)$ as follows.

$$\bar{c}_e = \begin{cases} c_e - \left(\frac{\beta_i+\beta_j}{2}\right) & \text{if } e \notin \delta(\{0\}) \\ c_e - \left(\frac{\beta_i+\gamma}{2}\right) & \text{if } e \in \delta(\{0\}) \end{cases} . \tag{V.3}$$

Now that the reduced cost is defined for a route and for an edge, some of the existing pricing relaxations will be presented.

(a) The q-Route Pricing Relaxation

Since an optimal solution of a routing problem does not include non-elementary routes, we would want to price, ideally, only elementary routes. This would correspond to solving the Elementary Shortest Path Problem with Resource Constraints (ESPPRC) as a pricing subproblem, which is known to be strongly $\mathcal{NP}$-hard [24]. However, if we choose not to deal with this level of complexity and relax the routes, allowing them to be non-elementary, the problem now corresponds to the Shortest Path Problem with Resource Constraints (SPPRC), which is weakly $\mathcal{NP}$-hard, and there are pseudo-polynomial algorithms available in the literature, as described in the seminal work of Christofides et al. [19].

We can solve this pricing relaxation using a forward dynamic programming algorithm, which is defined as follows. Let $T(d, i)$ be the minimum reduced cost of a path that starts at the depot vertex, visits a set of customers and ends at customer i with cumulative demand of d. We create a $(Q+1) \times |\mathcal{C}|$ dynamic programming matrix, where $|\mathcal{C}|$ is the number of customers and Q the total capacity of each vehicle. At first, this matrix is entirely filled with $+\infty$. Next, the recurrence shown below is used to solve the non-elementary pricing.

$$\begin{cases} T(d_i, i) = \bar{c}_{0i} \\ T(d, i) = \min\limits_{j \in \mathcal{C}} \{T(d - d_i, j) + \bar{c}_{ji}\} \end{cases} . \tag{V.4}$$

Notice that what we call $\bar{c}_{ji}$ is the reduced cost between a pair of customers. Hence, depending on the problem tackled, it can be a single edge (CVRP and GVRP) or a sequence of edges (CARP). Furthermore, it is easy to verify that the complexity of this algorithm is $\mathcal{O}(|\mathcal{C}|^2 Q)$.

We just presented what we are calling an unrestricted non-elementary route pricing, since this algorithm does not try to forbid any kind of cycle of customers which could possibly appear in its solution. In the work of

Christofides et al. [19], where this algorithm is presented, a label-setting extension is also described, which forbids non-elementary routes with 2-cycles $(i - j - i)$. Although it is a simple extension, which does not change the complexity of this algorithm, it can improve the lower bounds significantly. Furthermore, it is natural to infer that the greater is the length of the cycle prohibited by the algorithm, the better is the improvement on the lower bound. With this in mind, the work of Irnich and Villeneuve [45] presents the general case for this algorithm, where given an input parameter s, it returns only non-elementary routes without s-cycles. As expected, the complexity of this algorithm is factorial and, for this reason, it can only be used for small values of s, as shown by Fukasawa et al. [31] for the CVRP.

(b) The ng-Route Pricing Relaxation

Recently, the ng-route relaxation was introduced in the work of Baldacci et al. [5] for the CVRP and the CVRP with Time Windows (CVRPTW), and it was later extended to the GVRP by Bartolini et al. [7], where it was used to solve transformed CARP instances. This new relaxation aims at having a better compromise between efficiently pricing non-elementary routes and obtaining good lower bounds.

For each customer $i \in \mathcal{C}$, let $N_i \subseteq \mathcal{C}$ be a subset of customers which have a relationship with i. A possible representation for this relationship can be a neighborhood relationship, i.e., N_i contains the nearest customers of i, including i. These are the ng-sets and they work like a memory for each customer. For instance, when a route is being built, by the time it arrives at a customer i, it only remembers visiting a customer j before if j belongs to the ng-set of i, N_i. Thus, if the route visits a given customer i at some point, this customer can only be visited again – forming a cycle – if another customer which "forgets" customer i is visited before. At this point, we can conclude that the size of the ng-sets is an important factor on the quality of solutions, because the larger the ng-sets, the larger will be the smallest cycles which can appear in a route. The size of each set N_i is limited by $\Delta(N_i)$, which is a parameter defined *a priori*. Obviously, this size also changes the pricing complexity, as we will show further.

Let $P = (0, \ldots, i_k, \ldots, i_p)$ be a path starting at the depot, visiting customer i_k and ending at customer i_p, and $\mathcal{C}(P)$ be the list of customers visited by path P. We can define a function $\Pi(P)$ of prohibited extensions (the "memory") of path P as follows.

$$\Pi(P) = \left\{ i_k \in \mathcal{C}(P) : i_k \in \bigcap_{s=k+1}^{p} N_{i_s}, k = 1, \ldots, p-1 \right\} \cup \{i_p\}\,. \qquad \text{(V.5)}$$

We present two examples in Figures V.1 and V.2 to illustrate the update procedure of function Π. For both examples there are three customers and the depot. The path being built starts at the depot, goes through customers 1, 2 and 3 and then tries to go back to customer 1. In the first example, shown in Figure V.1, the ng-sets for each customer are $N_1 = \{1, 2\}$, $N_2 = \{2, 1\}$ and $N_3 = \{3, 1\}$. Notice that after the extension made in Figure V.1(d), customer 1 is still present in the "memory" of the path. Therefore the extension of Figure V.1(e) is not allowed.

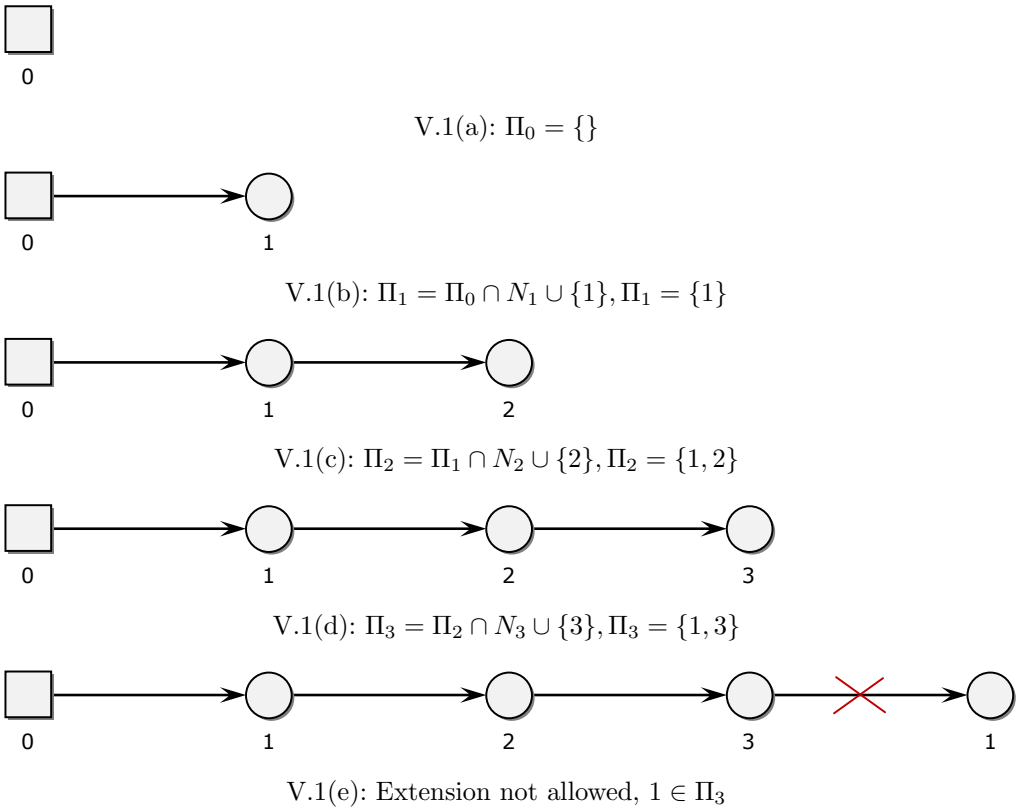


Figure V.1: Example of ng-route for $N_1 = \{1, 2\}$, $N_2 = \{2, 1\}$ and $N_3 = \{3, 1\}$.

On the other hand, in the second example, shown in Figure V.2, the ng-sets for each customer are now $N_1 = \{1, 2\}$, $N_2 = \{2, 1\}$ and $N_3 = \{3, 2\}$. The change on N_3 is enough for the path to "forget" customer 1 after the extension made in Figure V.2(d). Hence, in this case, the extension to customer 1 is now allowed, as shown in Figure V.2(e).

Given $d(P) = \sum_{i \in \mathcal{C}(P)} d_i$ as the total demand serviced by path P and $\bar{c}(P)$ as the total reduced cost of path P regarding equations (V.2) and (V.3), let $\mathcal{L}(P) = (i_p, d(P), \Pi(P), \bar{c}(P))$ be a label associated with a path P, which

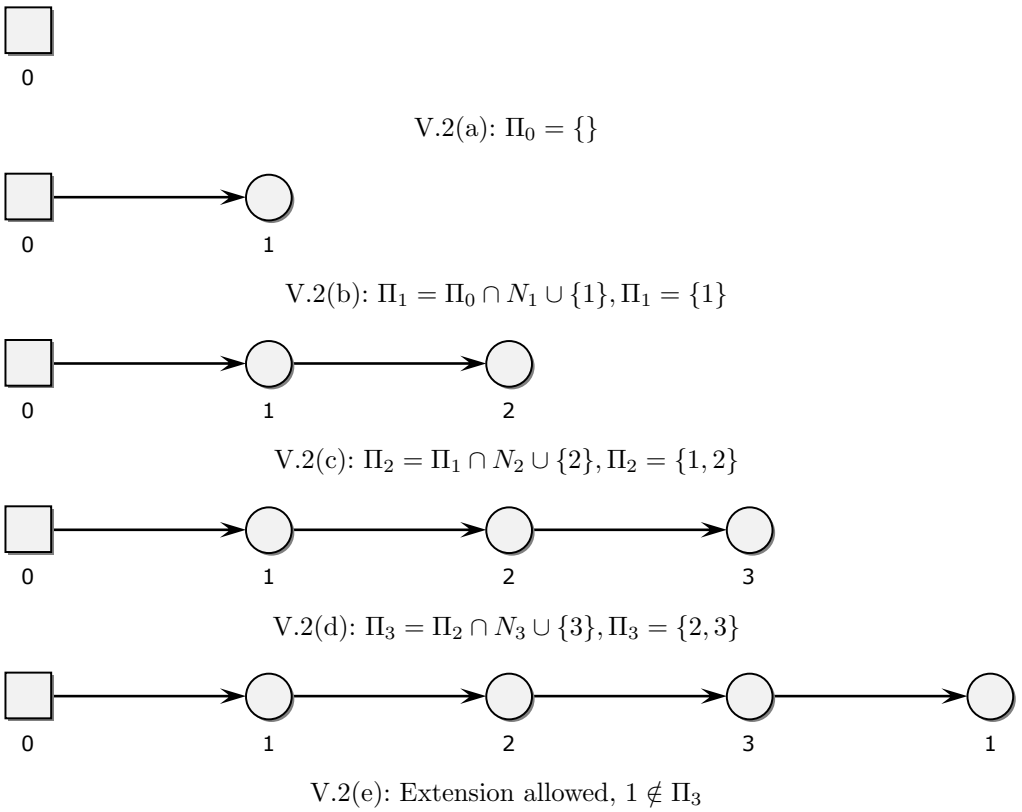


Figure V.2: Example of ng-route for $N_1 = \{1, 2\}$, $N_2 = \{2, 1\}$ and $N_3 = \{3, 2\}$.

ends at customer i_p, with total demand $d(P)$, prohibited extensions $\Pi(P)$, and total reduced cost $\bar{c}(P)$. We say that a label $\mathcal{L}(P)$ can be extended to a customer i_{p+1} if $i_{p+1} \notin \Pi(P)$ and $d(P) + d_{i_{p+1}} \leq Q$. After the extension, the customer i_{p+1} becomes the last customer of a new ng-path $P' = (0, \ldots, i_p, i_{p+1})$ and a new label $\mathcal{L}(P')$ can be obtained from the label $\mathcal{L}(P)$ by the following operations:

$$\mathcal{L}(P') = (i_{p+1}, d(P) + d_{i_{p+1}}, \Pi(P) \cap N_{i_{p+1}} \cup \{i_{p+1}\}, \bar{c}(P) + \bar{c}_{i_p i_{p+1}}). \quad \text{(V.6)}$$

These labels are computed using a forward dynamic programming algorithm which, in contrast to the q-route pricing, does not result in a pseudo-polynomial complexity. This pricing is exponential on the size of the subsets $\Delta(N_i)$ and its efficiency depends on the use of some techniques to speed up its execution.

In order to reduce the number of possible paths, a dominance rule is incorporated to the algorithm. Given the labels of two paths $\mathcal{L}(P_1) = (i_p, d(P_1), \Pi(P_1), \bar{c}(P_1))$ and $\mathcal{L}(P_2) = (i_p, d(P_2), \Pi(P_2), \bar{c}(P_2))$, path P_1 dominates path P_2 if and only if any possible extension from P_2 can be done from P_1 with a lower total reduced cost. For this to be true, the following three conditions must hold:

(i) $d(P_1) \leq d(P_2)$,

(ii) $\bar{c}(P_1) \leq \bar{c}(P_2)$ and

(iii) $\Pi(P_1) \subseteq \Pi(P_2)$.

In addition, Baldacci et al. [5] described another way to improve this dominance rule. When paths with a given capacity d are being computed, the best reduced costs for every $d' < d$ are stored in a dominance list, which is faster to check than to iterate through the dynamic programming matrix for each $d' < d$. We do not use this technique because it is not scalable, since the size of this dominance list is exponential in the value of $\Delta(N_i)$, reaching its size limit when $\Delta(N_i) \approx 13$.

(c) An Efficient Exact ng-Route Pricing Implementation

As discussed earlier, the basic ng-route pricing implementation does not allow the use of large ng-sets, which weakens the quality of the lower bounds found. To address this issue, we provide an efficient pricing implementation, adapting the Decremental State Space Relaxation (DSSR) for the ng-route pricing. This technique was introduced by Righini and Salani [72] to solve the ESPPRC. The original version of the algorithm helps reducing the number of labels to be managed during the dynamic programming algorithm which prices elementary routes. Firstly, it relaxes the elementarity of the routes and, at each iteration, identifies which customers are being repeated on the best routes found and then prohibits the repetition of these customers in subsequent iterations.

The main difference of our pricing is that instead of relaxing the elementarity of the routes, the new pricing relaxes the ng-set of each customer, therefore relaxing the ng-route restrictions. The algorithm is outlined as follows.

Basic Implementation

We start by creating a $(Q+1) \times |\mathcal{C}|$ dynamic programming matrix $\mathcal{M}$, where each entry $\mathcal{M}(d, i)$ is a bucket containing labels representing ng-paths which start at the depot and end at customer i with total demand exactly d. At first, we set $\mathcal{M}(d_i, i)$ with a single label $\mathcal{L}_i = (i, d_i, \{i\}, \bar{c}_{0i})$, $\forall i \in \mathcal{C}$, and all other entries with no label. Next, a forward dynamic programming is used to fill the matrix $\mathcal{M}$, running from $d = 1$ up to $d = Q$.

When processing the bucket $\mathcal{M}(d, i)$, the algorithm passes through all labels $\mathcal{L}(P')$ belonging to $\mathcal{M}(d - d_i, j)$, for all customers $j \in \mathcal{C}$, such that

$d - d_i \geq 0$. As the basic ng-route pricing algorithm, the extension from $\mathcal{L}(P')$ to i can only be done if $i \notin \Pi(P')$. If this condition holds, a new label $\mathcal{L}(P)$ is then created and it must be stored in the bucket $\mathcal{M}(d, i)$. Therefore, this is the right time to check for the dominance rule, which can be verified for all the labels in $\mathcal{M}(d', i), \forall d' \leq d$. This is the place where the dominance list mentioned earlier is used in the work of Baldacci et al. [5]. Surprisingly, for large values of $\Delta(N_i)$, we have found that the algorithm runs faster if the dominance rule is tested only for labels of the same bucket, i.e., for the labels from inside $\mathcal{M}(d, i)$.

This whole procedure is shown in Algorithm V.1, where the existence of a procedure called *buildRoutes* is taken into account. This procedure iterates over each bucket of the matrix $\mathcal{M}$, checking if the best path of the bucket results in a route with negative reduced cost when it goes back to the depot.

Algorithm V.1 The Basic Dynamic Programming Algorithm

```
1: procedure NG-ROUTEPRICING(M, N)
2:     input: matrix M and ng-sets N_i ⊆ C, ∀i ∈ C
3:     output: the best ng-routes with respect to ng-sets N_i

4:     M(d, i) ← ∅, ∀i ∈ C, d ∈ {0, ..., Q}
5:     M(d_i, i) ← {(i, d_i, {i}, c̄_0i)}, ∀i ∈ C
6:     for d := 1, ..., Q do
7:         for all i ∈ C do
8:             if d − d_i > 0 then
9:                 for all j ∈ C do
10:                    for all L(P') ∈ M(d − d_i, j) do
11:                        if i ∉ Π(P') then
12:                            L(P) ← (i, d, Π(P') ∩ N_i ∪ {i}, c̄(P') + c̄_ji)
13:                            insertLabel ← true
14:                            for all L(P'') ∈ M(d, i) do
15:                                if L(P) dominates L(P'') then
16:                                    delete L(P'')
17:                                else if L(P'') dominates L(P) then
18:                                    insertLabel ← false
19:                                    break
20:                            if insertLabel then
21:                                M(d, i) ← M(d, i) ∪ L(P)
22:    return buildRoutes(M)
```

Decremental State-Space Relaxation

The adapted DSSR is an iterative algorithm and it works by relaxing the state space of the original subsets N_i. At each iteration k, the algorithm uses the subsets $\Gamma_i^k \subseteq N_i$ as a replacement for the N_i subsets. These subsets Γ_i^k take the role of N_i in the definition of the function Π, described in (V.5), and in

the creation of new labels, shown in (V.6). Initially, the algorithm sets $\Gamma_i^0 = \emptyset$, $\forall i \in \mathcal{C}$, and executes the complete dynamic programming. As the best routes found by this dynamic programming are not necessarily ng-routes, they cannot be considered as the result of the pricing without verifying their feasibility. This test is performed during the update of the Γ_i^k subsets, as described as follows.

Let a cycle of customers be defined as a sub-path $H = (i, \ldots, j)$, where $i = j$, and let $\mathscr{H}(P)$ be the set of all cycles of customers in the path P. In order to evaluate if the best route R_k^* is an ng-route, the algorithm must check if there is no cycle of customers $H \in \mathscr{H}(R_k^*)$ which would not be allowed to be created if the original subsets N_i were being used. This happens only when the customer i (which creates the cycle H) is in all N_l, $\forall l \in \mathcal{C}(H)$, i.e., the customer i is in the "memory" of every other customer of the cycle. If any such cycle H is found, the subsets Γ_l^{k+1}, which are initialized as $\Gamma_l^{k+1} = \Gamma_l^k$, are then updated as $\Gamma_l^{k+1} = \Gamma_l^{k+1} \cup \{i\}$, $\forall l \in \mathcal{C}(H)$. If all cycles in the set $\mathscr{H}(R_k^*)$ respect the subsets N_i, no update is done and the subsets Γ_i^{k+1} and Γ_i^k are the same for all $i \in \mathcal{C}$. In this case, we can conclude that R_k^* is an ng-route and the pricing stops. On the other hand, if any cycle $H \in \mathscr{H}(R_k^*)$ does not respect the subsets N_i, there is at least one Γ_i^{k+1} larger than Γ_i^k and, for this reason, a new iteration of the algorithm is started.

Algorithm V.2 shows the DSSR procedure. In each iteration, it calls the basic ng-route pricing algorithm *ng-RoutePricing*, shown in Algorithm V.1, and also considers the existence of three other procedures, *selectBestRoute*, which returns the route with lowest reduced cost, *isNGRoute*, which tests if the given route is indeed an ng-route with respect to the original ng-sets and *updateNGSets*, which update the sets Γ in order to force the best route to be an ng-route.

Algorithm V.2 The DSSR Algorithm

```
1: procedure DSSRPRICING(M, N)
2:     input: matrix M and ng-sets N_i ⊆ C, ∀i ∈ C
3:     output: the best ng-routes with respect to ng-sets N_i

4:     Γ_i ← ∅, ∀i ∈ C, ng ← false, k ← 0
5:     while not ng do
6:         R ← ng-RoutePricing(M, Γ)
7:         R*_k ← selectBestRoute(R)
8:         if isNGRoute(R*_k, N) then
9:             ng ← true
10:        else
11:            updateNGSets(R*_k, N, Γ)
12:        k ← k + 1
13:    return R*_k
```

It is noteworthy to mention that if the best route found is indeed an ng-route, the procedure can stop and return only this route. This is how Algorithm V.2 is implemented. However, it can also return this route together with any other route with negative reduced cost and certified as being an ng-route. Furthermore, if the best route is not an ng-route, but there exists at least one route with negative reduced cost which is an ng-route, the algorithm can stop and return these ng-routes found. In this case, we consider it as being a heuristic run of the algorithm, not an exact one. It is not a problem for the column generation, except in cases where the reduced cost of the best route is necessary for further calculations.

Completion Bounds

In order to further speed up the DSSR algorithm, at some iteration k the completion bounds are calculated for each customer i with every capacity d. As mentioned before, the completion bounds are used to estimate a lower bound on the value of a route during its creation, thus discarding any route which would not lead to a negative reduced cost. Given $T_k^*(d,i)$, the best path which starts at customer i and ends at the depot with total capacity exactly d, the completion bounds $\widehat{T}(d,i)$ are calculated as shown in (V.7) and represent the best path which starts at customer i and ends at the depot with total capacity less than or equal d.

$$\widehat{T}(d,i) = \min_{d' \leq d} \left\{ T_k^*(d',i) \right\}. \tag{V.7}$$

It is important to observe that if the corresponding problem is represented by means of an undirected graph, $T_k^*(d,i)$ can be obtained directly from the dynamic programming matrix. On the other hand, if the problem is represented using a directed graph, in order to obtain these values, the direction of the edges has to be reversed and the last iteration of the DSSR algorithm has to be executed again. This occurs because when a route is traversed in the opposite direction on an asymmetric graph, it does not generate the same cost.

Considering a problem represented by an undirected graph, the procedure which builds the completion bounds is shown in Algorithm V.3. Notice that this procedure can be called anywhere after calling the *ng-RoutePricing* procedure on line 6 of algorithm V.2.

After calculating the completion bounds, at a subsequent iteration $k' > k$ of the DSSR, they can be used to avoid the extension of a given label $\mathcal{L}(P) = (i_p, d(P), \Pi(P), \bar{c}(P))$ to a customer i_{p+1} if $\bar{c}(P) + \bar{c}_{i_p i_{p+1}} + \widehat{T}(Q - d(P), i_{p+1}) \geq 0$. This equation calculates a lower bound on the value of the reduced cost of

Algorithm V.3 The Completion Bounds Generation

```
1: procedure generateCompletionBounds(M)
2:     input: matrix M
3:     output: completion bounds T̂

4:     T̂(d, i) ← ∞, ∀i ∈ C, d ∈ {0, ..., Q}
5:     T̂(0, 0) ← 0
6:     for all i ∈ C do
7:         for d := 1, ..., Q do
8:             T̂(d, i) ← min(M(d, i))
9:             if T̂(d − 1, i) < T̂(d, i) then
10:                T̂(d, i) ← T̂(d − 1, i)
```

any route the label can generate. Obviously, if this value is greater or equal than zero, the label cannot generate any route with a negative reduced cost, therefore it can be discarded. This test can be simply included in Algorithm V.1 together with the one in line 11.

Heuristic Pricing

Even with the improvements described in the last section, the exact ng-route pricing still takes a long time to be executed. In the view of this, a simple but effective heuristic is developed in order to quickly price a large initial set of routes with negative reduced cost. It was inspired on the heuristic pricing done for the elementary route pricing suggested by [67]. The purpose of this heuristic is to reduce the number of calls to the exact ng-route pricing. Therefore, the heuristic ng-route pricing is used as a hot-start for the exact ng-route pricing.

The heuristic closely resembles the q-route pricing without eliminating any cycle. The main difference between the pricing algorithms is that when extending one path, the heuristic ng-route pricing respects the subsets N_i. Its data structure is also an $(Q+1) \times |\mathcal{C}|$ matrix and each entry consists of just one label. For each customer and each capacity, this label is chosen as the best one with respect to the reduced costs. Also, as the subsets N_i must be respected, each label of the dynamic programming matrix must contain the Π sets for each customer and capacity.

Notice that unlike the exact algorithm, the heuristic algorithm uses neither the dominance rules nor the speed up techniques (DSSR and completion bounds). It is straightforward to verify that the resulting complexity of this algorithm is $\mathcal{O}(n^2Q)$.

V.2 Experimental Results

For the computational experiments, all algorithms were implemented using the same configuration as described in Chapter IV. The tests compare the bounds and performance of the pricing algorithms with a variety of different configurations. The column generation starts by calling the heuristic pricing at each iteration. The heuristic pricing returns the best 20 routes with negative reduced costs. If the heuristic pricing is no longer capable of finding routes with negative reduced cost, the column generation algorithm calls the exact pricing. If the later succeeds in obtaining at least one route with negative reduced cost, the column generation procedure restarts by calling the heuristic pricing. Otherwise, the column generation stops and the current value is returned as a lower bound. The exact pricing uses the DSSR approach and needs to calculate the completion bounds at some iteration in order to speed up the DSSR. Given $\bar{c}(R_k^*)$, the reduced cost of the best route at the kth iteration of the DSSR, the exact pricing calculates completion bounds when $\bar{c}(R_k^*) > \bar{c}(R_0^*)/3$.

The results for the column generation algorithm are shown in Tables V.1-V.4. At each table, columns `Ins`, `LB` and `UB` show the name, the lower bound and the upper bound of each instance. Following these columns, the results for different values of $\Delta(N_i)$ are shown. For each X, where $\Delta(N_i) = X$, `NG=X` consists of four columns, `Value`, `Routes`, `Cuts` and `Time`, which show the value found, the number of routes, the number of cuts and the total time of each instance.

For the CARP, the algorithms were applied to the same instance datasets described in Chapter IV, except for the *egl-large* dataset. The instances of this dataset are prohibitively large for column generation. Furthermore, as a large set of cuts is known for each instance due to what was done in the previous chapter, we decided to include all the cuts found by the dual ascent heuristic and the exact separations to the master problem prior to the execution of the column generation. During the column generation, before the exact pricing is called, the exact separation of the odd-degree cutset inequalities is called and, if any violated cut is found, this cut is inserted in the master problem and the column generation restarts. The lower and upper bounds shown in Table V.1 were taken from the works of Martinelli et al. [59], Bode and Irnich [16] and Bartolini et al. [7].

For the GVRP, we applied our algorithms to the instance datasets recently generated by Bektaş et al. [8]. These instance datasets are derived from the CVRP instance datasets `A`, `B`, `P` and `M`. The transformation is performed using a method similar to that of Fischetti et al. [29], which transforms TSP

instances into GTSP instances. The number of clusters is $t = \lceil n/\theta \rceil$, where θ is a parameter defined *a priori*. For each original CVRP instance dataset, two new instance datasets are created, using $\theta = 2$ and $\theta = 3$, resulting in 158 GVRP instances. All lower and upper bounds shown in Tables V.2 and V.3 were taken from the work of Bektaş et al. [8]. Analogously to the CARP, during the column generation, before the exact pricing is called, the capacity cuts are separated using the package CVRPSEP [56] and, if any violated cut is found, this cut is inserted in the master problem and the column generation restarts.

For the CVRP, we used the classical instance datasets `A`, `B`, `E`, `P` and `M`, available at `www.branchandcut.org` [71]. All the lower and upper bounds shown in Table V.4 were taken from the work of Baldacci et al. [4], except the upper bounds for instances `M-n200-k16` and `M-n265-k25`, which were found by running the hybrid algorithm proposed by Subramanian et al. [74].

Notice that the lower bounds obtained with `NG=32` and `NG=64` are identical for almost all instances and, in fact, these values are very close to those obtained by a column generation with an elementary route pricing algorithm, such as the results presented in [68] for the CVRP, which were found by running the algorithm developed by Pecin [67]. This is an empirical evidence that the routes found by the pricing with large ng-sets (greater than 32) are almost elementary when the average size is up to 12 or 13 customers. Moreover, the runtime is typically much smaller than that required if the elementarity constraint is imposed on the routes.

Our algorithm was not capable of solving some instances for large ng-sets within the time limit of three hours. This is probably due to the large average size of the routes that are part of an optimal solution, causing the number of labels to be treated by the dynamic programming algorithm to be prohibitive. In the work of Pecin [67], the results of elementary route pricing for classical CVRP instances with up to 101 vertices are shown, but none for larger instances. In contrast, our ng-route pricing algorithm allowed for calculating excellent lower bounds for instances greater than 100 vertices, such as `M-n151-k12`, `M-n200-k16`, and `M-n200-k17`, in which the best known lower bounds were found using complicated state-of-the-art column and cut generation algorithms.

Table V.1: Column Generation Results for the CARP

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
kshs1	14661	14661	14661.0	116	5	0.140	14661.0	92	5	0.094	14661.0	92	5	0.093	14661.0	92	5	0.094
kshs2	9863	9863	9863.0	124	8	0.125	9863.0	88	8	0.093	9863.0	88	8	0.078	9863.0	88	8	0.094
kshs3	9320	9320	9320.0	167	4	0.125	9320.0	93	4	0.109	9320.0	93	4	0.094	9320.0	93	4	0.110
kshs4	11498	11498	11498.0	134	5	0.156	11498.0	99	5	0.140	11498.0	99	5	0.125	11498.0	99	5	0.156
kshs5	10957	10957	10957.0	110	4	0.140	10957.0	112	4	0.141	10957.0	112	4	0.141	10957.0	112	4	0.219
kshs6	10197	10197	10197.0	234	4	0.172	10197.0	123	4	0.110	10197.0	123	4	0.110	10197.0	123	4	0.140
gdb1	316	316	316.0	259	12	0.015	316.0	235	12	0.016	316.0	201	12	0.016	316.0	201	12	0.016
gdb2	339	339	339.0	260	8	0.031	339.0	303	8	0.031	339.0	290	8	0.031	339.0	290	8	0.031
gdb3	275	275	275.0	217	5	0.016	275.0	200	5	0.016	275.0	191	5	0.016	275.0	191	5	0.016
gdb4	287	287	287.0	126	6	0.000	287.0	86	6	0.000	287.0	86	6	0.015	287.0	86	6	0.015
gdb5	377	377	377.0	228	8	0.031	377.0	200	8	0.016	377.0	197	8	0.031	377.0	197	8	0.016
gdb6	298	298	298.0	158	5	0.016	298.0	160	4	0.015	298.0	143	4	0.016	298.0	143	4	0.016
gdb7	325	325	325.0	219	10	0.016	325.0	147	10	0.016	325.0	192	10	0.016	325.0	192	10	0.016
gdb8	348	348	346.2	623	20	0.344	346.2	518	20	0.297	346.2	498	20	0.360	346.2	470	20	0.360
gdb9	303	303	303.0	628	18	0.438	303.0	609	17	0.422	303.0	649	17	0.453	303.0	468	17	0.375
gdb10	275	275	275.0	245	12	0.031	275.0	228	12	0.031	275.0	237	13	0.031	275.0	237	13	0.047
gdb11	395	395	395.0	1105	13	0.953	395.0	925	13	0.781	395.0	1105	13	0.969	395.0	645	13	0.703
gdb12	458	458	452.7	242	11	0.062	454.0	221	11	0.063	454.0	195	11	0.078	454.0	195	11	0.093
gdb13	536	536	536.0	412	5	0.157	536.0	400	5	0.156	536.0	291	5	0.125	536.0	291	5	0.141
gdb14	100	100	100.0	151	1	0.031	100.0	165	1	0.016	100.0	135	1	0.016	100.0	135	1	0.031
gdb15	58	58	58.0	112	1	0.063	58.0	130	1	0.031	58.0	93	1	0.031	58.0	93	1	0.046
gdb16	127	127	127.0	378	6	0.078	127.0	322	6	0.078	127.0	228	7	0.063	127.0	228	7	0.078
gdb17	91	91	91.0	134	7	0.046	91.0	134	7	0.047	91.0	74	7	0.031	91.0	74	7	0.047
gdb18	164	164	164.0	565	1	0.235	164.0	685	1	0.296	164.0	385	1	0.156	164.0	365	1	0.188
gdb19	55	55	55.0	55	4	0.016	55.0	60	4	0.000	55.0	60	4	0.015	55.0	60	4	0.016
gdb20	121	121	121.0	265	9	0.062	121.0	205	10	0.062	121.0	187	10	0.047	121.0	187	10	0.063

Continued on next page

Table V.1: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
gdb21	156	156	156.0	414	7	0.141	156.0	334	6	0.094	156.0	294	7	0.094	156.0	254	7	0.110
gdb22	200	200	200.0	352	7	0.156	200.0	332	6	0.156	200.0	292	5	0.125	200.0	252	6	0.171
gdb23	233	233	233.0	372	1	0.219	233.0	372	1	0.219	233.0	332	1	0.203	233.0	272	1	0.219
1A	173	173	173.0	2715	20	6.531	173.0	2704	20	7.610	173.0	2380	20	7.500	173.0	1396	20	5.750
1B	173	173	173.0	1762	20	2.937	173.0	1410	20	2.719	173.0	1483	20	2.906	173.0	1277	20	2.813
1C	245	245	241.3	483	18	0.406	241.3	603	18	0.547	241.3	425	18	0.531	241.3	373	18	0.656
2A	227	227	227.0	2107	12	3.906	227.0	2138	11	5.016	227.0	1250	11	6.047	227.0	1323	11	6.641
2B	259	259	257.5	2030	25	3.500	258.0	2021	25	5.063	258.0	1708	25	8.188	258.0	1559	25	35.95
2C	457	457	457.0	406	16	0.328	457.0	398	16	0.297	457.0	345	16	0.266	457.0	372	16	0.406
3A	81	81	81.0	2371	25	3.109	81.0	2555	23	4.016	81.0	2085	24	4.422	81.0	1378	29	2.891
3B	87	87	87.0	1740	23	1.562	87.0	1782	26	1.766	87.0	1378	26	1.391	87.0	1363	27	1.563
3C	138	138	136.0	462	11	0.156	136.0	473	12	0.172	136.0	447	12	0.172	136.0	495	13	0.219
4A	400	400	400.0	8603	37	86.44	400.0	9467	37	113.3	400.0	10767	37	192.8	400.0	8227	36	171.1
4B	412	412	412.0	4116	34	22.05	412.0	5396	36	37.22	412.0	6076	34	52.02	412.0	3936	34	32.20
4C	428	428	428.0	3105	33	13.33	428.0	3705	33	17.19	428.0	3065	33	14.41	428.0	3085	33	17.22
4D	530	530	524.9	2103	50	6.329	525.3	2037	51	6.297	525.4	1816	52	6.766	525.4	1813	52	15.33
5A	423	423	423.0	4147	25	24.13	423.0	5187	26	35.42	423.0	4827	26	37.08	423.0	3847	26	37.49
5B	446	446	444.2	3629	44	18.19	444.2	3848	41	22.63	444.2	4163	40	29.77	444.2	2703	40	39.89
5C	474	474	469.3	2716	42	11.34	469.5	3068	46	14.67	469.5	2805	39	15.20	469.5	2551	44	28.47
5D	577	577	572.1	1433	27	3.672	572.3	1233	19	3.735	572.8	1456	63	6.063	572.8	1450	58	11.80
6A	223	223	223.0	2603	20	7.531	223.0	2820	21	8.891	223.0	2650	21	8.813	223.0	2587	20	42.49
6B	233	233	229.0	1948	22	4.234	229.8	2168	75	6.344	229.8	2076	58	7.250	229.8	1919	65	8.938
6C	317	317	310.8	610	22	0.765	311.1	506	23	0.813	311.1	593	23	1.016	311.1	522	22	1.110
7A	279	279	279.0	4507	23	23.22	279.0	4019	21	24.31	279.0	5376	21	40.17	279.0	3619	21	34.83
7B	283	283	283.0	3616	21	14.11	283.0	3576	21	17.49	283.0	3815	21	19.36	283.0	3135	21	25.95
7C	334	334	328.5	1744	43	7.656	328.9	1655	53	6.625	328.9	1609	52	6.078	328.9	1593	60	18.05
8A	386	386	386.0	3167	23	15.57	386.0	3127	23	16.44	386.0	3567	23	21.36	386.0	2147	23	15.81

Continued on next page

Table V.1: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
8B	395	395	395.0	2496	24	9.610	395.0	3116	21	13.05	395.0	3216	22	14.92	395.0	1916	21	9.891
8C	521	521	517.2	1049	46	2.687	517.2	1054	46	2.688	517.2	1057	46	2.766	517.2	938	46	3.328
9A	323	323	323.0	9427	54	140.0	323.0	11226	54	213.3	323.0	10767	53	242.8	323.0	11467	54	395.7
9B	326	326	326.0	9556	49	127.5	326.0	10656	49	193.1	326.0	16016	50	638.9	326.0	18216	49	1626
9C	332	332	332.0	5325	44	43.17	332.0	6665	44	66.33	332.0	6025	45	61.63	332.0	6525	44	91.94
9D	391	391	383.9	3328	79	15.80	383.9	3318	94	20.71	383.9	3276	85	17.86	383.9	3283	112	27.11
10A	428	428	428.0	10329	44	212.9	428.0	15513	44	577.4	428.0	20027	44	1392	428.0	23827	44	3103
10B	436	436	436.0	7216	59	108.9	436.0	7696	59	140.6	436.0	8396	58	179.5	436.0	8876	58	234.4
10C	446	446	446.0	6485	52	83.19	446.0	6405	51	85.28	446.0	7465	50	115.8	446.0	7665	51	151.9
10D	526	526	524.6	2639	60	15.89	524.6	2521	50	15.30	524.6	2485	60	16.86	524.6	2395	58	18.02
C01	4105	4150	4094.5	3646	101	15.61	4100.3	3376	95	23.55	4101.3	3369	87	19.53	4101.3	2820	94	23.27
C02	3135	3135	3135.0	2081	32	3.469	3135.0	2013	32	4.094	3135.0	1853	32	3.578	3135.0	1640	32	4.281
C03	2575	2575	2548.7	2426	54	5.390	2548.7	2429	55	6.594	2548.8	1927	59	4.016	2548.8	1757	51	4.813
C04	3478	3510	3474.7	4164	57	13.78	3475.1	4149	62	17.69	3476.6	4413	72	22.53	3476.6	3224	62	47.99
C05	5365	5365	5319.9	1956	64	4.844	5323.3	1897	57	5.328	5323.4	1963	57	5.953	5323.5	1756	57	6.110
C06	2535	2535	2515.9	2720	56	5.219	2518.8	2426	63	8.016	2519.8	2514	63	11.94	2519.8	2092	56	93.89
C07	4075	4075	4022.1	1975	54	3.469	4022.5	2008	54	3.875	4022.5	1843	54	4.485	4022.5	1970	54	5.156
C08	4090	4090	4039.5	2262	91	6.313	4039.5	2198	81	6.406	4039.5	2001	74	5.938	4039.5	1636	76	7.891
C09	5233	5260	5222.5	4905	69	27.39	5224.3	3873	73	25.69	5224.8	3862	78	26.78	5224.8	3642	82	33.52
C10	4700	4700	4620.7	1410	69	3.031	4623.8	1398	65	4.016	4624.0	1216	65	3.219	4624.0	1109	65	3.422
C11	4583	4635	4573.7	5676	158	34.33	4574.2	5894	154	40.50	4575.0	5402	187	41.27	4575.0	5659	190	61.88
C12	4209	4240	4172.7	3392	72	11.59	4173.6	4067	76	15.13	4173.8	3558	71	14.49	4173.8	2773	71	14.84
C13	2955	2955	2911.0	1735	40	2.953	2911.0	1526	40	2.937	2911.0	1543	40	3.485	2911.0	1304	40	8.641
C14	4030	4030	3990.7	1799	35	3.219	3990.7	1985	35	4.188	3990.7	1778	37	4.719	3990.7	1439	35	4.922
C15	4912	4940	4890.2	6512	92	43.34	4892.3	6846	95	58.41	4892.7	7024	86	66.98	4892.7	6083	94	83.84
C16	1475	1475	1470.0	2201	25	3.594	1470.0	1559	28	4.297	1470.0	1256	25	2934	1470.0	1216	25	21.67
C17	3555	3555	3550.0	1444	44	1.922	3550.0	1245	44	1.813	3550.0	1032	44	1.672	3550.0	966	44	2.375

Continued on next page

Table V.1: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
C18	5577	5620	5557.9	8764	115	97.83	5559.6	9199	115	134.8	5559.6	9188	115	153.3	—	—	—	—
C19	3096	3115	3082.0	2791	80	7.937	3084.2	2568	77	16.34	3091.5	2467	103	32.00	3091.5	2144	81	223.0
C20	2120	2120	2120.0	4452	59	10.16	2120.0	3683	53	7.735	2120.0	3223	38	7.875	2120.0	2670	38	71.25
C21	3960	3970	3956.2	5333	50	18.81	3956.3	4448	50	16.92	3959.0	4693	53	23.49	3959.0	4470	54	36.86
C22	2245	2245	2245.0	2420	36	2.937	2245.0	2086	36	3.515	2245.0	1673	36	3.844	2245.0	2220	36	13.00
C23	4032	4085	**4035.9**	6725	134	50.80	**4042.8**	6287	133	225.6	**4044.4**	7200	135	4943	**4046.2**	6060	136	6000
C24	3384	3400	3379.4	4573	83	20.28	3379.4	4908	53	22.72	3381.3	5886	64	41.13	3381.2	4225	61	50.02
C25	2310	2310	2310.0	1406	28	1.313	2310.0	1584	28	1.610	2310.0	970	28	1.047	2310.0	1042	28	1.656
D01	3215	3215	3215.0	8421	54	67.35	3215.0	10840	53	118.5	3215.0	10856	61	134.9	3215.0	12990	54	316.2
D02	2520	2520	2520.0	3618	25	9.266	2520.0	4276	25	13.41	2520.0	4744	25	17.91	2520.0	2845	25	22.11
D03	2065	2065	2065.0	5906	31	19.06	2065.0	6161	30	23.08	2065.0	5749	31	26.06	2065.0	3228	31	22.94
D04	2785	2785	2785.0	7941	52	46.91	2785.0	9448	51	84.33	2785.0	10169	51	129.5	2785.0	7774	49	200.5
D05	3935	3935	3935.0	4870	49	22.80	3935.0	4790	49	23.66	3935.0	4870	49	26.53	3935.0	4170	49	27.88
D06	2125	2125	2125.0	5378	30	17.27	2125.0	5941	30	23.16	2125.0	6861	30	38.69	2125.0	3482	36	16.30
D07	3115	3115	3045.2	4114	52	25.03	3046.2	5164	58	41.95	3049.4	5437	60	396.6	3049.7	3187	51	938.8
D08	2995	3045	**3001.0**	7165	42	43.70	**3003.6**	7212	41	69.72	**3010.9**	7083	53	80.70	**3010.9**	5607	56	203.0
D09	4120	4120	4120.0	11462	50	121.0	4120.0	14270	50	267.5	4120.0	19502	48	698.3	4120.0	18342	50	931.4
D10	3340	3340	3331.6	5185	33	26.23	3331.7	4906	33	20.28	3331.8	4213	33	1111	—	—	—	—
D11	3745	3745	3745.0	12155	71	146.8	3745.0	11215	71	160.9	3745.0	12575	71	232.8	3745.0	10794	71	233.7
D12	3310	3310	3310.0	8855	49	61.59	3310.0	9775	49	86.56	3310.0	10688	48	121.1	3310.0	7015	48	70.17
D13	2535	2535	2535.0	4279	37	12.91	2535.0	4769	36	16.97	2535.0	3461	36	12.38	2535.0	2848	36	13.33
D14	3272	3280	3271.7	5179	46	28.95	3271.8	5510	50	42.08	—	—	—	—	—	—	—	—
D15	3990	3990	3990.0	18902	55	355.1	3990.0	20299	54	436.4	3990.0	24021	54	657.0	—	—	—	—
D16	1060	1060	1060.0	5622	19	20.44	1060.0	6662	16	132.7	—	—	—	—	—	—	—	—
D17	2620	2620	2620.0	2874	27	5.672	2620.0	2659	26	5.985	2620.0	1936	26	5.437	2620.0	1700	26	7.437
D18	4165	4165	4165.0	18322	67	392.2	4165.0	19485	66	530.6	4165.0	19922	67	658.8	4165.0	23942	67	1482
D19	2393	2400	2372.6	9522	45	54.30	—	—	—	—	—	—	—	—	—	—	—	—

Continued on next page

Table V.1: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
D20	1870	1870	1870.0	6911	33	23.36	1870.0	8744	34	58.75	1870.0	7562	34	46.17	1870.0	3979	34	1020
D21	2985	3050	2972.4	11003	60	121.4	2974.5	11880	58	124.0	2974.5	14951	38	311.2	2974.5	11555	41	287.5
D22	1865	1865	1865.0	6840	32	20.22	1865.0	9213	32	43.09	1865.0	5867	32	78.89	1865.0	3147	34	25.09
D23	3114	3130	**3116.8**	21716	97	671.3	**3116.8**	21246	99	10033	—	—	—	—	—	—	—	—
D24	2676	2710	2672.9	11907	65	105.7	2673.0	17033	89	410.8	—	—	—	—	—	—	—	—
D25	1815	1815	1815.0	3501	24	6.563	1815.0	3841	58	10.33	1815.0	3376	24	356.5	1815.0	1888	24	5.813
E01	4885	4910	4855.3	3275	73	14.19	4856.2	3788	72	19.61	4856.9	3350	74	17.50	4856.9	2886	76	38.80
E02	3990	3990	3965.4	2043	51	3.766	3965.8	1807	51	4.891	3966.0	1811	51	5.516	3966.0	1416	51	7.375
E03	2015	2015	2015.0	3253	30	6.860	2015.0	3372	30	5.563	2015.0	2839	30	6.313	2015.0	2353	32	11.77
E04	4155	4155	4132.7	4477	111	17.91	4133.6	4052	93	26.02	4133.7	4197	99	26.31	4133.7	3225	107	29.61
E05	4585	4585	4573.7	2301	78	7.000	4573.8	2308	65	5.953	4577.0	2129	65	6.531	4577.0	1785	65	6.563
E06	2055	2055	2055.0	1668	25	2.391	2055.0	1704	25	2.125	2055.0	1315	25	1.688	2055.0	1429	25	7.641
E07	4155	4155	4064.0	1146	64	2.188	4066.3	1114	66	3.000	4066.3	1031	63	4.437	4066.3	837	63	5.860
E08	4710	4710	4677.5	2219	72	4.985	4680.6	1821	73	4.328	4680.6	1681	73	4.250	4680.6	1622	73	4.891
E09	5780	5820	5773.4	5622	145	38.17	5773.9	4903	133	33.52	5774.7	5079	146	39.39	5774.7	3983	143	33.66
E10	3605	3605	3605.0	1411	58	2.094	3605.0	1486	58	3.313	3605.0	1098	58	1.969	3605.0	1294	58	2.703
E11	4637	4655	4631.1	5275	125	36.66	4632.2	4948	125	36.08	4632.6	5263	126	50.52	4632.6	5445	126	66.53
E12	4180	4180	4128.6	2576	87	6.954	4128.7	2512	87	10.00	4128.9	2761	88	8.906	4128.9	1948	86	9.453
E13	3345	3345	3312.5	1709	60	3.344	3313.0	1438	60	3.969	3313.6	1364	60	129.6	3313.6	1306	60	90.58
E14	4115	4115	4092.0	1868	61	3.579	4092.1	1755	56	3.485	4092.1	1336	55	10.88	4092.1	1365	54	12.61
E15	4189	4205	4182.0	7049	81	48.84	4184.5	7601	75	63.39	4186.6	7099	80	297.5	—	—	—	—
E16	3755	3775	**3759.8**	3205	69	6.969	**3760.1**	2619	69	7.203	**3760.1**	2810	69	7.672	**3760.1**	1808	69	16.24
E17	2740	2740	2740.0	1248	38	1.297	2740.0	1144	37	1.156	2740.0	1146	39	1.328	2740.0	890	38	1.188
E18	3825	3835	3825.0	6194	86	36.80	**3825.6**	7368	97	62.08	**3825.7**	6336	92	58.81	**3825.8**	6325	91	73.25
E19	3222	3235	3215.7	3881	109	14.03	3217.4	3735	99	14.70	3219.2	2860	98	16.08	3219.4	2606	94	81.28
E20	2802	2825	2798.9	3314	65	9.547	2798.9	2846	67	9.047	2798.9	2727	60	8.360	2798.9	2302	62	15.58
E21	3728	3730	3727.6	4189	54	12.91	3727.6	4015	54	13.95	3727.6	4019	54	16.03	3727.6	3667	57	647.8

Continued on next page

Table V.1: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
E22	2470	2470	2465.9	1880	60	6.437	2466.1	1823	56	10.06	2466.6	1636	56	43.70	2466.6	1494	56	130.1
E23	3686	3710	3684.8	5211	114	32.78	**3688.0**	5620	114	48.80	**3688.4**	5672	114	77.92	**3688.4**	4631	114	116.5
E24	4001	4020	3999.2	4386	95	20.80	**4004.8**	4566	98	24.38	**4006.4**	4789	106	30.30	**4006.4**	3731	101	25.47
E25	1615	1615	1615.0	956	20	0.578	1615.0	1155	21	1.063	1615.0	662	20	1.047	1615.0	662	20	1.875
F01	4040	4040	4040.0	7975	93	78.02	4040.0	10155	85	123.4	4040.0	10915	94	195.7	4040.0	9135	88	195.7
F02	3300	3300	3300.0	6067	49	26.00	3300.0	5381	49	24.27	3300.0	7296	49	52.39	3300.0	3718	49	20.05
F03	1665	1665	1665.0	6982	36	21.27	1665.0	7249	36	25.33	1665.0	4442	39	48.56	1665.0	2949	38	6398
F04	3476	3485	3475.8	11068	90	133.4	3476.0	9914	89	345.1	**3476.5**	10814	82	6415	—	—	—	—
F05	3605	3605	3605.0	4154	65	17.69	3605.0	5074	65	27.69	3605.0	5171	66	31.27	3605.0	3453	65	19.94
F06	1875	1875	1875.0	3149	28	6.469	1875.0	3353	28	7.860	1875.0	2812	28	7.765	1875.0	1973	28	7.906
F07	3335	3335	3335.0	2594	59	6.657	3335.0	2948	59	8.235	3335.0	2238	59	6.438	3335.0	1773	59	7.969
F08	3695	3705	3692.8	5015	69	23.53	3693.1	5542	69	31.17	3693.1	5781	69	60.55	3693.1	4492	69	68.80
F09	4730	4730	4730.0	11522	88	151.9	4730.0	15582	88	344.7	4730.0	15702	87	449.7	4730.0	15243	87	598.1
F10	2925	2925	2925.0	3345	44	8.406	2925.0	3680	44	10.70	2925.0	2588	44	8.250	2925.0	2248	44	18.61
F11	3835	3835	3835.0	12694	76	169.1	3835.0	15695	76	292.8	3835.0	20413	76	675.9	3835.0	21295	76	1484
F12	3390	3395	3386.1	7611	66	46.42	3386.1	9017	66	67.42	3386.2	7785	66	58.83	3386.3	4978	66	286.8
F13	2855	2855	2855.0	2723	48	6.922	2855.0	3004	48	8.687	2855.0	2730	48	8.125	2855.0	2106	48	8.156
F14	3330	3330	3330.0	4082	50	14.97	3330.0	5078	50	231.9	3330.0	5333	50	2018	3330.0	3024	50	885.8
F15	3560	3560	3560.0	16941	74	301.4	3560.0	23295	70	589.8	3560.0	21400	70	566.7	3560.0	19554	71	1027
F16	2725	2725	2725.0	7115	38	25.92	2725.0	7537	37	36.48	2725.0	8646	37	3381	—	—	—	—
F17	2055	2055	2055.0	2981	31	5.906	2055.0	3206	31	8.453	2055.0	1497	31	3.141	—	—	—	—
F18	3063	3075	3061.6	12888	68	136.4	3061.9	16706	65	703.6	—	—	—	—	—	—	—	—
F19	2500	2525	2490.9	8545	65	57.25	—	—	—	—	—	—	—	—	—	—	—	—
F20	2445	2445	2445.0	8308	54	46.80	2445.0	9674	54	74.11	2445.0	9913	54	92.11	2445.0	6610	54	92.63
F21	2930	2930	2930.0	9647	51	66.03	2930.0	11328	51	111.1	2930.0	11865	51	138.5	2930.0	6428	51	66.74
F22	2075	2075	2075.0	5529	50	17.830	2075.0	6951	56	84.11	2075.0	4756	49	323.6	—	—	—	—
F23	2994	3005	**2996.9**	19076	153	413.7	**2998.8**	27193	112	3012	—	—	—	—	—	—	—	—

Continued on next page

Table V.1: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
F24	3210	3210	3210.0	13141	88	173.9	3210.0	16598	102	960.1	3210.0	22045	88	9437	—	—	—	—
F25	1390	1390	1390.0	2023	22	2.610	1390.0	2092	22	4.031	1390.0	736	22	3.766	1390.0	736	22	7.562
e1-A	3548	3548	3548.0	3116	55	26.73	3548.0	2898	55	23.16	3548.0	2623	55	18.63	3548.0	1717	55	26.05
e1-B	4498	4498	4472.5	1877	78	9.297	4472.5	1531	78	10.88	4473.4	1555	78	8.828	4473.4	1356	78	10.10
e1-C	5595	5595	5541.8	1312	70	6.938	5544.8	1227	70	6.860	5544.8	1012	70	5.485	5544.8	894	70	9.922
e2-A	5018	5018	5006.0	4177	82	46.77	5008.3	3980	84	132.6	5008.3	3734	84	692.3	—	—	—	—
e2-B	6305	6317	6291.7	2623	81	22.53	6296.6	2773	78	32.39	6298.7	2576	78	380.5	6299.3	2508	78	96.41
e2-C	8335	8335	8270.6	1895	80	13.05	8274.4	1777	80	12.42	8274.4	1666	80	12.24	8274.4	1366	80	16.50
e3-A	5898	5898	5895.9	5959	75	103.7	5895.9	5989	75	104.1	5895.9	6794	75	179.7	5895.9	5567	75	535.5
e3-B	7711	7775	7696.9	3091	77	38.44	7701.0	3087	77	63.86	7702.8	2857	76	45.08	7704.6	2450	76	90.83
e3-C	10244	10292	10180.0	2034	84	17.86	10183.6	2183	84	20.55	10183.9	1882	84	17.86	10184.1	1708	84	20.44
e4-A	6408	6444	6389.7	6453	105	123.1	6390.7	7517	116	608.8	—	—	—	—	—	—	—	—
e4-B	8935	8961	8884.8	2714	72	51.38	8889.3	2755	71	47.66	8890.4	2576	71	52.72	8890.4	2547	71	92.75
e4-C	11493	11550	11465.3	2146	77	28.25	11467.5	2418	78	29.27	11467.5	2326	77	34.60	11467.5	1918	77	33.02
s1-A	5018	5018	5014.5	5674	143	100.3	5014.5	8792	143	188.0	5014.5	7544	143	167.2	5014.5	6216	143	413.2
s1-B	6388	6388	6377.4	3272	154	35.28	6378.0	2932	154	31.91	6378.0	3753	154	54.77	6378.0	3109	154	237.9
s1-C	8518	8518	8484.4	1821	156	17.03	8487.1	1663	153	14.34	8487.2	1903	153	17.56	8487.2	1452	153	37.09
s2-A	9825	9884	9800.4	7940	142	310.6	9801.6	7711	145	330.6	9801.6	8474	139	498.5	9801.6	7903	145	2464
s2-B	13017	13100	12965.9	5049	165	138.8	12972.5	5060	153	209.0	12976.3	4782	185	201.4	12978.6	4698	175	223.1
s2-C	16425	16425	16347.9	2979	164	70.55	16355.8	3268	155	88.17	16358.0	3622	156	93.00	16358.4	3009	156	106.8
s3-A	10145	10220	10140.7	9409	162	623.3	10144.7	9585	178	895.4	**10147.4**	9717	203	1964	—	—	—	—
s3-B	13648	13682	13618.7	5279	151	182.1	13621.9	5027	183	213.2	13623.3	5270	189	240.1	13623.5	4924	185	323.8
s3-C	17188	17188	17096.8	3615	138	103.0	17112.2	3165	146	89.74	17113.3	3199	150	90.34	17113.7	3011	145	102.6
s4-A	12143	12268	12127.4	9100	159	536.8	12133.0	8910	137	640.4	12136.6	9971	141	1639	12137.1	9402	141	4629
s4-B	16098	16283	16065.5	6326	325	299.0	16076.7	5636	384	377.3	16078.0	5283	343	376.4	16078.6	5585	358	497.1
s4-C	20430	20481	20384.8	4220	158	176.6	20394.2	4144	151	196.4	20397.2	4054	147	199.7	20397.2	3951	151	320.3

Table V.2: Column Generation Results for the GVRP ($\theta = 2$)

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
A-n32-k5-C16-V2	519.0	519	516.3	248	4	0.501	516.7	190	1	0.324	516.7	190	1	0.322	516.7	190	1	0.558
A-n33-k5-C17-V3	451.0	451	451.0	315	20	0.346	451.0	216	27	0.288	451.0	202	20	0.289	451.0	202	20	0.359
A-n33-k6-C17-V3	465.0	465	465.0	299	3	0.254	465.0	208	3	0.263	465.0	164	3	0.239	465.0	164	3	0.348
A-n34-k5-C17-V3	489.0	489	489.0	252	0	0.230	489.0	172	0	0.299	489.0	143	0	0.198	489.0	143	0	0.270
A-n36-k5-C18-V2	505.0	505	500.9	437	8	0.681	504.5	300	2	0.486	504.5	223	5	0.359	504.5	223	5	0.477
A-n37-k5-C19-V3	432.0	432	432.0	490	0	0.365	432.0	472	0	0.560	432.0	256	0	0.341	432.0	256	0	0.433
A-n37-k6-C19-V3	584.0	584	584.0	379	6	0.450	584.0	313	6	0.536	584.0	225	4	0.463	584.0	225	4	0.744
A-n38-k5-C19-V3	476.0	476	476.0	386	5	0.474	476.0	322	5	0.389	476.0	219	5	0.319	476.0	219	5	0.435
A-n39-k5-C20-V3	557.0	557	543.8	559	55	0.630	547.1	443	45	0.581	547.3	368	51	0.595	547.3	368	51	0.717
A-n39-k6-C20-V3	544.0	544	544.0	423	29	0.640	544.0	355	19	0.538	544.0	290	18	0.471	544.0	290	18	0.690
A-n44-k6-C22-V3	608.0	608	608.0	517	21	0.624	608.0	365	0	0.387	608.0	321	0	0.585	608.0	321	0	1.002
A-n45-k6-C23-V4	613.0	613	608.4	633	46	0.759	608.4	635	29	0.864	608.4	402	39	0.757	608.4	402	39	1.046
A-n45-k7-C23-V4	674.0	674	662.6	523	51	0.631	663.1	547	50	0.658	663.2	350	50	0.675	663.2	350	50	0.986
A-n46-k7-C23-V4	593.0	593	590.7	634	21	0.786	591.8	541	19	0.949	591.8	412	22	1.073	591.8	412	22	1.783
A-n48-k7-C24-V4	667.0	667	655.0	609	68	1.029	655.4	591	58	1.143	655.5	410	56	0.954	655.5	410	56	1.288
A-n53-k7-C27-V4	603.0	603	603.0	894	28	1.409	603.0	937	28	1.611	603.0	757	58	1.644	603.0	715	58	2.706
A-n54-k7-C27-V4	690.0	690	689.5	1010	111	1.971	690.0	899	91	2.009	690.0	625	54	1.253	690.0	625	54	1.961
A-n55-k9-C28-V5	699.0	699	699.0	644	36	1.082	699.0	658	31	1.775	699.0	467	32	2.281	699.0	467	32	3.951
A-n60-k9-C30-V5	769.0	769	769.0	834	12	1.229	769.0	734	21	1.469	769.0	596	20	1.138	769.0	596	20	1.701
A-n61-k9-C31-V5	638.0	638	635.3	784	68	1.548	636.2	807	101	1.909	636.3	756	111	1.708	636.3	756	111	2.080
A-n62-k8-C31-V4	740.0	740	740.0	1176	201	3.190	740.0	1150	157	2.701	740.0	915	127	2.677	740.0	915	127	3.848
A-n63-k10-C32-V5	801.0	801	794.0	808	68	2.057	794.0	858	65	1.384	794.0	741	57	1.446	794.0	725	66	1.766
A-n63-k9-C32-V5	900.3	912	**906.8**	904	109	2.430	**907.0**	807	104	1.578	**907.0**	740	99	3.004	**907.0**	642	92	2.770
A-n64-k9-C32-V5	763.0	763	763.0	958	0	1.800	763.0	978	0	1.460	763.0	827	0	1.553	763.0	675	0	1.462
A-n65-k9-C33-V5	682.0	682	681.6	997	66	1.547	681.5	1040	73	2.436	681.5	989	71	1.808	681.5	783	65	1.992
A-n69-k9-C35-V5	680.0	680	671.4	1026	94	1.766	672.6	940	63	2.317	672.6	948	72	2.053	672.6	782	63	3.729

Continued on next page

Table V.2: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
A-n80-k10-C40-V5	957.4	997	**982.4**	1595	71	4.184	**982.7**	1507	65	4.768	**983.0**	1619	56	6.152	**982.8**	1485	90	7.868
B-n31-k5-C16-V3	441.0	441	441.0	310	8	0.269	441.0	165	8	0.213	441.0	165	8	0.214	441.0	165	8	0.271
B-n34-k5-C17-V3	472.0	472	472.0	390	5	0.416	472.0	247	5	0.390	472.0	232	5	0.311	472.0	232	5	0.423
B-n35-k5-C18-V3	626.0	626	626.0	436	18	0.474	626.0	290	4	0.328	626.0	267	5	0.423	626.0	267	5	0.605
B-n38-k6-C19-V3	451.0	451	451.0	384	4	0.515	451.0	336	7	0.462	451.0	230	5	0.345	451.0	230	5	0.431
B-n39-k5-C20-V3	357.0	357	357.0	523	5	0.807	357.0	421	4	0.860	357.0	348	7	0.752	357.0	348	7	1.119
B-n41-k6-C21-V3	481.0	481	481.0	490	31	0.802	481.0	429	6	0.756	481.0	274	11	0.719	481.0	274	11	1.100
B-n43-k6-C22-V3	483.0	483	481.8	792	54	1.090	482.0	642	42	0.843	482.0	470	46	1.086	482.0	470	46	1.533
B-n44-k7-C22-V4	540.0	540	540.0	637	35	0.786	540.0	591	31	1.239	540.0	501	40	0.741	540.0	501	40	0.890
B-n45-k5-C23-V3	497.0	497	497.0	815	16	1.308	497.0	492	0	0.807	497.0	466	0	0.963	497.0	451	0	1.057
B-n45-k6-C23-V4	478.0	478	474.2	567	25	0.713	474.5	577	22	0.884	474.5	462	15	0.749	474.5	475	23	0.790
B-n50-k7-C25-V4	449.0	449	449.0	740	0	1.654	449.0	602	0	0.911	449.0	454	0	0.576	449.0	442	0	2.372
B-n50-k8-C25-V5	916.0	916	912.6	537	58	0.996	913.2	487	38	0.748	913.2	386	25	0.838	913.2	386	25	1.098
B-n51-k7-C26-V4	651.0	651	651.0	654	4	0.925	651.0	472	4	0.971	651.0	416	4	0.782	651.0	416	4	1.158
B-n52-k7-C26-V4	450.0	450	450.0	838	11	1.567	450.0	812	11	1.294	450.0	571	13	1.000	450.0	582	11	1.460
B-n56-k7-C28-V4	486.0	486	484.7	1022	29	2.461	486.0	1032	20	1.904	486.0	742	16	1.788	486.0	742	16	2.451
B-n57-k7-C29-V4	751.0	751	751.0	940	40	1.456	751.0	915	35	1.857	751.0	692	25	1.488	751.0	692	25	2.127
B-n57-k9-C29-V5	942.0	942	942.0	606	22	0.772	942.0	695	24	0.924	942.0	666	34	1.358	942.0	666	34	1.790
B-n63-k10-C32-V5	816.0	816	809.0	953	45	1.726	809.0	1084	58	2.609	809.0	863	53	2.430	809.0	862	52	1.887
B-n64-k9-C32-V5	509.0	509	509.0	952	5	1.560	509.0	900	1	2.240	509.0	770	1	2.045	509.0	740	1	2.310
B-n66-k9-C33-V5	808.0	808	808.0	1037	53	1.992	808.0	1055	50	2.065	808.0	978	78	2.317	808.0	922	55	2.017
B-n67-k10-C34-V5	673.0	673	667.7	1087	77	2.046	667.8	1192	67	2.136	667.7	1079	65	2.243	667.7	943	70	2.237
B-n68-k9-C34-V5	704.0	704	704.0	1020	18	2.747	704.0	998	3	2.370	704.0	920	0	7.059	704.0	1063	44	6.034
B-n78-k10-C39-V5	803.0	803	803.0	1420	57	3.339	803.0	1365	48	3.335	803.0	1246	36	3.237	803.0	1216	48	5.456
P-n16-k8-C8-V5	239.0	239	239.0	15	0	0.007	239.0	15	0	0.007	239.0	15	0	0.012	239.0	15	0	0.007
P-n19-k2-C10-V2	147.0	147	147.0	73	0	0.105	147.0	42	0	0.106	147.0	42	0	0.158	147.0	42	0	0.145

Continued on next page

Table V.2: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
P-n20-k2-C10-V2	154.0	154	154.0	82	0	0.168	154.0	38	0	0.126	154.0	38	0	0.133	154.0	38	0	0.164
P-n21-k2-C11-V2	160.0	160	160.0	130	0	0.321	160.0	42	0	0.253	160.0	42	0	0.251	160.0	42	0	0.415
P-n22-k2-C11-V2	162.0	162	162.0	120	0	0.216	162.0	65	0	0.229	162.0	65	0	0.230	162.0	65	0	0.351
P-n22-k8-C11-V5	314.0	314	314.0	48	0	1.240	314.0	47	0	1.241	314.0	47	0	1.230	314.0	47	0	1.245
P-n23-k8-C12-V5	312.0	312	312.0	54	10	0.026	312.0	57	8	0.028	312.0	57	8	0.028	312.0	57	8	0.030
P-n40-k5-C20-V3	294.0	294	294.0	506	12	1.000	294.0	434	21	0.912	294.0	225	27	0.741	294.0	225	27	1.233
P-n45-k5-C23-V3	337.0	337	337.0	576	19	1.003	337.0	597	9	1.344	337.0	369	13	1.240	337.0	369	13	2.064
P-n50-k7-C25-V4	353.0	353	350.5	572	42	1.244	350.6	560	35	1.520	350.6	404	31	1.369	350.6	404	31	2.036
P-n50-k8-C25-V4	378.4	392	**385.7**	372	18	0.913	**385.8**	391	17	0.749	**385.8**	304	11	1.052	**385.8**	304	11	1.457
P-n50-k10-C25-V5	410.0	410	407.9	392	56	0.974	409.0	412	44	1.241	409.0	341	27	0.820	409.0	341	27	1.252
P-n51-k10-C26-V6	427.0	427	427.0	406	0	0.424	427.0	391	3	0.482	427.0	325	0	0.368	427.0	325	0	0.528
P-n55-k7-C28-V4	361.0	361	355.7	749	30	1.771	355.7	712	27	1.963	355.7	494	28	1.380	355.7	494	28	1.605
P-n55-k8-C28-V4	361.0	361	359.6	658	60	1.644	359.7	717	37	1.937	359.6	467	39	1.637	359.6	467	39	2.494
P-n55-k10-C28-V5	415.0	415	411.9	522	51	1.035	412.1	515	45	1.037	412.2	374	47	1.155	412.2	374	47	1.609
P-n55-k15-C28-V8	545.3	555	**555.0**	219	4	0.399	**555.0**	212	4	0.385	**555.0**	216	3	0.350	**555.0**	216	3	0.538
P-n60-k10-C30-V5	433.0	443	**435.4**	519	39	1.760	**435.5**	491	43	2.339	**435.3**	426	39	2.540	**435.4**	426	39	3.621
P-n60-k15-C30-V8	553.9	565	**564.7**	374	32	0.607	**564.8**	336	21	0.614	**564.8**	294	26	0.591	**564.8**	294	26	0.674
P-n65-k10-C33-V5	487.0	487	484.8	750	99	2.051	485.4	669	87	2.042	485.4	688	78	1.922	485.4	579	69	3.345
P-n70-k10-C35-V5	485.0	485	485.0	765	87	2.366	485.0	695	60	2.889	485.0	741	63	2.033	485.0	595	61	17.53
P-n76-k4-C38-V2	383.0	383	379.4	2027	26	23.54	380.7	1835	11	24.45	380.7	1641	10	110.8	380.7	1383	10	2185
P-n76-k5-C38-V3	405.0	405	400.1	1596	60	14.63	401.4	1753	81	17.46	401.6	1588	65	24.58	401.6	1253	59	40.28
P-n101-k4-C51-V2	455.0	455	452.4	4739	51	146.0	452.8	5071	26	342.7	—	—	—	—	—	—	—	—
M-n101-k10-C51-V5	542.0	542	540.4	2077	22	19.85	540.4	2182	17	19.08	540.4	1953	17	19.57	540.5	2153	15	41.26
M-n121-k7-C61-V4	707.7	719	**710.3**	4773	156	112.9	**710.7**	6086	170	131.0	**710.8**	6850	176	1656	**710.9**	4887	177	7919
M-n151-k12-C76-V6	629.9	659	**649.5**	3272	296	52.39	**650.0**	3554	275	62.55	**649.9**	3356	208	52.22	**650.2**	2971	244	86.25
M-n200-k16-C100-V8	744.9	791	**777.8**	4180	382	88.92	**778.4**	4366	284	109.9	**778.8**	4313	231	128.7	**778.8**	4241	217	192.9

Table V.3: Column Generation Results for the GVRP ($\theta = 3$)

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
A-n32-k5-C11-V2	386.0	386	382.3	112	5	0.303	382.3	84	7	0.269	382.3	84	7	0.262	382.3	84	7	0.361
A-n33-k5-C11-V2	315.0	315	315.0	109	3	0.203	315.0	85	3	0.180	315.0	85	3	0.180	315.0	85	3	0.238
A-n33-k6-C11-V2	370.0	370	370.0	92	0	0.180	370.0	58	0	0.177	370.0	58	0	0.177	370.0	58	0	0.258
A-n34-k5-C12-V2	419.0	419	418.3	137	19	0.392	418.3	107	16	0.475	418.3	107	16	0.471	418.3	107	16	0.785
A-n36-k5-C12-V2	396.0	396	385.9	192	14	0.458	385.9	104	8	0.323	385.9	104	8	0.323	385.9	104	8	0.440
A-n37-k5-C13-V2	347.0	347	347.0	187	0	0.395	347.0	106	0	0.700	347.0	106	0	0.712	347.0	106	0	1.317
A-n37-k6-C13-V2	431.0	431	431.0	138	4	0.244	431.0	84	0	0.209	431.0	84	0	0.198	431.0	84	0	0.295
A-n38-k5-C13-V2	367.0	367	367.0	158	0	0.287	367.0	110	0	0.401	367.0	110	0	0.400	367.0	110	0	0.679
A-n39-k5-C13-V2	364.0	364	358.0	222	18	0.523	359.2	162	12	0.706	359.2	162	12	0.705	359.2	162	12	1.148
A-n39-k6-C13-V2	403.0	403	403.0	172	2	0.375	403.0	113	2	0.294	403.0	113	2	0.293	403.0	113	2	0.395
A-n44-k6-C15-V2	503.0	503	503.0	206	0	0.450	503.0	152	0	0.737	503.0	152	0	0.735	503.0	152	0	1.244
A-n45-k6-C15-V3	474.0	474	474.0	250	5	0.510	474.0	155	5	0.558	474.0	155	5	0.555	474.0	155	5	0.847
A-n45-k7-C15-V3	475.0	475	475.0	208	0	0.480	475.0	137	0	0.304	475.0	137	0	0.302	475.0	137	0	0.398
A-n46-k7-C16-V3	462.0	462	462.0	303	18	0.545	462.0	206	14	0.650	462.0	206	14	0.651	462.0	206	14	1.041
A-n48-k7-C16-V3	451.0	451	449.4	262	7	0.535	451.0	138	5	4.554	451.0	138	5	4.558	451.0	138	5	6.942
A-n53-k7-C18-V3	440.0	440	440.0	525	0	1.148	440.0	246	0	1.073	440.0	214	3	1.042	440.0	214	3	1.800
A-n54-k7-C18-V3	482.0	482	482.0	355	8	0.870	482.0	285	8	0.979	482.0	256	4	1.051	482.0	256	4	1.831
A-n55-k9-C19-V3	473.0	473	473.0	346	20	0.787	473.0	361	19	1.159	473.0	202	21	1.355	473.0	202	21	2.572
A-n60-k9-C20-V3	595.0	595	592.8	480	23	1.086	593.5	451	28	1.604	593.5	305	29	1.566	593.5	305	29	2.665
A-n61-k9-C21-V4	473.0	473	473.0	354	7	0.787	473.0	368	7	0.914	473.0	280	7	1.086	473.0	280	7	1.859
A-n62-k8-C21-V3	596.0	596	593.0	486	74	1.376	594.0	447	67	1.499	594.8	347	54	1.306	594.8	347	54	1.868
A-n63-k10-C21-V4	593.0	593	591.7	432	8	0.952	592.3	361	12	0.841	592.3	305	9	0.805	592.3	305	9	1.101
A-n63-k9-C21-V3	625.6	642	**629.7**	462	19	1.317	**636.3**	384	15	1.417	**636.3**	300	7	1.007	**636.3**	300	7	1.605
A-n64-k9-C22-V3	536.0	536	536.0	649	28	1.496	536.0	653	29	1.387	536.0	523	63	2.148	536.0	523	63	3.037
A-n65-k9-C22-V3	500.0	500	500.0	414	0	0.875	500.0	388	0	0.882	500.0	303	0	0.749	500.0	303	0	0.987
A-n69-k9-C23-V3	520.0	520	515.3	640	39	1.916	519.8	432	23	2.049	520.0	313	16	1.577	520.0	313	16	2.347

Continued on next page

Table V.3: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
A-n80-k10-C27-V4	679.4	710	**706.5**	779	17	2.132	**708.8**	797	32	2.326	**708.8**	636	23	2.394	**708.8**	636	23	3.278
B-n31-k5-C11-V2	356.0	356	354.5	133	4	0.187	354.5	66	2	0.147	354.5	66	2	0.147	354.5	66	2	0.180
B-n34-k5-C12-V2	369.0	369	369.0	154	8	0.305	369.0	94	4	0.334	369.0	94	4	0.334	369.0	94	4	0.546
B-n35-k5-C12-V2	501.0	501	501.0	171	4	0.434	501.0	108	7	0.389	501.0	108	7	0.389	501.0	108	7	0.618
B-n38-k6-C13-V2	370.0	370	370.0	144	0	0.277	370.0	93	0	0.255	370.0	93	0	0.262	370.0	93	0	0.356
B-n39-k5-C13-V2	280.0	280	280.0	193	0	0.402	280.0	149	0	0.481	280.0	149	0	0.471	280.0	149	0	0.799
B-n41-k6-C14-V2	407.0	407	407.0	156	4	0.316	407.0	146	4	0.547	407.0	146	4	0.559	407.0	146	4	0.907
B-n45-k5-C15-V2	410.0	410	410.0	297	0	0.860	410.0	153	0	0.419	410.0	153	0	0.422	410.0	153	0	0.606
B-n43-k6-C15-V2	343.0	343	343.0	347	21	1.017	343.0	165	14	1.044	343.0	165	14	1.041	343.0	165	14	1.956
B-n45-k6-C15-V2	336.0	336	336.0	227	0	0.388	336.0	150	0	0.373	336.0	150	0	0.374	336.0	150	0	0.547
B-n44-k7-C15-V3	395.0	395	394.3	239	4	0.615	394.3	156	4	0.516	394.3	156	4	0.515	394.3	156	4	0.790
B-n50-k8-C17-V3	598.0	598	598.0	310	27	0.530	598.0	240	21	0.457	598.0	216	20	0.447	598.0	216	20	0.532
B-n51-k7-C17-V3	511.0	511	511.0	285	1	0.560	511.0	204	1	0.574	511.0	200	1	0.525	511.0	200	1	0.789
B-n50-k7-C17-V3	393.0	393	393.0	312	1	0.778	393.0	244	3	0.831	393.0	178	1	1.195	393.0	178	1	1.970
B-n52-k7-C18-V3	359.0	359	359.0	469	2	1.026	359.0	370	5	1.157	359.0	309	7	1.022	359.0	309	7	1.569
B-n57-k9-C19-V3	681.0	681	674.0	527	51	1.667	674.0	419	30	1.265	674.0	372	35	1.185	674.0	372	35	1.495
B-n56-k7-C19-V3	356.0	356	355.0	544	6	1.375	355.5	411	6	1.310	355.5	238	6	0.874	355.5	238	6	1.326
B-n57-k7-C19-V3	558.0	558	558.0	460	11	1.423	558.0	410	6	1.464	558.0	316	10	1.560	558.0	316	10	2.717
B-n63-k10-C21-V3	599.0	599	599.0	425	0	1.519	599.0	340	0	1.565	599.0	257	0	1.974	599.0	257	0	3.888
B-n66-k9-C22-V3	609.0	609	595.8	611	44	1.531	596.2	637	22	1.601	596.5	451	26	2.232	596.5	451	26	3.226
B-n64-k9-C22-V4	452.0	452	452.0	424	0	1.333	452.0	363	0	1.013	452.0	268	0	0.964	452.0	268	0	1.492
B-n67-k10-C23-V4	558.0	558	551.0	475	8	1.387	551.0	423	7	1.137	551.0	338	8	1.190	551.0	338	8	1.921
B-n68-k9-C23-V3	523.0	523	523.0	667	35	1.621	523.0	495	12	1.760	523.0	326	7	2.202	523.0	326	7	4.188
B-n78-k10-C26-V4	606.0	606	606.0	756	35	1.724	606.0	721	15	3.002	606.0	568	10	1.954	606.0	568	10	3.034
P-n16-k8-C6-V4	170.0	170	170.0	17	0	0.013	170.0	17	0	0.008	170.0	17	0	0.007	170.0	17	0	0.014
P-n19-k2-C7-V1	111.0	111	111.0	26	0	0.137	111.0	26	0	0.078	111.0	26	0	0.077	111.0	26	0	0.160

Continued on next page

Table V.3: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
P-n20-k2-C7-V1	117.0	117	117.0	21	0	0.128	117.0	21	0	0.091	117.0	21	0	0.088	117.0	21	0	0.115
P-n21-k2-C7-V1	117.0	117	117.0	22	0	0.115	117.0	22	0	0.108	117.0	22	0	0.108	117.0	22	0	0.222
P-n22-k2-C8-V1	111.0	111	111.0	25	0	0.103	111.0	25	0	0.102	111.0	25	0	0.102	111.0	25	0	0.141
P-n22-k8-C8-V4	249.0	249	249.0	31	0	1.170	249.0	31	0	1.177	249.0	31	0	1.177	249.0	31	0	1.221
P-n23-k8-C8-V3	174.0	174	174.0	34	0	0.022	174.0	34	0	0.023	174.0	34	0	0.022	174.0	34	0	0.025
P-n40-k5-C14-V2	213.0	213	212.2	250	12	0.944	212.2	132	11	0.969	212.2	132	11	0.973	212.2	132	11	1.741
P-n45-k5-C15-V2	238.0	238	237.7	264	9	1.209	237.7	137	8	0.794	237.7	137	8	0.791	237.7	137	8	1.289
P-n50-k7-C17-V3	261.0	261	259.0	286	16	1.059	259.0	209	4	0.809	259.0	177	11	0.962	259.0	177	11	1.510
P-n50-k8-C17-V3	262.0	262	262.0	206	0	0.447	262.0	170	0	0.505	262.0	190	0	0.558	262.0	190	0	0.850
P-n50-k10-C17-V4	292.0	292	292.0	204	3	0.434	292.0	164	3	0.478	292.0	154	3	0.422	292.0	154	3	0.529
P-n51-k10-C17-V4	309.0	309	305.5	209	6	0.375	305.5	187	6	0.335	305.5	167	5	0.316	305.5	167	5	0.413
P-n55-k7-C19-V3	271.0	271	267.8	385	7	1.704	267.9	327	5	1.230	267.9	264	5	1.169	267.9	264	5	1.492
P-n55-k8-C19-V3	274.0	274	271.6	356	14	1.453	271.6	295	16	1.439	271.6	223	10	1.091	271.6	223	10	1.485
P-n55-k10-C19-V4	301.0	301	300.8	274	13	0.717	301.0	266	16	0.663	301.0	185	11	0.584	301.0	185	11	0.718
P-n55-k15-C19-V6	378.0	378	378.0	141	0	0.250	378.0	161	0	0.289	378.0	130	0	0.267	378.0	130	0	0.374
P-n60-k10-C20-V4	325.0	325	320.7	312	20	1.112	320.7	267	25	1.214	320.7	234	22	1.157	320.7	234	22	1.709
P-n60-k15-C20-V5	380.0	382	**380.3**	172	9	0.562	**381.5**	172	10	0.675	**381.5**	156	9	0.693	**381.5**	156	9	1.053
P-n65-k10-C22-V4	372.0	372	369.1	383	11	1.326	369.4	383	18	1.689	369.4	280	16	1.131	369.4	280	16	1.579
P-n70-k10-C24-V4	385.0	385	382.0	514	34	2.401	382.6	429	33	1.721	382.6	343	28	1.597	382.6	343	28	2.423
P-n76-k4-C26-V2	309.0	309	303.4	1114	28	13.39	303.5	1039	16	20.35	303.5	716	16	29.99	303.5	716	16	48.04
P-n76-k5-C26-V2	309.0	309	309.0	856	0	6.453	309.0	821	0	13.61	309.0	497	0	7.282	309.0	497	0	11.65
P-n101-k4-C34-V2	370.0	370	367.3	3014	43	85.14	367.8	3122	54	304.1	—	—	—	—	—	—	—	—
M-n101-k10-C34-V4	458.0	458	456.3	1181	50	11.09	456.9	1183	61	13.44	456.9	943	58	17.63	456.9	964	51	45.84
M-n121-k7-C41-V3	527.0	527	526.9	2883	42	69.99	527.0	2629	23	147.7	527.0	3287	34	10001	527.0	2241	32	11210
M-n151-k12-C51-V4	465.6	483	**482.3**	2229	107	37.62	**483.0**	2142	126	38.88	**483.0**	2095	105	56.54	**483.0**	1972	155	103.1
M-n200-k16-C67-V6	563.1	605	**592.4**	2504	161	78.75	**593.6**	2400	119	58.65	**594.0**	2445	147	68.53	**594.2**	2606	119	152.5

Table V.4: Column Generation Results for the CVRP

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
A-n32-k5	784	784	784.0	1017	61	0.948	784.0	1009	38	0.860	784.0	779	55	0.610	784.0	779	55	0.793
A-n33-k5	661	661	661.0	818	66	0.569	661.0	748	43	0.484	661.0	649	44	0.489	661.0	674	45	1.793
A-n33-k6	742	742	740.3	825	49	0.578	740.3	821	68	0.776	740.3	752	57	0.552	740.3	667	59	0.784
A-n34-k5	778	778	775.5	926	69	0.981	776.0	958	95	1.134	776.2	826	98	0.906	776.0	764	59	0.891
A-n36-k5	799	799	798.4	1221	77	1.130	798.7	1352	60	1.218	798.7	1221	61	1.135	798.7	981	53	1.298
A-n37-k5	669	669	667.3	1530	89	1.554	667.3	1560	76	1.703	667.2	1330	79	1.354	667.4	1178	69	1.901
A-n37-k6	949	949	937.0	915	63	1.181	940.4	997	44	0.893	940.1	985	55	3.378	940.7	883	83	3.055
A-n38-k5	730	730	723.4	1204	124	1.232	723.4	1290	83	1.367	723.4	1220	77	1.287	723.4	1279	85	1.893
A-n39-k5	822	822	818.0	1545	182	2.164	818.1	1529	271	2.804	819.2	1424	194	2.104	819.0	1261	199	1.804
A-n39-k6	831	831	824.5	1264	113	1.310	825.2	1225	84	1.262	825.2	1198	88	1.418	825.2	1024	91	1.737
A-n44-k6	937	937	934.9	1120	174	1.321	936.8	1311	142	1.586	936.8	1254	163	1.765	936.8	1219	183	1.903
A-n45-k6	944	944	941.2	1340	107	2.015	941.8	1384	170	2.218	942.5	1276	86	1.844	942.5	1181	134	3.841
A-n45-k7	1146	1146	1140.5	1329	203	1.970	1141.5	1224	236	2.036	1142.3	1323	167	2.016	1142.4	1143	192	2.171
A-n46-k7	914	914	914.0	1582	188	2.019	914.0	1294	103	1.873	914.0	1401	203	2.739	914.0	1353	203	5.508
A-n48-k7	1073	1073	1071.6	1460	157	2.100	1071.8	1482	173	2.365	1072.0	1626	170	3.144	1072.0	1381	157	5.438
A-n53-k7	1010	1010	1003.7	2233	223	4.908	1004.2	2159	134	3.824	1004.5	2389	111	4.539	1004.5	2110	122	8.414
A-n54-k7	1167	1167	1155.6	1852	268	4.217	1156.8	1999	275	5.150	1157.1	2219	245	4.688	1156.6	1860	279	5.446
A-n55-k9	1073	1073	1068.5	1384	158	2.551	1069.4	1332	204	2.974	1069.4	1461	200	3.928	1069.3	1387	179	5.175
A-n60-k9	1354	1354	1345.0	1891	186	3.847	1345.6	1817	173	3.703	1345.7	1925	198	4.327	1345.7	1947	171	6.334
A-n61-k9	1034	1034	1023.7	1577	255	3.501	1024.6	1478	288	4.066	1024.6	1534	312	5.264	1024.6	1770	295	8.005
A-n62-k8	1288	1288	1280.9	2567	288	8.051	1282.1	2690	204	9.503	1282.0	2489	212	9.422	1282.1	2577	207	19.43
A-n63-k9	1616	1616	1608.6	1888	259	4.420	1609.1	2051	232	5.168	1610.3	2199	206	5.681	1610.8	2153	224	6.641
A-n63-k10	1314	1314	1302.4	1715	183	3.868	1303.9	1794	149	4.163	1304.0	1817	132	4.730	1304.0	2046	147	7.308

Continued on next page

Table V.4: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
A-n64-k9	1401	1401	1387.8	2069	177	5.159	1389.6	2046	158	5.269	1390.0	2208	152	6.275	1390.1	2268	161	10.42
A-n65-k9	1174	1174	1168.5	2074	269	4.643	1169.0	1905	223	4.844	1169.0	1892	190	4.423	1169.0	2098	222	5.948
A-n69-k9	1159	1159	1144.2	2338	228	4.688	1145.2	2344	231	5.015	1145.4	2156	221	5.417	1145.4	2367	214	8.407
A-n80-k10	1763	1763	1756.5	3192	217	11.76	1757.0	3184	200	12.03	1756.7	3238	174	11.50	1756.8	3070	191	15.00
B-n31-k5	672	672	672.0	931	14	0.474	672.0	940	11	0.595	672.0	746	10	0.741	672.0	746	10	0.980
B-n34-k5	788	788	784.9	1435	76	1.259	785.0	1335	80	1.179	785.2	1202	95	1.298	785.2	1351	96	9.624
B-n35-k5	955	955	955.0	1347	22	1.030	955.0	1396	29	0.948	955.0	1164	26	0.897	955.0	1014	31	1.246
B-n38-k6	805	805	804.1	1187	51	0.907	804.1	1185	44	1.075	804.1	1146	43	1.340	804.1	1048	37	1.657
B-n39-k5	549	549	549.0	2003	46	3.173	549.0	1382	37	18.01	549.0	1408	46	5.338	549.0	1294	42	306.7
B-n41-k6	829	829	828.5	1243	86	1.384	828.6	1236	80	1.494	828.8	1316	70	1.467	828.8	1023	83	1.567
B-n43-k6	742	742	737.3	1414	109	1.821	737.6	1383	136	2.016	737.7	1637	97	2.083	737.6	1336	129	2.120
B-n44-k7	909	909	909.0	1588	132	1.861	909.0	1269	37	1.482	909.0	1484	49	1.997	909.0	1352	88	2.171
B-n45-k5	751	751	750.6	2301	81	3.802	751.0	2323	90	7.787	751.0	2276	98	31.35	751.0	1963	65	12.70
B-n45-k6	678	678	678.0	1612	98	3.213	678.0	1540	82	4.320	678.0	1466	95	5.385	678.0	1526	88	3.676
B-n50-k7	741	741	741.0	2156	58	3.114	741.0	2034	64	2.483	741.0	2265	57	3.885	741.0	1976	63	4.278
B-n50-k8	1312	1312	1303.4	1597	181	2.381	1303.6	1630	138	2.853	1303.7	1799	136	3.019	1303.7	1541	137	3.882
B-n51-k7	1032	1032	1026.8	1994	135	3.833	1027.2	1965	121	3.348	1027.3	1737	110	3.233	1027.2	1833	98	3.922
B-n52-k7	747	747	746.3	2304	109	4.313	746.5	2105	104	3.694	746.5	1968	79	3.027	746.5	1996	83	8.596
B-n56-k7	707	707	705.0	2562	65	3.994	705.0	2480	51	4.173	705.0	1928	24	3.901	705.0	2599	60	13.82
B-n57-k7	1153	1153	1153.0	2409	168	7.198	1153.0	2030	85	6.627	1153.0	2170	40	7.708	1153.0	1934	51	7.625
B-n57-k9	1598	1598	1596.0	2022	171	3.563	1596.2	1805	189	3.969	1596.2	2049	131	4.047	1596.2	1799	133	5.268
B-n63-k10	1496	1496	1487.2	1840	172	4.447	1487.4	1847	167	4.690	1487.4	1937	152	4.843	1487.4	2202	165	6.355
B-n64-k9	861	861	860.5	2645	199	5.895	860.5	2639	214	6.630	860.6	2369	137	5.117	860.6	2738	237	9.453

Continued on next page

Table V.4: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
B-n66-k9	1316	1316	1308.9	2552	236	7.893	1308.9	2525	247	7.698	1309.2	2342	218	8.401	1309.4	2667	193	20.40
B-n67-k10	1032	1032	1027.6	2369	128	6.346	1028.1	2283	147	5.750	1028.1	2511	131	6.055	1028.1	2499	136	9.254
B-n68-k9	1272	1272	1263.8	2854	246	9.085	1263.9	2698	206	10.15	1263.9	2546	256	8.934	1263.9	2979	273	18.90
B-n78-k10	1221	1221	1217.0	3134	257	12.22	1217.4	3133	187	11.37	1217.5	3377	182	13.95	1217.6	3478	187	15.54
E-n13-k4	247	247	247.0	61	0	0.020	247.0	78	0	0.025	247.0	78	0	0.025	247.0	78	0	0.027
E-n22-k4	375	375	375.0	403	35	0.150	375.0	368	35	0.148	375.0	310	35	0.158	375.0	310	35	0.220
E-n23-k3	569	569	569.0	889	2	19.73	569.0	544	13	18.14	569.0	328	13	14.71	569.0	328	13	15.70
E-n30-k3	534	534	518.5	1643	61	2.391	523.4	1535	58	2.882	523.4	910	46	2.619	523.4	910	46	3.692
E-n31-k7	379	379	378.3	918	13	0.572	378.5	996	14	0.648	378.5	560	14	0.409	378.5	560	14	0.491
E-n33-k4	835	835	835.0	1330	72	6.385	835.0	1154	66	5.374	835.0	1010	45	6.749	835.0	1117	48	14.79
E-n51-k5	521	521	519.3	2031	153	4.953	519.4	2004	116	4.358	519.4	1700	107	4.946	519.5	1725	146	6.393
E-n76-k7	682	682	671.1	3082	144	13.26	671.3	3177	112	15.86	671.9	2922	93	17.95	671.9	2888	121	30.19
E-n76-k8	735	735	726.8	2836	303	14.46	727.0	2908	261	15.07	727.3	2812	224	12.40	727.3	2770	199	18.58
E-n76-k10	830	830	817.1	1932	143	5.277	817.4	1921	190	6.704	818.0	1860	174	5.941	818.0	2317	161	11.12
E-n76-k14	1021	1021	1006.9	1514	47	3.578	1007.5	1353	70	3.145	1007.5	1381	72	3.145	1007.5	1321	74	4.510
E-n101-k8	815	815	804.4	5224	392	48.57	804.9	5180	451	58.38	805.1	5328	339	69.69	805.1	5213	415	171.8
E-n101-k14	1067	1067	1053.7	2898	151	10.31	1054.1	3031	154	11.86	1055.0	2835	168	11.02	1055.0	2878	168	14.09
P-n16-k8	450	450	448.0	61	7	0.017	448.0	60	7	0.016	448.0	60	7	0.021	448.0	60	7	0.017
P-n19-k2	212	212	212.0	436	1	0.278	212.0	289	1	0.398	212.0	207	2	0.242	212.0	207	2	0.273
P-n20-k2	216	216	213.0	559	3	0.398	215.5	383	5	0.365	215.5	288	4	0.325	215.5	288	4	0.414
P-n21-k2	211	211	211.0	606	6	0.511	211.0	409	0	0.447	211.0	298	0	0.440	211.0	298	0	0.672
P-n22-k2	216	216	216.0	581	14	0.522	216.0	475	5	0.672	216.0	361	5	0.565	216.0	361	5	0.851
P-n22-k8	603	603	603.0	198	10	0.033	603.0	150	0	0.028	603.0	165	0	0.032	603.0	165	0	0.038

Continued on next page

Table V.4: *Continued from previous page*

Ins	LB	UB	NG=8				NG=16				NG=32				NG=64			
			Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time	Value	Routes	Cuts	Time
P-n23-k8	529	529	529.0	193	0	0.040	529.0	162	0	0.033	529.0	129	0	0.029	529.0	129	0	0.032
P-n40-k5	458	458	457.9	1201	161	1.508	458.0	1310	143	1.658	458.0	1185	146	1.787	458.0	1156	140	1.879
P-n45-k5	510	510	507.0	1377	181	2.623	507.3	1353	150	2.676	507.3	1428	181	2.816	507.3	1308	189	3.847
P-n50-k7	554	554	551.5	1200	145	2.691	552.0	1273	148	2.978	551.7	1245	134	2.616	551.7	1240	137	3.689
P-n50-k8	631	631	617.2	877	73	1.475	618.0	895	47	1.561	618.0	854	52	1.683	618.0	886	55	2.449
P-n50-k10	696	696	689.4	779	39	1.009	689.6	783	40	1.085	689.6	759	47	1.082	689.6	797	43	2.897
P-n51-k10	741	741	736.4	850	92	1.169	736.3	951	120	1.080	736.3	1076	119	1.472	736.1	996	102	1.868
P-n55-k7	568	568	559.2	1476	55	3.391	559.8	1330	85	2.965	559.8	1384	97	4.326	559.8	1326	59	8.831
P-n55-k10	694	694	681.8	947	73	1.514	681.9	943	102	1.609	681.9	960	89	1.595	681.9	1038	100	2.415
P-n55-k15	989	989	972.5	587	103	0.937	972.8	584	67	0.844	972.8	560	70	0.911	972.8	591	81	1.009
P-n55-k8	588	588	572.4	1427	201	3.294	573.0	1423	96	4.594	573.1	1352	90	3.294	573.1	1441	90	4.394
P-n60-k10	744	744	738.9	1086	59	1.827	739.7	1038	65	1.977	739.7	1111	47	2.151	739.7	1037	73	4.801
P-n60-k15	968	968	963.5	830	77	1.164	963.6	773	42	1.246	963.6	704	44	1.002	963.6	808	41	1.258
P-n65-k10	792	792	787.1	1338	86	3.130	788.3	1277	101	2.855	788.3	1291	101	2.994	788.3	1378	95	4.135
P-n70-k10	827	827	814.3	1508	164	4.686	815.3	1503	138	5.588	815.3	1617	124	4.969	815.3	1704	144	6.800
P-n76-k4	593	593	589.2	5308	232	63.95	589.9	5480	183	85.47	590.1	5193	154	309.8	590.0	5340	179	6333
P-n76-k5	627	627	617.8	4091	297	32.15	617.9	4013	172	27.99	618.6	4017	199	40.30	618.6	3857	138	67.71
P-n101-k4	681	681	678.6	12741	555	613.5	678.7	12797	445	1097	—	—	—	—	—	—	—	—
M-n101-k10	820	820	820.0	4024	204	10.90	820.0	4948	48	16.04	820.0	4728	59	68.90	820.0	3718	4	333.3
M-n121-k7	1034	1034	1032.5	8244	213	147.5	1032.6	12719	193	6271	—	—	—	—	—	—	—	—
M-n151-k12	1003	1015	1000.4	6974	234	114.2	1001.4	6963	207	118.2	1002.2	7351	68	167.7	1002.7	7020	203	827.4
M-n200-k16	1256.4	1286	1252.5	8610	161	213.9	1252.8	8619	113	247.0	1253.3	8620	116	313.2	1254.1	8963	129	1879
M-n200-k17	1256.8	1275	1255.0	8288	323	232.6	1255.3	8669	319	227.0	1255.6	8832	316	318.5	1256.2	8740	304	10054

VI
A Branch-Cut-and-Price Algorithm

As mentioned in the former chapters, since the column generation algorithm solves the linear relaxation of the set partitioning formulation, another technique is needed in order to obtain an integer solution. In this chapter, we use Branch-and-Bound, a widely known technique, which enumerates all possible solutions through a search tree, discarding the branches with solutions greater than a known upper bound for the instance. This technique, when used together with column generation and cut separation is called *Branch-Cut-and-Price* (BCP). We devise a Branch-Cut-and-Price algorithm for the CARP using the column generation algorithm described in Chapter V and exactly separating the odd edge cutset and capacity cuts as described in Chapter IV.

VI.1 The Branch-and-Bound

The BCP algorithm starts by solving the relaxation problem defined at root node of the search tree. This operation consists in executing the column generation algorithm as described in Chapter V, which includes the exact separation of cuts. If the solution is integral, i.e., all variables are integers, the algorithm stops and returns this solution. Otherwise, this node is inserted into a list and the main loop of the BCP begins. At each iteration, the algorithm chooses from the list the next node to be opened. This choice of a node can be implemented in different ways, each one reflecting in a different search through the tree. For instance, if the list is a stack, the search will be a depth-first search. In our implementation, a priority queue is used, always returning the node with the lowest value. This approach is usually called best-first search and, each time a new node is chosen from the list, its value is considered the new current lower bound. This can be assured by the fact that there is no node with a lower value remaining in the list.

The opening of a node consists in selecting a fractional variable of its solution and creating two branches, following a defined branching rule, which will be explained further. Then, from each branch a new node is generated,

which will be solved in a similar manner as the root node. However, differently from the root node, the value is tested against the known upper bound and, if it is greater or equal than the bound, it is discarded. This operation is usually known as pruning the node of the search tree. On the other hand, if the value is less than the upper bound, the node is considered as a valid node. Furthermore, if the solution of a valid node is integral, its value is then considered as the new upper bound and there is no way to branch from this node anymore. When this is not the case and the solution of a valid node still contains any fractional variable, the node is inserted into the list.

It is important to notice that, at a given iteration, if the chosen node has a value that differs less than one unit from the known upper bound, the BCP algorithm stops and the solution which generated the upper bound is then returned as the overall best solution.

(a) Branching Rules

The most intuitive branching rule is, for a given variable λ_r from the set partitioning formulation with a fractional value $\bar{\lambda_r} \in [0, 1]$, create two branches, the first one with $\lambda_r = 0$ and other with $\lambda_r = 1$. However, this cannot be done due to the column generation algorithm. It is easy to see that if the variable associated with a route r is fixed to zero, eventually the pricing subproblem will find this route again on a subsequent iteration. This route will be returned to the restricted master which will consider it as a new route and will create a new variable λ_r for it.

There are some ways to cope with this difficulty. But instead of that, we prefer to branch on the deadheaded edges variables z_e from the one-index formulation. By the time it is necessary to branch, the values of the z_e variables are obtained using the relation $\sum_{r\in\Omega} b_r^e \lambda_r = z_e, \forall e \in E$, which is analogous to the one created for the two-index formulation, shown in (III.20).

The drawback of this solution is that when an integer solution is found, it is integer just on z_e variables and it may not be integer on the λ_r variables. This integer solution has the same properties of one from the one-index formulation, it may have a value less than the optimal solution, which would also be infeasible. However, it does not prevent the BCP algorithm from returning good solutions, as will be shown in the results.

The selection of the branching variable can follow different rules. The simplest rule is to search for the variable whose value in the solution $\bar{z}_e$ is closer to 0.5 and then create two branches, one with $z_e \leq \lfloor \bar{z}_e \rfloor$ and other with $z_e \geq \lceil \bar{z}_e \rceil$. Mapping these inequalities to λ_r variables results in the following two constraints.

$$\sum_{r\in\Omega} b_r^e \lambda_r \leq \lfloor \bar{z}_e \rfloor \tag{VI.1}$$

$$\sum_{r\in\Omega} b_r^e \lambda_r \geq \lceil \bar{z}_e \rceil \ . \tag{VI.2}$$

Instead of creating these two constraints for each new branch generated during the BCP algorithm, one bound constraint for each deadheaded edge $e \in E$ can be inserted *a priori* in the restricted master as shown below.

$$lb_e \leq \sum_{r\in\Omega} b_r^e \lambda_r \leq ub_e \quad \forall e \in E \ . \tag{VI.3}$$

With the introduction of the bounds constraints, the pricing subproblem does not change, but the reduced cost of an edge, shown in equations (V.2) and (V.3), must consider the dual values associated with these constraints. Therefore, given ρ_e, the dual variable associated with constraints (VI.3), this value must be subtracted from the reduced cost of the associated edge e for both equations.

In this work, the branching rule described above is used. However, there are some different branching rules which could be used. For instance, instead of choosing the variable closer to 0.5, the rule could be to choose the variable closer to any other value between 0 and 1, exclusive. Furthermore, a set of edges could be chosen. One way to do this is to select a vertex set S with $\sum_{e\in\delta(S)} z_e$ fractional and create the associated branches.

VI.2 Improving the Search

(a) Using Strong Branching

Strong Branching is a technique used in the attempt to find better lower bounds faster. Instead of choosing the best fractional variable w.r.t. the branching rule, it uses a candidate set of variables and tests which of them give the best progress before actually perform any branching. This test is done by solving the linear relaxations – in this case the column generation – of the two nodes which would be created from each candidate. If the candidate set is the full set of fractional variables and, for each candidate, both linear relaxations are solved to optimality, this strategy is called *full strong branching*. A benefit of this strategy is to always choose the locally best fractional variable to branch on. However, the effort needed for solving each node of the search tree is high. Alternatively, most strong branchings in literature tries to estimate quickly which variable is a good one to branch on.

One possibility to speed up the strong branching is to evaluate only a subset of the candidate set. In this work, we create the candidate set by choosing the best 5 fractional variables w.r.t. the branching rule. For each candidate s, an estimate on the values of both the linear relaxations is obtained by running the column generation using just the heuristic pricing described on Section V.1(c) and without separating any cut. Let $left_s$ and $right_s$ be these values. In order to choose which candidate is the one to branch, an evaluation criterion is used. The most common one is to select the one with largest $min\{left_s, right_s\}$ and, in the case of a tie, the one with largest $max\{left_s, right_s\}$. We use a more complex criterion, as follows, which usually gives better results than this simple one.

$$\sigma(s) = \alpha \min(left_s, right_s) + (1 - \alpha) \max(left_s, right_s) \tag{VI.4}$$

This function can be calibrated using different values for the parameter $\alpha \in [0, 1]$. During our experiments, we found $\alpha = 3/4$ to be a good choice. Therefore, we branch on the candidate with largest $\sigma(s)$.

As mentioned, during the strong branching evaluation the linear relaxations are not solved to optimality. This approach is called strong branching on pseudo-costs, since we do not know the optimal value of the linear relaxations. This was originally proposed for problems which do not use column generation. In this case, the pseudo-costs are obtained performing only a few iterations of the linear programming solver and these values are used to choose the branching variable. Anyway, in both cases, when the branching variable is finally chosen, the optimal values must be computed. This requires our algorithm to do a complete execution of the column generation for both new nodes of the search tree.

(b) Using Reduced Cost Fixing

Reduced Cost Fixing is a well-known technique which is used to estimate the limits of a variable using its current reduced cost and the known lower and upper bounds. We refer the reader to the 1998 book of Wolsey [79] where a detailed description of its applications is given. The benefit of using this technique is the possible reduction in the number of nodes during the branch-and-bound search if some variable fixings are done at each iteration of the algorithm.

Given a variable z_e with reduced cost $\bar{c}_e$ and solution value $\tilde{z}_e = lb_e$ equals to its current lower bound, it is correct to state that if the value of this variable is incremented by 1, the value of the current solution is incremented

by $\bar{c}_e$. Therefore, knowing the value of the current solution Z^* and an upper bound UB for the solution, it can be concluded that an upper bound for this variable is as follows.

$$ub'_e = lb_e + \left\lfloor \frac{UB - Z^*}{\bar{c}_e} \right\rfloor \tag{VI.5}$$

The upper bound for this variable is updated in case $ub'_e < ub_e$. Analogously, given a variable z_e with solution value $\tilde{z}_e = ub_e$ equals to its current upper bound and a known lower bound LB for the solution, it can also be concluded that a lower bound for the variable is as follows.

$$lb'_e = ub_e - \left\lceil \frac{UB - Z^*}{|\bar{c}_e|} \right\rceil \tag{VI.6}$$

Notice that these bounds can only be computed for variables with non-negative reduced costs $\bar{c}_e \neq 0$, otherwise it would result in a division by zero.

VI.3 Experimental Results

The computational experiments were done using the same configuration as Chapters IV and V. The BCP algorithm was tested on all CARP datasets described on section IV.4, except for the *egl-large* dataset. As mentioned before, the instances of this dataset are prohibitively large to column generation.

The results are shown in Table VI.1. Columns `Ins`, `LB` and `UB` show the name and the known lower and upper bounds for each instance. Following these columns, for each group of columns `NG=X`, where $\Delta(N_i) = X$, columns `Value`, `Routes`, `Cuts`, `Nodes` and `Time` show the solution cost, the number of routes found by the column generation, the number of cuts found by the separation algorithms, the number of nodes opened by the branch-and-bound and the overall time of computation in seconds, respectively. For each instance, a time limit of three hours was used. Furthermore, when a solution cost is better than the known lower bound, its value is highlighted in boldface, and when it is an optimal solution for the instance, it is underlined.

Notice that even with the small time limit used and for small values of $\Delta(N_i)$, the algorithm was able to improve several known lower bounds and it was also able to find some new optimal values for the beullens dataset.

Table VI.1: Branch-Cut-and-Price Results for the CARP

Ins	LB	UB	NG=8					NG=16					NG=32				
			Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time
kshs1	14661	14661	14661.0	116	5	1	0.110	14661.0	92	5	1	0.109	14661.0	92	5	1	0.094
kshs2	9863	9863	9863.0	124	8	1	0.109	9863.0	88	8	1	0.109	9863.0	88	8	1	0.078
kshs3	9320	9320	9320.0	167	4	1	0.141	9320.0	93	4	1	0.125	9320.0	93	4	1	0.125
kshs4	11498	11498	11498.0	134	5	1	0.156	11498.0	99	5	1	0.141	11498.0	99	5	1	0.188
kshs5	10957	10957	10957.0	110	4	1	0.156	10957.0	112	4	1	0.219	10957.0	112	4	1	0.203
kshs6	10197	10197	10197.0	234	4	1	0.172	10197.0	123	4	1	0.109	10197.0	123	4	1	0.188
gdb1	316	316	316.0	259	12	1	0.031	316.0	235	12	1	0.016	316.0	201	12	1	0.031
gdb2	339	339	339.0	260	8	1	0.031	339.0	303	8	1	0.031	339.0	290	8	1	0.031
gdb3	275	275	275.0	217	5	1	0.016	275.0	200	5	1	0.015	275.0	191	5	1	0.016
gdb4	287	287	287.0	126	6	1	0.000	287.0	86	6	1	0.000	287.0	86	6	1	0.016
gdb5	377	377	377.0	228	8	1	0.032	377.0	200	8	1	0.015	377.0	197	8	1	0.031
gdb6	298	298	298.0	158	5	1	0.015	298.0	160	4	1	0.016	298.0	143	4	1	0.016
gdb7	325	325	325.0	219	10	1	0.016	325.0	147	10	1	0.016	325.0	192	10	1	0.015
gdb8	348	348	347.0	944	28	13	1.688	347.0	828	24	11	1.266	347.0	885	25	13	1.672
gdb9	303	303	303.0	813	18	7	1.469	303.0	734	17	3	0.844	303.0	857	19	3	0.922
gdb10	275	275	275.0	245	12	1	0.032	275.0	228	12	1	0.031	275.0	237	13	1	0.047
gdb11	395	395	395.0	1105	13	1	0.953	395.0	925	13	1	0.782	395.0	1105	13	1	0.968
gdb12	458	458	454.0	253	11	5	0.203	455.0	236	11	3	0.156	455.0	202	11	3	0.141
gdb13	536	536	536.0	412	5	1	0.172	536.0	400	5	1	0.172	536.0	291	5	1	0.125
gdb14	100	100	100.0	151	1	1	0.016	100.0	165	1	1	0.031	100.0	135	1	1	0.015
gdb15	58	58	58.0	127	1	3	0.062	58.0	174	1	3	0.109	58.0	220	1	5	0.157
gdb16	127	127	127.0	378	6	3	0.125	127.0	322	6	3	0.110	127.0	228	7	3	0.093
gdb17	91	91	91.0	134	7	1	0.047	91.0	134	7	1	0.093	91.0	74	7	1	0.032
gdb18	164	164	164.0	565	1	1	0.219	164.0	685	1	1	0.297	164.0	385	1	1	0.156
gdb19	55	55	55.0	55	4	1	0.016	55.0	60	4	1	0.015	55.0	60	4	1	0.000
gdb20	121	121	121.0	265	9	1	0.047	121.0	205	10	1	0.063	121.0	187	10	1	0.047

Continued on next page

Table VI.1: *Continued from previous page*

Ins	LB	UB	NG=8					NG=16					NG=32				
			Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time
gdb21	156	156	156.0	454	7	3	0.219	156.0	354	6	3	0.281	156.0	314	7	3	0.172
gdb22	200	200	200.0	452	7	11	0.875	200.0	472	7	9	0.890	200.0	412	6	11	1.015
gdb23	233	233	233.0	412	1	17	1.484	233.0	432	1	7	0.703	233.0	352	1	9	0.859
1A	173	173	173.0	3067	20	3	8.516	173.0	3192	20	3	10.812	173.0	2754	20	3	10.750
1B	173	173	173.0	1762	20	1	2.985	173.0	1711	20	3	4.094	173.0	1809	20	3	4.641
1C	245	245	242.0	1094	28	31	5.750	242.0	696	18	5	1.484	242.0	713	18	5	1.672
2A	227	227	227.0	2107	12	1	4.063	227.0	2138	11	1	5.063	227.0	1250	11	1	6.109
2B	259	259	258.0	2662	25	5	7.515	258.0	2522	25	3	16.141	258.0	1708	25	1	8.219
2C	457	457	457.0	406	16	1	0.328	457.0	398	16	1	0.297	457.0	345	16	1	0.265
3A	81	81	81.0	2589	25	3	4.282	81.0	2867	23	3	5.969	81.0	2340	24	3	5.938
3B	87	87	87.0	2591	35	7	5.344	87.0	2755	36	7	5.797	87.0	2670	40	9	9.594
3C	138	138	136.0	764	13	9	0.750	136.0	792	15	9	0.953	136.0	775	17	9	0.921
4A	400	400	400.0	9283	37	3	110.140	400.0	10281	37	3	141.953	400.0	12046	37	3	247.156
4B	412	412	412.0	4576	34	3	29.515	412.0	5996	36	3	49.375	412.0	6596	34	3	64.203
4C	428	428	428.0	4297	33	3	26.390	428.0	4771	33	3	30.844	428.0	3846	33	3	28.406
4D	530	530	527.0	5302	75	85	149.984	528.0	10580	102	371	1224.310	528.0	6431	78	95	303.688
5A	423	423	423.0	4487	25	3	31.297	423.0	5547	26	3	42.781	423.0	5327	26	3	44.687
5B	446	446	446.0	4901	73	11	60.890	446.0	6615	125	11	102.406	446.0	7771	74	11	129.906
5C	474	474	471.0	8552	119	25	196.500	471.0	6127	127	13	97.890	471.0	8893	125	19	200.469
5D	577	577	573.0	4734	166	27	61.219	573.0	2564	110	11	20.532	573.0	2414	111	9	21.547
6A	223	223	223.0	3475	21	3	13.797	223.0	3849	21	3	16.907	223.0	3458	22	3	14.969
6B	233	233	231.0	6120	86	37	73.422	231.0	6068	125	19	75.266	231.0	2961	79	7	18.718
6C	317	317	313.0	1198	48	25	8.797	313.0	1623	46	111	45.422	313.0	1344	46	57	23.328
7A	279	279	279.0	4507	23	1	23.235	279.0	4019	21	1	24.547	279.0	5376	21	1	40.532
7B	283	283	283.0	3616	21	1	14.156	283.0	3576	21	1	17.563	283.0	3815	21	1	19.469
7C	334	334	330.0	6057	135	71	114.078	330.0	4987	133	47	96.735	330.0	4125	109	37	78.453
8A	386	386	386.0	3167	23	1	15.563	386.0	3127	23	1	16.531	386.0	3567	23	1	21.515

Continued on next page

Table VI.1: *Continued from previous page*

Ins	LB	UB	NG=8					NG=16					NG=32				
			Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time
8B	395	395	395.0	2496	24	1	9.609	395.0	3116	21	1	13.140	395.0	3216	22	1	15.000
8C	521	521	518.0	2715	71	23	23.735	518.0	2506	68	23	23.796	518.0	2108	60	15	15.750
9A	323	323	323.0	9967	54	3	173.109	323.0	11886	54	3	263.953	323.0	11746	54	3	303.500
9B	326	326	326.0	9956	49	3	149.875	326.0	11336	49	3	228.532	326.0	16736	50	3	701.781
9C	332	332	332.0	6025	44	5	64.593	332.0	7765	44	5	107.969	332.0	6485	45	3	77.156
9D	391	391	386.6	24915	753	545	*3h*	387.0	22472	599	429	6646.190	387.0	15283	438	201	2414.980
10A	428	428	428.0	10749	44	3	241.000	428.0	16333	44	3	640.140	428.0	21067	44	3	1516.860
10B	436	436	436.0	8756	59	5	183.438	436.0	9116	59	5	236.032	436.0	10176	58	5	304.297
10C	446	446	446.0	8563	60	9	209.859	446.0	8285	65	7	176.375	446.0	9752	63	7	245.609
10D	526	526	525.0	8806	132	33	260.328	525.0	7110	99	21	140.906	525.0	9474	154	31	303.156
C01	4105	4150	**4125.0**	33153	470	471	6974.130	**4125.0**	20467	345	265	2994.160	**4125.0**	18158	328	209	2917.990
C02	3135	3135	3135.0	2081	32	1	3.453	3135.0	2108	32	3	5.281	3135.0	2040	32	3	6.016
C03	2575	2575	2565.0	5145	113	23	48.016	2565.0	4668	125	27	52.515	2565.0	3938	134	17	45.219
C04	3478	3510	**3495.0**	7310	76	11	55.188	**3500.0**	8775	82	21	127.375	**3500.0**	7333	81	13	125.000
C05	5365	5365	5335.0	5022	90	27	64.078	5335.0	4266	73	23	46.547	5340.0	6215	119	73	179.031
C06	2535	2535	2525.0	6446	76	69	89.735	2525.0	4156	84	9	44.969	2525.0	4543	80	9	47.047
C07	4075	4075	4050.0	5224	106	37	54.485	4050.0	4479	94	31	46.485	4050.0	4690	99	37	55.015
C08	4090	4090	4060.0	4004	155	31	53.953	4060.0	4513	196	33	92.125	4060.0	3767	125	27	68.766
C09	5233	5260	**5240.0**	32727	495	305	6933.890	**5240.0**	19023	390	137	2180.630	**5240.0**	14748	335	87	1172.170
C10	4700	4700	4670.0	4057	176	59	87.563	4685.0	5753	241	119	278.281	4685.0	5022	221	125	285.094
C11	4583	4635	4580.0	10966	317	25	345.438	**4585.0**	11582	339	37	569.312	**4585.0**	9918	288	25	569.234
C12	4209	4240	4200.0	18606	267	271	2175.080	4200.0	18858	243	267	2093.520	4200.0	15971	248	189	1587.270
C13	2955	2955	2940.0	4229	65	17	29.578	2940.0	4129	61	15	27.734	2940.0	4171	63	19	39.500
C14	4030	4030	4010.0	3639	64	31	28.922	4010.0	4222	65	45	63.578	4010.0	2725	59	27	34.125
C15	4912	4940	4910.0	28469	163	209	2734.550	**4920.0**	43642	178	561	*3h*	**4913.3**	33902	151	307	*3h*
C16	1475	1475	1470.0	2201	25	1	3.594	1470.0	2065	28	3	6.735	1470.0	1256	25	1	2935.440
C17	3555	3555	3550.0	1635	49	3	3.218	3550.0	1421	48	3	3.485	3550.0	1205	48	3	3.797

Continued on next page

Table VI.1: *Continued from previous page*

Ins	LB	UB	NG=8					NG=16					NG=32				
			Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time
C18	5577	5620	5564.5	39731	273	150	*3h*	5567.6	33027	179	85	*3h*	5568.6	34423	197	85	*3h*
C19	3096	3115	**3115.0**	16707	290	147	1134.000	**3115.0**	9861	201	65	1142.050	**3115.0**	5337	120	25	1725.640
C20	2120	2120	2120.0	4452	59	3	11.641	2120.0	5022	53	5	29.703	2120.0	4197	38	5	18.953
C21	3960	3970	3960.0	6976	52	7	45.562	3960.0	6005	50	7	70.171	3960.0	5532	53	5	259.312
C22	2245	2245	2245.0	3137	36	3	5.109	2245.0	2244	36	5	5.484	2245.0	1673	36	1	3.828
C23	4032	4085	**4070.0**	38026	383	241	9081.970	**4067.0**	20228	242	60	*3h*	**4056.1**	8877	179	3	*3h*
C24	3384	3400	**3385.0**	18883	372	51	1042.030	**3385.0**	8924	139	15	144.703	**3385.0**	8856	173	25	255.703
C25	2310	2310	2310.0	1899	29	3	2.797	2310.0	1584	28	1	1.593	2310.0	1343	29	3	2.484
D01	3215	3215	3215.0	8421	54	1	66.453	3215.0	10840	53	1	118.704	3215.0	10856	61	1	135.250
D02	2520	2520	2520.0	3618	25	1	9.250	2520.0	4276	25	1	13.344	2520.0	4744	25	1	17.953
D03	2065	2065	2065.0	6715	31	3	30.704	2065.0	6929	30	3	33.984	2065.0	6413	31	3	39.532
D04	2785	2785	2785.0	12240	54	3	156.766	2785.0	12136	52	3	171.468	2785.0	12839	51	3	233.437
D05	3935	3935	3935.0	5850	49	3	35.188	3935.0	6540	52	3	41.734	3935.0	5870	49	3	43.031
D06	2125	2125	2125.0	5901	30	3	23.484	2125.0	6564	30	3	30.078	2125.0	7326	30	3	45.938
D07	3115	3115	3075.0	14750	137	53	600.516	3090.0	17065	179	57	1515.880	3090.0	14226	104	45	*3h*
D08	2995	3045	**3020.0**	17349	103	39	495.625	**3020.0**	16325	89	33	579.000	**3020.0**	14640	89	27	688.891
D09	4120	4120	4120.0	12842	50	3	172.812	4120.0	15510	50	3	319.016	4120.0	20382	49	3	756.891
D10	3340	3340	3335.0	5304	33	3	34.000	3335.0	5148	33	3	27.875	3335.0	4451	33	3	1132.190
D11	3745	3745	3745.0	18943	78	7	531.953	3745.0	20367	80	7	604.890	3745.0	19543	80	7	752.625
D12	3310	3310	3310.0	9794	49	3	88.734	3310.0	10748	49	3	123.860	3310.0	11227	49	3	151.266
D13	2535	2535	2535.0	4279	37	1	12.968	2535.0	4769	36	1	17.032	2535.0	3461	36	1	12.437
D14	3272	3280	**3280.0**	10278	59	11	125.859	**3280.0**	8839	82	9	120.266	—	—	—	—	—
D15	3990	3990	3990.0	19842	55	3	412.141	3990.0	21235	54	3	550.359	3990.0	24021	54	1	656.812
D16	1060	1060	1060.0	5622	19	1	20.500	1060.0	6662	16	1	131.390	—	—	—	—	—
D17	2620	2620	2620.0	3816	27	3	10.172	2620.0	3464	26	5	13.469	2620.0	2244	26	3	7.907
D18	4165	4165	4165.0	19955	67	3	534.844	4165.0	20805	66	3	680.625	4165.0	21352	67	3	850.312
D19	2393	2400	2385.0	20774	87	13	370.360	—	—	—	—	—	—	—	—	—	—

Continued on next page

Table VI.1: *Continued from previous page*

Ins	LB	UB	NG=8					NG=16					NG=32				
			Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time
D20	1870	1870	1870.0	7745	33	3	32.594	1870.0	9467	34	3	76.281	1870.0	8547	34	3	66.500
D21	2985	3050	**3010.0**	48420	116	67	3607.050	**3010.0**	31992	88	49	2229.050	**2995.3**	21954	60	9	*3h*
D22	1865	1865	1865.0	7724	32	3	26.594	1865.0	9213	32	1	43.844	1865.0	5867	32	1	79.453
D23	3114	3130	**3120.0**	33900	116	9	2355.740	**3116.8**	23785	110	3	*3h*	—	—	—	—	—
D24	2676	2710	**2705.0**	61167	218	41	8401.740	**2697.5**	61419	201	21	*3h*	—	—	—	—	—
D25	1815	1815	1815.0	3501	24	1	6.625	1815.0	3841	58	1	10.406	1815.0	3376	24	1	355.859
E01	4885	4910	4870.0	18143	413	439	2719.260	4870.0	15571	321	293	1823.880	4870.0	12827	311	195	1822.480
E02	3990	3990	3975.0	4301	100	25	37.766	3975.0	3950	79	17	31.516	3975.0	3334	65	21	34.922
E03	2015	2015	2015.0	3603	34	3	9.718	2015.0	3372	30	1	5.546	2015.0	2839	30	1	6.312
E04	4155	4155	4145.0	12606	307	31	273.859	4145.0	8560	196	41	1502.480	4145.0	8569	252	29	619.984
E05	4585	4585	4580.0	3110	85	5	15.297	4580.0	3001	75	5	13.828	4585.0	4153	106	17	46.687
E06	2055	2055	2055.0	1668	25	1	2.391	2055.0	1704	25	1	2.125	2055.0	1315	25	1	1.687
E07	4155	4155	4080.0	2961	145	39	42.547	4080.0	2494	130	31	44.953	4080.0	1851	92	13	22.937
E08	4710	4710	4700.0	5454	114	21	56.375	4710.0	5609	129	27	60.016	4710.0	5662	163	23	65.468
E09	5780	5820	5775.0	11761	194	39	492.093	5775.0	9151	173	21	244.031	5780.0	18003	277	213	3092.750
E10	3605	3605	3605.0	1411	58	1	2.078	3605.0	1486	58	1	3.313	3605.0	1098	58	1	1.969
E11	4637	4655	**4645.0**	10111	204	25	305.047	**4645.0**	9216	137	17	223.203	**4650.0**	14520	219	43	1687.860
E12	4180	4180	4165.0	10031	183	197	516.750	4165.0	8475	158	127	324.703	4165.0	8195	151	115	374.657
E13	3345	3345	3330.0	2686	63	5	10.203	3330.0	2519	63	5	11.422	3330.0	2279	63	5	179.797
E14	4115	4115	4105.0	7817	192	179	262.625	4105.0	6092	124	129	190.187	4105.0	5463	139	137	293.500
E15	4189	4205	**4197.6**	47931	177	369	*3h*	**4200.0**	41161	198	273	10450.800	**4197.7**	17654	125	51	*3h*
E16	3755	3775	3775.0	10561	124	53	176.422	3775.0	5599	123	39	118.672	3775.0	6866	115	45	801.266
E17	2740	2740	2740.0	1684	38	3	2.422	2740.0	1565	41	5	4.531	2740.0	1542	39	3	3.297
E18	3825	3835	3825.0	11053	96	13	173.219	**3830.0**	19493	232	43	1215.640	**3830.0**	16008	320	43	3447.670
E19	3222	3235	**3230.0**	10584	144	31	240.453	**3230.0**	8908	108	23	184.312	**3230.0**	7447	121	19	279.844
E20	2802	2825	**2810.0**	6629	110	21	71.672	**2810.0**	6644	126	23	87.172	**2810.0**	4840	73	15	53.235
E21	3728	3730	**3730.0**	8623	67	21	121.235	**3730.0**	8976	79	17	143.515	**3730.0**	7732	79	17	213.984

Continued on next page

Table VI.1: *Continued from previous page*

Ins	LB	UB	NG=8					NG=16					NG=32				
			Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time
E22	2470	2470	**2470.0**	3118	63	11	23.922	**2470.0**	2554	61	9	41.359	**2470.0**	2614	56	7	116.750
E23	3686	3710	**3705.0**	33482	564	143	5479.630	**3705.0**	27254	443	91	4330.250	**3696.2**	8536	178	13	*3h*
E24	4001	4020	**4010.0**	15192	199	71	926.172	**4010.0**	11096	155	21	266.094	**4015.0**	13671	191	47	699.579
E25	1615	1615	1615.0	956	20	1	0.578	1615.0	1155	21	1	1.063	1615.0	662	20	1	1.047
F01	4040	4040	4040.0	11375	93	3	210.640	4040.0	13103	92	3	242.062	4040.0	15593	94	3	451.547
F02	3300	3300	3300.0	6067	49	1	26.218	3300.0	5381	49	1	24.281	3300.0	7296	49	1	52.469
F03	1665	1665	1665.0	6982	36	1	21.141	1665.0	7249	36	1	25.219	1665.0	4442	39	1	48.594
F04	3476	3485	**3485.0**	27270	137	27	1471.310	**3485.0**	31918	109	11	*3h*	**3485.0**	17243	93	5	*3h*
F05	3605	3605	3605.0	4154	65	1	17.594	3605.0	5334	65	3	32.110	3605.0	5431	66	3	39.609
F06	1875	1875	1875.0	3149	28	1	6.500	1875.0	3353	28	1	7.875	1875.0	2812	28	1	7.875
F07	3335	3335	3335.0	5511	72	5	39.125	3335.0	4262	59	3	17.468	3335.0	3445	59	3	13.484
F08	3695	3705	**3705.0**	8869	103	21	163.062	**3705.0**	10759	79	27	249.140	**3705.0**	10168	84	19	271.016
F09	4730	4730	4730.0	12822	88	5	230.609	4730.0	17582	88	5	541.109	4730.0	18022	87	5	629.421
F10	2925	2925	2925.0	4130	44	3	15.016	2925.0	4788	44	3	20.218	2925.0	3695	44	3	16.828
F11	3835	3835	3835.0	14947	85	5	263.625	3835.0	19243	206	5	523.250	3835.0	21815	84	5	818.938
F12	3390	3395	3390.0	17590	92	19	436.734	3390.0	18161	94	19	513.000	3390.0	13644	99	13	409.156
F13	2855	2855	2855.0	3355	48	3	10.953	2855.0	3950	48	3	17.110	2855.0	3632	48	3	14.422
F14	3330	3330	3330.0	4443	78	3	23.687	3330.0	5078	50	1	231.485	3330.0	5333	50	1	2019.550
F15	3560	3560	3560.0	24625	74	9	924.421	3560.0	35989	70	13	2394.890	3560.0	29466	70	13	1497.230
F16	2725	2725	2725.0	7484	38	3	30.328	2725.0	7537	37	1	36.406	2725.0	8646	37	1	3384.970
F17	2055	2055	2055.0	2981	31	1	5.875	2055.0	3206	31	1	8.453	2055.0	1497	31	1	3.125
F18	3063	3075	**3065.0**	19142	74	3	329.297	**3065.0**	21413	66	3	2540.670	—	—	—	—	—
F19	2500	2525	2500.0	22195	148	11	595.969	—	—	—	—	—	—	—	—	—	—
F20	2445	2445	2445.0	9775	54	3	74.265	2445.0	10775	54	3	93.703	2445.0	10455	54	3	106.688
F21	2930	2930	2930.0	9647	51	1	65.797	2930.0	11328	51	1	110.875	2930.0	11865	51	1	138.672
F22	2075	2075	2075.0	5902	50	3	21.703	2075.0	7474	56	3	91.063	2075.0	4756	49	1	322.672
F23	2994	3005	**3005.0**	34350	213	10	*3h*	**3005.0**	34409	194	5	5923.860	—	—	—	—	—

Continued on next page

Table VI.1: *Continued from previous page*

Ins	LB	UB	NG=8					NG=16					NG=32				
			Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time	Value	Routes	Cuts	Nodes	Time
F24	3210	3210	3210.0	13914	88	3	215.125	3210.0	18096	102	3	2245.720	3210.0	22045	88	1	9429.840
F25	1390	1390	1390.0	2023	22	1	2.594	1390.0	2092	22	1	3.984	1390.0	736	22	1	3.750
e1-A	3548	3548	3548.0	3116	55	1	26.344	3548.0	2898	55	1	23.312	3548.0	2623	55	1	18.735
e1-B	4498	4498	4496.0	7402	94	131	449.797	4496.0	6400	86	85	344.562	4496.0	5498	87	89	549.750
e1-C	5595	5595	5556.0	2847	70	49	147.890	5556.0	2010	70	19	59.078	5556.0	1627	70	19	62.375
e2-A	5018	5018	5012.0	6531	88	5	160.000	5018.0	5912	94	7	655.797	5018.0	5472	90	7	5886.940
e2-B	6305	6317	6299.0	5529	93	49	340.343	6299.0	4615	89	21	275.172	6301.0	5326	105	53	1349.550
e2-C	8335	8335	8302.0	4882	99	281	810.688	8308.0	5156	123	439	1290.770	8308.0	4569	139	349	1209.700
e3-A	5898	5898	5898.0	7930	81	5	227.359	5898.0	8096	76	5	322.360	5898.0	9244	75	5	1002.020
e3-B	7711	7775	**7715.0**	15787	178	749	7309.340	**7722.0**	15523	210	831	10135.300	**7722.0**	10861	156	289	7833.480
e3-C	10244	10292	10207.0	4787	131	267	1088.140	10214.0	5750	137	407	1914.380	10214.0	5047	111	291	1538.280
e4-A	6408	6444	6393.0	11099	223	23	1060.330	6393.0	12182	163	17	6083.190	—	—	—	—	—
e4-B	8935	8961	8897.0	7552	110	229	1962.030	8900.0	7543	120	203	2289.560	8912.0	9889	218	689	*3h*
e4-C	11493	11550	11490.0	6519	238	901	5085.380	**11495.0**	6754	192	1189	7936.920	**11495.0**	6072	202	723	5176.520
s1-A	5018	5018	5018.0	6904	143	3	176.984	5018.0	10945	143	3	377.640	5018.0	8825	143	3	417.734
s1-B	6388	6388	6388.0	7400	166	35	465.281	6388.0	6510	160	21	369.219	6388.0	7626	182	29	712.125
s1-C	8518	8518	8511.0	5612	183	169	792.062	8511.0	4223	158	91	520.344	8511.0	4598	172	63	409.750
s2-A	9825	9884	9812.7	23671	767	102	*3h*	9816.8	25049	529	79	*3h*	9811.3	20759	300	45	*3h*
s2-B	13017	13100	12986.8	13353	634	357	*3h*	12991.3	12292	591	295	*3h*	12993.0	11357	488	263	*3h*
s2-C	16425	16425	16376.2	8257	397	725	*3h*	16380.0	8608	317	591	*3h*	16382.3	8381	327	601	*3h*
s3-A	10145	10220	**10153.2**	23485	368	93	*3h*	**10152.6**	14615	226	25	*3h*	**10149.7**	10637	224	3	*3h*
s3-B	13648	13682	13633.1	14969	588	275	*3h*	13635.7	11574	512	237	*3h*	13636.6	11146	469	193	*3h*
s3-C	17188	17188	17126.2	9987	614	531	*3h*	17134.1	8422	466	453	*3h*	17136.8	7842	396	473	*3h*
s4-A	12143	12268	12141.8	22904	387	118	*3h*	**12144.5**	18010	398	51	*3h*	**12143.1**	13142	183	15	*3h*
s4-B	16098	16283	16085.9	11926	789	229	*3h*	**16099.1**	10087	685	135	*3h*	**16098.6**	9591	520	125	*3h*
s4-C	20430	20481	20395.5	6455	438	256	*3h*	20407.3	6681	349	307	*3h*	20409.6	6278	339	309	*3h*

VII
Conclusions

This thesis dealt with exact algorithms for some VRPs. The focus of the research was the CARP, but some algorithms were also extended for the GVRP, which allowed them to be tested for another problem, the CVRP. A brief review of these problems was done at the beginning of the work and then some known formulations were presented for both the CARP and the GVRP.

A first contribution of our work was related to the exact separation of the capacity cuts for the CARP. A new exact separation based on a MIP formulation was presented. The computational tests showed that this separation can perform better in practice than the previous one. Furthermore, this approach was improved by the use of a dual ascent heuristic, which can generate a large set of cuts as a hot-start for the cutting plane. Lower bounds for large scale instances were presented for the first time. Although it has not improved the bounds of any of the classical instances for the CARP, this approach can be used to assist more complex algorithms due to its low running times, as done in the proposed column generation and BCP.

As another contribution we suggested an efficient implementation for the state-of-the-art pricing algorithm, which prices a kind of restricted non-elementary route, called the ng-route. It was done using a heuristic pricing and two techniques to improve the exact pricing. The heuristic pricing is a very simple algorithm, which can run as efficiently as a pricing of non-elementary routes without any restriction. The first technique used during the exact pricing was the DSSR, which relaxes the restrictions imposed by the ng-routes rules and incrementally rebuild these rules, until a feasible solution is found, reducing the overall computational time spent. Along with this technique, completion bounds were used. They act as an estimate for the lower bound on the reduced cost a path can have. It helps the algorithm discarding any path which would lead to routes which would not be in the solution. The results confirmed the efficiency of the proposed algorithms, as it was possible to run the pricing for ng-sets with sizes never used before. Furthermore, some new best lower bounds were obtained and new optimality certificates for GVRP instances were obtained.

Finally, all the proposed algorithms were put together in a branch-cut-and-price algorithm for the CARP. The help of some improvements such as strong branching and reduced cost fixing allowed the algorithm to obtain new lower bounds for open instances, as well as prove new optimal solutions. Even with ng-sets of small sizes and a small time limit, very good results could be obtained.

VII.1 Future Work and Extensions

This thesis provides a wide range of future works. Some of them are listed below:

- The proposed pricing implementation can be tested with other families of cuts as well as with an extended formulation with capacity-indexed variables as the one presented in Pessoa et al. [69];
- A complete branch-cut-and-price for the GVRP can be created in order to try to improve the results on both GVRP and CVRP.

Bibliography

[1] AHR, D.. **Contributions to Multiple Postmen Problems**. PhD thesis, Department of Computer Science, Heidelberg University, 2004. II.1, IV.1(b), IV.4(a)

[2] AMBERG, A.; VOSS, S.. **A Hierarchical Relaxations Lower Bound for the Capacitated Arc Routing Problem**. In: PROCEEDINGS OF THE 35TH ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 2002. II.1

[3] ASSAD, A. A.; PEARN, W. L.; GOLDEN, B. L.. **The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases**. American Journal of Mathematical Management Sciences, 7(1,2):63–88, 1987. II.1

[4] BALDACCI, R.; CHRISTOFIDES, N.; MINGOZZI, A.. **An Exact Algorithm for the Vehicle Routing Problem Based on the Set Partitioning Formulation with Additional Cuts**. Math. Program., 115(2):351–385, 2008. II.2, V.2

[5] BALDACCI, R.; MINGOZZI, A.; ROBERTI, R.. **New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem**. Operations Research, 59(5):1269–1283, 2011. II.2, V.1(b), V.1(b), V.1(c)

[6] BALDACCI, R.; TOTH, P.; VIGO, D.. **Exact algorithms for routing problems under vehicle capacity constraints**. Annals of Operations Research, 175:213–245, 2010. II.2

[7] BARTOLINI, E.; CORDEAU, J.-F.; LAPORTE, G.. **Improved Lower Bounds and Exact Algorithm for the Capacitated Arc Routing Problem**. Technical Report CIRRELT-2011-33, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, 2011. II.1, V.1(b), V.2

[8] BEKTAŞ, T.; ERDOĞAN, G.; RØPKE, S.. **Formulations and Branch-and-Cut Algorithms for the Generalized Vehicle Routing Prob-**

lem. Transportation Science, 45(3):299–316, 2011. II.3, III.3(a), III.3(a), V.2

[9] BELENGUER, J. M.; BENAVENT, E.. **Polyhedral Results on the Capacitated Arc Routing Problem.** Technical Report TR 1-92, Departamento de Estadística e Investigación Operativa, Universitat de València, 1992. II.1

[10] BELENGUER, J. M.; BENAVENT, E.. **A Cutting Plane Algorithm for the Capacitated Arc Routing Problem.** Computers & Operations Research, 30:705–28, 2003. II.1, III.2(b)

[11] BELENGUER, J. M.; BENAVENT, E.. **The capacitated arc routing problem: Valid inequalities and facets.** Computational Optimization and Applications, 10:165–187, 1998. III.2(a)

[12] BENAVENT, E.; CAMPOS, V.; CORBERAN, A.; MOTA, E.. **The Capacitated Arc Routing Problem: Lower Bounds.** Networks, 22:669–690, 1992. IV.4

[13] BEULLENS, P.; MUYLDERMANS, L.; CATTRYSSE, D.; VAN OUDHEUSDEN, D.. **A Guided Local Search Heuristic for the Capacitated Arc Routing Problem.** European Journal of Operational Research, 147(3):629–643, 2003. II.1, IV.4

[14] BILDE, O.; KRARUP, J.. **Besternmelse of optimal beliggenhed af produktionssteder.** Technical report, IMSOR, The Technical University of Denmark, 1967. IV.3

[15] BILDE, O.; KRARUP, J.. **Sharp lower bounds and efficient algorithms for the simple plant location problem.** In: P.L. Hammer, E.L. Johnson, B. K.; Nemhauser, G., editors, STUDIES IN INTEGER PROGRAMMING, v. 1 of **Annals of Discrete Mathematics**, p. 79–97. Elsevier, 1977. IV.3

[16] BODE, C.; IRNICH, S.. **Cut-First Branch-and-Price-Second for the Capacitated Arc Routing Problem.** Technical Report LM-2011-01, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University, Mainz, Germany, 2011. II.1, V.2

[17] BRANDÃO, J.; EGLESE, R.. **A Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem.** Computers & Operations Research, 35:1112–1126, 2008. II.1, IV, IV.4, IV.4(b)

[18] CHAPLEAU, L.; FERLAND, J. A.; LAPALME, G.; ROUSSEAU, J. M.. **A Parallel Insert Method for the Capacitated Arc Routing Problem**. Operations Research Letters, 3(2):95–99, 1984. II.1

[19] CHRISTOFIDES, N.; MINGOZZI, A.; TOTH, P.. **Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations**. Mathematical Programming, 20:255–282, 1981. V.1(a), V.1(a)

[20] CORDEAU, J.-F.; LAPORTE, G.; SAVELSBERGH, M. W. P.; VIGO, D.. **Transportation, v. 14 of Handbooks in Operations Research and Management Science**, ch. Vehicle Routing, p. 367–428. Elsevier, Amsterdam, 2007. II

[21] DANTZIG, G. B.; RAMSER, J. H.. **The truck dispatching problem**. Management Science, 6(1):80–91, 1959. I, II.2

[22] DEARMON, J. S.. **A Comparison of Heuristics for the Capacitated Chinese Postman Problem**. Master's thesis, University of Maryland, College Park, MD, USA, 1981. IV.4

[23] DOERNER, K.; HARTL, R.; MANIEZZO, V.; REIMANN, M.. **An Ant System Metaheuristic for the Capacitated Arc Routing Problem**. In: PROCEEDINGS OF THE 5TH METAHEURISTICS INTERNATIONAL CONFERENCE, TOKYO, JAPAN, 2003. II.1

[24] DROR, M.. **Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW**. Operations Research, 42(5):977–978, 1994. V.1(a)

[25] EDMONDS, J.; KARP, R.. **Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems**. J. ACM, 19:248–264, April 1972. IV.1(a)

[26] EGLESE, R. W.. **Routing Winter Gritting Vehicles**. Discrete Applied Mathematics, 48(3):231–244, 1994. II.1

[27] ERLENKOTTER, D.. **A dual-based procedure for uncapacitated facility location**. Operations Research, 26(6):992–1009, 1978. IV.3

[28] FISCHETTI, M.; LODI, A.. **Optimizing Over the First Chvátal Closure**. Mathematical Programming, 110:3–20, 2007. IV.2

[29] FISCHETTI, M.; SALAZAR-GONZÁLEZ, J.; TOTH, P.. **A Branch-And-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem**. Operations Research, 45(3):378–394, 1997. V.2

[30] FORD, L. R., J.; FULKERSON, D. R.. **A suggested computation for maximal multi-commodity network flows**. Management Science, 5(1):97–101, 1958. I

[31] FUKASAWA, R.; LONGO, H.; LYSGAARD, J.; POGGI DE ARAGÃO, M.; REIS, M.; UCHOA, E.; WERNECK, R. F.. **Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem**. Mathematical Programming, 106(3):491–511, 2006. II.2, IV.1(b), V.1(a)

[32] GAREY, M. R.; JOHNSON, D. S.. **Computers and Intractability; A Guide to the Theory of NP-Completeness**. W. H. Freeman, New York, 1979. I, II.2, III.2(b), III.3(a)

[33] GENDREAU, M.; POTVIN, J.-Y.; BRÄUMLAYSY, O.; HASLE, G.; LØKKETANGEN, A.. **Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography**. In: Golden, B.; Raghavan, S.; Wasil, E.; Sharda, R.; Voß, S., editors, THE VEHICLE ROUTING PROBLEM: LATEST ADVANCES AND NEW CHALLENGES, v. 43 of **Operations Research/Computer Science Interfaces Series**, p. 143–169. Springer US, 2008. II.2

[34] GHIANI, G.; IMPROTA, G.. **An Efficient Transformation of the Generalized Vehicle Routing Problem**. European Journal of Operational Research, 122(1):11–17, 2000. II.3

[35] GILMORE, P. C.; GOMORY, R. E.. **A linear programming approach to the cutting-stock problem**. Operations Research, 9(6):849–859, 1961. I

[36] GOLDEN, B. L.; ASSAD, A. A.. **Vehicle Routing: Methods and Studies**. North-Holland, Amsterdam, 1988. II

[37] GOLDEN, B. L.; DEARMON, J. S.; BAKER, E. K.. **Computational Experiments with Algorithms for a Class of Routing Problems**. Computers & Operations Research, 10(1):47–59, 1983. II.1, IV.4

[38] GOLDEN, B. L.; WONG, R. T.. **Capacitated Arc Routing Problems**. Networks, 11:305–315, 1981. II.1

[39] GOLDEN, B.; RAGHAVAN, S.; WASIL, E.. **The Vehicle Routing Problem: Latest Advances and New Challenges**. Springer, New York, 2008. II

[40] GÓMEZ-CABRERO, D.; BELENGUER, J. M.; BENAVENT, E.. **Cutting Plane and Column Generation for the Capacitated Arc Routing Problem**. In: ORP3, VALENCIA, 2005. II.1

[41] GOMORY, R. E.; HU, T. C.. **Multi-Terminal Network Flows**. Journal of the Society for Industrial and Applied Mathematics, 9(4):551–570, 1961. IV.1(a)

[42] HERTZ, A.; LAPORTE, G.; MITTAZ, M.. **A Tabu Search Heuristic for the Capacitated Arc Routing Problem**. Operations Research, 48(1):129–135, 2000. II.1

[43] HIRABAYASHI, R.; NISHIDA, N.; SARUWATARI, Y.. **Node Duplication Lower Bounds for the Capacitated Arc Routing Problems**. Journal of the Operations Research Society of Japan, 35(2):119–33, 1992. II.1

[44] HIRABAYASHI, R.; NISHIDA, N.; SARUWATARI, Y.. **Tour Construction Algorithm for the Capacitated Arc Routing Problem**. Asia?Pacific Journal of Operational Research, 9:155–75, 1992. II.1

[45] IRNICH, S.; VILLENEUVE, D.. **The Shortest-Path Problem with Resource Constraints and k-Cycle Elimination for $k \geq 3$**. INFORMS Journal on Computing, 18(3):391–406, 2006. V.1(a)

[46] KIUCHI, M.; SHINANO, Y.; HIRABAYASHI, R.; SARUWATARI, Y.. **An Exact Algorithm for the Capacitated Arc Routing Problem Using Parallel Branch and Bound Method**. In: ABSTRACTS OF THE 1995 SPRING NATIONAL CONFERENCE OF THE OPERATIONAL RESEARCH SOCIETY OF JAPAN, p. 28–29, 1995. in Japanese. IV.4

[47] KRUSKAL, JR., J.. **On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem**. Proceedings of the American Mathematical Society, 7(1):48–50, 1956. IV.3(b)

[48] LACOMME, P.; PRINS, C.; RAMDANE-CHÉRIF, W.. **A Genetic Algorithm for the Capacitated Arc Routing Problem and Its Extensions**. In: Boers, E., editor, APPLICATIONS OF EVOLUTIONARY COMPUTING, v. 2037 of **Lecture Notes in Computer Science**, p. 473–483. Springer-Verlag, Berlin, 2001. II.1

[49] LACOMME, P.; PRINS, C.; RAMDANE-CHÉRIF, W.. **Competitive Memetic Algorithms for Arc Routing Problems**. Annals of Operations Research, 131:159–185, 2004. II.1

[50] LAGANÁ, D.; GHIANI, G.; MUSMANNO, R.; LAPORTE, G.. **A Branch-and-Cut Algorithm for the Undirected Capacitated Arc Routing Problem**. Les Cahiers du GERAD, (G-2007-39), 2007. II.1, III.2(a)

[51] LAPORTE, G.. **Fifty years of vehicle routing**. Transportation Science, 43(4):408–416, 2009. II.2

[52] LETCHFORD, A. N.. **Polyhedral Results for Some Constrained Arc-Routing Problems**. PhD thesis, Lancaster University, 1996. II.1

[53] LETCHFORD, A.; OUKIL, A.. **Exploiting Sparsity in Pricing Routines for the Capacitated Arc Routing Problem**. Computers & Operations Research, 36:2320–2327, 2009. II.1, III.2(a), III.2(a), III.2(b)

[54] LI, L. Y. O.. **Vehicle Routing for Winter Gritting**. PhD thesis, Department of Management Science, Lancaster University, 1992. IV.4

[55] LI, L. Y. O.; EGLESE, R. W.. **An Interactive Algorithm for Vehicle Routing for Winter-Gritting**. Journal of the Operational Research Society, 47:217–228, 1996. IV.4

[56] LYSGAARD, J.. **CVRPSEP: A Package of Separation Routines for the Capacitated Vehicle Routing Problem**. Technical report, Department of Management Science and Logistics, Aarhus School of Business, 2003. V.2

[57] LYSGAARD, J.; LETCHFORD, A. N.; EGLESE, R. W.. **A new branch-and-cut algorithm for the capacitated vehicle routing problem**. Mathematical Programming, 100:423–445, 2004. II.2

[58] MARTINELLI, R.; PECIN, D.; POGGI, M.; LONGO, H.. **Column generation bounds for the capacitated arc routing problem**. In: Proceedings of the XLII Simpósio Brasileiro de Pesquisa Operacional, 2010. I.1, II.1

[59] MARTINELLI, R.; PECIN, D.; POGGI, M.; LONGO, H.. **A Branch-Cut-and-Price Algorithm for the Capacitated Arc Routing Problem**. In: Pardalos, P.; Rebennack, S., editors, EXPERIMENTAL ALGORITHMS, v. 6630 of **Lecture Notes in Computer Science**, p. 315–326. Springer-Verlag, Berlin, 2011. I.1, II.1, V.2

[60] MARTINELLI, R.; POGGI, M.; SUBRAMANIAN, A.. **Improved Bounds for Large Scale Capacitated Arc Routing Problem**. Technical Report MCC14/11, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Brazil, 2011. I.1, II.1

[61] MEI, Y.; TANG, K.; YAO, X.. **A Global Repair Operator for Capacitated Arc Routing Problem**. IEEE Transactions on Systems, Man and Cybernetics, 39(3):723–734, 2009. II.1

[62] MEI, Y.; TANG, K.; YAO, X.. **Improved Memetic Algorithm for Capacitated Arc Routing Problem**. In: PROCEEDINGS OF THE ELEVENTH CONFERENCE ON CONGRESS ON EVOLUTIONARY COMPUTATION, CEC'09, p. 1699–1706, Piscataway, NJ, USA, 2009. IEEE Press. II.1

[63] PADBERG, M. W.; RAO, M. R.. **Odd Minimum Cut-sets and b-matchings**. Mathematics of Operations Research, 7:67–80, 1982. IV.1(a)

[64] PEARN, W. L.. **Approximate Solutions for the Capacitated Arc Routing Problem**. Computers & Operations Research, 16(6):589–600, 1989. II.1

[65] PEARN, W. L.. **Augment-Insert Algorithms for the Capacitated Arc Routing Problem**. Computers & Operations Research, 18(2):189–198, 1991. II.1

[66] PEARN, W. L.. **New Lower Bounds for the Capacitated Arc Routing Problem**. Networks, 18:181–191, 1988. II.1

[67] PECIN, D. G.. **Uso de Rotas Elementares no CVRP**. Master's thesis, Instituto de Informática, Universidade Federal de Goiás, 2010. V.1(c), V.2

[68] PECIN, D.; PESSOA, A.; POGGI, M.; UCHOA, E.. **Experiments with New Cuts on the VRP**. In: EUROPEAN CONFERENCE ON OPERATIONS RESEARCH XXIV, 2010. V.2

[69] PESSOA, A.; UCHOA, E.; POGGI, M.. **Robust branch-cut-and-price algorithms for vehicle routing problems**. In: Golden, B.; Raghavan, S.; Wasil, E.; Sharda, R.; Voß, S., editors, THE VEHICLE ROUTING PROBLEM: LATEST ADVANCES AND NEW CHALLENGES, v. 43 of **Operations Research/Computer Science Interfaces Series**, p. 297–325. Springer US, 2008. II.2, VII.1

[70] POGGI DE ARAGÃO, M.; UCHOA, E.; WERNECK, R. F.. **Dual heuristics on the exact solution of large steiner problems**. Electronic Notes in Discrete Mathematics, 7:150–153, 2001. Brazilian Symposium on Graphs, Algorithms and Combinatorics. IV.3

[71] RALPHS, T.. **Branch-cut-and-price resource web**. www.branchandcut.org. Visited on August, 2012. V.2

[72] RIGHINI, G.; SALANI, M.. **New Dynamic Programming Algorithms for the Resource Constrained Elementary Shortest Path Problem**. Networks, 51(3):155–170, 2008. V, V.1(c)

[73] SANTOS, L.; COUTINHO-RODRIGUES, J.; CURRENT, J.. **An improved ant colony optimization based algorithm for the capacitated arc routing problem**. Transportation Research Part B, 44(2):246–266, 2010. II.1

[74] SUBRAMANIAN, A.. **Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems**. PhD thesis, Instituto de Computação, Universidade Federal Fluminense, 2012. V.2

[75] TOTH, P.; VIGO, D.. **The Vehicle Routing Problem**. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2002. I, II

[76] WELZ, S. A.. **Optimal Solutions for the Capacitated Arc Routing Problem Using Integer Programming**. PhD thesis, Department of QT and OM, University of Cincinnati, 1994. III.2(a)

[77] WØHLK, S.. **Contributions to Arc Routing**. PhD thesis, Faculty of Social Sciences, University of Southern Denmark, 2005. II.1

[78] WØHLK, S.. **New Lower Bound for the Capacitated Arc Routing Problem**. Computers & Operations Research, 33(12):3458–3472, 2006. II.1

[79] WOLSEY, L. A.. **Integer Programming**. Wiley-Interscience, 1998. VI.2(b)