UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGINEERING

Master Degree in Computer Engineering

On the Exact Solution of the Capacitated Arc Routing Problem

Supervisor
Prof. Roberto Roberti

Candidato
Nicola Farronato

Graduation Date : November 30, 2023
Academic Year: 2022-2023

Summary

Vehicle Routing Problems (VRPs) are fundamental combinatorial optimization challenges that arise in various real-world applications, encompassing delivery services, transportation logistics, and supply chain management. These problems involve efficiently routing a fleet of vehicles to service a set of geographically dispersed locations while minimizing overall costs, such as travel distance, time, or fuel consumption. The *Capacitated Arc Routing Problem* (CARP) is a specific variant where vehicles are required to traverse predefined arcs to deliver goods or provide services.

This thesis examines the transformation of the *Capacitated Arc Routing Problem* to a modified version of the *Capacitated Vehicle Routing Problem* (CVRP) formulation, first introduced by Baldacci and Maniezzo [7] in 2006. *Capacitated Vehicle Routing Problem* is the most extensively studied version concerning vehicle routing: its objective is to transport products from a depot to customers while reducing total transportation expenses, and adhering to capacity constraints on each vehicle.

The exact solution to these complex problems today is addressed to *Branch-and-Cut-and-Price* (BCP) algorithms, that integrate a branch-and-cut framework with column generation techniques, addressing the inherent complexity of the problem through a decomposition approach. The state-of-the-art B*ranch-and-Cut-and-Price* algorithms are represented by the *VRPSolver* framework, by Pessoa et al. [68].

The aim of this study is to modify the *Capacitated Vehicle Routing Problem* to model it as a *Resource Constraint Shortest Path Problem* (RCSPP) and combine it with features from the capacitated arc routing literature, such as valid inequalities and branching properties, to adapt it to the *VRPSolver*.

Computational experiments demonstrate promising results by solving almost all classical instances from the literature. Furthermore, the efficacy of this solution could be compared to the cutting-edge approach for the capacitated arc routing problem proposed by Pecin and Uchoa [63].

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a fundamental combinatorial optimization challenge in the field of operations research and logistics. It deals with efficiently distributing goods or services from a central depot to a set of geographically dispersed customers or delivery points using a fleet of vehicles. The objective is to design optimal routes for each vehicle to minimize the total cost, which is often measured in terms of distance traveled, time taken, fuel consumption, or other related metrics.

The VRP is paramount in various industries and applications due to its potential to optimize logistical operations and reduce operational costs. It is a foundational problem in transportation and distribution, impacting industries such as e-commerce, retail, waste management, healthcare, and public transportation. By finding efficient delivery routes, VRP contributes to reduced fuel consumption, lower transportation costs, improved customer satisfaction through timely deliveries, and overall enhancement of operational efficiency. Consequently, it serves as a crucial tool for organizations striving to streamline their supply chain and logistics management.

The VRP is a rich problem with many variations and extensions, each presenting its own set of challenges. Some common variants include the Capacitated VRP (CVRP), where each vehicle has a fixed capacity limit; the VRP with Time Windows (VRPTW), which adds time constraints for customer visits; the Multiple Depot VRP (MDVRP), involving multiple depots; the VRP with Split Deliveries (SDVRP), allowing deliveries to be split among multiple vehicles; and the Capacitated Arc Routing Problem (CARP), that focuses on finding minimum cost set of tours that services a subset of streets with positive demand under capacity constraints.

Among all VRPs, the CVRP is of primary interest in literature due to its generality. The next sections will highlight several problems that can be addressed by the CVRP, including the significant issue of CARP. Despite its different roots, the exact solution now utilizes many features found in CVRP solvers.

1.1.1 Capacitated Vehicle Routing Problem (CVRP)

The problem was proposed in the 1959 paper developed by Dantzig and Ramser [28], under the name *truck dispatching problem*, as the optimal planning of routes for a fleet of gasoline delivery trucks traveling between a bulk terminal and a multitude of service stations supplied by the terminal. The paper proposes a mathematical programming formulation and an algorithmic approach. Clarke and Wright [23] offered the first successful approach in 1964, utilizing a greedy heuristic to achieve an approximated solution.

Over the next sixty years, CVRP became one of the most studied problems and assumed a major role in conferences and papers. For instance, Google Scholar found 728 works published in 2017 containing both "vehicle" and "routing" in the title. Researchers over the years have proposed both exact and heuristic solutions, solving increasingly complex problems of different versions of the CVRP. However, despite the steps taken and the real-world relevance, CVRP can only be solved by exact algorithms for relatively small instances, on the order of hundreds of customers.

The problem is indeed NP-hard in the strong sense since CVRP is a generalization of the well-known Travelling Salesman Problem (TSP). Given a list of cities and the distance between each pair of cities, the TSP [38] aims to find the shortest route that visits each city exactly once and returns to the starting city; in other words, it's about finding the most efficient way for a traveling salesman to visit all the cities while minimizing the total distance traveled. TSP is an NP-hard problem [42], that has been widely studied in literature, achieving very complex and efficient algorithms, reaching to solve exactly a 85900-city tour instance [3] using *Concorde* software. The TSP is thus a special case of the CVRP with one vehicle and no capacity constraints. The CVRP, on the other hand, is a more practical and realistic problem for many real-world applications and requires more complex routing strategies and techniques to achieve the solution.

The complexity of CVRP inspired researchers to devise extremely intricate algorithms: starting from the initial branch-and-bound methods, to the branch-and-cut algorithms that dominated for several years, now branch-and-cut-and-price produces optimal outcomes. Such algorithms require resolving intricate sub-problems via dynamic programming and demand extensive effort to implement.

The reader is invited to consult book *The Vehicle Routing Problem* by Toth and Vigo [73], for an overview of the subject, methodologies, and algorithms that facilitated the current problem-solving solution.

1.1.2 Capacitated Arc Routing Problem (CARP)

The CARP finds its roots in the Arc Routing Problem, which arose from Euler's study of the Königsberg Bridge Problem. The problem centered on whether anyone could discover a closed walk that covered all seven bridges, without retracing any bridge. Euler devised a graphical representation of the problem and demonstrated

the infeasibility of locating such a path. He extended his research to other graphs, identifying the Eulerian cycle as the path in a graph that traverses each edge.

Another closely related problem was introduced by the Chinese mathematician Meigu Guan [49], who in 1962 described it as "A mailman has to cover his assigned segment before returning to the post office. The problem is to find the shortest walking distance for the mailman". The problem is now called the Chinese Postman Problem (CPP) and aims to find the minimum total distance subset of edges on a connected undirected graph that, when added to the original graph, results in an Eulerian graph.

A significant variant of the CPP is the Rural Postman Problem (RPP), introduced in 1974 by Oroloff [61], in which certain arcs necessitate service while others can only be used for traversal purposes. The RPP was demonstrated to be NP-hard [51].

CARP was first described in 1981 by Golden and Wong [45] introducing a demand for each required arc in an RPP. In the context of the CARP, the goal is to identify a series of routes for a fleet of vehicles with limited capacity originating from a central location. Each route must fulfill the demand while adhering to the vehicle's specific capacity and minimizing the overall cost.

The application range of such problem includes:

- Street Cleaning - trucks are equipped with a rotating brush, sweeping the material into the truck that must be emptied;
- Salt Spreading - Special vehicles spread salt on icy roads. The capacity can be understood as the amount of salt available for each vehicle. Additionally, the routes can be prioritized based on the level of traffic to determine which areas are in greatest need of cleaning;
- Road maintenance - In this scenario, a fleet of vehicles has to inspect a predetermined set of roads to visually assess the operational status of the said routes, identify any existing damages, and perform any other necessary tasks. The time available for each vehicle every day is restricted, resulting in the development of a model based on a CARP, in which the available time dictates the maximum length of each route.

Regarding the CVRP, the CARP solution employs identical procedures. Additionally, numerous algorithms include particular cuts and characteristics associated with the CVRP to resolve the CARP with efficiency, as indicated in chapter 3 for the reader's reference. Today, branch-and-cut-and-price algorithms are the most noteworthy, as they lead to the solution of almost all classical literature instances.

The reader is finally referred to the book [26] for further information regarding Arc Routing Problems.

1.2 Contributions

Although their origins lay in different problems, today the solution of CARP is closely linked to the progress made with CVRP. This work proposes a revisitation of the transformation from CARP to CVRP, proposed by Baldacci and Maniezzo in 2006 [7]. This transformation, outlined in chapter 5, translates a CARP instance with $|R|$ required edges in a CVRP one with $2|R| + 1$ nodes. The authors created a *Branch-and-cut*(BC) algorithm, which was considered the most advanced technique for the CVRP during that period. The algorithm achieved improved lower bounds for certain classical instances and managed to solve instances requiring up to 98 edges at optimality.

Nowadays *Branch-and-Cut-and-Price*(BPC), a technology based on *Column-Generation*(CG), is the state-of-the-art technique for solving the CVRP. Since the transformation of Baldacci and Maniezzo is very general, this study aims to test it using the BPC technique. To do that, the *VRPSolver* from Pessoa et al. [68] was employed. VRPSolver is a generic solver that incorporates all major contributions developed through the years for the VRPs solution. Extensive experimentation across various variants has demonstrated that the all-purpose solver has an exceptional performance overall, surpassing the best specific algorithms in numerous problems.

The proposed solution will be implemented through VRPSolver, tested on all classical instances, and will be compared to the most successful methods for the exact solution of CARP.

1.3 Outline

The initial chapter of this paper introduces and explores two related issues, CVRP and CARP, tracing their origins and revealing their interdependence. In chapter 2 both issues are formalized using mathematical and integer programming models. Chapter 3 provides a comprehensive review of essential stages in literature, from classical *branch-and-cut* techniques implemented in the CVRP to cutting-edge methods for addressing CVRP and CARP, i.e. *Branch-and-cut-and-Price* (BCP). Chapter 4 presents an outline of the BCP method, showcasing its salient features and recent advancements. Chapter 5 describes the proposed experiment, detailing the transformation from CVRP to CARP, and also the details of the implementation through the VRPSolver. The results are presented in chapter 6, revealing the overall performance of the proposed experiment along with a comparison with the state-of-the-art results of CARPs. Finally, Chapter 7 summarises the work by presenting the conclusion, along with potential future improvements.

Chapter 2

Problem Statement

Integer Programming (IP) is a mathematical optimization method employed to resolve decision and allocation issues where some or all of the decision variables should take on integer values. It expands on the concept of linear programming, which manages continuous decision variables, by stipulating that certain variables must be integers. This additional constraint makes integer programming particularly well-suited for problems requiring whole number decisions, such as determining the number of items to manufacture, routes for vehicles to take, or projects to undertake. IP formulation is composed of some fundamental components such as decision variables, an objective function to optimize, and a set of constraints.

To solve an IP problem, one must discover the best values for the decision variables that meet all constraints while also achieving the highest(lowest) possible objective function value. Certain IP solvers and algorithms have been developed specifically to tackle these types of issues. Some examples of well-known solvers are CPLEX, Gurobi, and SCIP. These solvers employ techniques such as branch and bound, cutting planes, and heuristics to address the problem efficiently.

In this chapter, the (Mixed) Integer Programming (MIP and IP) models for CVRP and CARP will be described, following the basic mathematical description of these problems.

2.1 CVRP Problem Description

The CVRP requires to transport of goods from a single *depot*, denoted as point 0, to a set of N *costumers* described by n points, $N = \{1, \dots, n\}$. Each customer $i \in N$ requires a given amount or weight of goods called *demand*, formulated as $q_i \geq 0$. The *fleet* is composed by $M = \{1, \dots, m\}$ identical vehicles; each vehicle has a fixed *capacity* $Q > 0$. A vehicle that satisfies a customer set $S \subseteq N$, would start at the depot, visit each customer in the set S and finally return to the depot. Each move from i to j has a cost d_{ij}.

Therefore the CVRP can be described as follows. An undirected graph $G = (V', E)$, where $V' = \{0,1,\ldots,n\}$ is the set of $n+1$ vertices and E is the set of edges. The depot is represented by node 0 while the costumers are described by the set $V = V' \setminus \{0\}$. G is complete, with edge set $E = \{e = (i,j) = (j,i) : i,j \in V, i \neq j\}$, and edge cost d_{ij} for $\{i,j\} \in E$. An instance of CVRP is thus defined by the weighted graph $G = (V, E, d_{ij}, q_i)$.

A *route* is defined as a least-cost elementary cycle $R = (0, i_1, \ldots, i_h, 0)$, in which the set $S = \{i_1, \ldots, i_h\} \subseteq N$ of costumers is visited. A route requires that the total demands of the customers visited do not exceed the vehicle capacity i.e. $q(S) = \sum_{i \in S} q_i \leq Q$; moreover, no customer is visited more than once.

Finally, the CVRP aims to design M least-cost routes, one for each vehicle, such that all customers are visited exactly once.

Many mathematical formulations have been proposed for the CVRP, but only a few of them were used for implementing exact algorithms. In this section, after introducing fundamental notation, the most important formulations are explained. It is possible to divide the formulations, depending on the exact algorithm to solve the problem, between *branch-and-cut* algorithms and *set partitioning formulation* algorithms.

As example, in Figure 2.1 instance E-n51-k5 is depicted. This instance has 51 nodes and 5 vehicles. In the image, the five optimal routes are highlited with different colors. The paths from and to the depot node are denoted by dashed lines.

Notation

Let $S \subseteq V$ be a subset of vertices. Given an undirected graph, a *cutset* is defined as the set of edges with exactly one (both) endpoint(s) in S, $\delta(S) = \{\{i,j\} \in E : i \in S, j \notin S || i \notin S, j \in S\}$. For a customer subset $S \subseteq N$, let $r(S)$ be the minimum number of vehicle routes needed to serve S. In the CVRP the number $r(S)$ can be computed solving a *bin packing* problem, with N items of weight $q_i, i \in N$ and bins of size Q.

2.1.1 Two-Index Vehicle Flow Formulations

The first formulation is the **two-index vehicle flow** introduced by Laporte, Mercure, and Nobert in 1985 [50]. This formulation has a polynomial number of variables with respect to $n = |N|$ and an exponential number of constraints. The two-index vehicle flow is suited to solve simple VRPs with mathematical programming techniques i.e. using a direct MIP solver and/or branch-and-cut algorithms.

Let $\mathcal{S} = \{S : S \subseteq V, |S| \geq 2\}$, and let $q(S) = \sum_{i \in S} q_i$ be the total demand of costumers in $S \in \mathcal{S}$ and $k(s)$ the minimum number of vehicles of capacity Q needed to service all costumers in $\mathcal{S}$.

Let x_{ij} be an integer variable that takes value $\{0,1\}$ $\forall \{i,j\} \in E \setminus \{\{0,j\} : j \in V\}$

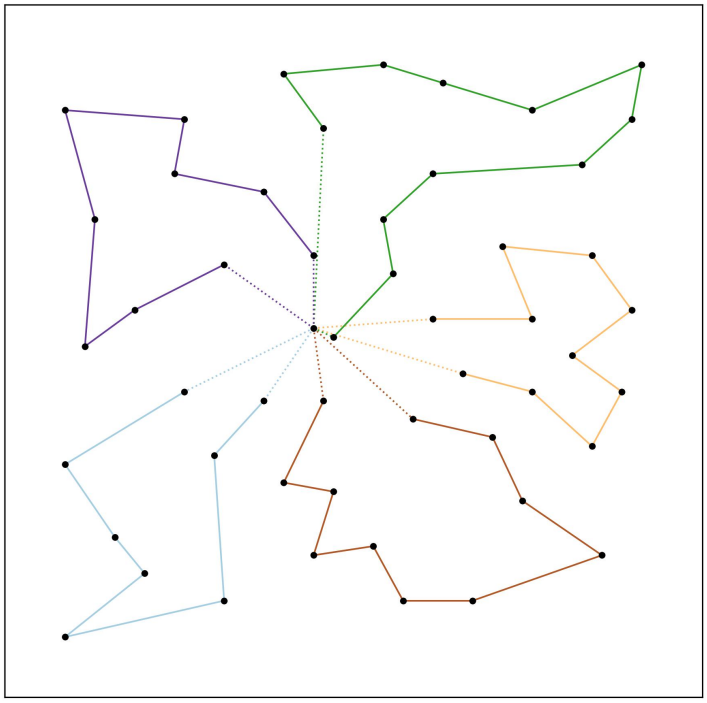


Figure 2.1: CVRP instance E-n51-k5 solution, with 51 nodes and 5 vehicles.

and value {0,1,2} $\forall\{0,j\} \in E, j \in V$. Note that $x_{0j} = 2$ when a route including the single customer j is selected in the solution.

The two-index vehicle flow formulation for the CVRP can be described as the following integer programming (IP).

$$\begin{aligned}
z_{F1} = \text{minimize} \quad & \sum_{ij \in E} d_{ij} x_{ij} && && (2.1a)\\
\text{subject to} \quad & \sum_{ij \in \delta(\{h\})} x_{ij} = 2, && \forall h \in V, && (2.1b)\\
& \sum_{ij \in \delta(S)} x_{ij} \geq 2k(S), && \forall S \in \mathcal{S}, && (2.1c)\\
& \sum_{j \in V} = 2m, && && (2.1d)\\
& x_{ij} \in \{0,1\}, && \forall\{i,j\} \in E \setminus \{\{0,j\} : j \in V\}, && (2.1e)\\
& x_{0j} \in \{0,1,2\}, && \forall\{0,j\}, j \in V && (2.1f)
\end{aligned}$$

The objective value to minimize is given by (2.1a). Constraints (2.1b) are the degree constraints stating that each customer is connected to a predecessor and a successor. (2.1c) serves at the same time as *capacity constraints* and *Subtour Elimination Constraints* (SECs) which, for any subset S of costumer that does not include the depot, impose that $r(S)$ vehicles enter and leave S. For instance, consider an infeasible route over set $S \subseteq \mathcal{S}$, with demand $q(S) > Q$. Since $r(S) > 1$ at least two routes must connect S with the complement $V \setminus S$; in such a way any infeasible route is excluded. Moreover, any subtour over $S \subseteq \mathcal{S}$ has no connection with the complement and due to $r(S) > 1$ this tour is also eliminated. Note that the formulation remains valid if $r(S)$ is replaced by the lower bound given by:

$$r(S) = \lceil q(S)/Q \rceil. \tag{2.2}$$

In this case (2.1c), using (2.2), takes the name *rounded capacity constraints*.

Constraint (2.1d) states that M vehicles must leave and return to the depot. Finally (2.1e) and (2.1f) are the integrality constraints.

Laporte at al. [50] replaced constraint (2.1c) of F1 with the following constraint, obtained from combining (2.1b) and (2.1c):

$$\sum_{i \in S} \sum_{\substack{j \in S \\ i<j}} x_{ij} \leq |S| - r(S) \quad \forall S \in \mathcal{S} \tag{2.3}$$

In all cases, the number of constraints grows exponentially with the number of vertices. A basic technique used to handle this number of constraints is the Branch-and-Cut(BC). The first step is to exclude SECs from the problem and solve the *linear relaxation*, i.e. relaxing the integrality constraint (2.1e) and (2.1f). Therefore the optimal solution obtained by an LP solver, z_{LP}, will be a lower bound value of the optimal value z_{F1}, i.e. $z_{LP} \leq z_{F1}$. The linear relaxation is then solved by *cutting plane* technique, adding at each iteration the violated SEC identified, until no more violated SEC are found. For a more extensive discussion about BC, the reader is invited to read section (3.1) for a literature review.

2.1.2 Set Partitioning Formulation

The Set Partitioning (SP) formulation for CVRP was first proposed by Balinsky et al. in 1964 [9] and is based on an extended set partitioning or set covering model. Different from the formulations proposed in section (2.1.1), SP formulation employs a small number of constraints and an exponential number of variables.

Let Ω be the index set of all feasible routes and a_{ij} a binary coefficient that is equal to 1 if costumer i belongs to route $j \in \Omega$, and takes value 0 otherwise. Each route $r \in \Omega$ has an associated cost $\hat{c}_r$ which corresponds to the sum of the edges traversed. Let δ be a binary variable which is equal to 1 if and only if route $j \in \Omega$ is in the optimal value. The SP formulation is then:

$$z_{SP} = \text{minimize} \quad \sum_{r\in\Omega} \hat{c}_r \delta_r \tag{2.4a}$$

$$\text{subject to} \quad \sum_{r\in\Omega} a_{ir}\delta_r = 1, \quad \forall i \in V, \tag{2.4b}$$

$$\sum_{r\in\Omega} \delta_r = m, \tag{2.4c}$$

$$\delta_r \in \{0,1\}, \quad \forall r \in \Omega \tag{2.4d}$$

In (2.4a) the selected routes are minimized. (2.4b) are the set partitioning constraints stating that each customer is visited exactly once. Constraint (2.4c) requires that each vehicle is utilized.

Due to the exponential number of variables, model SP, cannot be used directly to solve instances of large size. However, model SP is highly versatile and can consider various route constraints, such as time windows. This is due to the implicit consideration of route feasibility in defining the route set Ω. Moreover, as pointed out in [19], the linear relaxation provides excellent lower bounds.

Although an outdated model, SP formulation is utilized by the state-of-the-art CVRP solvers, the *Branch-and-cut-and-price* method, as described in Chapter 4.

2.2 CARP Problem Description

The CARP can be modeled as an undirected graph $G = (V, E)$. $V = \{0, \ldots, n\}$ is the set of $|V| = n + 1$ nodes, where node 0 is the *depot*. E is the set of edges with endpoints in V. Each edge $e \in E$ has an associated *demand* $q_e \geq 0$ and a travel *cost* $c_e \geq 0$. The set of edges that require service is called *required edges* and denoted by $E_R \subseteq E$. Be $m = |E_R|$ the number of required edges and assume that the demand is strictly positive while is zero for each non-required edges $e \in E \setminus E_R$. Denote by i_e,j_e the two endpoints of each edge e and by $A = \{(i_e, j_e), (j_e, i_e) : e \in E\}$ the set of arcs associated with E. Conversely given $a = (i_e, j_e) \in A$, the mapping $e(a)$ gives the corresponding edge $e \in E$. $A_R \subseteq A$ is then the set of arcs associated with the required edges E_R.

A walk $R = (v_0, e_0, v_1, e_1, \ldots, v_p)$ in G is an alternating sequence of vertices $v_0, \ldots, v_p \in V$ and edges $e_k = \{v_k, v_{k+1}\} \in E$, $k = 0, \ldots, p-1$. The edges traversed in a walk R are denoted by $E(R)$. A CARP *route* is a walk R, where $v_0 = v_p = 0$ that serves a subset $S(R) \subseteq E(R) \cap E_R$ of required edges such that the total demand does not exceed Q. An edge that is traversed by a route but does not receive service, or an edge that is traversed multiple times, is considered *deadheaded* by that route. Likewise, a path that corresponds to a sequence of edges that are deadheaded by the same route is also considered deadheaded by that route.

The objective of CARP is to identify K routes within Graph G such that each

necessary edge is serviced by precisely one route, minimizing the total cost of all edges traversed during the corresponding walks.

2.2.1 Set Partitioning Formulation

Similarly, as 2.1.2 CARP can be formulated as an SP model. Let $\mathcal{R}$ be the index set of all routes and α_{le} the number of times that edge $e \in E_R$ is serviced by route R_l, $l \in \mathcal{R}$ and β_{le} the number of times that edge $e \in E$ is traversed by route R_l. Let S_l the set of required edges serviced by route R_l and $d_l = \beta_{le} - \alpha_{le}$ the number of times that route R_l deadheads edge $e \in E$. Define $c_l = \sum_{e \in E} \beta_{le} c_e$ as the cost of route $R_l, l \in \mathcal{R}$. Let ξ_l be a binary coefficient equal to 1 if and only if route R_l is in the solution. The set partitioning model can be formulated as an IP model:

$$
\begin{aligned}
z_{\text{CARP}_{\text{CARP}}} = \text{minimize} \quad & \sum_{l \in \mathcal{R}} c_l \xi_l && \text{(2.5a)} \\
\text{subject to} \quad & \sum_{l \in \mathcal{R}} \alpha_{le} \xi_l = 1, \quad \forall e \in E_R, && \text{(2.5b)} \\
& \sum_{l \in \mathcal{R}} \xi_l \geq K, && \text{(2.5c)} \\
& \xi_l \in \{0,1\}, \quad \forall l \in \mathcal{R} && \text{(2.5d)}
\end{aligned}
$$

In the CARP literature, K is either fixed, free or bounded above. Similarly to CVRP is possible to use a lower bound $K = \lceil \sum_{e \in E_R} q_e / Q \rceil$ as in (2.2). Constraint (2.5b) specifies that each required edge is serviced by exactly one route and constraint (2.5c) imposes the lower bound on the number of routes needed to service all required edges.

SP formulation (2.5) is very general and is given to the reader in order to formalize the problem. However, differently from CVRP, for the CARP the formulations in literature are more specific to the type of algorithm proposed. Among the others, notable formulations are:

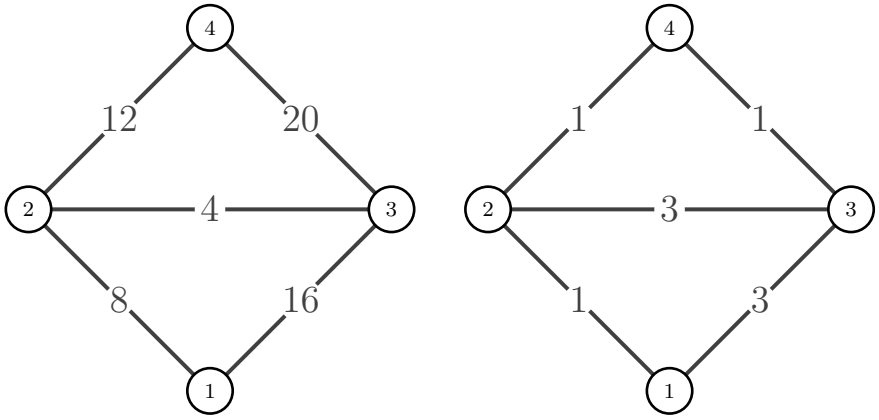
1. The **one-index formulation** was first considered independently by Letchford [52] and Belenguer and Benavent [12]. The method can generate lower bounds, proven to be optimal or very precise for small and medium-sized instances. However, the one-index formulation is considered a relaxation of the CARP, due to the integer polyhedron associated with it generally containing infeasible solutions.

2. **Transformations to the CVRP** as in Pearn, Assad, and Golden [62], Baldacci and Maniezzo [7]; Longo, Poggi de Aragão and Uchoa [55] and Foulds, Longo, and Martins [40]. These transformations exploit both formulations (2.1),(2.4) and are explained in chapter 3.

3. Pecin and Uchoa [63] developed a **flow formulation** by amalgamating a selection of features derived from the CARP literature with some features adjusted from the most accomplished contemporary algorithms for node routing.

4. Pessoa et al. [68] modeled the CARP as a **Resource Constraint Shortest Path** problem (RCSP) over a directed graph. The general transformation in RCSP is suited for the proposed VRPSolver.

CARP Example



(a) CARP with costs on edges. (b) CARP with demands on edges.

Figure 2.2: Example of a CARP, with 4 nodes and 5 required edges.

In figure (2.2), a CARP network is represented, with 4 nodes and 5 edges, all required. In the sub-figure (2.2(a)), the costs are pictured on the edges while in (2.2(b)), the demands are reported. Node 1 is the depot node where the vehicles must leave and return. Moreover, each vehicle has a capacity of 5.

The two optimal paths are then

$$1 - 3 - 2 - 1,$$

$$1 - 2 - 4 - 3 - 2 - 1$$

and are reported in figure (2.3) outlined by the two colors. The dashed lines represent the edges that are *deadheaded*. Thus, the final cost is 80 as the sum of the edge costs of the two paths. Note that each edge's demand is met by only one vehicle, indicated by the edge's color. If an edge is deadheaded, the vehicle is not required to meet the capacity constraint.

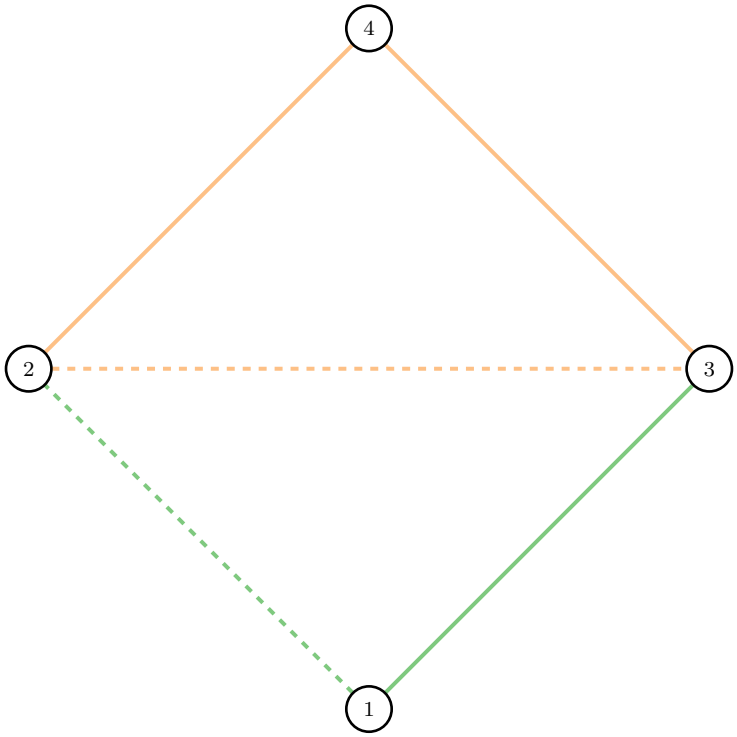


Figure 2.3: Solution of CARP example.

Chapter 3

Literature Review

This chapter details the significant advancements made in exact algorithms for CPRV and CARP. For the CPRV, there will be two phases:

1. **Classical exact algorithms** span from problem formulation to the early 2000s. Firstly, the main improvements will be explained, starting from branch-and-bound (BB) to branch-and-cut (BC) approaches, including the initial application of the column generation (CG) method.

2. **New exact algorithms** : from the Early 2000s to Present Day. The focus will be on the Branch-and-cut-and-price (BCP) algorithms, which represent state-of-the-art techniques for the exact solution of CVRP.

The literature on the Capacitated Arc Routing Problem (CARP) intertwines with that of the CVRP, using the same methods up to the present. Today's state-of-the-art approach uses the most advanced CVRP techniques to solve the CARP exactly.

3.1 CPRV Classical Exact Algorithms

After the proposal of the CVRP problem by Dantzig and Ramser [28], numerous authors sought an exact solution for it. The solution followed hand-in-hand the progress made in several fields of IP problems. In this context, the advancements made for the *Travel Salesman Problem* were fundamental for the success of the first exact algorithms.

3.1.1 Branch-and-Bound

Branch-and-bound represents one of the first successful techniques employed to solve exactly the CVRP. The first attempts follow the idea exploited for the TSP solution. The main contributions are listed:

1. **Christofides et al.** [21] developed a BB algorithm relaxing the problem by dropping the SECs in a resulting *Transportation Problem*(TP) by following Little et al. [54] combinatorial relaxation based on the *Assignment Problem* (AP) or *Shortest Spanning Tree* (SST). The authors were able to solve two small instances with 6 and 13 customers. Similarly **Laporte et al.** [50] solved randomly generated *Asymmetrical Capacitated Vehicle Routing Problem* (ACVRP) instances up to tens costumer and four vehicles.

2. **Fischetti, Toth, and Vigo** [36] in 1994 utilized an AP relaxation as a basis for a bounding process founded on a disjunction of infeasible arc subsets, improving the ACVRP relaxation bounds. The result was solving random ACVRP instances up to 300 customers and four vehicles.

3. More sophisticated techniques were proposed by **Miller** [60] and **Fisher** [37] to reinforce the CPRV relaxation by dualizing some relaxed constraints. Fisher started from his K-tree relaxation and included in the objective function the degree constraints (2.1b) and some SECs (2.1c) as *Capacity Cut Constraints*(CCC) (see equation (2.2)). Miller added a set of SECs constraints that were removed to obtain the relaxation. Since the number SEC/CCC constraints is exponential (see section 2.1.1) both authors proposed to include iteratively only a limited set of them. This technique allowed to solve a problem with $n \leq 100$ costumers within 60,000s on an Apollo Domain 3000 (0.071 Mflops) for Fisher, and problems with $n \leq 50$ costumers within 15,000s on a Sun Sparc 2 (4 Mflops) for Miller.

3.1.2 Column Generation

Using the alternative *Set Partitioning* (SP) formulation proposed in section 2.1.2 **Bramel and Simchi-Levi** [19] developed a *Column Generation* (CG) approach in order to deal with the exponential number of constraints. Their procedure starts from a small subset of routes and solves the linear relaxation of the corresponding reduced model (where constraint (2.4b) is substituted by $\sum_{r\in\Omega} a_{ir}\delta_r \geq 1$) deriving the optimal dual variables associated with the constraints. This work inspired further literature leading to modern *Branch-and-cut-and-price* algorithms.

The first approach with SP formulation was proposed by **Agawal et al.** in 1989 [2] where the authors considered an unlimited number of vehicles, by removing constraint (2.4c). To solve the resulting model, they implemented a CG approach in which the pricing problem is faced through a dedicated BB algorithm. The algorithm successfully solved Euclidean CVRP instances up to 25 customers.

3.1.3 Branch-and-Cut

Most of the works for solving the CVRP from 1980 to the early 2000s were addressed to *Branch-and-Cut* algorithms. These works are based on the two-index formulation (see section 2.1.1) and are presented below.

1. **Laporte et al.** [50] developed a BC algorithm, not considering SEC constraints (2.1c) and the integrality constraints (2.1e),(2.1f).From the relaxed model, the violated SEC constraints were retrieved using a heuristic separation procedure. All generated constraints were added to the model. Additionally, Gomory cuts were introduced at the root node and a branching procedure was implemented when no more constraints were found by the heuristic.

2. **Augerat** [4] in his PhD thesis separated four families of valid inequalities: (i) *the rounded capacity inequalities*; (ii) *the generalized capacity constraints*; (iii) *the comb inequalities*; (iv) the *hypotour inequalities*. Furthermore, a tabu search-based heuristic was used to generate an initial upper bound and update it based on the fractional solutions visited during the course of the algorithm.

3. **Ralphs et al.** [70], in 2003, proposed a BC algorithm with a new approach. They first separated the capacity constraints, utilizing three heuristics. If the heuristics could not detect a violated inequality, they suggested implementing a decomposition algorithm to establish further constraints.

4. **Lysgaard, Letchford and Eglese** [57] developed a new BC algorithm in 2004. The authors exploited the family of inequality presented in [4], with the same branching scheme. Furthermore, they disturbed the present fractional solution by employing mixed-integer Gomory cuts at the root node.

5. Other significant contributions for BC algorithms were carried out by **Achuthan et al.** [1], that developed a heuristic to separate rounded capacity constraints; **Blasum et al.** [15] that developed new family of valid inequalities and **Baldacci, Hadjiconstantinou and Mingozzi** [6] that presented a formulation based on a two-commodity network flow.

3.2 CVRP New Exact Algorithms

Since Desrosiers, Soumis, and Desrochers [31] conducted their initial work in 1984, there has been a significant amount of research dedicated to enhancing the efficiency of Branch-and-Price (BP) and Branch-and-Price-and-Cut (BPC) algorithms, making them the prominent algorithms for resolving many categories of VRPs.

This section shows a list of major contributions of BC and BPC algorithms that are all based on CG procedure, using the SP model described in section 2.1.2. The

concepts pertaining to BPC, as discussed, will be comprehensively addressed in chapter 4.

1. **Fukasawa et al.** [41] developed an exact algorithm where the variables correspond to the set of q-routes, described in [22]. Valid inequalities are exploited as *rounded capacity inequalities*, *framed capacity*, *strengthend comb*, *multistar*, *partial multistar*, *generalized large multistar* and *hypotour inequalities*, as presented in [57]. This algorithm can achieve optimal solutions for all instances documented in the literature, up to 135 customers, with significant improvements compared to previous methods.

2. **Baldacci, Christofides, and Mingozzi** [5] proposed *strengthen capacity inequalities* and *clique inequalities* in order to strengthen the SP model. The algorithm does not branch and instead concludes at the root node by listing all elementary routes with a reduced cost less than the duality gap. This creates an SP problem that includes all of these routes and is provided to a Mixed-Integer Programming (MIP) solver. This method solved almost all instances solved by [41] with less time. However, the algorithm could fail on instances with long routes.

3. **Pessoa, Poggi de Aragão, and Uchoa** [67] improved the work from Fukusawa et al. [41] using the concept of basic route enumeration and MIP solving to complete a node that was adopted from [5]. To prevent premature failure when the root gap is too significant, a traditional branching approach was incorporated into the process.

4. **Baldacci, Mingozzi, and Roberti** [8] proposed the ng-routes, a relaxation that outperforms the q-routes with no k-cycles. ng-routes are used in the earlier bounding procedures and also to speed up the pricing/enumeration of elementary routes. The resulting algorithm was more stable and faster with respect to [5], solving instances with long routes.

5. **Contardo et al.** [24] introduced innovations in the application of non-robust cuts and route enumeration. The authors solved for the first time a hard instance with 151 vertices and 12 vehicles.

6. **Pecin et al.** [64] developed a BPC algorithm, including all major improvements from previous algorithms combined with new enhancements. The algorithm was able to solve all classical instances from literature up to 200 vertices.

7. **Pessoa et al.** [68] proposed a generic solver for the vehicle routing problem and related. The authors developed the state-of-the-art solution for the CVRP. The *VRPsolver* found the exact solution for six opened instances of up to 548 customers. This tool can handle various VRPs and has outperformed the previous state-of-the-art, thanks to extensive development efforts.

3.3 CARP Major Contributions

This section provides an overview of the most successful exact method for solving the CARP.

1. **Belenguer and Benavent** [12] proposed a vehicle-indexed compact formulations. The authors developed a cutting plane algorithm, based on the aggregation of the deadheaded variables using *Odd-Edge-Cutset* and *CARP Rounded Capacity* as families of cuts. However, the proposed algorithm solved very small instances with a small number of required edges.

2. **Belenguer and Benavent** [11] developed a BC algorithm exploiting the families of cut of [12] and introducing *Disjoint-Path* inequalities, obtaining significative lower bounds. Before this study, heuristic algorithms were the primary method for discovering the best-known lower bounds for the CARP.

3. Several authors developed transformations from CARP to CPRV. **Pearn et al.** in 1987 [62] were the first to propose such transformation, resulting in a CPRV with $3|R| + 1$ nodes. **Baldacci and Maniezzo** [7] and **Longo, Poggi de Aragão, and Uchoa** [55] proposed independently, in 2006, a transformation with $2|R| + 1$ nodes. A BC algorithm was used by Baldacci and Maniezzo and a BPC by Longo, Poggi de Aragão, and Uchoa. The transformation was beneficial as the literature on the CVRP was more advanced at that time.

4. **Bartolini et al.** [10] devloped a cut-and-column based technique combined with a set partitioning approach. The authors solved for the first time an instance with 140 vertices and 190 edges (159 required edges).

5. **Bode and Irnich** [17] improved a proposed algorithm that introduced the so-called follower/non-follower branching scheme [16]. The resulting BPC solved to optimum 29 opened instances.

6. **Pecin and Uchoa** [63] reviewed previous formulations in the literature, highlighting their advantages and disadvantages. The authors developed a BCP based on the main contributions of the CARP. This algorithm was successful in solving nearly all classical instances and claimed a new state-of-the-art.

7. **Pessoa et al.** [68] modeled the CARP as a *Resource Constrained Shortest Path* (RCSP) problem. The generic *VRPsolver* proposed by the authors was able to solve almost all Eglese instances [33] as fast as [63].

Chapter 4

Branch and Price and Cut

This chapter will introduce the *Branch-and-Price*(BP), *Branch-and-Price-and-Cut* (BPC) algorithms and the *Column Generation* (CG) procedure, which are fundamental components of contemporary solvers for VRPs. BP algorithms are a form of BB algorithms where the linear relaxation is solved by CG. CG is an iterative procedure that can efficiently handle linear programs when the number of variables is large.

For further readings, the reader is referred to Feillet et al. [34] for a tutorial on CG and BP for VRPs, Costa et al. [27] for an overview on BCP and the best techniques developed in the literature, and Desrosiers and Lübbecke [30] for a more general BP and BPC formulations and description.

In the following section the main components of a BPC will be presented, i.e. CG (4.1), pricing problem (4.2), cutting planes (4.3) and branching decisions (4.4).

4.1 Column Generation

In the SP model, as described in section 2.1.2, the number of variables exhibits exponential growth with the number of customers n, rendering it unmanageable by BB approaches. This assessment can alternatively be addressed with a column generation approach, thereby converting the BB to BP. At the root note of the BB search tree, the *Master Problem* (MP) is defined as:

$$
\begin{aligned}
z_{\text{MP}} = \text{minimize} \quad & \sum_{r \in \Omega} \hat{c}_r \delta_r && && \text{(4.1a)}\\
\text{subject to} \quad & \sum_{r \in \Omega} a_{ir} \delta_r = 1, && \forall i \in V, && \text{(4.1b)}\\
& \sum_{r \in \Omega} \delta_r = m, && && \text{(4.1c)}\\
& \delta_r \geq 0, && \forall r \in \Omega && \text{(4.1d)}
\end{aligned}
$$

(4.1) is the linear relaxation of (2.4), obtained by relaxing the integrality constraint (2.4d). An optimal solution to the MP provides a valid dual bound at the root node of the search tree. The *Restricted Master Problem*(RMP) is obtained from the MP replacing Ω with a smaller subset of variables (possibly none) $\Omega' \subseteq \Omega$.

Consider now the dual problem associated with the RMP as the *Dual Restricted Master Problem* (DRMP):

$$\begin{aligned} z_{\text{DRMP}} = \text{maximize} \quad & m\pi_0 + \sum_{i \in V} \pi_i && && \text{(4.2a)} \\ \text{subject to} \quad & \pi_0 + \sum_{i \in V} a_{ir}\pi_i \leq \hat{c}_r, && \forall r \in \Omega', && \text{(4.2b)} \\ & \pi_0 \in \mathbb{R}, && && \text{(4.2c)} \\ & \pi_i \in \mathbb{R}, && \forall i \in V && \text{(4.2d)} \end{aligned}$$

In each iteration of a CG procedure, an optimal solution of RMP and DRMP is obtained, named respectively δ^* and π^*. If the solution is feasible for the *Dual Matster Problem*(DMP), then both DMP and MP are solved. If not means that one or more constraints were violated in $\Omega \setminus \Omega'$. The principle behind the column generation method is to find one or more of these constraints by solving an appropriate optimization sub-problem, called *Pricing Problem*(PP), in order to integrate the corresponding variables in the set Ω'. The algorithm terminates when no violated constraint is identified. In a formal way, at each iteration, the following equation is evaluated:

$$\bar{c}_r = \hat{c}_r - \sum_{i \in V} a_{ir}\pi_i, \tag{4.3}$$

when $\bar{c}_r$ is negative, then the variable δ and its coefficient column are added to the RPM, and the process is reiterated. On the contrary, if $\bar{c}_r$ is positive, the current solution δ^* is an optimal solution for the MP.

4.2 Pricing Problem

The PP aims to find a feasible $r \in \Omega$ with negative reduced cost $\bar{c}_r$. Let set $\pi_0 = 0$ for notation conciseness, then the reduced cost $\bar{c}_r$ of variable δ_r is:

$$\bar{c}_r = \hat{c}_r - \sum_{i \in V} a_{ir}\pi_i = \sum_{ij \in A} (\hat{c}_{ij} - \pi_i) = \sum_{ij \in A} \bar{c}_{ij} \tag{4.4}$$

with $\bar{c}_{ij} = \hat{c}_{ij} - \pi_i$ as modified arc cost. PP is often expressed as finding the route with the least reduced cost: this problem corresponds to an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC). ESPPRC finds the shortest path from the origin to the origin with the elementary constraint, i.e. each customer must be visited at most once, and resource constraints as the load.

The algorithm is shown to be strongly NP-hard [32]. Feillet et al. [35] proposed a dynamic programming algorithm to solve the basic version of the problem. However, for pricing the authors used a relaxation known as the (non-elementary) *resource constrained shortest path problem* (RCSPP). The RCSPP allows routes with cycles (i.e. multiple visits to customers). The RCSPP can be resolved through a pseudopolynomial algorithm, as demonstrated in the work by Desrochers et al. [29].

The underlying problem then can be solved using a dynamic programming algorithm known as *labeling algorithm*. In this algorithm, partial paths commence from the depot node and are denoted by multi-dimensional resource vectors, known as labels. Starting with a label assigned to the depot, *resource extension functions* (REFs) are used to propagate labels forwardly through the network graph.

4.3 Cutting Planes

To enhance the lower bound for the MP, a typical technique in BPC includes utilizing valid inequalities to reinforce the MP. There are two types of cuts, according to Poggi and Uchoa [69]:

- **Robust cuts**: these cuts do not change the structure of the pricing problem and can be expressed as arc-flow variables. Robust cuts are quite effective at closing the integrality gap but may be not sufficient for hard instances of VRP.
- **Non-robust cuts**: are defined directly on the MP variables δ_r. These cuts increase the problem's complexity by modifying the pricing problem e.g. adding additional resources and modifying the dominance rule. Non-robust cuts can be effective in addressing complex instances, but they must be utilized with caution.

CVRPs valid inequalities, such as *rounded capacity*, *framed capacities*, *strengthened combs*, *multistars*, and *extended hypotours* played a significant role in BC algorithms, e.g. in Lysgaard et al. [57]. However, for BPC, most of the cuts are already implied through constraints (2.4b) and (2.4c). Letchford and Salazar González [53] have demonstrated, in fact, that the cited constraints imply all *generalized large multistar* cuts. Only *rounded capacity cuts*(RCC) (2.2) (see section 2.1.1) can contribute to improving the lower bound at the root node.

In this section, a list of useful valid inequalities is presented. The reader is referred to [27] for an overview of robust and non-robust cuts used in literature.

1. **Strengthened Capacity Cuts**(SCCs) introduced by Baldacci, Christofides, and Mingozzi [5] exhibit greater strength w.r.t. RCC as they remain objective to routes that may enter and exit set $S \subseteq V$ more than once. On the other hand, SCCs are non-robust cuts and affect the pricing problem.

2. **Subset Row Cuts**(SRCs), developed by Jepsen et al. [48] are non-robust cuts obtained as a subset of the Chvátal-Gomory rank-1 cuts. These cuts are defined as:
$$\sum_{r\in\Omega} \lfloor p \sum_{i\in C} a^{ir} \rfloor \delta_r \leq \lfloor p|C| \rfloor, \tag{4.5}$$
where $C \subseteq V$ and $0 < p < 1$. These cuts are NP-hard to separate (see [48]), therefore, the subset C is typically separated through enumeration, with $|C| < 5$. When $|C| = 3$ and $p = 1/2$ are named 3-SRCs and are the most popular in literature, used in [25] and [8].

3. **Limited-Memory SRCs** are a viable solution for reducing the adverse effects of SRCs on the PP. Pecin et al. [65] proposed these cuts, which are dependent on a coefficient. The coefficient weakens the impact of SRCs over the pricing.

4.4 Branching Decisions

Branching is the last step in order to obtain an integer solution if the CG ends with a duality gap. A first direction could be to choose δ_r as the variable for branching, however, this solution would complicate the PP, generating an ESPPRC with forbidden paths [74]. Moreover, such a branching procedure would result in a completely unbalanced search tree.

A possible solution is to branch on the edge variables of the two-index formulation (2.1.1). This solution can be implemented by branching either on the single edges $e \in E$, such that $x_e \leq 0$ or $x_e \geq 1$, otherwise over sets $S \subseteq V$. In Fukasawa et al. [41] the authors developed a BPC that branches on the sets. To enhance the selection of the branching set, BCP implemented a **strong branching** strategy. The primary characteristic of strong branching is the assessment through a heuristic process of the relevant variables to generate a score using some functions. The set or variable with the highest score is then chosen for branching. This method can decrease the quantity of nodes to examine in the search tree, at the cost of an increase in branching time.

This schema was improved by many authors. In Pecin et al. [65], for instance, a three-step procedure was implemented. The first step (i) involves the candidate's construction by a pseudocost and new candidates each time. In the refinement phase (ii) a lower bound and an upper bound are calculated for each candidate. The small set of candidates retrieved from the second phase is finally evaluated using a heuristic CG procedure (iii).

Another branching technique is the *Ryan and Foster branching* [71]. This branching rule complicates the PP since is non-robust, on the other hand, results in a more balanced search tree. Gélinas et al. [43] proposed another strategy, called *branching over the accumulated resource consumption* that involves branching on

resource variables (time for CVRPTW or capacity) rather than on network flow variables. This solution does not increase the PP difficulty.

Chapter 5

Experiment

In this chapter, the proposed experiment is exposed. As outlined in section 1.2, the main idea is to exploit a transformation from CARP to CVRP and implement it using *VRPSolver*, which is currently the most advanced tool for exact VRPs solution.

5.1 CARP to CVRP Transformation

The transformation from CARP to CVRP, employed in this work, was proposed by Baldacci and Maniezzo [7] in 2006. The authors developed a solution in order to transform each required edge of a CARP instance into two nodes of a CVRP network. Before Baldacci and Maniezzo, Pearn et al. [62] proposed to transform each required edge in a triplet of nodes. However, the large number of nodes and a network structure not completely connected discourages further developments.

Given a CARP network defined by $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes (plus a node for the depot). Let R be the set of required edges $R \subseteq E$ and $V_R \subseteq V$ be the set of edges containing endpoints of the required edges and the depot. Finally, the cost of each edge is given by c_{ij} and the demand by q_{ij} for each $\{i, j\} \in E$.

Let define the CVRP graph as $\hat{G} = (\hat{V}', \hat{E})$. The set of nodes $\hat{V}' = \{0,1, \ldots ,2|R|\}$ corresponds to the endpoint nodes of CARP required edges R. Each node in V_R is replicated as many times a required edge is incident to it. The set of edges $\hat{E}$ makes the graph complete.

For each edge $\{i, j\} \in R$, the corresponding edge in the CVRP graph is given by $\{s_{ij}, s_{ji}\} \in \hat{E}$. Let define $\hat{R}$ as the set of edges in $\hat{G}$ corresponding to the set R in G. Moreover, let the CARP instance have a known upper bound, UB.

Given two edges in the CVRP graph, named $s_{ij} \in \hat{E}$ and $s_{kl} \in \hat{E}$, the corresponding cost $\hat{c}_{s_{ij}s_{kl}}$ with $s_{ij}, s_{kl} \neq 0$ is given by:

$$\hat{c}_{s_{ij}s_{kl}} = \begin{cases} -\text{UB} & \text{if } \{s_{ij}, s_{kl}\} \in \hat{R} \\ \frac{1}{2}c_{ij} + \text{dist}(i,k) + \frac{1}{2}c_{kl} & \text{otherwise} \end{cases}. \tag{5.1}$$

The cost $\hat{c}_{0s_{ij}}$ from the depot node to the node s_{ij} is:

$$\hat{c}_{0s_{ij}} = \text{dist}(0,i) + \frac{1}{2}c_{ij}. \tag{5.2}$$

The element $\text{dist}(i,k)$ is the shortest path from $i \in V$ to $j \in V$ in the CARP network. The demand a node $s_{ij} \in \hat{V}$ in the CVRP graph is then defined by $\hat{q}_{s_{ij}}$ with value:

$$\hat{q}_{s_{ij}} = \hat{q}_{s_{ji}} = \frac{1}{2}q_{ij} \qquad \forall\{s_{ij}, s_{ji}\} \in \hat{R} \tag{5.3}$$

Using a coefficient β that takes values in $0 \leq \beta \leq \frac{1}{2}$, equations (5.1) and (5.2) can be merged into:

$$\hat{c}_{s_{ij}s_{kl}} = \begin{cases} -\text{UB} + (1-2\beta)c_{ij} & \text{if } \{s_{ij}, s_{kl}\} \in \hat{R} \\ \beta c_{ij} + \text{dist}(i,k) + \beta c_{kl} & \text{otherwise} \end{cases}. \tag{5.4}$$

This process does not generate a CVRP instance that is general. Altering the parameter β and varying the upper bound UB values can result in distinct mappings, which consequently change the solver's computational burden. The authors suggested fixing $\beta = \frac{1}{2}$, eliminating the upper bound and adding a mandatory edge between each $\{s_{ij}, s_{kl}\} \in \hat{R}$ instead. The final formulation is presented:

$$\hat{c}_{s_{ij}s_{kl}} = \begin{cases} 0 & \text{if } \{s_{ij}, s_{kl}\} \in \hat{R} \\ \frac{1}{2}c_{ij} + \text{dist}(i,k) + \frac{1}{2}c_{kl} & \text{otherwise} \end{cases}. \tag{5.5}$$

Equation (5.5) was then implemented with a modified two-index vehicle flow formulation (see section 2.1.1), named *Edge CVRP* (ECVRP) as follows:

$$\begin{aligned} z_{\text{ECVRP}} = \text{minimize} \quad & \sum_{ij \in E} \hat{c}_{ij}x_{ij} && && \text{(5.6a)} \\ \text{subject to} \quad & (2.1b), (2.1c), (2.1d) \text{ and } x_{ij} = 1, && \forall\{i,j\} \in \hat{R}, && \text{(5.6b)} \\ & x_{ij} \in \{0,1\}, && \forall\{i,j\} \in \hat{E} && \text{(5.6c)} \end{aligned}$$

in which (5.6b) introduces the mandatory edges $x_{ij} = 1, \forall\{i,j\} \in \hat{R}$ with zero cost. Moreover, since each CARP edge is represented by two nodes in the CVRP graph, there is no possibility of having the situation represented by equation (2.1f).

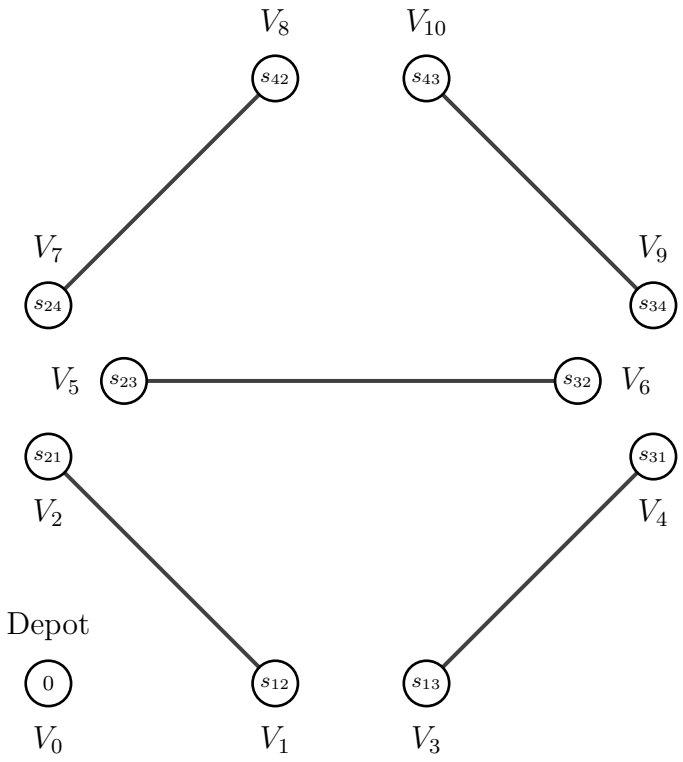


Figure 5.1: CARP to CVRP transformation of the graph of Figure 2.2.

Transformation Example

In order to understand deeply this transformation an example is given, starting from the one explained in section (2.2.1). In figure (5.1) the transformed network of figure (2.2) is depicted. Each required edge is now represented by two nodes, named after the two nodes that define the CARP edges, e.g. s_{12},s_{21} represent the original edge e = (1,2). Moreover, near each node the vertex ID of the CVRP network is reported as $\hat{V}' = \{V_0, \ldots, V_{10}\}$. The special node V_0, is the depot, where each vehicle must leave and return. The edges represented in the figure are the mandatory edges, defined by equation (5.6b). The cost of such edges, as defined in equation (5.5), is zero. Despite the edges represented, the real CVRP graph is complete and the cost of the edges is reported as the distance matrix in Table 5.1.

The solution of such a network is shown in Figure (5.2). The two colors represent the two vehicles' paths, where the dashed line means that a vehicle is leaving or returning to the depot. Finally, as in the example of section 2.2.1, the total cost is 80.

		V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}
		0	s_{12}	s_{21}	s_{13}	s_{23}	s_{23}	s_{32}	s_{24}	s_{42}	s_{34}	s_{42}
V_0	0	–	4	12	8	20	10	14	14	26	22	30
V_1	s_{12}	4	–	0	12	24	14	18	18	30	26	34
V_2	s_{21}	12	0	–	20	16	6	10	10	22	18	26
V_3	s_{13}	8	12	20	–	0	18	22	22	34	30	38
V_4	s_{31}	20	24	16	0	–	14	10	18	30	18	34
V_5	s_{23}	10	14	6	18	14	–	0	8	20	16	24
V_6	s_{32}	14	18	10	22	10	0	–	12	24	12	28
V_7	s_{24}	14	18	10	22	18	8	12	–	0	20	28
V_8	s_{42}	26	30	22	34	30	20	24	0	–	32	16
V_9	s_{34}	22	26	18	30	18	16	12	20	32	–	0
V_{10}	s_{42}	30	34	26	38	34	24	28	28	16	0	–

Table 5.1: Distance matrix for the example of Figure 5.1.

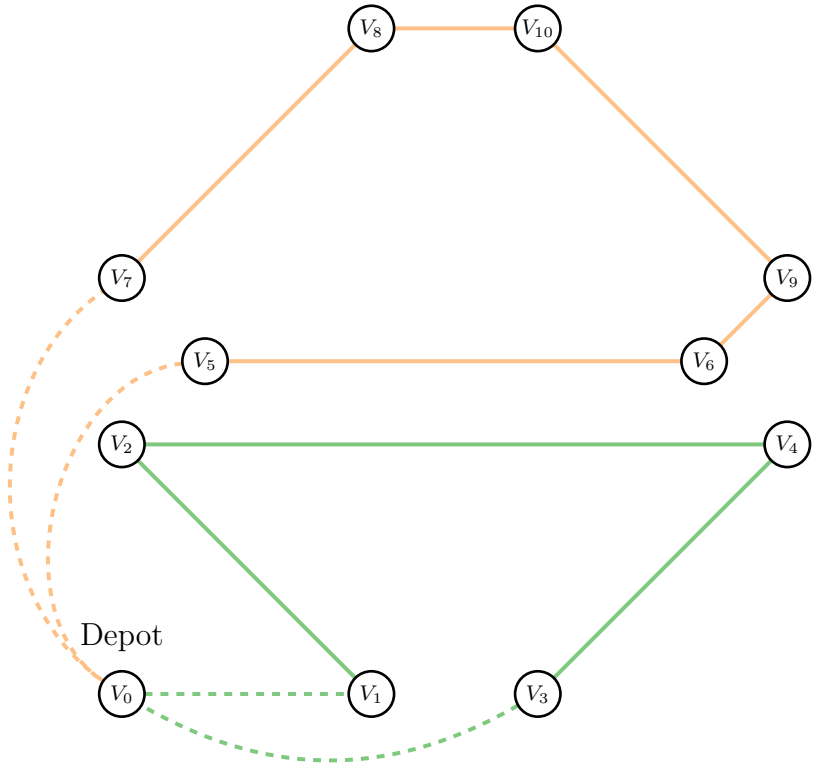


Figure 5.2: Solution of the example of Figure 5.1.

5.2 VRPSolver

Pessoa et al. in 2020 presented their work "A Generic Exact Solver for Vehicle Routing and Related Problems" [68]. The exact solver named *VRPSolver* aims to find a solution for multiple VRPs using the BCP technique. The main idea was to incorporate the best methodologies in the literature as *ng-path relaxation*, *rank-1 cuts with limited memory*, *path enumeration*, and *rounded capacity cuts*. These features are all generalized through the concepts of *packing set* and *elementarity set*.

5.2.1 Basic Model Formulation

In the *VRPSolver* a generic model can be implemented as a *Resource Constrained Shortest Path* (RCSP) problem. The basic model is now described, for more information the reader is referred to the author's paper [68].

Define a set R of resources, divided into main resources R^M and secondary resources R^N. Let K be the set of subproblems, then for each $k \in K$ define direct graphs $G^k = (V^k, A^k)$. For each sub-problem, there are two special vertices, named v^k_{source} and v^k_{sink} that represent the first and last vertices visited in the path. Note that vertex v^k_{source} can be equal to v^k_{sink}. For each $r \in R^k$ and for each $a \in A^k$, $q_{ar} \in \mathbb{R}$ is the consumption of resource r in arc a. Moreover, assume that q_{ar} is non-negative for $r \in R^M$ and unrestricted for $r \in R^N$. There are finite accumulated resource consumption intervals $[l_{ar}, u_{ar}]$ for each $r \in R^k$ and $a \in A^k$. The consumption can also be defined on vertices instead of arcs. Define finally $V = \bigcup_{k \in K} V^k$ and $A = \bigcup_{k \in K} A^k$.

A Path $p = (v_{\text{source}} = v_0, a_1, v_1, \ldots, a_n, v_n = v_{\text{sink}})$ over G^k is a **Resource Constrained Path** if and only if for every $r \in R$, the accumulated resource consumption S_{jr} at visit j, $0 \leq j \leq n$, where $S_{0r} = 0$ and $S_{jr} = \max\{l_{a_j r}, S_{j-1,r} + q_{a_j r}\}$ does not exceed $u_{a_j r}$.

Let P^k for each $k \in K$ be the set of all resource-constrained paths in G^k, and $P = \bigcup_{k \in K} P^k$. Then the variables of the problem can be defined as either integer and/or continuous. There are two types of variables:

- Mapped x variables : each variable $x_j, 1 \leq j \leq n_1$ is mapped into a non-empty set $M(x_j) \subset A$. The inverse mapping of arc a is $M^{-1}(a) = \{j | a \in M(j)\}$.
- Additional non-mapped y variables.

For all $a \in A$ and $p \in P$, let h^p_a indicate how many times arc a appears in path p. Given $c \in \mathbb{R}^n_1, f \in \mathbb{R}^n_2, \alpha \in \mathbb{R}^{m \times n_1}, \beta \in \mathbb{R}^{m \times n_2}$ and $d \in \mathbb{R}^m$, the model is then defined.

$$\text{minimize} \quad \sum_{j=1}^{n_1} c_j x_j + \sum_{s=1}^{n_2} f_s y_s \tag{5.7a}$$

$$\text{subject to} \quad \sum_{j=1}^{n_1} \alpha_{ij} x_j + \sum_{j=1}^{n_2} \beta_{is} y_s \geq d_i, \qquad i = 1, \ldots, m, \tag{5.7b}$$

$$x_j = \sum_{k \in K} \sum_{p \in P^k} \left(\sum_{a \in M(j)} h_a^p \right) \lambda_p, \quad j = 1, \ldots, n_1, \tag{5.7c}$$

$$L^k \leq \sum_{p \in P^k} \lambda_p \leq U^k, \qquad k \in K, \tag{5.7d}$$

$$\lambda_p \in \mathbb{Z}_+, \qquad p \in P, \tag{5.7e}$$

$$x_j \in \mathbb{N}, y_j \in \mathbb{N}, \qquad j = 1, \ldots, \bar{n}_1, s = 1, \ldots, \bar{n}_2 \tag{5.7f}$$

Equation (5.7a) is the objective function to minimize, with c and f cost vectors. The inequalities (5.7b) define m general linear constraints over those variables, α and β are the coefficient matrices, and d the right-hand side vector. (5.7b) may contain an exponential number of constraints, provided that suitable separation routines are given.

Then substituting x variable and relaxing integrality:

$$\text{minimize} \quad \sum_{k \in K} \sum_{p \in P} \left(\sum_{j=1}^{n_1} c_j + \sum_{a \in M(j)} h_a^p \right) \lambda_p + \sum_{s=1}^{n_2} f_s y_s, \tag{5.8a}$$

$$\text{subject to} \quad \sum_{k \in K} \sum_{p \in P} \left(\sum_{j=1}^{n_1} \alpha_{ij} + \sum_{a \in M(j)} h_a^p \right) \lambda_p + \sum_{s=1}^{n_2} \beta_{is} y_s \quad i = 1, \ldots, m, \tag{5.8b}$$

$$L^k \leq \sum_{p \in P^k} \lambda_p \leq U^k, \qquad k \in K, \tag{5.8c}$$

$$\lambda_p \geq 0, \qquad p \in P \tag{5.8d}$$

Master LP (5.8) is solved by column generation. Denote the dual variables of (5.8b) as $\pi_i, 1 \leq i \leq m$ and the dual variables of (5.8c) as ν_+^k and ν_-^k, $k \in K$. Then the reduced cost of arc $a \in A$:

$$\bar{c}_a = \sum_{j \in M^{-1}(a)} c_j - \sum_{i=1}^{m} \sum_{j \in M^{-1}(a)} \alpha_{ij} \pi_i. \tag{5.9}$$

The reduced cost of path $p = (v_0, a_1, v_1, \ldots, a_n, v_n) \in P^k$ is:

$$\bar{c}_p = \sum_{i=1}^{m} \bar{c}_{a_j} - \nu_+^k - \nu_-^k. \tag{5.10}$$

The pricing sub-problem corresponds to finding, for each $k \in K$, a path $p \in P^k$ with minimum reduced cost. The basic model leads to a standard BP algorithm, or to a robust BCP if some constraints in (5.7b) are separated.

Packing Sets and Elementarity Sets

One of the main contributions of this work is a generalization of the key additional concepts found in those state-of-the-art algorithms, leading to the construction of a powerful and still quite generic BCP algorithm. In order to do that, the new concepts of packing sets and elementarity sets are introduced.

Let $\mathcal{P} \subset 2^A$ be a collection of mutually disjoint subsets of A such that the constraints:

$$\sum_{p \in P} \left(\sum_{a \in S} h_a^p \right) \lambda_p \leq 1 \quad S \in \mathcal{P}, \tag{5.11}$$

are satisfied by at least one optimal solution (x^*, y^*, λ^*) of formulation (5.7). Then $\mathcal{P}$ is a collection of packing sets. The arcs in each $S \in \mathcal{P}$ must not appear more than once in any of the paths that form an optimal solution. The packing set's definition is integral to the problem's modeling. Note that packing sets can be also defined on vertices as follows. Let $\mathcal{P}^\nu \subset 2^{V'}$ be a collection of mutually disjoint subsets of V' such that the constraints:

$$\sum_{p \in P} \left(\sum_{v \in S} h_v^p \right) \lambda_p \leq 1 \quad S \in \mathcal{P}^\nu, \tag{5.12}$$

are satisfied by at least one optimal solution x^*, y^*, λ^* of formulation (5.7).

Let $\mathcal{E} \subset 2^A$ be a collection of mutually disjoint subsets of A such that the constraints:

$$\sum_{a \in S} h_a^p \lambda_p \leq 1, \quad S \in \mathcal{E}, p \in P \tag{5.13}$$

are satisfied by at least one optimal solution (x^*, y^*, λ^*) of formulation (5.7). Each element in $\mathcal{E}$ is then an elementarily set. That condition can be expressed as the arcs in each $S \in \mathcal{E}$ must not appear more than once in any of the paths that form an optimal solution.

Generalizing ng-paths

The bounds obtained by linear relaxation (5.8) can often be improved by only working with elementary paths. However, the resulting pricing problems may be intractable. The ng-paths [8] (see section 4.2) obtain a good compromise between the bound quality and pricing difficulty.

For each arc $a \in A$, let $\mathrm{NG}(a) \subseteq \mathcal{P}$ denote the ng-set of a. An ng-path may use two arcs belonging to the same packing set S, but only if the subpath between those two arcs passes by an arc a such that $S \notin \mathrm{NG}(a)$. This concept represents *Generalized ng-paths*.

Limited Memory Rank-1 Cuts

Pecin et al. [65] proposed the *Rank-1 Cuts* (R1C) that are a generalization of the Subset Row Cuts proposed by Jepsen et al. [48] in 2008 (see section 4.3). VRPsolver further generalizes the concept as follows.

Consider a collection of packing sets $\mathcal{P}$ and multiplier $\rho_S \in \mathbb{R}_+$, for each $S \in \mathcal{P}$. A Chvátal-Gomory rounding of constraints (5.11) produces a generalized R1C:

$$\sum_{p \in P} \left\lfloor \sum_{S \in \mathcal{P}} \rho_S \sum_{a \in S} h_a^p \right\rfloor \lambda_p \leq \left\lfloor \sum_{S \in \mathcal{P}} \rho_S \right\rfloor . \tag{5.14}$$

R1C cuts are very effective, on the other hand are not robust, so can impact on the pricing problem. Limited memory technique is used to mitigate the negative impact.

Path Enumeration

Path Enumeration is a technique proposed by Baldacci et al. [5] and improved by Contardo and Martinelli [25]. It involves enumerating all possible paths within a set P^k that could be part of an improved solution. Subsequently, the related pricing sub-problem k can be efficiently solved through inspection, resulting in time saved. VRPSolver tries to enumerate all paths $p \in P^k_{\text{elem}}$ such that $\bar{c}(p) \leq \text{UB} - \text{LB}$ where UB is the best-known integer solution cost, and LB the value of the current linear relaxation.

Branching

VRPSolver suggests branching over x and y in order to not change the structure of the pricing subproblem. This solution may work in many models. However, if it is not sufficient, the solver offers two ways to branch over λ variables, based on *packing sets*:

- A generalization of *Ryan and Foster branching rule* [71] (see section 4.4) can provide a balanced branching tree making the PP harder.
- The second strategy is *branching over the accumulated consumption of a resource*, introduced by Gélinas et al. [43]. It does not increase the difficulty of pricing, and it could be successful in practice by delaying (or even preventing) the need for a branching approach that complicates pricing.

Rounded Capacity Cut Separators

Laporte and Norbert [50] proposed RCC cuts in 1986 as valid inequalities for the CVRP. VRPSolver implements those cuts by exploiting the heuristic separation routines developed in the library CVRPSEP by Lysgaard [56]. The RCC separator feature can be used only if packing sets are defined on vertices.

5.3 Additional robust cuts

The initial formulation was not robust enough to solve multiple instances. Therefore, an additional valid inequality was embedded through a callback function as *Lifted Odd Edge Cutset*. Odd Edge Cutset was introduced by Belenguer et al. [12],[11] and are described as follows. Given a feasible solution ξ to the SP CARP model (2.5), let $z_e, e \in E$ be an integer variable that represents the number of time edge e is *deadheaded*:

$$z_e = \sum_{l \in \mathcal{R}} d_{le} \xi_l. \tag{5.15}$$

Define for each set of nodes $S \subseteq V$, $\delta_R(S) = \delta(S) \cap E_R$, where $\delta(S)$ is a cutset, as defined in section 2.1. Define $\mathcal{S} = \{S \subseteq V : |\delta_R(S)| \text{ is odd}\}$. Then odd edge cutset inequalities are defined as:

$$\sum_{e \in \delta(S)} z_e \geq 1, \quad \forall S \in \mathcal{S}. \tag{5.16}$$

Equation (5.16) states that if $|\delta_R(S)|$ is odd then at least one edge incident to S must be deadheaded. This is due to the fact that any feasible solution induces an Eulerian graph. In order to strengthen the formulation let's define the aggregate variable y_{ij} representing the number of times that both shortest paths P_{ij} and P_{ji} between $i, j \in V$ are deadheaded:

$$y_{ij} = \sum_{l \in \mathcal{R}} \zeta_{ij}^l \xi_l, \quad \forall i, j \in V. \tag{5.17}$$

Lifted odd edge cutset inequalities are then defined by:

$$\sum_{\substack{i \in S \\ j \in V \setminus S}} y_{ij} \geq 1, \quad \forall S \in \mathcal{S}. \tag{5.18}$$

Since $|\delta_R(S)|$ is an odd number, every route will cross S an even number of times, and each necessary edge will be serviced by only one route. As a result, there must be at least one path-crossing node set S that needs to be deadheaded. Graph G is symmetric so $P_{ij} = P_{ji}$, then z_e can be expressed in terms of y_{ij} as:

$$z_e = \sum_{\substack{i,j \in V \\ i<j \\ e \in E(P_{ij})}} y_{ij}, \quad \forall e \in E. \tag{5.19}$$

Finally, equations (5.16) and (5.19) can be formulated as:

$$\sum_{\substack{i,j \in V \\ i<j \\ E(P_{ij} \cap \delta(S)) \neq \emptyset}} y_{ij} \geq 1, \quad \forall S \in \mathcal{S}. \tag{5.20}$$

5.4 Implementation

The proposed algorithm was coded in order to exploit the VRPSolver. At the website https://vrpsolver.math.u-bordeaux.fr/, VRPSolver can be downloaded only for academic use. The solver back-end is coded in C++ and given as an extension of *BaPCod* a generic BCP solver [72]. The authors developed a Julia interface in order to facilitate the implementation while not losing efficiency. For this purpose, the code is written in Julia v1.4.2, available at the website https://github.com/NicolaFarronato/CVRP_CARP. Moreover *BaPCod* employs CPLEX 12.10 as MIP solver.

The machine used for the implementation is a 3.2GHz Intel Core i9-12900K, with 32GB of RAM and Ubuntu 22.04.2 LTS OS. All tests are performed using a single core.

5.4.1 VRPSolver model

The implementation follows the considerations done in section 5.2. The proposed CVRP is modeled as a single graph $G = (\hat{V}', \hat{A})$, where $A = \{(i,j),(j,i) : (i,j) \in \hat{E}\}$, $v_{\text{source}} = v_{\text{sink}} = 0$. Only a main resource $R^M = \{1\}$: the resource consumption is defined as $q_{a1} = (d_i + d_j)/2$, where $d_i, i \in \hat{V}'$ is the demand of a node. The consumption interval is then defined as $l_{i1} = 0$ for the lower bound and $u_{i1} = Q$ for the upper bound for $i \in V$. Moreover, in order to embed equation (5.6b), a secondary resource $R^N = \{2\}$ that will be explained in section 5.4.2 was implemented. Finally, $x_e, e \in E$ is the integer variable. The model formulation is:

$$\text{minimize} \quad \sum_{e \in E} c_e x_e \tag{5.21a}$$

$$\text{subject to} \quad \sum_{e \in \delta(i)} x_e = 2 \quad i \in V, \tag{5.21b}$$

$$x_e \leq 1 \qquad e \in E \tag{5.21c}$$

Equation (5.21a) is the objective function that defines equation (5.7a). The degree constraints, by equation (5.21b) define (5.7b). Note that x_e cannot be more than 1 since every CARP edge is mapped into two nodes of the CVRP. Moreover the mapping of x is given by $M(x_e) = \{(i,j),(j,i)\}, e = (i,j) \in \hat{E}$ and defines constraint (5.7c). The lowerbound $L = \lceil \sum_{i=1}^{n} d_i/Q \rceil$ and the upper bound $U = n$ on the number of vehicles define constraint (5.7d). Variables λ are implicitly defined by the set of resource-constrained paths in G. The packing sets are defined on vertices as $\mathcal{P}^\nu = \mathcal{E}^\nu = \bigcup_{i \in V} \{\{i\}\}$.

5.4.2 CARP to CVRP Transformation Details

The CARP instances are transformed into CVRPs one using the procedure in section 5.1. The fundamental steps in the implementation are now listed:

- In order to calculate the shortest distance, required for each node, the *Floyd-Warshall algorithm* [39],[47] was implemented. Two dictionaries are produced as output: the *distance* dictionary shows the cost of the shortest path for two given nodes, while the *next* dictionary provides an array as value, containing the path from one node to the other, for a given pair of nodes as the key.

- Two functions, called v and v^{-1}, have been introduced for the purpose of enabling a straightforward change of problem domain from CARP to CVRP and vice versa. These functions have been implemented as dictionaries, with the keys for v being a tuple consisting of two nodes of the CARP graph and the direction of the path as the source node. The value will be the node ID in the CVRP graph. Conversely, v^{-1} keys refer to node IDs in the CVRP graph, and their values are the two vertices and the direction in the CARP domain.

- The additional *Lifted Odd Edge Cutset* inequalities are implemented using the *Gomory-Hu tree* algorithm [46] for separate the cuts. At the end of each CG phase, the CVRP instance is mapped to the CARP instance, calculating the number of times in which an edge is deadheaded. After the separation of the cuts, using function v, the new cuts are obtained as CVRP constraints.

- A direct implementation of (5.6b) as a constraint for model (5.21) leads to a very high time for the CG procedure. Therefore, in order to embed constraint (5.6b), a secondary resource $R^N = \{2\}$ was used. The secondary resource consumption is defined for each arc $a = (i, j) \in A$ as:

$$q_{a,2} = \begin{cases} -1.5 & \forall \{i,0\} \in A \\ -1 & \forall \{i,j\}, \{j,i\} \in \hat{R}, i \neq 0, j \neq 0 \\ 1 & \forall \{i,j\}, \{j,i\} \notin \hat{R}, i \neq 0, j \neq 0 \\ 1.5 & \forall \{0,i\} \in A \end{cases}. \tag{5.22}$$

  The consumption intervals are then defined

$$l_{i,2} = \begin{cases} 0.5 & \forall i \in V \\ 0 & \text{otherwise} \end{cases} \qquad u_{i,2} = \begin{cases} 1.5 & \forall i \in V \\ 0 & \text{otherwise} \end{cases}. \tag{5.23}$$

  The idea is to model constraint (5.6b) as a modified *Pickup and Delivery*(PD) problem. The PD problem involves transporting goods from a set of vertices known as pickup nodes to another set known as delivery nodes. Every pickup node has a corresponding delivery node that must be reached before returning

to the depot. In this scenario, vertices $i, j \in V$, s.t. $\{i, j\} \in \hat{R}$ can serve as both pickup or depot nodes, depending on which node is reached first in the current path. Therefore, after reaching the first node, it can be identified as pickup and the following node must be the second node, recognized as delivery.

- A strong branching procedure is employed using two types of branching with the same priority:

  1. Branching on the CVRP x variables that is the original CVRP branching. This type of branching does not affect the PP and is very effective, especially deeper in the BB tree when other types of branching variables can be not available.
  2. Branching on the degree of vertices $i \in V$ of the CARP network. This type of branching was successfully employed in [68], [63], [17]. The PP is not impacted, and these types of branching variables are generally preferred in the initial nodes of the BB tree.

Chapter 6

Results

6.1 Instances

Several standard benchmarks are employed to assess the efficacy of the algorithms put forward by the authors. The website of Prof. S. Irnich proposes all classical instances with the associated optimal values found in several literature works. The instances adhere to a standard format that specifies the number of vertices, the count of required and non-required arcs, the depot node, the number of vehicles required, and a list of arcs that include associated costs and demands (if applicable)[1]. The following instances are used to evaluate the proposed solution:

- **Eglese Instances** with 24 problems [33], based on real-world data of streets in Lancashire (UK). The set was originally built for a gritting issue. The graphs contain between 98 and 190 edges, with the required number of edges varying between 51 and 190. This specific set of instances appears consistently in all recent literature on CARP.
- **BBCM Instances a.k.a. VAL Instances** were introduced by Benavent et al. [13] in 1992. Every CARP instance is defined over a randomly generated graph with a number of vertices between 24 and 50 and a number of edges between 39 and 97, all required.
- **BMCV Instances** with 100 problems [14], based on a partitioning of the inter-city road network of Flanders (Belgium) into two districts, for solving a winter gritting problem. The instances are divided into four groups C, D, E, and F of 25 problems each. C and E present a capacity of $Q = 300$ while D, and F have a capacity of $Q = 600$. Moreover, all the costs are multiple of 5, thus the solutions will be multiples of 5.

[1]For further information about the standard format for CARP instances consult https://www.uv.es/~belengue/carp/READ_ME

Three other classes of instances are not reported: **KSHS** [58] and **GDB** [44] are the easiest instances and are solved in a fraction of a second, while **Egl-large** [20] are too hard for an exact solver.

6.1.1 Implementation Setup

For the algorithm that was implemented and described in Chapter 5, two parameter settings were given:

1. For the Eglese, the parameters include the use of Rank-1 Cuts with 50 cuts per round, 120 candidates for the first strong branching phase, and 3 for the second stage. The R1C is necessary for solving these hard instances even if the PP is penalized.

2. For C, E from BMCV Rank-1 Cuts with 20 cuts per round are implemented, then 100 candidates for the first strong branching phase and 2 for the second stage. In this case, the R1C has more impact on the PP with respect to Eglese instances and since the problems are easier, the number in each round was reduced.

3. For Val, D, and F the Rank-1 Cuts are disabled. Since the routes in these categories are lengthier, the Limited Memory R1Cs may not be as efficient and could significantly prolong the pricing process. The strong branching was reduced with respect to the other setting, using 50 candidates for the first stage and 2 for the second in order to speed up the process.

The evaluations occur with a time limit of 14,400 seconds (4 hours) for the most straightforward instances and 130,000 seconds for the most challenging ones. Moreover, no *Ryan and Foster branching rule* or *branching over the accumulated consumption of a resource* was used.

Finally, the initial upper bounds for each experiment were chosen as the best available bounds in the literature.

6.2 Empirical Results

6.2.1 Other Benchmarks

To facilitate comparison of the results obtained, a list of several algorithms that were previously discussed in Chapter 3 is given. Each algorithm is accompanied by its corresponding abbreviation and algorithm type.

- Longo, Poggi, and Uchoa [55] [**LPU06**] : BCP
- Bode and Irnich [16] **BI12** : Cut-First Branch-and-Price

- Bartolini, Cordeau, Laporte [10] **BLC13** : Cut and column generation
- Bode and Irnich [18] **BI14** : Cut-First Branch-and-Price Second
- Foulds, Longo, Martins [40] **FLM15** : BCP
- Bode and Irnich [17] **BI15** : Cut-First Branch-and-Price Second
- Martinelli, Malta, Poggi [59] **MMP16** : Cut and column generation
- Diego Pecin, Eduardo Uchoa [63] **PU19** : BCP

6.2.2 Summary of CARP instances

A summary comparison of the cited instances is now given. The tables contain columns **RGap** that represent the average percent gap in the root node. This is calculated based on the root lower bounds and in relation to the same upper bounds that are best known. The table shows the average root times (in seconds) for each processor listed as **RT**. To provide an idea of the processors' relative speed, the scores from the PassMark site are listed in table 6.1. The **Opt** columns indicate the number of instances that the algorithm solved to optimality using either branching, enumeration, or both methods. The count always encompasses situations wherein the algorithm discovers a lower bound that demonstrates the present upper bound's optimality.

The results illustrate how the proposed algorithm solved nearly all classical instances to the optimum. Additionally, among the solutions for the Eglese, C, and E instances, the RGap column displays the lowest values. However, the remaining three categories of instances demonstrate difficulties in solving the examples that are conventionally viewed as easier, with high values for the root node evaluation and root node gap. The forthcoming section, section 6.2.3, will elaborate on the findings to address the encountered challenges.

6.2.3 Detailed CARP Instances

Tables (6.3),(6.4),(6.6),(6.5),(6.7),(6.8) show the details for each instance presented in section 6.1. The tables focus on the notable contribution of Pecin and Uchoa's **PU19** work, as it presently dominates as the leading CARP algorithm. Therefore, the key parameters for each instance are presented for comparison with those obtained by the proposed algorithm. For the other authors described in section 6.2.1 only the total time is given. The following list describes those parameters:

- **IUB** : best upperbound available. The value was used to run the algorithm, as the upper bound.
- **OPT** : value obtained at the end of the execution of the algorithm.

Table 6.1: Processors used in the experiments.

Algorithm	Processor	Single-thread Score
LPU06	Pentium IV 2.4 GHz	-
BI12	i7-2600 3.4 GHz	1740
BLC13	Xeon E5310 1.6 GHz	639
BI14	i7-2600 3.4 GHz	1740
FLM15	Core i3 3.0 GHz	1764
BI15	i7-2600 3.4 GHz	1740
MMP16	i7-3960X 3.3 GHz	1793
PU19	Xeon E5-2637 3.5 GHz	2203
Proposed BPC	i9-12900K 3.2 GHz	4189

- **RLB** : root node lower bound for the proposed algorithm. This lower bound is achieved by separating robust cuts (RCC and Lifted Odd-Edge-Cutset), and non-robust R1C if applicable.
- $\mathbf{RLB_{PU19}}$: root node lowerbound for the PU19 algorithm.
- **T1** : time to evaluate the root node for this work in seconds.
- $\mathbf{T1_{PU19}}$: time to evaluate the root node for PU19 in seconds.
- **Nds** : number of nodes in the BB tree explored by the proposed algorithm.
- $\mathbf{Nds_{PU19}}$: number of nodes in the BB tree explored by the PU19 algorithm.
- **TT** : total time, in seconds, of running for the proposed algorithm.
- $\mathbf{TT_{PU19}}$: total time, in seconds, of running for PU19 algorithm.

The rightmost columns represent the other algorithms' results. If a value is present then the instance is solved. BI15 does not display the total time in seconds but rather indicates whether the instance was resolved within 4 hours or 12 hours. Moreover, BI12, BI14, and BI15 do not employ external upper bounds, instead relying on the algorithm to search for and demonstrate the feasibility and optimization of a solution within a specified time limit. In cases where the algorithm concludes within the time limit, the authors note whether the lower bound established is sufficient to demonstrate the optimality of previously reported solutions. Finally, the bold values in RLB columns highlight when the algorithm exceeds the value provided in Pecin and Uchoa's work.

Table 6.2: Summary of CARP benchmarks.

	LPU06			BI12			BCL13		
Class/NP	RGap	RT	Opt	RGap	RT	Opt	RGap	RT	Opt
Eglese/24	0.57	1,334	2	0.77	582	5	0.30	3,268	10
Val/34	0.23	1,939	26	0.24	11	34	0.16	755	29
C/25						0.41	2,293	14	
D/25						0.30	1,629	19	
E/25						0.26	2,860	14	
F/25						0.10	1,468	19	
Total/158			28			39			105
	BI14			FLM15			BI15		
Class/NP	RGap		Opt	RGap	RT	Opt	RGap	RT	Opt
Eglese/24	0.61		6	0.96	61	4	0.51	2,601	7
Val/34									
C/25	0.88		17				0.83	215	20
D/25	0.44		22				0.41	3,978	23
E/25	0.57		19				0.51	304	23
F/25	0.13		23				0.13	4,610	23
Total/158			87			4			96
	MMP16			PU19			Proposed BPC		
Class/NP	RGap	RT		RGap	RT	Opt	RGap	RT	Opt
Eglese/24	0.29	2,228		0.24	2,216	23	0.23	988	19
Val/34				0.18	45	34	0.56	89	34
C/25				0.29	462	25	0.28	173	24
D/25				0.30	74	25	0.66	254	25
E/25				0.17	489	25	0.17	283	25
F/25				0.08	55	24	0.40	378	24
Total/158						156			151

Table 6.3: Detailed results for the Eglese instances.

Ins	IUB	OPT	RLB	RLB_{PU19}	T1	$T1_{PU19}$	Nds	Nds_{PU19}	TT	TT_{PU19}	LPU06	BI12	BCL13	BI14	FLM15	BI15
egl-e1-A	3,548	3,548	3,548	3,548	34.2	33	1	1	34	33	144	1,621	16	589	775	$\leq$ 4h
egl-e1-B	4,498	4,498	**4,498**	4,492	75.64	389	1	1	75	389		1,836	2,620	3,827	2,513	$\leq$ 4h
egl-e1-C	5,595	5,595	**5571**	5,565	362.75	1386	3	1	652	1,386			733		9,715	
egl-e2-A	5,018	5,018	5,018	5,018	96.82	81	1	1	96	81		1,991		6,345		$\leq$ 4h
egl-e2-B	6,417	6,317	**6,305**	6,302	304.73	425	5	3	941	629						$\leq$ 4h
egl-e2-C	8,335	8,335	**8,315**	8,307	206.56	489	13	1	644	489			4,204			
egl-e3-A	5,898	5,898	5,898	5,898	430.48	233	3	1	1,318	233	924	844	73	761		$\leq$ 4h
egl-e3-B	7,775	7,775	7,738	7,744	527	10,649	215	17	98,146	35,702						
egl-e3-C	10,292	10,292	**10,260**	10,233	263.9	465	135	65	11,851	5,590						
egl-e4-A	6,444	6,444	6,403	6,410	6,419.1	5248	275	163	127,588	94,657						
egl-e4-B	8,961	8,961	**8,940**	8,930	1056.2	1958	27	17	11,690	8,530						
egl-e4-C	11,529	11,529	**11,506**	11,498	192.09	263	19	9	595	609						
egl-s1-A	5,018	5,018	5,018	5,018	80.21	209	1	1	80	209		8,670	1,232	4,312		$\leq$ 4h
egl-s1-B	6,388	6,388	**6,388**	6,385	47.49	82	1	1	47	82			344	12,250		$\leq$ 4h
egl-s1-C	8,518	8,518	**8,518**	8,514	22.77	44	1	1	22	44			196		20,879	
egl-s2-A	9,868		9,838	9,842	4,375.8	9955		145		94,774						
egl-s2-B	13,057	13,057	13,014	13,021	1,349.61	2,988	181	53	117,874	76,094						
egl-s2-C	16,425	16,425	**16,396**	16,395	247.8	285	51	25	1,875	872			15,083			
egl-s3-A	10,201		10,156	10,167	2508.0	6,310		145		89,406						
egl-s3-B	13,682		13,643	13,651	861.0	2,989		67		51,609						
egl-s3-C	17,188	17,188	**17,163**	17,152	682.5	551	117	3	3,456	913			13,202			
egl-s4-A	12,216		12,151	12,168	2,907.0	3,990										
egl-s4-B	16,187		16,113	16,142	2275.0	5695		315		337,962						
egl-s4-C	20,461	20,461	**20,446**	20,433	864.56	936	7	1	1,326	936						
Solved									**19**	**23**	**2**	**5**	**10**	**6**	**4**	**7**

Table 6.4: Detailed results for the C instances.

Ins	IUB	OPT	RLB	RLB_{PU19}	T1	$T1_{PU19}$	Nds	Nds_{PU19}	TT	TT_{PU19}	BCL13	BI14	BI15
C01	4,150	4,150	**4,138**	4,133	621.97	281.0	5	7	1,889	934			$\leq$ 12h
C02	3,135	3,135	3,135	3,135	28.81	5.3	1	1	28	5	5	60	$\leq$ 4h
C03	2,575	2,575	2,566	2,569	73.64	40.0	5	1	185	40	548	252	$\leq$ 4h
C04	3,510	3,510	3,494	3,494	158.66	95.0	27	5	1,435	188		4,403	$\leq$ 4h
C05	5,365	5,365	5,365	5,365	118.31	10.0	1	1	118	10	2,462	272	$\leq$ 4h
C06	2,535	2,535	2,524	2,525	38.01	28.0	5	3	161	63	1,278	196	$\leq$ 4h
C07	4,075	4,075	**4,075**	4,033	52.67	239.0	1	13	52	387	827	724	$\leq$ 4h
C08	4,090	4,090	4,090	4,090	38.56	12.0	1	1	38	12	95	634	$\leq$ 4h
C09	5,260	5,260	5,239	5,239	177.25	134.0	135	17	6,016	10,343			
C10	4,700	4,700	**4,666**	4,660	48.47	30.0	3	21	60	100	128	1,493	$\leq$ 4h
C11	4,630		4,593	4,595	584.74	2,570.0		39		22,889			
C12	4,240	4,240	**4,210**	4,209	118.99	137.0	3	3	202	193			$\leq$ 12h
C13	2,955	2,955	**2,940**	2,939	55.69	59.0	9	3	209	102	4,200	589	$\leq$ 4h
C14	4,030	4,030	**4,021**	4,020	41.48	8.2	3	1	44	8	381		$\leq$ 4h
C15	4,940	4,940	**4,928**	4,924	221.42	161.0	69	11	4,939	788			
C16	1,475	1,475	1,475	1,475	22.64	1.5	1	1	22	1	24	196	$\leq$ 4h
C17	3,555	3,555	3,555	3,555	12.05	1.6	1	1	12	1	3	23	$\leq$ 4h
C18	5,605	5,605	5,580	5,584	626.35	963.0	25	27	3556	9,391			
C19	3,115	3,115	3,115	3,115	89.12	31.0	1	1	89	31		6,706	$\leq$ 4h
C20	2,120	2,120	2,120	2,120	35.24	5.8	1	1	35	5	12	392	$\leq$ 4h
C21	3,970	3,970	3,962	3,963	232.75	53.0	23	3	5,561	75		3,284	$\leq$ 4h
C22	2,245	2,245	2,245	2,245	7.56	4.3	1	1	7	4	5	256	$\leq$ 4h
C23	4,085	4,085	4,040	4,072	786.21	6,214.0	76	7	22,317	7,783			
C24	3,400	3,400	3,392	3,392	130.81	227.0	9	3	479	388		2,325	$\leq$ 4h
C25	2,310	2,310	2,310	2,310	5.15	1.5	1	1	5	1	2	20	$\leq$ 4h
Solved									**24**	**25**	**14**	**17**	**20**

6.3 Discussion

In this section the results presented in tables (6.3)-(6.8) are exposed. It is worth noting that the only comparable literature results come from the state-of-the-art algorithm proposed by Pecin and Uchoa in [63]. The other results, particularly for Eglese instances, are not as effective as the proposed solution in both RLB and final results.

Eglese Instances

The Eglese instances, table (6.3), are a widely adopted benchmark in literature, featuring a high volume of nodes, edges, and vehicles. The routes typically span short to medium paths. The proposed algorithm presents good results in solving the instances, also outperforming the state-of-the-art PU19 algorithm in some experiments. The most valuable result is that the RLB is greater than the competitor in many instances, opening new possibilities for further developments. However, in some instances e.g. egl-e3-B the total time presents a very high value: this is due to the non-convergence in some CG iterations. The issue was identified also in the study by Pessoa et al. [68], who also discovered similar problems in challenging Generalized

Table 6.5: Detailed results for the E instances.

Ins	IUB	OPT	RLB	RLB_{PU19}	T1	$T1_{PU19}$	Nds	Nds_{PU19}	TT	TT_{PU19}	BCL13	BI14	BI15
E01	4,910	4,910	4,886	4,886	179	574.0	19	21	1,329	2,053			
E02	3,990	3,990	**3,990**	3,981	59	31.0	1	1	59	31	252	862	$\leq$ 4h
E03	2,015	2,015	2,015	2,015	12	6.3	1	1	12	6	9	23	$\leq$ 4h
E04	4,155	4,155	4,152	4,155	154	30.0	3	1	168	30	2,042	2,789	$\leq$ 4h
E05	4,585	4,585	4,585	4,585	20	7.4	1	1	20	7	61	68	$\leq$ 4h
E06	2,055	2,055	2,055	2,055	10	4.0	1	1	10	4	4	54	$\leq$ 4h
E07	4,155	4,155	**4,128**	4,126	111	135.0	3	9	121	197	319	6,684	$\leq$ 4h
E08	4,710	4,710	**4,706**	4,702	58	13.0	3	3	83	5,781	200	198	$\leq$ 4h
E09	5,810	5,810	5,787	5,787	294	231.0	53	13	14,403	1,712			
E10	3,605	3,605	3,605	3,605	4	2.5	1	1	4	2	4	48	$\leq$ 4h
E11	4,650	4,650	**4,645**	4,644	865	414.0	7	1	1,731	414		1,440	$\leq$ 4h
E12	4,180	4,180	4,160	4,161	67	33.0	9	5	150	52	6,585		$\leq$ 4h
E13	3,345	3,345	**3,336**	3,334	79	69.0	9	3	337	101	1,362	437	$\leq$ 4h
E14	4,115	4,115	4,115	4,115	20	5.9	1	1	20	5	67	2,089	$\leq$ 4h
E15	4,205	4,205	4,190	4,194	747	1,028.0	1	59	77	35,224			$\leq$ 12h
E16	3,775	3,775	3,771	3,775	249	89.0	3	1	324	89		380	$\leq$ 4h
E17	2,740	2,740	2,740	2,740	11	2.3	1	1	11	2	3	10	$\leq$ 4h
E18	3,835	3,835	3,825	3,828	268	2,174.0	15	7	14987	2,648			$\leq$ 4h
E19	3,235	3,235	3,235	3,235	148	47.0	1	1	148	47		13,935	$\leq$ 4h
E20	2,825	2,825	2,807	2,807	108	292.0	3	3	329	378		3,112	$\leq$ 4h
E21	3,730	3,730	3,728	3,730	254	20.0	7	1	612	20		13,935	$\leq$ 4h
E22	2,470	2,470	2,470	2,470	32	5.8	1	1	32	5	149	74	$\leq$ 4h
E23	3,710	3,710	3,701	3,710	3,233	1,161.0	69	1	12,275	1,161			$\leq$ 4h
E24	4,020	4,020	4,009	4,011	113	81.0	53	7	11,955	186		13,135	$\leq$ 4h
E25	1,615	1,615	1,615	1,615	2	0.7	1	1	2		1	2	$\leq$ 4h
Solved									**25**	**25**	**14**	**19**	**23**

Table 6.6: Detailed results for the D instances.

Ins	IUB	OPT	RLB	RLB_{PU19}	T1	$T1_{PU19}$	Nds	Nds_{PU19}	TT	TT_{PU19}	BCL13	BI14	BI15
D01	3,215	3,215	3,215	3,215	393.71	90.0	15	1	1,510	90	84	4,117	$\leq$ 4h
D02	2,520	2,520	2,520	2,520	56.63	12.0	17	1	246	12	19	286	$\leq$ 4h
D03	2,065	2,065	2,010	2,065	38.31	12.0	4,320	1	16,721	12	49	1,472	$\leq$ 4h
D04	2,785	2,785	2,785	2,785	133.23	24.0	35	1	451	24	70	9,022	$\leq$ 4h
D05	3,935	3,935	3,911	3,935	96.7	20.0	17	1	241	20	22	166	$\leq$ 4h
D06	2,125	2,125	2,095	2,125	32.7	6.8	29	1	222	6	16	1,615	$\leq$ 4h
D07	3,115	3,115	3,047	3,054	46.23	20.0	211	45	1,599	247	10,446		
D08	3,045	3,045	2,995	3,004	44.3	45.0	99	25	683	328		3,730	$\leq$ 4h
D09	4,120	4,120	4,120	4,120	163.0	52.0	55	1	1,760	52	103	1,654	$\leq$ 4h
D10	3,340	3,340	3,332	3,333	36.7	15.0	25	1	202	15	107	493	$\leq$ 4h
D11	3,745	3,745	3,745	3,745	1,086.55	71.0	27	1	9,231	71	123	11,009	$\leq$ 4h
D12	3,310	3,310	3,310	3,310	234.31	105.0	19	1	1,377	105	78	198	$\leq$ 4h
D13	2,535	2,535	2,525	2,535	54.41	15.0	13	1	332	15	18	605	$\leq$ 4h
D14	3,280	3,280	3,225	3,272	63.37	23.0	361	3	7,001	32		1,804	$\leq$ 4h
D15	3,990	3,990	3,990	3,990	795.5	247.0	3	1	9,214	247	336	602	$\leq$ 4h
D16	1,060	1,060	1,060	1,060	21.37	161.0	5	1	35	161	9	667	$\leq$ 4h
D17	2,620	2,620	2,562	2,620	19.76	23.0	7	1	80	23	9	42	$\leq$ 4h
D18	4,165	4,165	4,165	4,165	834.8	139.0	63	1	3,103	139	455	2,951	$\leq$ 4h
D19	2,400	2,400	2,392	2,395	89.91	18.0	125	3	2,257	35		13,090	$\leq$ 4h
D20	1,870	1,870	1,870	1,870	59.81	13.0	19	1	805	13	27	2,004	$\leq$ 4h
D21	3,050	3,050	2,986	2,988	163.53	61.0	1,235	237	51,108	30,560			
D22	1,865	1,865	1,865	1,865	128.61	7.8	7	1	345	7	20	3,200	$\leq$ 4h
D23	3,130	3,130	3,117	3,120	1,023	249.0	37	17	1,515	13,521		14,399	$\leq$ 4h
D24	2,710	2,710	2,675	2,676	236.73	385.0	42	15	12,458	2,428			$\leq$ 12h
D25	1,815	1,815	1,815	1,815	28.87	8.2	9	1	122	8	11	155	$\leq$ 4h
Solved									**25**	**25**	**19**	**22**	**23**

Table 6.7: Detailed results for the F instances.

Ins	IUB	OPT	RLB	RLB_{PU19}	T1	$T1_{PU19}$	Nds	Nds_{PU19}	TT	TT_{PU19}	BCL13	BI14	BI15
F01	4,040	4,040	3,955	4,040	511	82.0	39	1	7,525	82	88	2,170	$\leq$ 4h
F02	3,300	3,300	3,300	3,300	53	12.0	11	1	305	12	19	1,957	$\leq$ 4h
F03	1,665	1,665	1,665	1,665	72	20.0	9	1	297	20	23	507	$\leq$ 4h
F04	3,485	3,485	3,475	3,476	119	173.0	639	13	13,982	745		9,654	$\leq$ 4h
F05	3,605	3,605	3,605	3,605	190	47.0	13	1	453	47	28	1,023	$\leq$ 4h
F06	1,875	1,875	1,830	1,875	22	7.2	25	1	368	7	12	350	$\leq$ 4h
F07	3,335	3,335	3,335	3,335	32	8.3	7	1	164	8	11	226	$\leq$ 4h
F08	3,705	3,705	3,691	3,693	38	42.0	17	3	202	54		1,927	$\leq$ 4h
F09	4,730	4,730	4,730	4,730	553	88.0	29	1	3,721	88	137	526	$\leq$ 4h
F10	2,925	2,925	2,920	2,925	21	9.5	383	1	8,134	9	8	373	$\leq$ 4h
F11	3,835	3,835	3,820	3,835	546	122.0	49	1	8,937	122	134	4,889	$\leq$ 4h
F12	3,395	3,395	3,390	3,390	410	75.0	167	3	4,890	85		2,341	$\leq$ 4h
F13	2,855	2,855	2,855	2,855	40	12.0	15	1	351	12	14	306	$\leq$ 4h
F14	3,330	3,330	3,330	3,330	34	36.0	27	1	316	36	26	822	$\leq$ 4h
F15	3,560	3,560	3,560	3,560	1,017	158.0	245	1	23,591	158	279	277	$\leq$ 4h
F16	2,725	2,725	2,725	2,725	30	30.0	11	1	88	30	27	543	$\leq$ 4h
F17	2,055	2,055	2,055	2,055	10	5.4	3	1	27	5	6	90	$\leq$ 4h
F18	3,075		3,060	3,063	279	74.0							
F19	2,525	2,525	2,499	2,504	3,952	87.0	71	9	5,305	6,571			
F20	2,445	2,445	2,417	2,445	151	15.0	143	1	10,022	15	42	2,210	$\leq$ 4h
F21	2,930	2,930	2,930	2,930	396	26.0	35	1	4,084	26	70	3,582	$\leq$ 4h
F22	2,075	2,075	2,060	2,075	57	5.6	25	1	567	5	18	141	$\leq$ 4h
F23	3,005	3,005	2,994	3,005	471	135.0	51	1	3,932	135			$\leq$ 4h
F24	3,210	3,210	3,200	3,210	444	117.0	131	1	8,484	117	146	10,106	$\leq$ 4h
F25	1,390	1,390	1,390	1,390	7	2.7	5	1	21	2	2	89	$\leq$ 4h
Solved									**24**	**24**	**19**	**23**	**23**

Table 6.8: Detailed results for the Val instances.

Ins	IUB	OPT	RLB	RLB_{PU19}	T1	$T1_{PU19}$	Nds	Nds_{PU19}	TT	TT_{PU19}	LPU06	BI12	BCL13
1A	247	247	247	247	11	8.7	37	1	251	8	98	17	5
1B	247	247	247	247	13	5.9	19	1	144	5	55	4	4
1C	319	319	317	319	15	0.8	23	1	47	1	8,917	56	134
2A	298	298	294	298	26	5.1	19	1	110	5	79	10	5
2B	330	330	328	329	20	8.1	13	5	57	18	671	9	290
2C	528	528	528	528	1	0.3	1	1	1	1	1	1	
3A	105	105	105	105	8	4.7	21	1	110	4	128	6	5
3B	111	111	111	111	8	3.0	11	1	49	3	134	3	3
3C	162	162	161	161	7	0.3	9	1	20	1	329	2	3
4A	522	522	522	522	177	59.0	57	1	1,686	59	2,475	413	80
4B	534	534	534	534	152	47.0	55	1	1,142	47	1,178	73	50
4C	550	550	550	550	121	40.0	23	1	787	40	825	32	28
4D	650	650	647	648	36	20.0	337	5	1,985	133		716	
5A	566	566	566	566	101	31.0	75	1	5,913	31	629	76	34
5B	589	589	587	588	114	40.0	237	5	8,985	65		25	
5C	617	617	612	613	55	25.0	1,273	55	23,283	306		406	
5D	718	718	714	716	19	7.5	163	9	1,145	27		28	702
6A	330	330	326	330	42	11.0	279	1	8,876	11	159	20	11
6B	340	340	336	337	31	16.0	737	13	6,263	66		209	
6C	424	424	419	419	6	1.5	25	125	40	54		4,027	56
7A	382	382	377	382	46	56.0	213	1	7,731	56	319	$\leq$ 4h	25
7B	386	386	381	386	42	28.0	67	1	2,315	28	164	35	13
7C	437	437	433	433	44	19.0	133	15	2,529	71	132	$\leq$ 4h	1,760
8A	522	522	522	522	66	24.0	25	1	1,654	24	359	38	21
8B	531	531	531	531	30	18.0	49	1	1,869	18	169	18	16
8C	657	657	653	654	25	6.7	127	19	940	85		25	2,496
9A	450	450	442	450	91	187.0	429	1	9620	187	17,723	685	213
9B	453	453	448	453	78	72.0	373	1	21,876	72	4,520	154	115
9C	459	459	453	459	268	85.0	81	1	7,989	85	1,461	75	63
9D	515	515	509	512	98	21.0	1,219	45	22,718	289		$\leq$ 4h	
10A	637	637	637	637	466	237.0	321	1	25,584	237	13,337	542	215
10B	645	645	640	645	322	204.0	278	1	23,369	204	13,719	7,555	157
10C	655	655	655	655	326	181.0	97	1	3,827	181	5,079	172	97
10D	734	734	733	734	182	68.0	157	1	4,962	68	474	196	61
Solved									**34**	**34**	**26**	**34**	**29**

Assignment Problem (GAP) tasks. To address this, VRPSolver utilizes stabilization techniques explained in Pessoa et al.'s previous work [66]. These techniques prove effective for simpler instances, but they fall short for the most difficult ones. This problem is preventing the resolution of the five unresolved instances in a lengthy loop.

C and E Instances

C and E instances, tables (6.4),(6.5), present a moderate level of difficulty, as they offer mostly medium-length paths with few long ones. For these instances, the same principles used in Eglese's case are applied. The RLB value surpasses that of the PU19 outcomes, although the overall results are similar in most cases. However, the total time can be considerably greater than that of the competitor. This issue can be tackled in the branching process. Notably, the number of explored nodes is significantly higher in these instances, indicating that the proposed algorithm faces difficulties in demonstrating whether the obtained result is optimal. The proposal to reduce R1C cuts compared to Eglese instances has been suggested to address the issue of convergence. This approach has demonstrated encouraging outcomes for nearly all cases. Furthermore, the only unresolved instance, namely C11, suffers from the aforementioned non-convergence problem.

D and F Instances

The D and F instances, tables (6.6),(6.7), have double the capacity and approximately half the vehicles of the C and E instances. Furthermore, their routes are considerably longer in comparison to those of Eglese, C, and D. Although most of the instances were resolved, the total time required was significantly high when compared to PU19. Furthermore, the RLB exhibits inferior values compared to its competitor, indicating poorer performance. The main issue is that the proposed model faces difficulties in producing long routes and needs a different implementation to face such a problem.

Val Instances

Val instances, table (6.8), have a smaller number of arcs, nodes, and vehicles. Moreover, the routes are considerably longer than those in other instances. In line with the D and F considerations, the algorithm exhibits limitations in creating long routes for Val instances. However, the RLB value is almost always comparable to the optimal value, which is conclusive in confirming the bound's optimality. On the contrary, although the optimal bound can be generated in a few seconds, creating the route itself requires a significant amount of time since a large number of nodes in the BB tree must be explored before the appropriate columns can be generated.

Chapter 7

Conclusions

This dissertation presents a study on the exact resolution of the *Capacitated Arc Routing Problem* (CARP). To achieve this, an algorithm is proposed that has its origins in the method established by Baldacci and Maniezzo [7] in 2006. The transformation technique developed in their research was used to convert the CARP to a *Capacitated Vehicle Routing Problem* (CVRP).

The improvements made over time for the CVRP have made it possible to solve the problem for large instances within a limited timeframe. The implementation of the *Branch-and-Cut-and-Price* (BCP) method was instrumental in enabling the use of exact solvers in real-world scenarios. Numerous studies have focused on the *Branch-and-Cut-and-Price* method for *Vehicle Routing Problems* (VRPs) in the past two decades. As a result of these efforts, the state-of-the-art tool *VRPSolver* has been developed [68].

After recent advancements, the Capacitated Arc Routing Problem was reformulated to a Capacitated Vehicle Routing Problem. It was then modelled as a Resource Constrained Shortest Path Problem (RCSPP) to make use of the VRPSolver. Two dictionaries were used to effectively translate the problem domain and enable branching over the CARP nodes degree, as well as the implementation of robust cuts such as *Lifted Odd-Cutsets*. Furthermore, the utilization of an adapted *Pickup and Delivery* problem formulation enabled the integration of additional constraints resulting from the transformation, without affecting the *Column Generation* Procedure.

Pecin and Uchoa [63] introduced the cutting-edge method for the Capacitated Arc Routing Problem (CARP) exact resolution. The authors created a solution that encompasses all the outstanding solutions for CARP up to 2019 and resolved 22 out of 24 unresolved instances. The algorithm proposed in this work exhibits exceptionally favorable results despite not utilizing all the features of CARP and is comparable to that of Pecin and Uchoa.

Among the classical literature instances, the Eglese's one are the most widely

used in CARP literature, with medium-long paths and high difficulty in the resolution. The proposed solution efficiently solved the majority of these problems, outperforming Pecin and Uchoa's results in terms of the root lower bound for most of the results and total time for certain instances.

On the flip side, this solution has several shortcomings. Instances with lengthy routes, despite being solved, demonstrated considerably long runtimes and inferior root node lowerbounds when compared to those of other authors. Additionally, hard instances demonstrate that the column generation procedure does not always converge, which significantly increases the running time. Lastly, the branching procedure, particularly when a solution is discovered, lacks balance, resulting in a lengthy amount of time required to close the duality gap.

Finally, these results are promising and require further research to address the listed limitations. The ongoing development of the VRPSolver may alleviate some of these issues, e.g. the convergence problem. Additionally, a reassessment of the transformation using a more RCSPP approach could improve the solver's efficiency, when combined with a suitable branching method.

Bibliography

[1] N. R. Achuthan, L. Caccetta, and S. P. Hill. «A new subtour elimination constraint for the vehicle routing problem». In: *European Journal of Operational Research* 91.3 (1996), pp. 573–586. URL: https://EconPapers.repec.org/RePEc:eee:ejores:v:91:y:1996:i:3:p:573-586.

[2] Yogesh Agarwal, Kamlesh Mathur, and Mehmet Salkin. «A set-partitioning-based exact algorithm for the vehicle routing problem». In: *Networks* 19 (Dec. 1989), pp. 731–749. DOI: 10.1002/net.3230190702.

[3] David L. Applegate et al. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006. ISBN: 9780691129938. URL: http://www.jstor.org/stable/j.ctt7s8xg.

[4] Philippe Augerat. «Approche polyèdrale du problème de tournées de véhicules». Theses. Institut National Polytechnique de Grenoble - INPG, June 1995. URL: https://theses.hal.science/tel-00005026.

[5] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. «An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts». In: *Math. Program.* 115 (Oct. 2008), pp. 351–385. DOI: 10.1007/s10107-007-0178-5.

[6] Roberto Baldacci, Eleni Hadjiconstantinou, and Aristide Mingozzi. «An Exact Algorithm for the Capacitated Vehicle Routing Problem Based on a Two-Commodity Network Flow Formulation». In: *Operations Research* 52 (Oct. 2004), pp. 723–738. DOI: 10.1287/opre.1040.0111.

[7] Roberto Baldacci and Vittorio Maniezzo. «Exact methods based on node-routing formulations for undirected arc-routing problems». In: *Networks* 47 (Jan. 2006), pp. 52–60. DOI: 10.1002/net.20091.

[8] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. «New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem». In: *Operations Research* 59.5 (2011), pp. 1269–1283. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/41316029.

[9] Michel Balinski and R. E. Quandt. «On an Integer Program for a Delivery Problem». In: *Operations Research* 12.2 (1964), pp. 300–304. URL: https://EconPapers.repec.org/RePEc:inm:oropre:v:12:y:1964:i:2:p:300-304.

[10] Enrico Bartolini, Jean-François Cordeau, and Gilbert Laporte. «Improved lower bounds and exact algorithm for the capacitated arc routing problem». In: *Mathematical Programming* 137 (Feb. 2011). DOI: 10.1007/s10107-011-0497-4.

[11] José M. Belenguer and Enrique Benavent. «A cutting plane algorithm for the capacitated arc routing problem». In: *Computers & Operations Research* 30.5 (2003), pp. 705–728. ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(02)00046-1. URL: https://www.sciencedirect.com/science/article/pii/S0305054802000461.

[12] José-Manuel Belenguer and Enrique Benavent. «The Capacitated Arc Routing Problem: Valid Inequalities and Facets». In: *Computational Optimization and Applications* 10 (1998), pp. 165–187. URL: https://api.semanticscholar.org/CorpusID:5605316.

[13] Enrique Benavent et al. «The Capacitated Arc Routing Problem: Lower bounds». In: *Networks* 22 (Dec. 1992), pp. 669–690. DOI: 10.1002/net.3230220706.

[14] Patrick Beullens, Luc Muyldermans, and Dirk Cattrysse. «A guided local search heuristic for the capacitated arc routing problem». In: *European Journal of Operational Research* 147 (June 2003), pp. 629–643. DOI: 10.1016/S0377-2217(02)00334-X.

[15] Ulrich Blasum and Winfried Hochstättler. «Application of the Branch and Cut Method to the Vehicle Routing Problem». In: (July 2004).

[16] Claudia Bode and Stefan Irnich. «Cut-First Branch-and-Price-Second for the Capacitated Arc-Routing Problem». In: *Operations Research* 60.5 (2012), pp. 1167–1182. DOI: 10.1287/opre.1120.1079. eprint: https://doi.org/10.1287/opre.1120.1079. URL: https://doi.org/10.1287/opre.1120.1079.

[17] Claudia Bode and Stefan Irnich. «In-Depth Analysis of Pricing Problem Relaxations for the Capacitated Arc-Routing Problem». In: *Transportation Science* 49.2 (May 2015), pp. 369–383. DOI: 10.1287/trsc.2013.0507. URL: https://ideas.repec.org/a/inm/ortrsc/v49y2015i2p369-383.html.

[18] Claudia Bode and Stefan Irnich. «The shortest-path problem with resource constraints with (k,2)-loop elimination and its application to the capacitated arc-routing problem». In: *European Journal of Operational Research* 238.2 (2014), pp. 415–426. DOI: 10.1016/j.ejor.2014.04.00. URL: https://ideas.repec.org/a/eee/ejores/v238y2014i2p415-426.html.

[19] Julien Bramel and David Simchi-Levi. «On the Effectiveness of Set Covering Formulations for the Vehicle Routing Problem with Time Windows». In: *Operations Research* 45.2 (1997), pp. 295–301. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/171746.

[20] José Brandão and Richard Eglese. «A Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem (CARP)». In: *Computers & Operations Research* 35 (Apr. 2008), pp. 1112–1126. DOI: 10.1016/j.cor.2006.07.007.

[21] N. Christofides and S. Eilon. «An Algorithm for the Vehicle-Dispatching Problem». In: *OR* 20.3 (1969), pp. 309–318. ISSN: 14732858. URL: http://www.jstor.org/stable/3008733.

[22] Nicos Christofides, Aristide Mingozzi, and Paolo Toth. «Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations». In: *Mathematical Programming* 20 (1981), pp. 255–282. URL: https://api.semanticscholar.org/CorpusID:18086758.

[23] G. Clarke and J. W. Wright. «Scheduling of Vehicles from a Central Depot to a Number of Delivery Points». In: *Operations Research* 12.4 (1964), pp. 568–581. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/167703.

[24] Claudio Contardo and Bernard Gendron. «A Branch-And-Cut-And-Price Algorithm for the Capacitated Location Routing Problem». In: 2009. URL: https://api.semanticscholar.org/CorpusID:14697050.

[25] Claudio Contardo and Rafael Martinelli. «A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints». In: *Discrete Optimization* 12 (2014), pp. 129–146. ISSN: 1572-5286. DOI: https://doi.org/10.1016/j.disopt.2014.03.001. URL: https://www.sciencedirect.com/science/article/pii/S1572528614000097.

[26] Ángel Corberán and Gilbert Laporte. *Arc Routing*. Ed. by Ángel Corberán and Gilbert Laporte. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2015. DOI: 10.1137/1.9781611973679. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611973679. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611973679.

[27] Luciano Costa, Claudio Contardo, and Guy Desaulniers. «Exact Branch-Price-and-Cut Algorithms for Vehicle Routing». In: *Transportation Science* 53.4 (2019), pp. 946–985. DOI: 10.1287/trsc.2018.0878. eprint: https://doi.org/10.1287/trsc.2018.0878. URL: https://doi.org/10.1287/trsc.2018.0878.

[28] G. B. Dantzig and J. H. Ramser. «The Truck Dispatching Problem». In: *Management Science* 6.1 (1959), pp. 80–91. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2627477.

[29] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. «A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows». In: *Operations Research* 40.2 (1992), pp. 342–354. DOI: 10.1287/opre.40.2.342. eprint: https://doi.org/10.1287/opre.40.2.342. URL: https://doi.org/10.1287/opre.40.2.342.

[30] Jacques Desrosiers and Marco Lübbecke. «Branch-Price-and-Cut Algorithms». In: Jan. 2011. ISBN: 9780470400531. DOI: 10.1002/9780470400531.eorms0118.

[31] Jacques Desrosiers, François Soumis, and Martin Desrochers. «Routing with time windows by column generation». In: *Networks* 14.4 (1984), pp. 545–565. DOI: https://doi.org/10.1002/net.3230140406. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230140406. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230140406.

[32] Moshe Dror. «Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW». In: *Operations Research* 42.5 (1994), pp. 977–978. DOI: 10.1287/opre.42.5.977. eprint: https://doi.org/10.1287/opre.42.5.977. URL: https://doi.org/10.1287/opre.42.5.977.

[33] R. W. Eglese and L. Y. O. Li. «Efficient Routeing for Winter Gritting». In: *Journal of the Operational Research Society* 43.11 (1992), pp. 1031–1034. DOI: 10.1057/jors.1992.160. eprint: https://doi.org/10.1057/jors.1992.160. URL: https://doi.org/10.1057/jors.1992.160.

[34] Dominique Feillet. «A tutorial on column generation and branch-and-price for vehicle routing problems». In: *4OR* 8 (Dec. 2010), pp. 407–424. DOI: 10.1007/s10288-010-0130-z.

[35] Dominique Feillet et al. «An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems». In: *Networks* 44.3 (2004), pp. 216–229. DOI: https://doi.org/10.1002/net.20033. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.20033. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20033.

[36] Matteo Fischetti, Paolo Toth, and Daniele Vigo. «A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs». In: *Operations Research* 42 (Oct. 1994), pp. 846–859. DOI: 10.1287/opre.42.5.846.

[37] Marshall L. Fisher. «Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees». In: *Operations Research* 42.4 (1994), pp. 626–642. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/171617.

[38] Merrill M. Flood. «The Traveling-Salesman Problem». In: *Operations Research* 4.1 (1956), pp. 61–75. URL: https://EconPapers.repec.org/RePEc:inm:oropre:v:4:y:1956:i:1:p:61-75.

[39] Robert W. Floyd. «Algorithm 97: Shortest Path». In: *Commun. ACM* 5.6 (June 1962), p. 345. ISSN: 0001-0782. DOI: 10.1145/367766.368168. URL: https://doi.org/10.1145/367766.368168.

[40] Les Foulds, Humberto Longo, and Jean Martins. «A compact transformation of arc routing problems into node routing problems». In: *Annals of Operations Research* 226 (Feb. 2015), pp. 177–200. DOI: 10.1007/s10479-014-1732-1.

[41] Ricardo Fukasawa et al. «Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem». In: *Math. Program.* 106 (July 2006), pp. 491–511. DOI: 10.1007/s10107-005-0644-x.

[42] M. R. Garey, D. S. Johnson, and R. Endre Tarjan. «The Planar Hamiltonian Circuit Problem is NP-Complete». In: *SIAM Journal on Computing* 5.4 (1976), pp. 704–714. DOI: 10.1137/0205049. eprint: https://doi.org/10.1137/0205049. URL: https://doi.org/10.1137/0205049.

[43] Sylvie Gélinas et al. «A new branching strategy for time constrained routing problems with application to backhauling». In: *Annals of Operations Research* 61 (Dec. 1995), pp. 91–109. DOI: 10.1007/BF02098283.

[44] B.L. Golden, J.S. Dearmon, and E.K. Baker. «Computational experiments with algorithms for a class of routing problems». In: *Computers & Operations Research* 10.1 (1983), pp. 47–59. ISSN: 0305-0548. DOI: https://doi.org/10.1016/0305-0548(83)90026-6. URL: https://www.sciencedirect.com/science/article/pii/0305054883900266.

[45] Bruce L. Golden and Richard T. Wong. «Capacitated arc routing problems». In: *Networks* 11.3 (1981), pp. 305–315. DOI: https://doi.org/10.1002/net.3230110308. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230110308. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230110308.

[46] R. E. Gomory and T. C. Hu. «Multi-Terminal Network Flows». In: *Journal of the Society for Industrial and Applied Mathematics* 9.4 (1961), pp. 551–570. DOI: 10.1137/0109047. eprint: https://doi.org/10.1137/0109047. URL: https://doi.org/10.1137/0109047.

[47] P. Z. Ingerman. «Algorithm 141: Path Matrix». In: *Commun. ACM* 5.11 (Nov. 1962), p. 556. ISSN: 0001-0782. DOI: 10.1145/368996.369016. URL: https://doi.org/10.1145/368996.369016.

[48] Mads Jepsen et al. «Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows». In: *Operations Research* 56 (Apr. 2008), pp. 497–511. DOI: 10.1287/opre.1070.0449.

[49] M. K. Kuan. «Graphic programming using odd or even points». In: (1962).

[50] Gilbert Laporte, Hélène Mercure, and Yves Nobert. «An exact algorithm for the asymmetrical capacitated vehicle routing problem». In: *Networks* 16.1 (1986), pp. 33–46. DOI: https://doi.org/10.1002/net.3230160104. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230160104. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230160104.

[51] J. K. Lenstra and A. H. G. Rinnooy Kan. «On general routing problems». In: *Networks* 6.3 (1976), pp. 273–280. DOI: https://doi.org/10.1002/net.3230060305. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230060305. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230060305.

[52] Adam Letchford. «Polyhedral Results for Some Constrained Arc-Routing Problems». In: (Jan. 1996).

[53] Adam Letchford and Juan José Salazar González. «Projection Results for Vehicle Routing». In: *Math. Program.* 105 (Mar. 2006), pp. 251–274. DOI: 10.1007/s10107-005-0652-x.

[54] John D. C. Little et al. «An Algorithm for the Traveling Salesman Problem». In: *Operations Research* 11.6 (1963), pp. 972–989. DOI: 10.1287/opre.11.6.972. eprint: https://doi.org/10.1287/opre.11.6.972. URL: https://doi.org/10.1287/opre.11.6.972.

[55] Humberto Longo, Marcus Poggi, and Eduardo Uchoa. «Solving capacitated arc routing problem using a transformation to the CVRP». In: *Computers & Operations Research* 33 (June 2006), pp. 1823–1837. DOI: 10.1016/j.cor.2004.11.020.

[56] Jens Lysgaard. «CVRPSEP: A package of separation routines for the Capacitated Vehicle Routing Problem». In: (Jan. 2003).

[57] Jens Lysgaard, Adam Letchford, and Richard Eglese. «A New Branch-and-Cut Algorithm for the Capacitated Vehicle Routing Problem». In: *Mathematical Programming* 100 (June 2004), pp. 423–445. DOI: 10.1007/s10107-003-0481-8.

[58] R. Hirabayashi M. Kiuchi Y. Shinano and Y. Saruwatari. «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 Oper. Res. Soc. of Japan, 28-29.* (1995).

[59] Rafael Martinelli, Marcus Poggi, and Anand Subramanian. «Improved bounds for large scale capacitated arc routing problem». In: *Computers & Operations Research* 40.8 (2013), pp. 2145–2160. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2013.02.013. URL: https://www.sciencedirect.com/science/article/pii/S0305054813000531.

[60] Donald L. Miller. «A Matching Based Exact Algorithm for Capacitated Vehicle Routing Problems». In: *ORSA Journal on Computing* 7.1 (1995), pp. 1–9. DOI: 10.1287/ijoc.7.1.1. eprint: https://doi.org/10.1287/ijoc.7.1.1. URL: https://doi.org/10.1287/ijoc.7.1.1.

[61] C. S. Orloff. «A fundamental problem in vehicle routing». In: *Networks* 4.1 (1974), pp. 35–64. DOI: https://doi.org/10.1002/net.3230040105. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230040105. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230040105.

[62] Wen-Lea Pearn, Arjang Assad, and Bruce L. Golden. «Transforming arc routing into node routing problems». In: *Computers & Operations Research* 14.4 (1987), pp. 285–288. ISSN: 0305-0548. DOI: https://doi.org/10.1016/0305-0548(87)90065-7. URL: https://www.sciencedirect.com/science/article/pii/0305054887900657.

[63] Diego Pecin and Eduardo Uchoa. «Comparative Analysis of Capacitated Arc Routing Formulations for Designing a New Branch-Cut-and-Price Algorithm». In: *Transportation Science* 53.6 (2019), pp. 1673–1694. DOI: 10.1287/trsc.2019.0900. eprint: https://doi.org/10.1287/trsc.2019.0900. URL: https://doi.org/10.1287/trsc.2019.0900.

[64] Diego Pecin et al. «Improved Branch-Cut-and-Price for Capacitated Vehicle Routing». In: *Integer Programming and Combinatorial Optimization*. Ed. by Jon Lee and Jens Vygen. Cham: Springer International Publishing, 2014, pp. 393–403. ISBN: 978-3-319-07557-0.

[65] Diego Pecin et al. «Improved branch-cut-and-price for capacitated vehicle routing». In: *Mathematical Programming Computation* 9 (June 2016). DOI: 10.1007/s12532-016-0108-8.

[66] A. Pessoa et al. «Automation and Combination of Linear-Programming Based Stabilization Techniques in Column Generation». In: *INFORMS Journal on Computing* 30.2 (2018), pp. 339–360. DOI: 10.1287/ijoc.2017.0784. eprint: https://doi.org/10.1287/ijoc.2017.0784. URL: https://doi.org/10.1287/ijoc.2017.0784.

[67] Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. «Robust Branch-Cut-and-Price Algorithms for Vehicle Routing Problems». In: vol. 43. Jan. 2008, pp. 297–325. ISBN: 978-0-387-77777-1. DOI: 10.1007/978-0-387-77778-8_14.

[68] Artur Pessoa et al. «A generic exact solver for vehicle routing and related problems». In: *Mathematical Programming* 183 (2020), pp. 483–523.

[69] Marcus Poggi and Eduardo Uchoa. «Integer Program Reformulation for Robust Branch-and-Cut-and-Price Algorithms». In: (Dec. 2003).

[70] Ted Ralphs et al. «On the Capacitated Vehicle Routing Problem». In: *Mathematical Programming* 94 (Jan. 2003), pp. 343–359. DOI: 10.1007/s10107-002-0323-0.

[71] David Ryan and Brian Foster. «An Integer Programming Approach to Scheduling». In: *Computer Scheduling of Public Transport* 1 (Jan. 1981), pp. 269–.

[72] Ruslan Sadykov and François Vanderbeck. *BaPCod - a generic branch-and-price code*. Technical Report. Inria Bordeaux Sud-Ouest, Nov. 2021. URL: https://inria.hal.science/hal-03340548.

[73] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Ed. by Paolo Toth and Daniele Vigo. Society for Industrial and Applied Mathematics, 2002. DOI: 10.1137/1.9780898718515. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718515. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.

[74] Daniel Villeneuve and Guy Desaulniers. «The shortest path problem with forbidden paths». In: *European Journal of Operational Research* 165.1 (Aug. 2005), pp. 97–107. URL: https://ideas.repec.org/a/eee/ejores/v165y2005i1p97-107.html.