UNIVERSITÀ DEGLI STUDI DI MILANO

Facoltà di Scienze Matematiche, Fisiche e Naturali

Dipartimento di Tecnologie dell'Informazione

Dottorato di Ricerca in Informatica, XXIV Ciclo

Settore scientifico/disciplinare INF-01

Location and Routing Problems: a Unified Approach

PhD Thesis of:

Emanuele Tresoldi

Tutor: Prof. Alberto Ceselli

Co-tutor: Prof. Giovanni Righini

Coordinator: Prof. Ernesto Damiani

A.A. 2010/2011

Abstract

This thesis is about location and routing problems. We propose a unified algorithmic approach, based on the branch-and-cut-and-price paradigm, for the exact solution of general location and routing problems involving both costs and profits. In particular three different types of NP-hard problems are taken into account: the first is an extension, arising in the context of waste collection management, of the well studied Vehicle Routing Problem. The second is based on the Multi-Depot Vehicle Routing Problem with profits and has applications in the exploration of planetary surfaces. The last problem is about the distribution of drugs in emergency situations. For every problem a detailed description and a mathematical formulation are given.

The largest part of the thesis is dedicated to the careful explanation of how our method can be efficiently implemented in every of the problems taken into account. In particular we propose new algorithmic ideas and several modifications and extensions to many procedures already presented in the literature. However, all components of our algorithms are fully presented and analyzed pointing out every methodological and practical issue.

Extensive computational experiments and comparisons are carried out to evaluate the performance of our approach and the tractability of the problems addressed.

Contents

Chapter 1

Introduction

1.1 Location and Routing Problems

Location and *routing* are two of the most common words in operations research. In fact, they represent, from the early 50's, two of the main and most successful fields of application of mathematical modeling and combinatorial optimization techniques.

Location problems have a long tradition, their formal origin and the starting point of what has been later called *the location analysis* dates back to 1909 when Alfred Weber published a book titled "Theory of the Location of Industries" [1]. Generally speaking, location problems usually concern the optimal placement of points within a limited space, in order to optimize a quantity that is a function of the relationship, e.g. distance, among points in the same area. A classical application of this concept is the facility location problem in which a set of potential facility sites, where a facility can be opened, and a set of demand points that must be serviced are given. The problem requires to find the optimal placement for the facilities to minimize the sum of the distance from each demand point to its closest facility (see Love et al. [2] or more recently Farahani and Hekmatfar [3] and Church and Murray [4]). Several additional requirements and constraints along with different, more complex, and multiple criteria objective functions (see Farahani et al. [5]) may be taken into account giving birth to a multitude of variations and extensions that arose during the last 50 years spanning countless contexts and fields of application from classical supply chain management (see Melo et al. [6]), to health-care (see Gough and McCarthy [7]), to telecommunication (see Yaman [8]) passing by the optimization of maritime hub locations (see Takano [9]). Other notable location problems are the p-median problem (see Reese [10]),

the p-center problem (see Drezner [11]) and the quadratic assignment problem (see Loiola et al. [12]), they, together with the facility location problem, represent the core problems in the location area and have been addressed using both heuristics (see Mladenović et al. [13] and Arya et al. [14]) and exact (see for instance the branch-and-price algorithms proposed by Klose and Görtz [15] or Ceselli and Righini [16]) methods available in operations research.

Routing problems are instead related to the design of paths, usually referred to as routes, in a network, often represented by a graph, in order to optimize the circulation among the network points, frequently called clients. These problems, as well as the location ones, have a long history, in fact they were already studied, as part of the developing graph theory, during the 19th century by mathematicians as Sir William Rowan Hamilton and Thomas Penyngton Kirkman (see the book by Biggs et al. [17]). However, the routing area grew at a much faster pace after the formalization, in the early 30's, of what become its most representative problem that is the Traveling Salesman Problem (TSP, see Schrijver [18] for the history of the TSP before 1960). Given a collection of cities and the cost of traveling between each pair of them, the traveling salesman problem requires to find the shortest possible tour that visits each city exactly once and returns to the starting point. From this simple definition originated one of the most famous, important and well studied problem in the whole operations research, we refer the interested reader to the book by Applegate et al. [19] for a detailed analysis of the TSP and to the book by Gutin and Punnen [20] for its variations. The routing problems reached a new level after the formalization, proposed by Dantzig and Ramser in 1959 (see [21]), of the Vehicle Routing Problem (VRP). This is a generalization of the TSP in which, on the same network, multiple actors are taken into account and a tour for each of them has to be designed. A book by Toth and Vigo [22] and more recently another book by Golden et al. [23] offer a detailed description of the problem including several variations and the most common and successful solving approaches. A peculiar and interesting extension of both the TSP and the VRP are the routing problem with profits (see Feillet [24] and Vansteenwegen [25]). In this case the problems do not require all clients in the network to be reached, instead profits are associated with every client and they are collected when clients are visited. The objective function is consequently switched from the minimization of the traveling costs to the maximization of the collected profits. All these routing problems and their extensions have many applications from the most obvious in the transportation field (see for instance Serna and Bonrostro [26], Cordeau [27]) and supply chain management (see Rajmohan and Shahabudeen [28]) to waste management (see

Santos et al. [29]) and health-care (see Savelsbergh et al. [30]). Solution approaches, as for the location problems, are the most different and goes from the heuristic ones, we refer to a paper by Pisinger and Ropke [31] for a general heuristic algorithm and to a survey by Bräysy and Gendreau [32] for metaheuristic approaches, to the exact ones (see Liong [33]) among which the column generation based algorithms seem to be the most effective (see for instance Baldacci et al. [34] and Ceselli et al. [35]).

Although location and routing represent two distinct and well defined fields in operations research, in real life the problems involve, most of the time, both aspects simultaneously. In fact, when designing, for instance, a distribution system, the selection of the optimal location for the warehouses influence the design of the routes traveled by the vehicles to deliver the goods, a the same time the optimization of the routes can influence the placement of the warehouses as it may be convenient to place them in particular locations due to the availability of especially cheap routes starting from that places. Problems like this, in which both location and routing aspects are involved, are usually referred to as Location and Routing Problems (LRP). Although the need for taking into account the routes when selecting the location for the facilities was already considered starting from the early 60's (see Maranzana [36]) and has been frequently remarked during the years (see for instance Salhi and Rand [37]), LRP are often addressed in a hierarchical way. We refer the interesting reader to the survey by Nagy and Salhi [38] for a discussion about the motivation behind this hierarchical approach. In particular, locations decisions are usually perceived as strategic ones, taken once for all and not modifiable while routing choices are considered as tactical decisions that are subject to changes and can be daily re-arranged. This traditional viewpoint had been suited to deal with very classical location-routing problem as the Round-Trip Location Problem (see Chan and Hearn [39] for the rectilinear distance version or Laporte and Norbert [40] for its general version) when no exact algorithmic approaches were available for the exact solution of more general LRP. However, the progress in the optimization science makes today possible to tackle such problems in their wholeness. Furthermore, the traditional hierarchical view does not fit in many problems where no clear priority exists between the location and the routing aspects. This may be the case when setting up one-time distribution/collection systems in which routes and depots share the same lifespan. In fact, in these scenarios clients are visited only once and depots become useless after all clients in their area are served. In addition, speaking in general terms, since a LRP involves two distinct components whose behavior is mutually dependent an

exact optimal solution of the complete problem cannot be obtained without considering both aspects at the same time, indeed if location decisions are taken independently from the routes the best solution of the combined problem may be dropped.

Moreover, as already pointed out in the survey by Nagy and Salhi [38], the vast majority of the LRP in literature are only concerned about the minimization of the costs, while real-life objective functions involve both costs and profits.

For all these reasons in this thesis we present exact methods that are able to deal with general LRP taking into account both location and routing with both costs and profits at the same time, allowing the possibility for the clients to be skipped. In the reminder of this Chapter we delve into the nature of the LRP, give a general description of our approach and point out the innovative aspects.

1.2 The relationship between Location and Routing

The optimization problems taken into account in this thesis combine routing and location features with both costs and profits in order to model complex logistic system. Different levels of interaction and combination between these two aspects may lead to various kinds of problems; in particular in this thesis we focus on three types of problems with increasing relative importance of the location decisions over the routing ones. In the remainder of this Section we give a brief description of these three typologies by means of very general mathematical models.

1.2.1 No Location Decisions

The first type of problem we addressed is characterized by the lack of location decisions. In this case the focus is totally on the routing part since the problem involves only a single depot located in a fixed position (see Figure 1.1). Typical examples are the VRP and several of its variations. The following model (1.1-1.4) describes a standard routing problem with profits in which, given a set of clients $\mathcal{N}$ and a fixed location for the depot, we aim at the maximization of the profits (p_r) collected along the K routes that can be used to reach the clients. Only two types of variables are used: s_i with $i \in \mathcal{N}$ and z_r with $r \in \Omega$, where Ω is the set of all possible routes in the problem. The former variables are used to model the possibility to avoid clients when they are not perceived as useful while the latter represent the routes. Thus, on the one hand each s_i assumes value 1 when the client i is skipped

and 0 if it is visited, on the other hand each z_r takes value 1 if the route $r \in \Omega$ is used in the solution and 0 otherwise. Two families of constraints are needed to describe the requirements of the problem: partitioning constraints (1.2), in which a_{ir} is equal to 1 if the client i is visited by the route r and 0 otherwise, imposing that no client is visited more than once and inequalities (1.3) putting an upper bound on the total number of routes that can be used.

This simple general formulation allows for the inclusion of many additional constraints modeling more complex situations arising in real problems and represents the starting point of our analysis.

$$\max \sum_{r\in\Omega} p_r z_r \tag{1.1}$$

$$\text{s.t. } s_i + \sum_{r\in\Omega} a_{ir} z_r = 1 \qquad \forall i \in \mathcal{N} \tag{1.2}$$

$$\sum_{r\in\Omega} z_r \leq K \tag{1.3}$$

$$s_i \in \{0,1\} \qquad \forall i \in \mathcal{N}$$

$$z_r \in \{0,1\} \qquad \forall r \in \Omega \tag{1.4}$$

In Chapter 2 we consider an optimization problem of this type arising in the context of waste collection management and called Vehicle Routing Problems with Different Service Constraints (VRPDSC). It is an extension of the VRP and takes into account several additional constraints. This problem, first presented by Macedo et al. [41], does not involve any location choice and is not a problem with profits since its objective function calls for the minimization of the total service costs, nonetheless, due to its peculiar requirements, it allows for the skipping of the clients.

1.2.2 Passive Location Decisions

The second type of problems we considered brings in the first real location components and indeed it involves a set of potential locations $\mathcal{L}$ among which facilities, that represent the starting points of the routes, can be placed (see Figure 1.2). The introduction of multiple candidate locations for the facilities triggers a modification in the model. In fact,

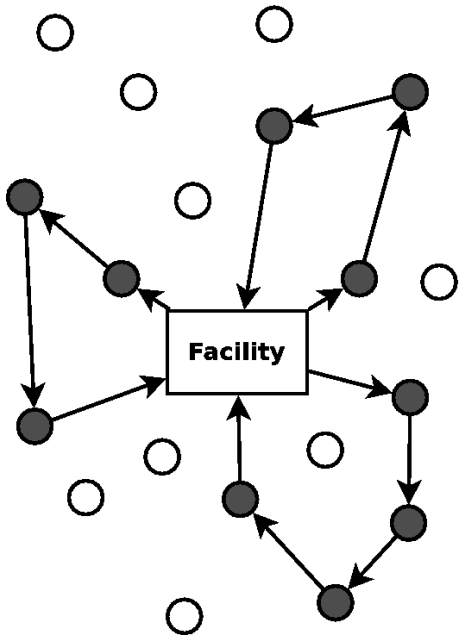

Figure 1.1: No locations

new variables are needed to model location choices, in particular we use variables y_l with $l \in \mathcal{L}$ that take value 1 if a facility is opened in location $l \in \mathcal{L}$ and zero if the location is unused. Since the design of the routes depends on their starting point the set Ω has to be split in subsets Ω_l, one for every potential facility location $l \in \mathcal{L}$, identifying sets of routes starting from the same location. This separation is also reflected in the constraints that limit the number of routes. Indeed, since different locations can be characterized by different limitations, an inequality (1.3) has to be considered for every used location as modeled by the following constraints:

$$\sum_{r \in \Omega_l} z_r \leq K_l y_l \qquad \forall l \in \mathcal{L} \tag{1.5}$$

where K_l represents an upper bound on the total number of routes departing from location l.

Moreover, additional constraints are needed in order to model situation in which the availability of resources needed to build the facilities is restricted. Indicating with M the total number of facilities that can be opened, this limitation can be formally stated with

the introduction of the following inequality:

$$\sum_{l \in \mathcal{L}} y_l \leq M \tag{1.6}$$

Taking into consideration all changes and additions required when multiple candidate locations for the facilities are considered, the model of this second type of problems is the following.

$$\begin{aligned}
\max \ & \sum_{l \in \mathcal{L}} \sum_{r \in \Omega_l} p_r z_r && && (1.7)\\
\text{s.t. } & s_i + \sum_{l \in \mathcal{L}} \sum_{r \in \Omega_l} a_{ir} z_r = 1 && \forall i \in \mathcal{N} \\
& \sum_{r \in \Omega_l} z_r \leq K_l y_l && \forall l \in \mathcal{L} \\
& \sum_{l \in \mathcal{L}} y_l \leq M \\
& s_i \in \{0,1\} && \forall i \in \mathcal{N} \\
& y_l \in \{0,1\} && \forall l \in \mathcal{L} \\
& z_r \in \{0,1\} && \forall l \in \mathcal{L}, \forall r \in \Omega_l
\end{aligned}$$

We have described the influence of the location decisions in this problem as *passive* because they do not give a direct contribution to the value of the objective function, indeed there is no y_l variable in (1.7) as they do not have any control over the nature of the distribution system.

An example of this type of problem is the multi-depot VRP with profits in which, left out possible fixed costs for opening the depots, the value of the objective function depends only on the routes employed. At the same time, the design of the routes may depend on the placement of the depots.

This type of problems represent the classical mix of location and routing features that can be found in several academic and practical problems.

In Chapter 3 we tackle a location and routing problem called Generalized Location Routing Problem with Profits (GLRPP) introduced by Ahn et al. [42] and applied to the

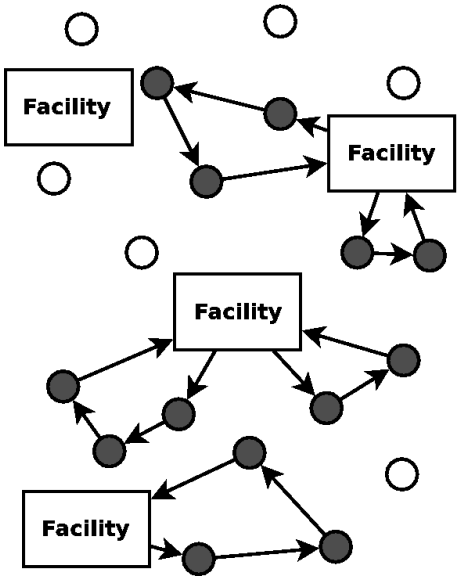

Figure 1.2: Passive locations

exploration of planetary surfaces. It fits into the description of the typical "Passive Locations Decisions" problem and in fact it represents an extension of the multi-depot VRP with profits. In particular, in this problem a new level of choice, due to the need of taking into account different strategies and limitations in building the routes, is considered.

1.2.3 Active Location

The last kind of problem taken into account (see Figure 1.3) brings in an additional degree of freedom. In this case location decisions have a more active impact on the structure of the problem and indeed they consist not only in choosing which places have to be opened but also, for every place used, in selecting which distribution strategy to employ. In fact, in this problem multiple distribution methods (two in the model (1.8-1.13)) are available and can be mixed together in order to maximize, as stated in the objective function (1.8), the total value of the collected profits. Every distribution strategy may be characterized by different limitations and employs a different method for reaching a subset of customers: routes, self-service points, direct delivery etc.

Each distribution method is modeled with a different variable. In particular, in the formulation (1.8-1.13), the variables z_r, with $r \in \Omega$, and y_w, with $w \in \Theta$, identify clusters

of clients served by the two different distribution strategies. The sets Ω and Θ, containing all the possible clusters of clients served with respectively the first and the second strategy, can be split in subsets Ω_l and Θ_l one for every location $l \in L$ to identify only clusters originating from location l.

Several modifications are needed in the model in order to handle the inclusion of multiple distribution strategies. Variables y_w with binary coefficients b_{iw}, taking value 1 if the client i belongs to cluster w and 0 otherwise, have to be added to the partitioning constraints (1.9) to model a situation where a client can be either reached with only a distribution method or is not served. Limitations on the total number of facilities that can be opened at the same time are now separated for every kind of distribution system as stated in inequalities (1.11) and (1.12) the former dealing with variables z_r while the latter consider only variables y_w . Finally, constraints (1.10) are used to link together the two delivery strategies and to prevent a location from being used by both distribution methods at the same time. In particular, these constraints impose that, on the one hand if delivery strategy associated with variables z_r is employed at location l then at most T clusters of clients can be served starting from that location, on the other hand if, in l, the strategy associated with variables y_w is used then only a single cluster of client can be served.

This formulation describes a Generalized Location and Distribution Problem (GLDP) which, to the best of our knowledge, is a new optimization problem. It pushes the complexity of the classical LRP one step forward allowing the design of more elaborated and realistic distribution networks.

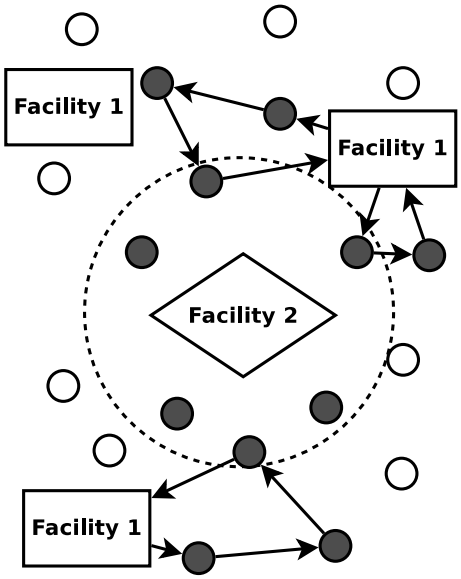

Figure 1.3: Active locations

$$\max \sum_{l\in\mathcal{L}}\sum_{r\in\Omega_l} p_r z_r + \sum_{l\in\mathcal{L}}\sum_{w\in\Theta_l} p_w y_w \tag{1.8}$$

$$\text{s.t. } s_i + \sum_{l\in\mathcal{L}}\sum_{r\in\Omega_l} a_{ir} z_r + \sum_{l\in\mathcal{L}}\sum_{w\in\Theta_l} b_{iw} y_w = 1 \qquad \forall i \in \mathcal{N} \tag{1.9}$$

$$\sum_{r\in\Omega_l} z_r + \sum_{w\in\Theta_l} T y_w \leq T \qquad \forall l \in \mathcal{L} \tag{1.10}$$

$$\sum_{l\in\mathcal{L}}\sum_{r\in\Omega_l} z_r \leq K \tag{1.11}$$

$$\sum_{l\in\mathcal{L}}\sum_{w\in\Theta_l} y_w \leq M \tag{1.12}$$

$$s_i \in \{0,1\} \qquad \forall i \in \mathcal{N}$$

$$y_w \in \{0,1\} \qquad \forall l \in \mathcal{L}, \forall w \in \Theta_l$$

$$z_p \in \{0,1\} \qquad \forall l \in \mathcal{L}, \forall p \in \Omega_l \tag{1.13}$$

In Chapter 4 we propose a problem, belonging to the "Active Location Decisions" type, arising in the context of drugs distribution in case of emergency. We took the basic idea behind this problem from a paper by Shen et al. [43]. However we elaborated the problem

adding several requirements and the possibility to employ a new distribution system that can be combined with the classical VRP-like delivery method to describe a more diverse logistic system.

1.3 Our Unified Approach

For the solution of the three types of problems previously identified we propose a unified approach based on the branch-and-cut-and-price paradigm (see Ladányi et al. [44]). The branch-and-cut-and-price is an algorithmic framework that represents the most advanced technique available today to solve Mixed Integer Programs (MIP) like the ones coming from the mathematical formulations for the three proposed families of problems. The idea behind branch-and-cut-and-price algorithms is to consider only a small subset of variables describing the problem and to iteratively enlarge this set introducing step by step only useful variables. This method is then embedded in a classical branch and bound scheme (see Balas and Toth [45]) in order to ensure that the optimal integer solution is found. Moreover, to improve the bounds additional valid inequalities are dynamically generated during the process (see Mitchell [46] and Balas et al. [47]). An overview of a general branch-and-cut-and-price algorithm is reported in Figure 1.4.

Such a complex algorithm is composed by several different components that in the case of our unified approach are the following:

Pricing algorithms: they are procedures used to dynamically introduce the variables into the problem. In our approach we implemented a multiple pricing technique consisting in using several different pricers at the same time. They may be either heuristic or exact algorithms, however since our approach aims at finding the optimal solution for the problem it always includes at least one exact pricing procedure.

Cuts generators: they represent algorithmic routines to find violated inequalities that can be added to the problem in order to accelerate the convergence of the algorithm and to possibly limit the size of the branching tree.

Primal heuristic algorithms: they are sub-optimal methods employed in order to early find good feasible integer solutions during the execution of the branch-and-cut-and-price algorithm.

Stabilization method: it is an algorithmic procedure employed in order to mitigate the possible convergence issues of the branch-and-cut-and-price algorithm.

Branching Strategies: they represent the policies followed to build and visit the search tree used in order to ensure the integrality of the optimal solution.

To demonstrate the effectiveness of our approach we have selected three real problems, one for each type of problem identified in the previous section, that are solved within our framework in an uniform way.

1.4 Original Contribution

In this thesis we present a general framework for solving problems involving location and routing aspects and considering costs and profits through the possibility to leave customers uncovered. Our contributions can be divided into modeling, methodological and practical ones.

From the modeling point of view our contribution lies in the analysis of the relationship between the two aspects of the location and routing problems. We started from problems that do not involve location features and we gradually introduced location components showing, by means of mathematical formulations, how they impact on the structure of the problems. Three main types of problems have been identified, each one represents a general scheme from which many different problems can be derived adding new requirements and limitations.

Moving to the methodological contributions we have defined a unified approach to address LRP with costs and profits. Our approach is based on the branch-and-cut-and-price framework, it is able to take both location and routing aspects into account at the same time and presents an original way, especially suited for being used in conjunction with algorithms based on column generation, to deal with the possibility to avoid clients. The approach we present is made up of several algorithmic components some of them are new versions of the ones already presented in the literature that we modified, extended and improved to efficiently work in new contexts. Other components represent entirely new algorithms and define new procedures that can be adapted to other, completely different, problems. In particular, on the one hand the greedy and the dynamic programming pricing procedures

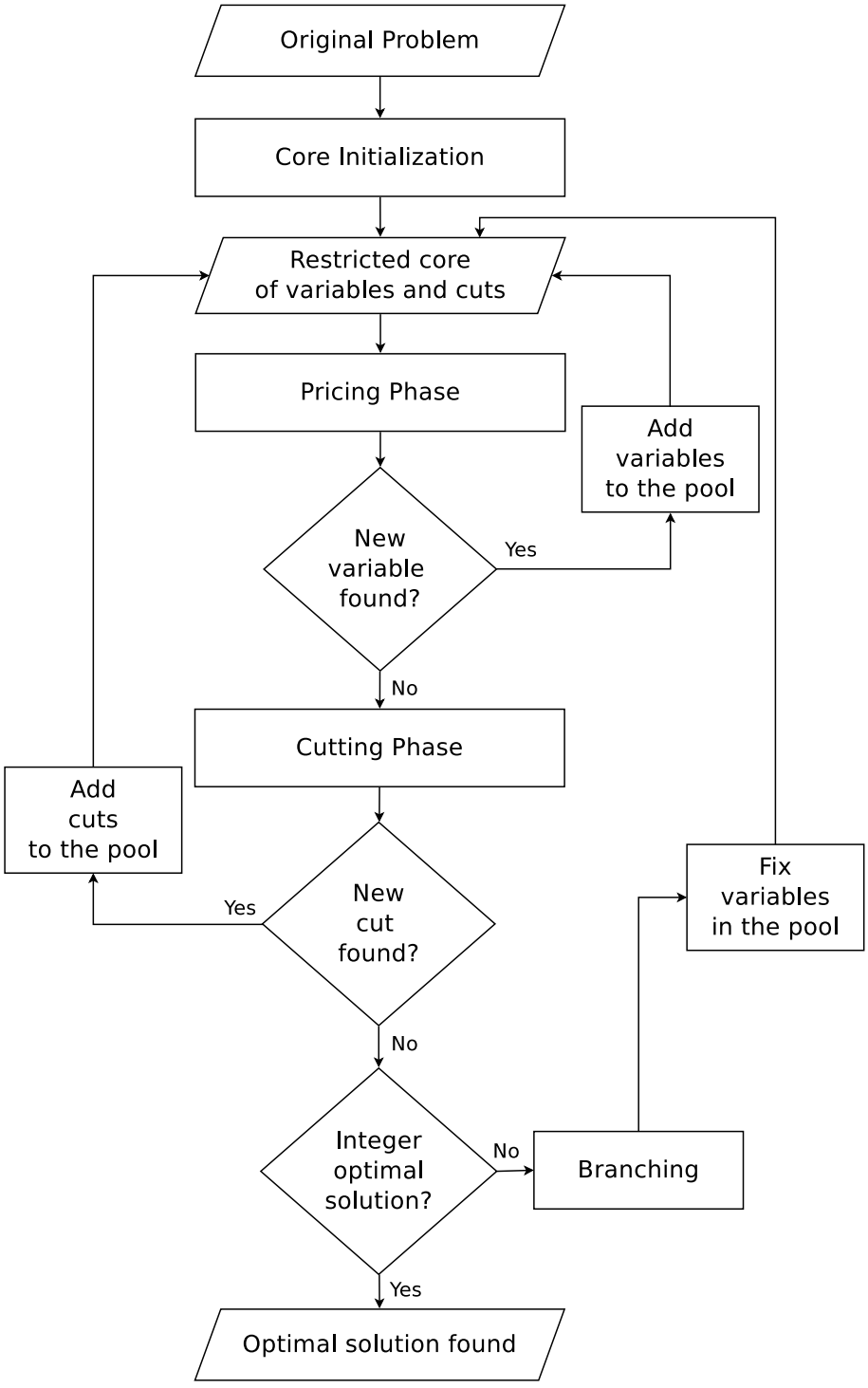

Figure 1.4: The outline of a branch-and-cut-and-price algorithm

required very extensive and non-trivial modifications to properly handle all additional constraints and peculiar requirements of every problem. The tabu search pricer on the other hand is characterized by a completely new initialization phase which is complemented with a primal heuristic procedure looking for feasible solutions during the pricing stage. Cut generation procedures are mainly new since we borrowed valid inequalities presented in literature for classical routing problems and we translated in the context of LRP with profits. Subset-row inequalities and 2-path cuts were adapted in order to work when multiple depots and multiple distribution strategies are available; in particular, separation procedures have to be totally redesigned. In addition, we presented the consistency cuts: they are new inequalities, whose structure is general enough to fit into the description of many LRP, that have never been used in conjunction with column generation algorithms and that required the design of a custom data-structure to keep a high level of efficiency in the computation. The introduction of the consistency cuts triggers a modification in the very nature of the branch-and-cut-and-price paradigm mixing together the pricing and the cutting phase and leading to the definition of simultaneous column and row generation algorithms.

The idea behind the stabilization method employed in our approach comes from the literature, however this thesis represents the first attempt to use this stabilization strategy in complex problems and provides extensive details of its implementations.

Going into the analysis of our contribution problem by problem a few things are worth mentioning: for the Vehicle Routing Problems with Different Service Constraints we provide the first algorithmic approach able to take properly into account all the requirements of the problem. In fact, despite the problem was already presented in literature the algorithms proposed for its exact solution do not consider all its constraints. About the Generalized Location Routing Problem with Profits, our approach is the first exact procedure for this problem and introduces the use of a new type of cuts. Finally the Generalized Location and Distribution Problem is a completely new location and routing, or more properly location and distribution, problem that generalized the classical concepts behind the LRP and mixes different delivery strategies on the same level. In this case our approach represents the first and only exact algorithm for this new problem.

From a more practical point of view this thesis demonstrates the possibility to handle general LRP using a unified framework. Such problems describe more realistic scenarios compared to the ones in which only a single aspect is considered and consequently methods able to solve them properly are more suited to be implemented in real life application.

In particular in the case of the GLRPP arising in the context of planetary surface exploration we took into account a lot of additional constraints and we had to deal with very large instances that may be well representative of the real underlying problem. However an extensive campaign of tests has been performed for all problems and, if possible, our results were compared with the state of the art showing that our general approach remains competitive even against algorithms especially designed for a single specific problem.

1.5 Outline

In the following three chapters of this thesis we carefully present the three problems taken into account, Chapter 2 is dedicated to the VRPDSC, the GLRPP is addressed in Chapter 3 while in Chapter 4 we presented the GLDP and its application in the context of drug distribution in case of emergency. In details, in each chapter we give a description and a complete mathematical formulation for every problem then we explain how our branch-and-cut-and-price approach can be applied to the specific problem, what components are used and which modifications they may require pointing out possible implementation issues. Then we report the results obtained during the experimental campaigns and our conclusions. Finally in Chapter 5 we draw a few final remarks and general conclusions.

Chapter 2

Waste Collection

2.1 Introduction

The world population has increased in the last 10 years by almost one billion people growing at steady pace of about 1.1% every year. In particular the total number of living humans on Earth went form six billions in 1999 [48] to almost seven billions at the end of 2011 [49]. The last century has seen an extremely fast increase in the population (see Figure 2.1) mainly due to the advancements achieved in medicine and agricultural productivity. If the world is able to sustain this growing rate the world population is going to reach about nine billions by the end of 2040 [50].

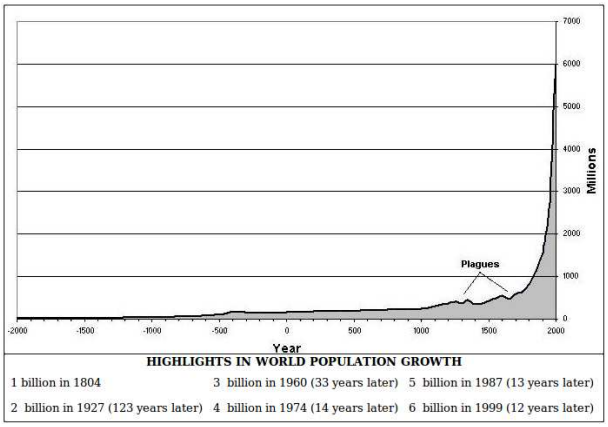

Figure 2.1: World Population Growth

Together with the unprecedented population growth after World War II the world assisted to the migration of people from rural areas to the cities and nowadays it is estimated than almost half of the total world population lives in urban areas as, for instance, in Japan where the metropolitan area of Tokyo counts about 35 millions people and represents more than one third of the entire country population.

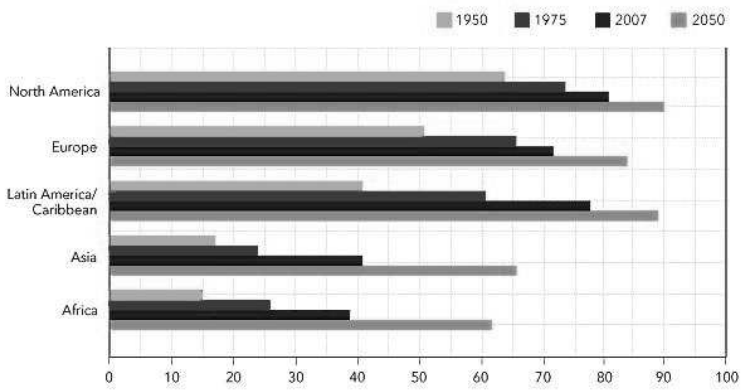

Figure 2.2: Percentage of Population Living in Urban Areas

These two factors make clear the reason why one of the wold main concern in the last couple of decades is the wast management. Urban areas are the zones where the largest amount of waste is generated and require an efficient system for the garbage collection and disposal. Maintaining a good waste management system is a difficult and very expensive task (e.g. the solid waste management services budget for the city of Toronto in 2011 totals 342.631 million dollars [51]) that, especially today, goes badly along with the lower and lower budgets available in highly interconnected, rapidly and irregularly growing urban areas that characterized many of the metropolis in the world. In fact, this problem is particularly significant in the western European countries where, despite a lot of successful education and awareness campaigns, the annual production of solid waste exceeds 550kg per capita [52]. In this case a careful and effective waste management is essential in order to ensure the necessary service level that may guarantee the population health, that does not compromise the roads condition and that allows for the reuse and recycle of as much garbage as possible. Unfortunately, the need of an efficient waste management system conflicts with the limited resources available to the local administrations that are usually charged with the garbage removal and disposal. However, when the service quality, due to

a multiplicity of motives, drops beyond the minimum satisfactory level we may assist to regrettable situations as what as been called "Naples waste management crisis" that has afflicted, and partially is still plaguing, one of the most beautiful Italian cities.

In this chapter we focus on the optimization of the routes performed by the vehicles used for the collection of the garbage in order to both minimize the costs and guarantee an acceptable level of service.

In Section 2.2 we give a general description of the problem, that is mathematically formalized, starting from the model introduced by Macedo et al. [41], in Section 2.3. Our algorithmic approach is presented in Section 2.4. In Section 2.5 we report the computational results achieved while our conclusions are finally drawn in Section 2.6.

2.2 Problem Description

The problem analyzed in this chapter can be described as follows. There is a company that has been charged with collecting the waste in a metropolitan area. The company has at his disposal a depot and a fleet of identical vehicles. Every day the vehicles go out to collect the contents of the garbage bins spread in the area managed by the company and at the end of the day they come back, full of waste, to the depot. The public administration, knowing the average garbage production of that urban area, in order to ensure a constant level of service, negotiate with the company a minimum amount of garbage to be collected every day. Moreover, to avoid the concentration of waste in the same place, it sets a minimum filling level beyond which a bin must be imperatively emptied. At the same time, the company tries to optimize the cost of the service minimizing the distance traveled by every vehicle and, in order to manage at best its fleet of vehicles, it imposes that a vehicle cannot go back to the depot before having collected a minimum quantity of waste.

Therefore, given the described scenario, we would like to design daily plans for the waste collection such that the following conditions hold:

I The number of vehicles used does not exceed a given limit.

II Every vehicle can only perform a single route collecting waste starting from the depot and cannot go back to the depot before having collected at least a predefined quantity of garbage.

III The contents of a garbage bin is collected in a single visit by a single vehicle.

IV The total amount of waste collected in a day must not be lower than a fixed quantity.

V All garbage bins whose filling level exceeds the value set by the public administration have to be compulsory emptied.

VI The other bins can be collected if they are needed to fulfill conditions II and IV.

Among all the feasible distribution plans, one that minimizes the total traveling cost is perceived as an optimal one.

This problem represents a variation of the classical VRP (see [23]) and indeed has been first described as VRP with Different Service Constraints (VRPDSC) by Macedo et al. [41]. It belongs to the first type of problem we have identified in the introduction (see Chapter 1) in fact it does not involve any location choice since only a single depot is available.

2.3 Mathematical Formulation

Borrowing the notations from Macedo et al. [41], in this section we present an extended formulation for the VRPDSC that makes use of an exponential number of variables representing routes performed by vehicles starting and ending at the depot and collecting all the garbage bins met along the way.

Formally, a complete graph $G = (\mathcal{V}, \mathcal{A})$ symbolizing the metropolitan area is given. Together with a special vertex o representing the starting depot a set of vertexes $\mathcal{V}$ indicating locations where the garbage bins are placed is given; with each of them a value l_i, indicating the amount of waste that is contained in it, is associated. A bin is forced to be emptied only if $l_i \geq W_{max}$ where W_{max} is a predefined value. This allows for the splitting of the vertex set in two subsets $\mathcal{N}_1 = \{i \in \mathcal{V} : l_i \geq W_{max}\}$ and $\mathcal{N}_2 = \{i \in \mathcal{V} : l_i < W_{max}\}$ being respectively the set of the mandatory and the optional bins. The set $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}\}$ is the arc set representing the distance to be traveled to go from node $i \in \mathcal{V}$ to node $j \in \mathcal{V}$. With every arc $(i, j) \in \mathcal{A}$ a cost c_{ij}, function of the traveling time to go from i to j, is associated. Given a fleet of K identical vehicles with capacity W and a value L_{min} representing the minimum amount of waste that is requested to be collected by each vehicle before coming

back to the depot, the VRPDSC reads as follows:

$$\min \quad \sum_{p\in\Omega} c_p z_p \tag{2.1}$$

$$\text{s.t.} \quad \sum_{p\in\Omega} a_{ip} z_p = 1 \qquad \forall i \in \mathcal{N}_1 \tag{2.2}$$

$$s_i + \sum_{p\in\Omega} a_{ip} z_p = 1 \qquad \forall i \in \mathcal{N}_2 \tag{2.3}$$

$$\sum_{p\in\Omega} z_p \leq K \tag{2.4}$$

$$\sum_{p\in\Omega} l_p z_p \geq L^T_{min} \tag{2.5}$$

$$z_p \in \{0,1\} \qquad \forall p \in \Omega \tag{2.6}$$

$$s_i \in \{0,1\} \qquad \forall i \in \mathcal{V} \tag{2.7}$$

In this formulation Ω is the set of all feasible routes that can be done using one vehicle of the fleet. A route $p \in \Omega$ is considered feasible if it starts and ends at the depot and the total amount of waste collected along the route l_p does not exceed the capacity W but, at the same time, is greater than L_{min}.

The model makes use of two types of binary variables s and z. The former are called skip variables and are used to model the possibility for a bin location to be skipped. Indeed there is one s_i for every optional bin $i \in \mathcal{N}_2$ that takes value one if that bin is not collected and zero otherwise. The latter z_p represent the routes, in particular a z_p is equal to one if the route $p \in \Omega$ belongs to the solution. Moreover, three coefficients are employed in this model: a_{ip} that assumes value 1 if the bin $i \in \mathcal{V}$ is picked up by the vehicle performing the route $p \in \Omega$, l_p indicating the total amount of waste collected during route $p \in \Omega$ that is $l_p = \sum_{i\in\mathcal{R}(p)} l_i$ where $\mathcal{R}(p)$ is the set of all the bins picked up during route $p \in \Omega$. The last coefficient is c_p that represents the traveling cost paid for visiting all nodes belonging to the route $p \in \Omega$, more formally c_p is equal to the value of the TSP solution on the graph derived from node $\mathcal{R}(p) \cup o$.

The objective function (2.1) of the problem involves only z_p variables and aims at the minimization of the total traveling costs. The first two constraints (2.2) and (2.3) represents the covering requirements respectively over mandatory and optional bins, in the former no skip variable is employed and in fact none of the bins $i \in \mathcal{N}_1$ is avoidable. These two

constraints formalize the conditions V and VI. Constraints (2.4) model the conditions I saying that the number of vehicles available is equal to K. Finally the last constraints (2.5) impose that the total amount of waste collected by all vehicles used must be greater than a predefined quantity L^T_{min} as stated in condition IV. In Figure 2.3 a visualization of an instance and a solution for this problem is reported.

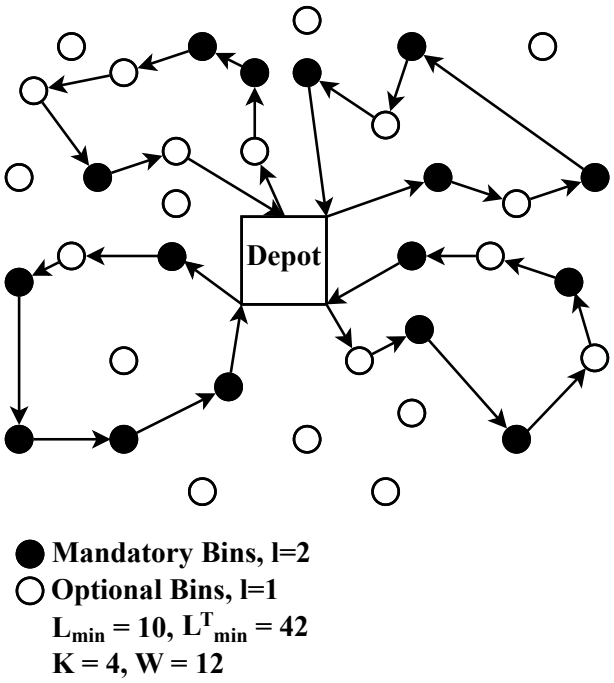

Figure 2.3: Waste Collection Problem

Additional Inequality. As shown by Macedo et al. [41] dual feasible functions (see Carvalho et al. [53]) can be exploited to derive valid additional constraints for the problem that can be useful for improving the dual bound and accelerating the convergence. In particular for the VRPDSC the following inequality is valid:

$$\sum_{p \in \Omega} f_p z_p \leq (K \cdot W) - L^T_{min} \tag{2.8}$$

where f_p represents the free space in a vehicle going through route $p \in \Omega$.

2.4 Branch-and-cut-and-price

The algorithm we propose for exactly solving the VRPDSC is based on the branch-and-price-and-cut paradigm (see Desrosiers and Lübbecke [54]). This methodology starts from the linear relaxation of the problem, usually called Master Problem (MP), that is obtained by substituting constraints (2.6) and (2.7) with

$$0 \leq z_p \leq 1 \qquad \forall p \in \Omega$$
$$0 \leq s_i \leq 1 \qquad \forall i \in \mathcal{V}.$$

The MP is solved using a technique called Column Generation (CG, see Desaulniers et al. [55]). Moreover to improve the bounds when the optimal value of the MP is achieved we look for valid inequalities that can cut out the optimal LP solution and improve the lower bound. Finally, when neither useful columns nor rows can be added, if the optimal solution of the MP is fractional branching is performed in order to recover the integrality and the derived search tree is explored recomputing the lower bound at every node. In this section we present the main components of our branch-and-cut-and-price algorithm namely: column generation, pricing, cut generation, branching and stabilization.

2.4.1 Column Generation

The column generation technique proved to be extremely useful in solving huge Linear Programming (LP) problems (see Barnhart et al. [56]) that is the case for the VRPDSC which involves an exponential number of variables. The idea behind this method is that when a LP problem is too large, only a subset of variables can be considered in order to optimally solve it and in particular only those assuming a value different from zero in the optimal solution. This result can be achieved starting with a Restricted Master Problem (RMP) initialized with a suitable subset of variables and adding, in subsequent iterations, useful columns. When no more variables can be added, the value of the RMP is optimal also for the MP and the process is stopped. The usefulness of a variable not already in the RMP is estimated considering its reduced cost, in particular in a minimization problem the more negative the reduced cost of a variable is the more profitable is its insertion in the MP. The process of finding the most convenient variable to be added to the RMP is called pricing and consists in exploiting the dual representation of the problem in order to find

the variable with the minimum reduced cost without the need of explicitly considering all of them.

In particular in our problem the RMP is initialized with:

(a) One dummy column representing a feasible solution with an infinite cost. This column in inserted with the only purpose to ensure that a feasible solution exists when the RMP is solved for the first time allowing the column generation process to start.

(b) All $|\mathcal{N}_2|$ skip columns one for every skip variable $s_i \in \mathcal{N}_2$.

(c) All one-node routes, corresponding to path starting from the depot, picking up only one bin $i \in V$ having $L_{min} \leq l_i \leq W$ and immediately coming back to the depot.

(d) A set of routes obtained in a single iteration by the greedy pricing algorithm described in Section 2.4.5.

The pricing finds the most useful variables evaluating their reduced costs. This is done by means of the dual variables associated with the dual representation of the MP whose formulation reads as follows:

$$\begin{aligned}
\max \ & \sum_{i \in \mathcal{V}} \lambda_i - K\mu + L_{min}^T \sigma + (L_{min}^T - KW)\rho & & \\
\text{st } & \lambda_i s_i \leq 0 & \forall i \in \mathcal{N}_2 & \\
& \sum_{i \in \mathcal{V}} a_{ip}\lambda_i - \mu + l_p\sigma - f_p\rho \leq c_p & \forall p \in \Omega & (2.9) \\
& \lambda_i \quad \text{unrestricted} & \forall i \in \mathcal{N} & \\
& \mu, \sigma, \rho \quad \geq 0 & &
\end{aligned}$$

where λ, σ, μ and ρ represent vectors of non-negative dual variables related respectively to constraints (2.2, 2.3), (2.5) and to constraints (2.4), (2.8) rewritten as $\geq$ inequalities. From this formulation is clearly visible that looking for the z_p variables with minimum reduced costs in the MP is equivalent to seeking the most violated constraints (2.9) in the dual problem.

The expression of the reduced costs ξ_p associated with the z_p variables reads as follows:

$$\xi_p = c_p - \sum_{i \in \mathcal{V}} a_{ip}\lambda_i + \mu - l_p\sigma + f_p\rho \quad \forall p \in \Omega. \tag{2.10}$$

The problem to be solved in order to identify the variables with the minimum reduced cost (pricing problem) consists in finding a feasible route satisfying conditions II and III whose associated reduced cost is as low as possible. Moreover, since it is never convenient to visit a bin location more than once, we improve the bound imposing that the route must be elementary. More formally we have to solve the following problem:

$$\begin{aligned} \min_{p\in\Omega} \xi_p &= c_p - \sum_{i\in\mathcal{R}(p)} \lambda_i + \mu - \sum_{i\in\mathcal{R}(p)} l_i\sigma + f_p\rho \\ \text{st } & L_{min} \leq l_p \leq W \\ & f_p = W - l_p. \end{aligned}$$

Where the set $\mathcal{R}(p)$ has to be determined. This pricing problem turns out to be the well studied NP-hard Resource Constraint Elementary Shortest Path Problem (RCESPP, see Dror [57]) that we solve using two heuristic and one exact algorithms.

2.4.2 Pricing Algorithms

In our branch-and-price-and-cut procedure we follow the approach detailed by Salani et al. [58], it makes use of three different algorithms to price out the columns and find the ones with the minimum reduced cost: a greedy one, a heuristic dynamic programming one and an exact dynamic programming one. Starting from the greedy algorithm, they are called in order only if the previous algorithm is not able to find any column with a negative reduced cost. For the purpose of better illustrating these algorithms, we give their detailed description following the reverse order with respect to their execution.

2.4.3 Exact Dynamic Programming Algorithm

Let us first consider the case in which no requirements are imposed on the minimum amount of garbage collected along a single route and let s and t be two distinct copies of the depot representing, respectively, the starting and the ending point of the route. In order to obtain the exact solution to the RCESPP we use a dynamic programming algorithm which consists of a bi-directional extension of node labels. It associates labels, which encode partial paths, with every node $i \in \mathcal{V}$ of the graph G. Every label is iteratively considered and the corresponding path is extended to sites not already visited.

Label Structure. Each label is defined by the tuple $(\mathcal{S}, q, C, i)$, that encodes a path of cost C starting from the depot, visiting all sites in $\mathcal{S}$ and arriving to the node i picking up a total amount of garbage equal to q. In this problem the optimal solution is represented by the minimum cost path going from s to t collecting at most a quantity $q \leq W$ of waste.

Extension. The algorithm starts creating a label $(\emptyset, 0, W \cdot \rho, s)$, the cost of the initial empty label is not zero due to the price we have to pay, according to the constraint (2.5), for the empty space in the vehicle. Then, at every iteration of the algorithm, a label $l' = (\mathcal{S}', q', C', i')$ is iteratively selected and extended from site i' to each site $i'' \in (\mathcal{V} \cup \{t\}) \backslash \mathcal{S}'$ generating another label $l'' = (\mathcal{S}'', q'', C'', i'')$. The following rules are used to carry out the labels extension:

$$\mathcal{S}'' = \mathcal{S}' \cup \{i''\} \tag{2.11}$$

$$q'' = q' + l_{i''} \tag{2.12}$$

$$C'' = C' + c_{i'i''} - \frac{\lambda_{i'}}{2} - \frac{\lambda_{i''}}{2} - l_{i''}\sigma - l_{i''}\rho \tag{2.13}$$

with $\lambda_s = \lambda_t = 0$ and $l_s = l_t = 0$. At every iteration the cost of the path is updated taking into account the distance between the two nodes and their associated dual variables, moreover the same quantity $l_{i''}$ is subtracted twice, with two different coefficient, from the reduced cost. The first time, with coefficient σ, representing a bonus for using the vehicle capacity, the second time with coefficient ρ, in order to reduce the penalty payed for the empty space left in the vehicle. The new label l'' represents a feasible state if $q'' \leq W$, in other words a label can be extended from i to i'' if there is enough free space on the vehicle to contain all the garbage in the bin located in i''. It is worth noting that, as requested by the conditions detailed in Section 2.2, this extension rule does not allow for split delivery and enforce the elementary of the path as no sites already visited are taken into account for the extension.

Dominance Test. The extension can, in principle, generate an exponential number of labels. Thus, in order to reduce the total amount of labels generated, a dominance test is performed to fathom labels that cannot lead to an optimal solution. Let $l' = (\mathcal{S}', q', C', i)$ and $l'' = (\mathcal{S}'', q'', C'', i)$ be two generic labels associated with the same pickup site i. Then

the former dominates the latter if the following conditions hold:

$$\mathcal{S}' \subseteq \mathcal{S}'' \tag{2.14}$$

$$q' \leq q'' \tag{2.15}$$

$$C' \leq C'' \tag{2.16}$$

None of the inequalities (2.14), (2.15) and (2.16) can be left out, in fact they represent a set of necessary dominance conditions and dropping any of them can lead to the discarding of an optimal label. Furthermore, as shown in Feillet et al. [59], it is sometimes possible to identify a node $u \in \mathcal{V}$ that cannot be reached by any feasible extension of a given label, because of resource limitations. In this case it is useful to insert u in the set $\mathcal{S}$ of that label: it is easy to check that enlarging set $\mathcal{S}''$ helps satisfying condition (2.14); at the same time, if a node cannot be reached by extending label l' due to resource limitations, it cannot be reached by extending label l'' either since resource consumption in l'' is not lower. Therefore, enlarging each set $\mathcal{S}$ allows the dynamic programming algorithm to fathom a larger number of labels consequently reducing the computation time.

Decremental state space relaxation. A technique we use to speed up the solution process in the dynamic programming algorithm is the decremental state space relaxation as described by Salani and Righini [60]. The idea behind such a relaxation is simple: the state space of the problem is reduced projecting it to a smaller one by discarding the elementary constraints, and then iteratively reintroducing them until a feasible solution for the RCESPP is found. More formally, given a set of bins locations $\tilde{\mathcal{V}} \subseteq \mathcal{V}$ called critical set, the extension rule (2.11) can be replaced with

$$\mathcal{S}'' = \mathcal{S}' \cup \{i''\} \cap \tilde{\mathcal{V}} \tag{2.17}$$

This relaxed problem can be solved more efficiently, since more labels can be compared in the dominance test. In order to identify a good critical node set, $\tilde{\mathcal{V}}$ is initialized to $\emptyset$ then the state space relaxation of the pricing problem is solved and all the nodes visited more than once in the optimal path are added to the set $\tilde{\mathcal{V}}$. In our algorithm, this procedure is iterated until either an elementary path with a negative reduced cost is found or the optimal value of the pricing subproblem is nonnegative.

Bidirectional dynamic programming. Another method that was already used by Dijkstra (see for instance Goldberg et al. [61]) to speed up the computation of shortest path is the bidirectional extension of labels. In this algorithm, as described by Righini and Salani [58], two kind of labels are associated with every delivery site: forward labels and backward labels. They, respectively, represent paths going from s to its successors and from t to its predecessors. The forward set is initialized with label $(\emptyset, 0, W \cdot \rho, s)$ while the backward set with label $(\emptyset, 0, W \cdot \rho, t)$. Extension and dominance are then composed by two steps, in which forward and backward labels are treated independently. The updating rules and feasibility tests for backward extension are symmetrical to those for the forward labels. Since the consumption of capacity, is monotone along the path a bounding technique can be employed together with the bidirectional dynamic programming in order to reduce the generation of backwards and forwards labels encoding the same paths. In particular, the extension of forward and backward labels can be limited and useless duplication of paths can be reduced, imposing that a state l, either forward or backward, can be extended only if its associated consumption of capacity q is $\leq \frac{W}{2}$. The use of this bounding technique requires to change the label initialization and in particular the initial forward label will be $(\emptyset, 0, \frac{W}{2} \cdot \rho, s)$ while the backward $(\emptyset, 0, \frac{W}{2} \cdot \rho, t)$.

Join. A complete route going from s to t can then be obtained joining together a forward and a backward partial path. Each forward path $(\mathcal{S}^{fw}, q^{fw}, C^{fw}, i^{fw})$ can be joined with a backward path $(\mathcal{S}^{bw}, q^{bw}, C^{bw}, i^{bw})$, where $i^{fw} \neq i^{bw}$, if the complete path is elementary and the total amount of garbage picked up does not exceed the capacity of the vehicle. More formally:

$$\begin{aligned} &\mathcal{S}^{fw} \cap \mathcal{S}^{bw} = \varnothing \\ &q^{fw} + q^{bw} \leq W \end{aligned} \tag{2.18}$$

If both conditions are satisfied the path from s to t is feasible and its associated values $\mathcal{S}, q$ and C can be computed as follows:

$$\mathcal{S} = \mathcal{S}^{fw} \cup \mathcal{S}^{bw}$$
$$q = q^{fw} + q^{bw}$$
$$C = C^{fw} + c_{i^{fw}i^{bw}} - \frac{\lambda^{i^{fw}}}{2} - \frac{\lambda^{i^{bw}}}{2} + C^{bw}$$

This label $l^* = (\mathcal{S}, q, C, t)$ represents a feasible route that corresponds to a variable in the MP having a reduced cost $\xi_{l^*} = C + \mu$.

The join operation is run when no more feasible extensions can be performed; the minimum cost path after join is the optimal solution of the pricing problem.

Handling the minimum filling constraints. The algorithm described in the previous paragraphs does not take into account condition II, which imposes that every vehicle must pick up at least an amount of garbage equal to L_{min} before going back to the depot. In order to satisfy the minimum filling constraints the dominance and joining rules have to be modified. In particular on the one hand the dominance condition (2.15) is substituted by a more restrictive one $q' = q''$ limiting the possibility for domination only among labels using exactly the same capacity. On the other hand the join condition on the resource consumption (2.18) is replaced by $L_{min} \leq q^{fw} + q^{bw} \leq W$ ensuring that no route violating the minimum filling constraints are generated. The new dominance rules are weaker: only a few labels can dominate one another slowing down the solution process. However, the objective function of the MP and the structure of the reduced costs associated with the generated variables naturally leads to routes that, in general, use as much capacity as possible. Therefore we devise the following procedure. At first, we generate routes, using the dynamic programming algorithm described in the previous paragraphs, discarding the minimum filling constraints. If the resource consumption in the optimal route is already $\geq L_{min}$ no further computation is needed. Otherwise we introduce the new dominance and joining rules described in this paragraph and the algorithm is re-executed to either find a new feasible and useful route or to prove that no variable with negative reduced cost exists. It is worth noting that by removing the minimum filling constraints we are relaxing the pricing problem. Therefore, if no variable with negative reduced cost is found in this way the column generation process can be stopped.

2.4.4 Heuristic Dynamic Programming Algorithm

A heuristic dynamic programming algorithm can be easily obtained relaxing the dominance rules by removing the condition (2.14). This modified dominance test allows for the fathoming of much more labels, consequently reducing the computational time required to find a solution for the RCESPP. At the same time, since all conditions (2.14), (2.15) and (2.16) are necessary, with the relaxed dominance test an optimal partial path could be discarded leading to sub-optimal solutions. In the heuristic pricing we do not take into account explicitly the minimum filling constraints, however variables generated with this algorithm are added to the MP only if they encode routes whose capacity consumption is at least L_{min}.

2.4.5 Greedy Algorithm

In this pricing algorithm a route going from s to t is generated by repeated extensions of a single label. In details, a label $l = (\mathcal{S}, q, C, i)$, whose cost and resource consumption are initialized to 0 and $W\rho$ respectively, is considered and iteratively extended to a single node with a nearest neighbor policy. In order to find the neighborhood of a node and to select the best extension the set of reachable nodes $\mathcal{R}$ is computed. A bin location $j \in \mathcal{V}$ belongs to $\mathcal{R}$ if all the following conditions hold:

$$j \notin \mathcal{S} \tag{2.19}$$

$$q + l_j \leq W \tag{2.20}$$

where $l_s = l_t = 0$. If $\mathcal{R} = \emptyset$ the path is closed going back to the depot and the search is stopped. If the value q associated with this path is greater or equal to L_{min} a corresponding variable is generated and added to the MP, otherwise the path is discarded and no variable is generated. If instead $\mathcal{R} \neq \emptyset$, among the sites in $\mathcal{R}$, we select the one that minimizes the path cost, that is

$$\bar{j} = argmin_{j \in \mathcal{R}}\{c_{ij} - \frac{\lambda_i}{2} - \frac{\lambda_j}{2} - l_j\sigma - l_j\rho\}$$

with $\lambda_q = \lambda_v = 0$. The label is extended to node $\bar{j}$ using the same update rules described for the exact pricing algorithm, and we iterate.

In order to find, in one iteration of the column generation algorithm, several variables to add to the RMP the greedy pricing procedure is repeated many times until no other feasible

path with negative reduced cost is found. In particular, every time the greedy algorithm terminates finding a new useful variables it is executed again on a restricted graph that does not includes all nodes visited in all the feasible paths already generated, with this algorithm, in the same iteration of the column generation process.

This same algorithm is used also to produce a portion of the starting pool of variables used as initialization for the column generation algorithm. The algorithm runs as described but with a different objective function: in this case we aim at the maximization of the amount of waste collected and so among all bin locations in $\mathcal{R}$ we select the one with the greatest l_i.

2.4.6 Additional Inequalities

The VRPDSC extends the VRP, for which several additional inequalities have been presented in the literature. They can be employed to improve the value of the lower bound and to consequently speeding up the convergence of the algorithm (see for instance Lysgaard et al. [62] and Battarra et al. [63]). In particular we have taken into account and adapted two families of inequalities: a special case of the subset-row inequalities introduced by Jepsen et al. [64] and the 2-path inequalities that were originally introduced by Kohl et al. [65].

Subset-row inequalities.

Subset-row inequalities are a special case of Chvatal-Gomory rank-1 cuts, suited for use in column generation based algorithms (see for instance Petersen et al. [66]). In particular, we implemented a specific case of subset-row inequalities in which only subsets made up of three rows are considered. The basic idea behind these cuts is the following: for every set of three nodes there must be at most a single route visiting at least two of them. More formally, let $\mathcal{G} = \{G \subseteq \mathcal{V} : |G| = 3\}$ be the set made of all possible clusters of three nodes. For $G \in \mathcal{G}$ let $J(G) \subseteq J$ be the set of all feasible routes that visit at least two of the three sites belonging to G. Then the following inequalities are valid:

$$\sum_{j \in J(G)} x_j \leq 1 \qquad \forall G \in \mathcal{G} \tag{2.21}$$

The separation of these inequalities could be done by complete enumeration since their number is polynomial ($|V|^3$). However to speed up the separation process only a subset

of the nodes set $V_s \subseteq \mathcal{V}$ is taken into account for the generation of the clusters set $\mathcal{G}$, in particular a node $i \in \mathcal{V}$ belongs to V_s if the value of its associated skip variable s_i is strictly lower than 1 in the last LP iteration. In the worst case $|\mathcal{V}_s| = |\mathcal{V}|$ but experimentally we saw a significant reduction in the cardinality of G. Moreover, the cardinality of G can be further reduced considering all triplets of nodes and discarding the ones that, due to capacity constraints, do not include at least two nodes that can be visited in the same route. This shrink in the cardinality of G can be performed in a preliminary preprocessing phase as it does not depend on variable values.

The introduction of these additional inequalities causes an alteration of the pricing subproblem and therefore it requires the modification of the pricing algorithms. In particular, identified with $\mathcal{P}$ the index set of the valid subset row inequalities added to the MP, the state label used in the exact dynamic programming procedure has to be extended with the introduction of a new variable b_p for every additional cut $p \in \mathcal{P}$ to handle its associated negative dual value ψ_p and, consequently the extension, dominance and join rules have to be modified.

In details, given label $l' = (\mathcal{S}', q', b'_p, C', i')$, label $l'' = (\mathcal{S}'', q'', b''_p, C'', i'')$ and defining $\bar{\mathcal{S}}'' \subset \mathcal{S}''$ as the set containing all the sites visited along the path encoded by label l'' without the inclusion of unreachable sites coming from the application of the technique by Feillet et al. [59], the extension of label l' to l'' is done in the following way:

$$\begin{aligned}
\mathcal{S}'' =& \mathcal{S}' \cup \{i''\} \\
b''_p =& |G_p \cap \bar{\mathcal{S}}''| \, mod \, 2 \qquad \forall p \in \mathcal{P} \\
q'' =& q' + l_{i''} \\
C'' =& C' + c_{i'i''} - \frac{\lambda_{i'}}{2} - \frac{\lambda_{i''}}{2} - l_{i''}\sigma - l_{i''}\rho - \sum_{p \in \mathcal{P}} \psi_p \lfloor \frac{|G_p \cap \bar{\mathcal{S}}''|}{2} \rfloor
\end{aligned} \tag{2.22}$$

where $i' \neq i''$ and G_p is the subset of $\mathcal{N}$ representing the triplet of sites associated with subset-row inequality $p \in \mathcal{P}$. Dominance rules have to be relaxed since the cost of a path

is no longer monotonic and so label l' dominates l'' if:

$$\mathcal{S}' \subseteq \mathcal{S}''$$
$$q' \leq q''$$
$$C' - \sum_{p \in \mathcal{W}} \psi_p \leq C''$$

with $i' = i''$ and $W = \{w \in \mathcal{P} : \psi_w < 0, b'_w > b''_w\}$. Finally the value of b_p, $\forall p \in P$, have to be taken into account when two partial path are joined together to compose a single complete route and in particular we check if $b'_p = 1 \wedge b'_p = 1 \, \forall p \in \mathcal{P}$ the value ψ_p is subtracted from the cost of the route.

2-path Inequalities

In addition to the subset-row inequalities we propose a modified version of the classical 2-path inequalities that works well with VRPDSC. In their traditional version the 2-path inequalities impose that for every set $\mathcal{S} \subseteq \mathcal{V}$ of nodes that cannot be served by a single route at least two routes must be used.

In order for these cuts to be adapted to the VRPDSC we have to take into account that the problem allows for some node to be skipped and that the only resource available is the vehicle capacity. Thus, for every set $\mathcal{S} \subseteq \mathcal{V}$ of sites whose amount of waste requires at least two vehicles to be collected, the following inequalities hold:

$$\sum_{i \in \mathcal{S}} s_i + \sum_{k \in \mathcal{K}(\mathcal{S})} z_k \geq 2 \tag{2.23}$$

where $\mathcal{K}(\mathcal{S})$ is the set of routes serving at least a site belonging to $\mathcal{S}$. It is worth noting that, regardless the inclusion of the skip variables, inequalities 2.23 differ from the original version presented Kohl et al. in [65]. In fact, like for instance in Archetti et al. [67], the coefficient of the variables z_k is always equal to one while in the original version the coefficients of the variables representing routes depend on the number of times a route enters the set $\mathcal{S}$.

The separation procedure is based on a revision of the heuristic algorithm proposed by Kohl et al. [65]. In particular, let $m(\mathcal{S})$ be the minimum number of vehicles necessary to serve all clients in $\mathcal{S}$ and let $q(\mathcal{S}) = \sum_{k \in \mathcal{K}(\mathcal{S})} z_k$ be the actual, possibly fractional, number of routes serving nodes in $\mathcal{S}$ then the separation problem requires to find sets $\mathcal{S} \subseteq \mathcal{N}$

with $\sum_{i\in\mathcal{S}} s_i + q(\mathcal{S}) < 2$ and $m(\mathcal{S}) \geq 2$, that is: sets of bins locations that cannot be cleaned up using a single route but that are visited, in the current fractional LP solution, by less than two routes. In our problem, this can be done quite easily: given a set $\mathcal{S}$ with $\sum_{i\in\mathcal{S}} s_i + q(\mathcal{S}) < 2$ we check if the sum of the amount of waste contained in the bins belonging to $\mathcal{S}$ exceed the capacity of the vehicles, in other worlds if $\sum_{i\in\mathcal{S}} l_i > W$ a cut is generated and added to the MP. Sets $\mathcal{S}$ are heuristically individuated computing the flow over the graph $G = (V, A)$ exactly as described in the paper by Kohl et al. [65].

The addition of these inequalities introduce a new vector χ of variables in the dual problem that requires the modification of the pricing algorithm.

In details, identified with $\mathcal{P}$ the index set of all 2-path cuts added to the MP and $p(\mathcal{S})$ the index of the cut corresponding to the set $\mathcal{S}$, we modified the exact dynamic programming algorithm for the RCESPP adding a new resource in the labels. Such resource is initialized at 0; after the first visit to a customer in $\mathcal{S}$, the resource is set to 1, and the reduced cost of the label is decreased by the value of the corresponding dual variable $\chi_{p(\mathcal{S})}$. In the joining phase if the resource is equal to 1 for either the forward or the backward paths the dual variable associated with the corresponding cuts is subtracted from the reduced cost of the complete path. Finally dominance rules are modified with an additional condition saying that no label having one of these resources at 1 can dominate a label having the corresponding resource at 0.

It is worth noting that, unlike subset-row inequalities, in order to not modify the structure of the problem a 2-path inequality added to the MP must be carefully updated every time a new column is inserted in the problem. Indeed the optimal solution may be dropped if newly generated variables are not taken into account in the existing 2-path inequalities.

2.4.7 Branching

At the end of the column generation process the solution found may be fractional, in this case branching policies are applied in order to recover the integrality. In particular, we use four different strategies that are considered and executed in the order presented thereafter. In the following we indicate as $\bar{z}$ and $\bar{s}$ the values assumed by the variables in the optimal solution of the MP.

Branching on the Number of Vehicles. The first rule taken into account is the branching on the number of vehicles. Indicate with m the number of routes used in the fractional

LP solution computed as $m = \sum_{p \in \Omega} \bar{z}_p$ that is the sum of the values of all variables encoding routes. If m is not integer the branching is executed and two nodes are generated, in the first it is imposed to use at least $\lceil m \rceil$ vehicles while in the other the maximum number of routes is limited to $\lfloor m \rfloor$. This result is obtained modifying the left and right hand side of the constraints (2.4), in particular in the first case the modified constraint becomes as follows

$$m \leq \sum_{p \in \Omega} z_p \leq K$$

in the other node instead the constraint is

$$\sum_{p \in \Omega} z_p \leq m.$$

This branching technique leaves the pricing subproblem unchanged: dual variable μ still appears as constant in the objective function of the pricing problem, but it is now unrestricted in sign. No modifications in the pricing algorithms are required to handle this branching technique.

Branching on Skip Sites. When the number of vehicles employed in the optimal LP solution of the MP is integer a second branching rule is taken into account. Let $s_{\bar{i}}$ be the variable whose value is fractional and closest to 1. The branching is performed in the following way: two children nodes are generated and in the first the visit of the site $\bar{i}$ is forbidden imposing $s_{\bar{i}} = 1$ on the contrary in the other node $s_{\bar{i}} = 0$ and therefore the site $\bar{i}$ must be visited.

Branching on Arcs. If all skip variables are integer branching on arcs is considered. We choose the site $i \in \mathcal{V}$ that is split among the largest number of routes in the optimal fractional solution of MP and that has at least two open outgoing arcs. Then we forbid half of its outgoing arcs to be used in the first child node, and the other half to be used in the second child node. To handle these branching decisions in the pricing problem it is enough to set travel distance of forbidden arcs to $+\infty$.

Branching on Routes Finally when none of the previous rules can be applied branching on variables encoding routes is performed. Among all z variables in the RMP let $z_{\bar{i}}$ be the

one whose value, in the optimal LP solution of the MP, is fractional and closest to $0,5$. The branching is performed in the following way: two children nodes are generated. In the former the usage of the route $z_{\bar{i}}$ is forced setting $z_{\bar{i}} = 1$, while in the latter the route $z_{\bar{i}}$ is forbidden imposing $z_{\bar{i}} = 0$.

2.4.8 Stabilization

We have implemented a stabilization technique based on the methods presented by Uchoa et al. [68] in order to improve the rate of convergence of the column generation algorithm. A detailed description of the stabilization method itself is given in the Appendix A.

The computation of the optimal solution to the Lagrangian relaxation of the MP is a crucial point in this stabilization technique. This is not a difficult task. In fact, given dual vectors λ, μ, σ and ρ the Lagrangian relaxation of our problem reads:

$$\begin{aligned}
\min \ & \sum_{p\in\Omega}(c_p - \sum_{i\in\mathcal{V}} a_{ip}\lambda_i + \mu - l_p\sigma + f_p\rho) - \sum_{i\in\mathcal{N}_2}\lambda_i s_i + \\
& + \sum_{i\in\mathcal{V}}\lambda_i - K\mu + L^T_{min}(\sigma+\rho) - WK\rho \\
\text{s.t.} \quad & 0 \le s_i \le 1 \qquad \forall i \in \mathcal{N}_2 \\
& 0 \le z_p \le 1 \qquad \forall p \in \Omega
\end{aligned}$$

The objective function comes from the relaxation of all constraints in the MP but the bounds on the variables values. This problem has the integrality property and its optimal solution is easily found by inspection analyzing the coefficients of the variables s_i, z_p. In particular, every variable whose coefficient is negative is set to 1, any other is set to 0. It is worth noting that the coefficients of the z_p in the Lagrangian relaxation are their reduced costs exactly as computed in (2.10).

2.4.9 Primal Heuristic Algorithm

We have implemented a primal heuristic algorithm in order to quickly find feasible solutions during the execution of the algorithm. In particular the solutions found by our algorithm come from the combination of variables that are already in the RMP. The algorithm is made up of three phases: initial solution, improvement and feasibility and is executed once for every node of the branching tree.

In details defined $\mathcal{I}$ as the set of the variables composing the heuristic solution, the algorithm runs as follows.

Initial solution. In the first phase variables that visit the mandatory nodes are selected. In particular the mandatory nodes $i \in N_1$ are considered in lexicographic order and if i is not covered by any of the variables in the partial solution $\mathcal{I}$ the most useful variable visiting i is introduced in $\mathcal{I}$. In order to evaluate the usefulness of the variables they are sorted in ascending order by the ratio between the traveling cost and the total amount of waste collected in the route encoded. The lower is this ratio the more useful is a variable. The variables are considered following this order and the first variable visiting i and not visiting any of the node in the routes encoded by the variables already in the partial solution is selected and added to $\mathcal{I}$. This phase terminates when either all mandatory nodes are covered or $|\mathcal{I}| = K$. It produces a, possibly unfeasible, solution whose cost C is given by the sum of the routing costs of all the variables in $\mathcal{I}$.

Improvement. In the second phase every variable in $\mathcal{I}$ is tentatively removed from the solution and replaced, in a greedy way, by one or more variables whose combination is cheaper. In particular removing i from $\mathcal{I}$ gives a solution with cost $C - c_i$ then, among all variables involving only nodes not visited by the variables in $\mathcal{I}/\{i\}$, the first compatible variables whose total cost is less than c_i are added to $\mathcal{I}$. This phase terminates when there are no more useful substitutions.

Feasibility. The last phase of the algorithm tries to ensure the feasibility of the solution. The algorithm considers all variables in $\mathcal{I}$ and computes the set of the sites served $\mathcal{S}(\mathcal{I}) = i \in R_j : j \in \mathcal{I}$; then it adds to $\mathcal{I}$ all skip variables s_i associated with optional sites not in $\mathcal{S}(\mathcal{I})$. It is worth noting that sometimes this procedure may not generate a feasible solution due to constraints (2.2) and (2.8). In these cases the algorithm fails.

2.5 Experimental Analysis

Implementation. The algorithms described in this chapter have been implemented in C++, using SCIP 2.1 [69] as branch-and-cut-and-price framework, linked to CPLEX 12.2 as pure LP solver. SCIP takes care of the management of the column and row pool and it is able to remove and re-insert them as needed. All presolving algorithms, propagators and

the automatic cut generator embedded in SCIP along with the following general purpose heuristic algorithms: shift-and-propagate, DINS, fix-and-infer, int-diving and shifting have been disabled as they were considered either incompatible or useless for our problem. All remaining SCIP parameters were kept at their default values including the branching tree exploration strategies.

The experimental campaign has been performed on a single core of a PC Intel® Core 2 Duo™ CPU T7300 (2.00 GHz) with 4 GB RAM and running Ubuntu 10.04 operating system. A time limit of 1 hour was imposed to each run.

Benchmark Instances. Our approach has been tested on a set of 75 instances. This test set is the same used in [41] and includes instances with a number of nodes ranging from 15 to 100 characterized by a very high variability in all the input parameters such as the number of mandatory and optional nodes, the vehicle capacity W, the values L^T_{min}, l_i and W_{max}. Moreover the instances do not state any value for the fleet size K and for the minimum filling requirement L_{min}. Since results published by Macedo et al. [41] are achieved with $L_{min} = 0$, we first tested our procedure with the same settings in order to compare our results with the state of the art and then in Subsection 2.5.4 we analyzed the behavior of our algorithm with $L_{min} > 0$.

Fleet Size. Since no information on the fleet size is given in the instance we adopted an iterative procedure to find out the optimal number of vehicles to be employed to minimize the routing costs. We started setting $K = \lceil \frac{(\sum_{i \in \mathcal{V}} l_i)}{W} \rceil$ then we solved the problem. If the optimal solution used a number of vehicles equal to K we incremented K by one and iterated until the optimal solution found used at most $K - 1$ vehicles, this means that no more than $K - 1$ vehicles are needed to reach the optimal solution. However if we were not able to reach the optimal solution within the time limit we set $K = |V|$ to ensure that proper lower bounds were found.

2.5.1 Evaluation of the Stabilization Methods

In order to evaluate the impact of the stabilization method on the performance of the column generation algorithm we solved all the instances with six different values of the parameter α: $0.0, 0.02, 0.05, 0.1, 0.2$ and 0.5. The computation is stopped at the route node and is carried out in the cleanest way possible using only the exact pricer and no cuts in

order to not distort the impact of the stabilization introducing heuristic components.

An overview of the results achieved is shown in Table 2.1 and Table 2.2. In Table 2.1 computational times obtained using different values of α are compared. In details, the table is made up of six columns, one for every possible value of α, fifteen rows reporting aggregated average, maximum, and minimum values computed on instances with the same number of nodes and three additional rows showing average, maximum, and minimum values considering all instances. Table 2.2 reports the difference in the number of pricing iterations from the execution without stabilization, in this case only five columns are employed one for every value of $\alpha > 0$. It is not easy to draw a general picture by analyzing the results achieved varying α from 0, that is equivalent to the algorithm without stabilization, to 0.5. In fact, although for every instance it is almost always possible to find a value for α that improves the performance of the column generation compared to the execution with $\alpha = 0$, we were not able to identify a single value of α bringing a constant improvement in the majority of the instances. Moreover, it seems that the larger is the instance, the less effective is the stabilization methods. Indeed on instances with 100 nodes it turned out to be very often harmful in terms of computational time. However for instance with less than 50 nodes setting α to 0.02 gives on the majority of the instances a reduction in the number of the pricing iterations even if this does not always come together with a reduction in the computational time. In fact when the stabilization method is applied, at every pricing iteration may correspond several calls to the pricing algorithm in order to handle possible mispriced columns, in particular in this problem we assisted to an increment in the number of calls to the exact dynamic programming by, on average, 10%. This value is lower (about 5%) when α is below 0.1 and raises up to 35% when $\alpha = 0.5$. Considering these results and the difficulty in extracting from these data a proper value for α we decided to disable the stabilization method in the other tests.

2.5.2 Lower Bounds

To evaluate the quality of the linear relaxation of our model and to investigate the impact of the cuts on the quality of the lower bound we tested every possible combination of cuts stopping the computation at the route node. All pricing algorithms were activated. In detail the four configurations tested are: No Cuts (NC) in which none of the additional inequalities presented in Section 2.4.6 is taken into account, Subset-row Cuts (SC) in which only the subset-row inequalities are considered, 2-Path Cuts (2C) in which only 2-path

Inst.	α					
	0.00	0.02	0.05	0.10	0.20	0.50
Avg 15	0.03	0.04	0.04	0.04	0.04	0.05
Max 15	0.11	0.14	0.15	0.15	0.13	0.16
Min 15	0.01	0.02	0.01	0.01	0.02	0.02
Avg 25	98.94	238.96	147.27	147.27	124.93	53.04
Max 25	1220.01	3286.03	1981.41	1981.41	1672.93	635.66
Min 25	0.02	0.02	0.03	0.03	0.03	0.05
Avg 50	3.59	3.38	3.51	3.51	3.94	3.75
Max 50	24.06	20.23	21.94	21.94	29.43	27.85
Min 50	0.07	0.08	0.08	0.08	0.08	0.08
Avg 75	431.59	419.01	412.13	412.13	396.13	357.72
Max 75	3600.00	3600.00	3600.00	3600.00	3600.00	3600.00
Min 75	0.05	0.10	0.11	0.11	0.16	0.15
Avg 100	58.33	63.03	64.54	64.54	63.00	59.74
Max 100	469.15	505.48	529.35	529.35	426.23	279.85
Min 100	0.26	0.30	0.30	0.30	0.32	0.33
Avg TOT	123.70	172.23	139.78	139.78	128.07	93.97
Max TOT	3600.00	3600.00	3600.00	3600.00	3600.00	3600.00
Min TOT	0.01	0.02	0.01	0.01	0.02	0.02

Table 2.1: Stabilization Methods: Time [s] comparison with different values of α

Inst.	α				
	0.02	0.05	0.10	0.20	0.50
Avg 15	-0.48	0.84	1.14	5.00	9.82
Max 15	12.50	12.50	12.50	28.57	64.29
Min 15	-12.50	-7.14	-12.12	-12.12	-18.18
Avg 25	-0.84	-0.61	-0.16	1.90	4.07
Max 25	5.41	11.32	7.55	20.00	40.00
Min 25	-17.50	-15.00	-10.68	-20.25	-23.93
Avg 50	-0.28	-0.47	2.69	2.68	5.24
Max 50	9.02	7.96	17.21	10.13	21.21
Min 50	-15.38	-13.19	-9.89	-13.64	-8.45
Avg 75	4.80	10.16	12.34	9.79	74.31
Max 75	13.70	66.67	83.33	33.33	1083.33
Min 75	-1.93	-0.50	-1.67	0.00	-6.25
Avg 100	2.09	1.80	1.88	5.12	4.65
Max 100	16.54	13.54	14.15	27.78	16.67
Min 100	-3.49	-8.22	-5.00	-10.00	-9.03
Avg TOT	0.86	2.06	3.31	4.74	18.13
Max TOT	16.54	66.67	83.33	33.33	1083.33
Min TOT	-17.50	-15.00	-12.12	-20.25	-23.93

Table 2.2: Stabilization Methods: variation in the number of pricing iterations with different values of α

inequalities are generated and finally the last configuration including both cuts (BC). The results obtained are shown in Table 2.3 in which, for every configuration of cuts, the average and maximum number of cuts found (N.C.), the percentage improvement (Imp%) in the value of the lower bound compared with the bound achieved with the NC configuration and the computational time in seconds (T[s]) computed aggregating all instances with the same number of nodes are reported.

Analyzing the results the subset row inequalities turn out to be the most effective cuts able to increment the bound in almost every instance. The contribution of the two path inequalities is usually lower but not negligible and more importantly the two contributions can be often usefully combined together as shown by the results obtained with the BC configuration. From the efficiency point of view the analysis is more complex and in fact although the impact of the additional inequalities is always evident in the computational time it is not uniform on all instances. Delving into the causes of such increment in the computational time several aspects must be taken into account; every time a cut is found the pricing algorithms are called again to look for new variables with negative reduced costs, every cut induces a modification in the exact dynamic programming procedure that weakens the dominance rules and finally every time a new column is found if additional inequalities are employed they need to be updated with the newly generated variables. In particular, looking at the detailed results for every instance we found out that on the one hand the separation procedures are quiet fast taking only a few seconds to generate the cuts. On the other hand the exact pricing algorithm is clearly slowed down by the introduction of the cuts and on top of that when additional inequalities are introduced the exact dynamic programming procedure is often called several more times than without the cuts. This is exemplified in instance "$Inst\_25\_03''$ in which without cuts the exact pricer is called 53 times and use a total of 29.43 seconds; when the SC configuration is used 18 subset-row inequalities are introduced, the separator runs for a total of 2.06 seconds while the exact dynamic pricer is called 123 times consuming more than 1200 seconds. In this case a single iteration of the exact pricer costs almost twenty times the computational time required when no cuts are considered.

Although the introduction of additional inequalities can have a huge impact on the computational time spent during the pricing procedures they are a viable way to improve the lower bound and can also be useful to narrow down the size of the branching tree. For these reasons we use the BC configuration in the evaluation of the global performance of

	Cuts Configurations									
	NC	SC			2C			BC		
	T[s]	N.C.	Imp%	T[s]	N.C.	Imp%	T[s]	N.C.	Imp%	T[s]
Avg 15	0.02	1.80	0.42	0.06	0.67	0.17	0.05	1.87	0.43	0.06
Max 15	0.04	10.00	2.32	0.11	3.00	0.83	0.12	10.00	2.48	0.13
Avg 25	3.26	4.73	0.59	106.50	1.47	0.42	7.56	4.60	0.59	106.31
Max 25	29.63	18.00	1.76	1215.77	5.00	1.76	67.84	18.00	1.76	1215.99
Avg 50	0.57	5.00	0.22	2.80	3.67	0.17	1.74	7.27	0.25	3.47
Max 50	5.20	14.00	0.57	25.03	10.00	0.46	15.07	18.00	0.57	28.86
Avg 75	3.48	4.00	0.07	12.90	7.40	0.11	7.91	9.93	0.13	16.67
Max 75	17.16	11.00	0.29	44.60	21.00	0.29	29.84	29.00	0.29	100.41
Avg 100	3.24	7.07	0.23	26.89	5.87	0.18	9.71	8.40	0.24	17.32
Max 100	10.67	20.00	1.35	199.02	18.00	0.76	28.77	21.00	1.35	62.10
Avg TOT	7.33	9.56	0.32	163.37	7.61	0.21	16.86	12.81	0.34	155.13
Max TOT	29.63	20.00	2.32	1215.77	21.00	1.76	67.84	29.00	2.48	1215.99

Table 2.3: Lower Bounds Comparison

our algorithm. In particular we look for violated inequalities at the root node and in every other node that improves the global lower bound.

2.5.3 Global Performances

We have tested our algorithm on all instances, with $\alpha = 0.0$ and the BC configuration, imposing a time limit of 1 hour to the computation. The results obtained by this test are reported in Tables 2.4 and 2.5 in which the following convention for columns headers is employed:

Inst. is the instance name, the number following the "W" indicates the number of nodes in the instance.

B.B.N. is the number of nodes explored in the branch and bound tree.

L.B.R. is the lower bound achieved at the end of the root node.

L.B.F. is the final lower bound obtained at the end of the computation.

Best is the best feasible solution found during the computation.

Gap% is the residual dual gap at the end of the computation.

N.Cuts is the total number of cuts added to the MP.

Time[s] is the computational time in seconds.

Our approach achieves an optimal solution in 56 instances out of 75 and in particular all the instances in Table 2.4 but one, that turned out to be one of the most difficult instances of the whole set. Among the instances not solved to optimality the residual gap goes from 0.01% to 6.83% with an average value around 1.1%. Twenty of the instances are solved at the root node in a few seconds showing that the pricing procedures are, in general, pretty efficient. On the other hand the instances not optimally solved require to branch a lot and very often explore a few thousands of nodes. During the branching the lower bound increase very slowly and indeed the average improvement in the lower bound on instance not closed is only about 0.48%, we observed that the branching on number of vehicles and branching on skipping clients strategies (see Section 2.4.7) are the main responsible for this improvement while the branching on arcs rule gives a very little contribution and indeed when only this rule can be used hundreds of nodes can be explored without an appreciable increment in the value of the lower bound. On top of that such, already poor, improvement comes often not directly by the decision taken through branching but by the introduction of a cut. Branching on Routes has never been applied.

The computational time spent by our algorithm is on average about 1000 seconds considering all instances and goes down to 124 second if only completely solved instances are considered. Most of the time is spent in the exploration of the branching tree and in fact the time employed at the root node is, on average, shorter than 6 seconds with a few exceptions among with the most noticeable is the instance $W25\_03$ that requires 95 seconds in order to compute the exact solution of the MP. Analyzing the distribution of the computational times among all procedures we found out that the exact and heuristic pricers together use on average 70% of the computational time equally shared between them, the greedy pricing algorithm instead is responsible for only 2% of the computational effort. As already noted in the previous section the time spent by the separation procedures is not negligible and in fact the subset-row separation routine consumes as much as 12% of the computational time while the 2-path cuts on average 5%.

The VRPDSC is not an easy problem from the primal point of view. Indeed, also finding a feasible solution is often computationally very expensive. This is entirely reflected in the performance of both our custom primal heuristic algorithm and in all the SCIP heuristics (see Berthold [70]). Our procedure is able to find a feasible solution at the end of the root node in about 90% of the instances while with the SCIP heuristics the percentage goes down to about 70%. This situation changes drastically during the exploration of the

branching tree, in fact in this case our primal procedure performs poorly generating very often solutions that do not match the branching decisions on the contrary SCIP heuristics keep producing feasible integer solution with the same rate throughout all the branching tree. For this reason we stop the execution of our primal algorithm at the root node.

Inst.	N.B&B	LB R.	LB F.	Best	Gap %	Cuts	Time
W15_01	1	2688.00	2688.00	2688.00	0.00	0	0.03
W15_02	1	4887.00	4887.00	4887.00	0.00	0	0.01
W15_03	45	3331.88	3516.00	3516.00	0.00	5	0.93
W15_04	25	6643.00	6727.00	6727.00	0.00	2	0.10
W15_05	1	5395.00	5395.00	5395.00	0.00	0	0.02
W15_06	91	2535.33	2673.00	2673.00	0.00	5	0.51
W15_07	28	1568.05	1796.00	1796.00	0.00	0	0.12
W15_08	3	3762.67	3993.00	3993.00	0.00	11	0.11
W15_09	68	2216.00	2307.00	2307.00	0.00	4	0.32
W15_10	1	5961.00	5961.00	5961.00	0.00	0	0.03
W15_11	1	5747.00	5747.00	5747.00	0.00	0	0.02
W15_12	3	3086.00	3096.00	3096.00	0.00	19	0.14
W15_13	1	2276.00	2276.00	2276.00	0.00	0	0.02
W15_14	1	4169.00	4169.00	4169.00	0.00	1	0.03
W15_15	170	5189.00	5284.00	5284.00	0.00	13	0.96
W25_01	98	7008.28	7065.00	7065.00	0.00	23	3.35
W25_02	1	741.00	741.00	741.00	0.00	0	0.01
W25_03	4474	3170.80	3246.57	3427.00	5.66	12	3600.00
W25_04	15	840.33	873.00	873.00	0.00	5	0.09
W25_05	9	2097.12	2136.00	2136.00	0.00	0	0.08
W25_06	359	5794.17	5817.00	5817.00	0.00	37	4.63
W25_07	1	5626.00	5626.00	5626.00	0.00	0	0.13
W25_08	3	6395.00	6474.00	6474.00	0.00	2	0.03
W25_09	3	8788.00	8807.00	8807.00	0.00	8	0.17
W25_10	201	8001.00	8142.00	8142.00	0.00	33	1.21
W25_11	2883	4322.28	4367.00	4367.00	0.00	19	1353.29
W25_12	1	4063.00	4063.00	4063.00	0.00	2	0.07
W25_13	3	7906.67	7958.00	7958.00	0.00	6	0.16
W25_14	1	6167.00	6167.00	6167.00	0.00	0	0.03
W25_15	3	2829.57	2943.00	2943.00	0.00	5	97.18

Table 2.4: Waste Collection Results: 15 - 25 nodes

In [41] the authors claim to reach an optimal solution for 27 instances up to 50 nodes imposing a time limit of 1000 seconds and using a configuration similar to our test machine. Our algorithm is able to find within the same time limit an optimal solution for 40 of the

45 instances with 15, 25 and 50 nodes. Moreover comparing the computational time on the 27 instances solved to optimality by both approaches we observed that their solving time is on average about 140 seconds while ours is less than 1 second marking an improvement of 2 orders of magnitude.

2.5.4 The minimum filling constraints

All results presented so far, in order to be compatible with the solutions reported by Maced et al. [41], are obtained imposing $L_{min} = 0$. In this section we analyze how the performance of our algorithm varies modifying the minimum filling requirements and in particular we tested our approach with three different values of L_{min}: $0.6 \cdot W$, $0.8 \cdot W$ and $0.99 \cdot W$. Detailed results for every instance and every value of L_{min} can be found in the Appendix C.

Before going further in the analysis of the impact of the minimum filling constraints on the performance of our algorithm we would like to point out that such constraints can make an instance infeasible, in particular 4 instances became infeasible in case $L_{min} = 0.8 \cdot W$ and 16 when $L_{min} = 0.99 \cdot W$; e.g. instance $W15\_04$: all 15 bins are mandatory, 7 of them have an amount of waste equal 3 while in the others is 2. W is equal to 5 and so if $L_{min} = 0.8 \cdot W$ then $L_{min} = 4$. In this case there is no possible solution because there is always a bin with l_i equal to 2 that is not collected.

An overview of the computational results is contained in Table 2.6 where the average number of nodes explored in the branch and bound tree (B.B.N.), the average residual gap for instances not solved to optimality (Gap %), the average number of variables added to the MP (N.Vars), the number (N. Solved) and the percentage (Solved %) of the instances optimally solved and the average computational time in seconds (Time[s]) are reported.

The overall problem becomes as expected easier with the introduction of the minimum filling constraints indeed the stronger are the constraints the higher is the percentage of problems solved to optimality. However this does not come for free and require a clear additional effort in solving, in an exact way, the associated pricing subproblem, indeed when $L_{min} = 0.99 \cdot W$ although the size of the branching tree is about half of the one reached when $L_{min} = 0$ the average computational time is one and half time the computational effort required when the minimum filling constraint is not considered. Such a behavior is partially due to the fact that only a fraction of the variables generated with $L_{min} = 0$ is used in the other cases and this leads to smaller search trees to be explored. Despite the

Inst.	B.B.N	L.B.R.	L.B.F.	Best	Gap %	N.Cuts	Time[s]
W50_01	1	12335.00	12335.00	12335.00	0.00	0	0.13
W50_02	1	13233.00	13233.00	13233.00	0.00	2	0.19
W50_03	117	13372.83	13410.00	13410.00	0.00	35	18.62
W50_04	239	17584.04	17627.00	17627.00	0.00	60	122.73
W50_05	3	23862.50	23927.00	23927.00	0.00	5	0.27
W50_06	1	12839.00	12839.00	12839.00	0.00	0	0.10
W50_07	51	13983.50	14047.00	14047.00	0.00	37	15.84
W50_08	161	2508.18	2554.00	2554.00	0.00	15	16.30
W50_09	17679	13592.50	13838.12	14145.00	2.22	404	3600.00
W50_10	69	5973.76	6018.00	6018.00	0.00	4	3.04
W50_11	3	15639.33	15658.00	15658.00	0.00	6	0.28
W50_12	3	12158.00	12205.00	12205.00	0.00	1	0.07
W50_13	3273	10406.83	10548.12	10741.00	1.83	69	3600.00
W50_14	9742	17260.25	17315.25	17325.00	0.06	530	3600.00
W50_15	367	14733.43	14789.00	14789.00	0.00	47	97.55
W75_01	1178	23105.15	23146.14	24728.00	6.83	232	3600.00
W75_02	1	17219.00	17219.00	17219.00	0.00	3	1.06
W75_03	1	38713.00	38713.00	38713.00	0.00	9	0.29
W75_04	1133	20969.80	21044.20	21124.00	0.38	204	3600.00
W75_05	1072	6236.75	6353.00	6353.00	0.00	64	585.17
W75_06	1	14255.00	14255.00	14255.00	0.00	0	0.22
W75_07	3498	24600.00	24628.95	24672.00	0.17	295	3600.00
W75_08	1148	21248.22	21335.40	21415.00	0.37	128	3600.00
W75_09	2934	21242.96	21306.82	21363.00	0.26	367	3600.00
W75_10	1	9004.00	9004.00	9004.00	0.00	0	0.05
W75_11	686	23656.84	23707.00	23709.00	0.01	260	3600.00
W75_12	1253	21101.13	21151.80	21204.00	0.25	256	3600.00
W75_13	1	18226.00	18226.00	18226.00	0.00	0	0.37
W75_14	3	27803.50	27807.00	27807.00	0.00	7	4.34
W75_15	1905	25575.55	25603.11	25628.00	0.10	262	3600.00
W100_01	3461	18656.15	18704.83	18742.00	0.20	235	3600.00
W100_02	75	29008.00	29020.00	29020.00	0.00	11	37.27
W100_03	3	24737.00	24741.00	24741.00	0.00	0	1.63
W100_04	7	118.15	143.00	143.00	0.00	0	1.98
W100_05	1938	37523.67	37549.00	37689.00	0.37	161	3600.00
W100_06	1204	24449.08	24528.64	24674.00	0.59	187	3600.00
W100_07	910	23009.03	23099.70	23485.00	1.67	179	3600.00
W100_08	1	29278.00	29278.00	29278.00	0.00	6	1.48
W100_09	43	5857.26	5910.00	5910.00	0.00	8	16.50
W100_10	1473	2103.16	2216.00	2216.00	0.00	53	1538.30
W100_11	11145	50281.50	50344.50	50410.00	0.13	357	3600.00
W100_12	902	32015.24	32074.89	32342.00	0.83	134	3600.00
W100_13	2325	6666.42	6719.00	6719.00	0.00	339	2777.04
W100_14	103	20424.59	20504.00	20504.00	0.00	17	205.00
W100_15	495	32008.75	32033.54	32164.00	0.41	119	3600.00

Table 2.5: Waste Collection Results: 50 - 100 nodes

fact that the percentage of optimally solved problems grows with the increase of L_{min} we would like to point out that the problems result more difficult from the primal point of view. In fact, unlike the case when $L_{min} = 0$ where a feasible primal solution is always found at the end of the root node, when the minimum filling constraints are taken into account is often difficult to find (as shown in the tables in Appendix C) a feasible solution. Indeed, in several instances the first primal solution is found after more than one hundred nodes and in one instance we are not able to provide any feasible integer solution within one hour of computational time.

L_{min}	B.B.N.	Gap %	N.Vars	N. Solved	Solved %	Time[s]
0.00	1054.73	1.24	1203.95	56	74.67	1004.13
0.60	2866.32	1.03	871.87	58	77.33	930.24
0.80	841.32	0.47	854.76	55	77.46	1038.77
0.99	570.87	1.55	693.35	48	81.36	1429.70

Table 2.6: Waste Collection - Minimum Filling Constraints

Table 2.7 brings the focus of the analysis on the impact of the minimum filling constraints on the exact algorithm for the RCESPP (see Section 2.4.3). In particular in the table are reported: the average number of iterations (N.I.), the number of iterations when the minimum filling constraints are explicitly taken into account (N.M.I.) and the average computational time, in seconds, for every iteration (Time per I.).

L_{min}	N.I.	N.M.I.	Time per I.
0.00	2722.22	0.00	0.04
0.60	1181.03	473.47	2.48
0.80	1013.63	472.63	2.89
0.99	698.58	600.56	6.49

Table 2.7: Waste Collection - Exact DP Pricer with Minimum Filling Constraints

The situation showed in Table 2.7 is quite clear; the higher is the value of L_{min} the more difficult is to find the exact solution of the pricing subproblem. In fact when $L_{min} = 0.99 \cdot W$ the exact dynamic programming algorithm for the RCESPP may require up to 200 times the computational effort needed when $L_{min} = 0$. Such a big difference is due to the use of a weaker dominance rule in the iterations when the minimum filling constraints have to be explicitly considered. The ratio between iterations with stronger and weaker dominance

rule is about 60% when L_{min} is either equal to $0.60 \cdot W$ or $0.80 \cdot W$ and increases to more than 80% when $L_{min} = 0.99 \cdot W$ meaning that very few feasible variables may be generated without forcing the minimum filling constraint in the pricing subproblems.

2.6 Conclusions

In this chapter we have proposed a new solving approach for the VRPDSC, an extension of the VRP concerning the optimization of the waste collection system in a metropolitan area.

We have applied our unified framework for LRP to the VRPDSC in order to show that our approach is general enough to be competitive even on a problem when the location part is missing. In details, we have devised a mathematical formulation for the problem that is compatible with our branch-and-cut-and-price algorithm and we have designed pricing algorithms, cuts, branching rules and a stabilization method to provide exact solutions for the VRPDSC. On top of that, we proposed a method to explicitly take into account minimum filling constraints that require the modification of the pricing procedures: although these constraints are part of the problem no solution attempt is given in the literature.

We have tested our approach against instances taken from the literature showing that our algorithm performs better than the previous approaches from both the efficacy and the efficiency point of view.

Chapter 3

Planetary Surface Exploration

3.1 Introduction

In the last couple of decades the exhaustive surface exploration of planetary bodies has become one of the main concerns of any space agency on Earth. In particular, starting from the late 90's the surface of Mars has attracted renewed attention after almost fifteen years from the conclusion of the NASA's highly successful Viking program [71] that brought two lander on the Martian ground and that was, from 1975 and for the following 20 years, the main source of facts and information about the red planet. Next step in the exploration of the planetary surface was made again by the US' space agency on the summer of 1997 when, as part of the Discovery program [72], the Pathfinder spacecraft [73] succeeded in landing and deploying a base station with roving probe, called "Sojourner", on Mars. After that day there have been several other attempts to bring on the Martian soil robotized vehicles and in particular the famous rovers "Spirit", "Opportunity" [74] represent a very successful story and a giant advancement in the exploration of the red planet. Nowadays, after a few other missions, among which the ESA's "Mars Express" is one of the most representative [75], the interest is not over. In fact, two other rovers are scheduled to be deployed on Mars, "Curiosity" at the end of 2011 and later, in 2018, a vehicle designed by ESA and called "ExoMars" [76]. All the data gathered by rovers and landers along with the informations collected by the several spacecrafts orbiting around Mars allowed NASA to identify more than one hundred and fifty sites of interest on the Martian surface that would be worth being carefully explored [77]. Every site represents a, potentially enormous, source of scientific knowledge and so various stakeholders are competing in order to head

the exploration toward the zones of the red planet that they perceive as more appealing for their scientific interests. Unfortunately, due to technological and budget limitations, it would be impossible to visit all the sites and for this reason decisions have to be made about which sites to examine and how carrying on the exploration. Moreover, it is worth noting that Mars is only the first major planetary body whose soil has received lot of attention, in fact, the exploration of the surface of other planets has already been taken into account, E.G. ESA's mission "BepiColombo" was supposed to bring a rover on Mercury in 2014 [78], and will become one of the main space activities involving collaborations among different space agencies.

Such a strong interest in very complex tasks as campaigns of planetary surface explorations, along with the increasing concern about economics constraints, most certainly calls for the use of operations research methodologies and techniques. These methods can be very useful in optimizing the exploration plans in order to maximize the scientific profits collected without exceeding scheduled budgets. In particular, Ahn et al. [79] and [42] first approached, from the operations research point of view, the surface exploration problem and formalized it as a Generalized Location Routing Problem with Profits (GLRPP). The authors introduce an extended Integer Linear Programming formulation for this GLRPP and propose single-phase and three-phase decomposition heuristics, which are able to provide feasible solutions for instances involving up to 100 potential base locations and 1000 exploration sites. The quality of these solutions is assessed a posteriori, through the computation of suitable metrics. In this chapter we propose an approach based on the revision of the GLRPP model presented by Ahn et al. [42], and we design exact optimization algorithms; our algorithms are able to tackle real size instances, providing also a priori guarantees on the optimality of the solutions produced.

In Section 3.2 the key problem features are revised; in Section 3.3 a mathematical formulation of the GLRPP is provided and in Section 3.4 we present our algorithms. Finally, in Section 3.5 we discuss computational results on realistic instances, and in Section 3.6 we briefly draw some conclusions.

3.2 Problem Description

Before diving into the description of our surface exploration problem, we review the terminology used by Ahn et al. [42], as these terms will be extensively used in this chapter.

Sites. A *site* is a place of interest that has been identified on the surface of the planet. Visiting a site is not mandatory but with every site is associated a profit, that represents a numerical quantification of the scientific value collected by exploring the site itself; since additional visits add no scientific value, it is useless to visit each site more than once.

Agents. An *agent* is usually a robotic vehicle designed to operate on the Martian ground. It performs the visit of the sites and collects the associated profits.

Bases. A *base* is the starting and ending point for the exploration of agents. Possible base locations are previously designated on the soil of the planet, but in general only a few of them are actually used.

Resources. Agents have both limited *local resources*, like fuel, and *collective resources*, like overall activity time, which are consumed during explorations. Local resources can be restored when the agent is at the base station, while collective ones cannot. E.G. On the one hand the fuel can be easily and quickly re-supplied when agents stand in a base, on the other hand more complex operations such as mechanical repairs needed after extended exposure to hostile environment may either require longer time or may not even be possible on the surface of the planet.

Routes and Routing tactics. A *route* is defined as a sequence of sites visited by an agent. Every route starts from a base and ends with the agent coming back to the same base. A *routing tactic* characterizes a route by imposing a single route constraint that limits the consumption of local resource and a maximum number of routes that can be performed with the selected routing tactic. A route is defined as feasible if it does not violate the constraint imposed by the routing tactic.

Mission and Mission Strategies. A *mission* is the set of all routes with a common base. Routes belonging to the same mission are usually performed in sequence by the same agent. There may be different technologies for carrying out the exploration of the planet ground: fixing a *mission strategy* consists in selecting a particular technology, and therefore in imposing limits on the collective resource consumption, defining costs and listing all the available routing tactics. All routes belonging to a mission must use one of the routing tactics available for the mission.

Campaign. A *campaign* is defined as the collection of all missions with the same objective and sharing the same budget. The total cost of a campaign is given by the sum of all the costs of missions belonging to it.

The GLRPP requires to find a campaign exploration plan described by a pool of routes starting from a subset of the potential base locations and feasible with respect to the selected routing tactic of the employed mission strategy. Such a plan must be optimized in order to gather as many scientific information as possible respecting all the technological and economics limitations. In particular, a campaign is feasible if it does not violate any of the following conditions:

I a mission strategy and a corresponding routing tactic is selected for each open base;

II the total cost of a campaign, that is the sum of mission costs, does not exceed the available budget;

III the collective resource consumption for routes in the same mission does not exceed a given limit, which depends on the mission strategy chosen;

IV the number of routes in the same base employing the same routing tactic do not exceed a given limit, which depends on the tactic chosen;

V the local resource consumption of each route does not exceed a given limit, which depends on the tactic chosen;

VI each route begins and ends at the same base;

VII no site is visited more than once.

Condition (VII) follows from the observation that it is never convenient to visit each site more than once: such a condition helps in formulating the mathematical models presented in Section 3.3. Among the feasible campaigns, we consider optimal those maximizing the total profit collected by the agents.

This problem fits in the general description of the second type of problem that we have identified in the introduction (see Chapter 1) indeed, in the GLRPP although location decisions have to be taken and represent an important component in the optimization they do not directly contribute to the value of the objective function but act in a more indirect way modifying the extent of the routing choices. In fact the GLRPP can be seen

as an extension and a generalization of the Multi-Depot Heterogeneous-fleet VRP with Profits [23].

3.3 Mathematical Formulation

Formally, our GLRPP for planetary surface exploration can be modeled as follows.

A complete graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ is given where $\mathcal{N} = \mathcal{B} \cup \mathcal{E}$ is the set of nodes and $A = \{(i, j) : i \in \mathcal{N}, j \in \mathcal{N}\}$ is the set of arcs. Set $\mathcal{N}$ is made up of two distinct subsets: $\mathcal{B}$, representing the set of potential base locations, and $\mathcal{E}$, representing the set of sites. For every potential site $i \in \mathcal{E}$ two values are given: p_i and t_i. The former represents the profit obtained visiting the site i while the latter is the time required to gather the data. Moreover a cost f_{ij} is associated with every arc $(i, j) \in \mathcal{A}$, it is an expression of the distance to be traveled, starting from node $i \in N$, to reach node $j \in \mathcal{N}$.

The set $\mathcal{S}$ made up of all possible mission strategies is also given. Every strategy $s \in \mathcal{S}$ is defined by a cost C^s and a set of available routing tactics $\mathcal{T}^s$, and for each $k \in \mathcal{T}^s$ a maximum number of routes $n^{s,k}$ that can be performed using tactic k.

Mission strategies limit collective resource consumption: when strategy $s \in \mathcal{S}$ is selected, coefficients d_0^s, d_d^s, d_τ^s and l_c^s represent respectively the fixed collective resource consumption of a route, the collective resource consumption per unit of distance traveled by an agent, the collective resource consumption per unit of time spent in acquiring data, and the amount of collective resource available for the whole mission. Every route j associated with a base b and a strategy s is therefore characterized by a consumption of collective resource $h_j^{b,s}$ defined as follows:

$$h_j^{b,s} = \underbrace{d_0^s}_{per-route} + \underbrace{TSP^b \cdot d_d^s}_{per-arc} + \underbrace{(\sum_{i \in \mathcal{R}_j} t_i \cdot d_\tau^s)}_{per-site} \tag{3.1}$$

where $\mathcal{R}_j \subseteq \mathcal{E}$ is the set of sites visited in j, and TSP^b is the travel distance of the TSP solution on the graph derived from nodes $\{b\} \cup \mathcal{R}_j$.

Routing tactics, instead, limit the local resource consumption: for every tactic $k \in \mathcal{T}^s$ coefficients $c_0^{s,k}$, $c_d^{s,k}$, $c_\tau^{s,k}$ and $l_r^{s,k}$ respectively represent the fixed local resource consumption of every route, the local resource consumption per unit of distance traveled by the agent, the local resource consumption per unit of time spent in acquiring data and the amount of

local resource available for each route. Therefore, the local resource consumption of a route j using tactic k of strategy s and starting from a base b can be computed as follows:

$$\underbrace{c_0^{s,k}}_{per-route} + \underbrace{TSP^b \cdot c_d^{s,k}}_{per-arc} + \underbrace{(\sum_{i \in \mathcal{R}_j} t_i \cdot c_\tau^{s,k})}_{per-site} \tag{3.2}$$

where $\mathcal{R}_j$ and TSP^b keep the same meaning as in (3.1). Defined J as the set of all possible routes over the graph G then set $J^{b,s,k} \subset \mathcal{J}$ is the set of feasible routes related to base $b \in \mathcal{B}$, mission strategy $s \in \mathcal{S}$ and routing tactics $t \in \mathcal{T}^s$. We say that a route is feasible if its local resource consumption does not exceed $l_r^{s,k}$ (condition V), and it starts and ends at the same base vertex $b \in \mathcal{B}$ (condition VI); therefore the set $J^{b,s,k}$ can be formally defined as follows:

$$J^{b,s,k} = \{j \in J : \underbrace{c_0^{s,k}}_{per-route} + \underbrace{TSP^b \cdot c_d^{s,k}}_{per-arc} + \underbrace{(\sum_{i \in \mathcal{R}_j} t_i \cdot c_\tau^{s,k})}_{per-site} \leq l_r^{s,k}\}. \tag{3.3}$$

A campaign is the collection of all missions with the same goal defined over the same graph $\mathcal{G}$ and sharing the same limited budget M that is used for the allocation of resources to the selected bases in order to set up a mission strategy. A global overview of the decision hierarchy involving all constraints and requirements associated with a campaign is depicted in Fig. 3.1 (taken from [42]).

The GRLPP can be described using a mathematical formulation as follows:

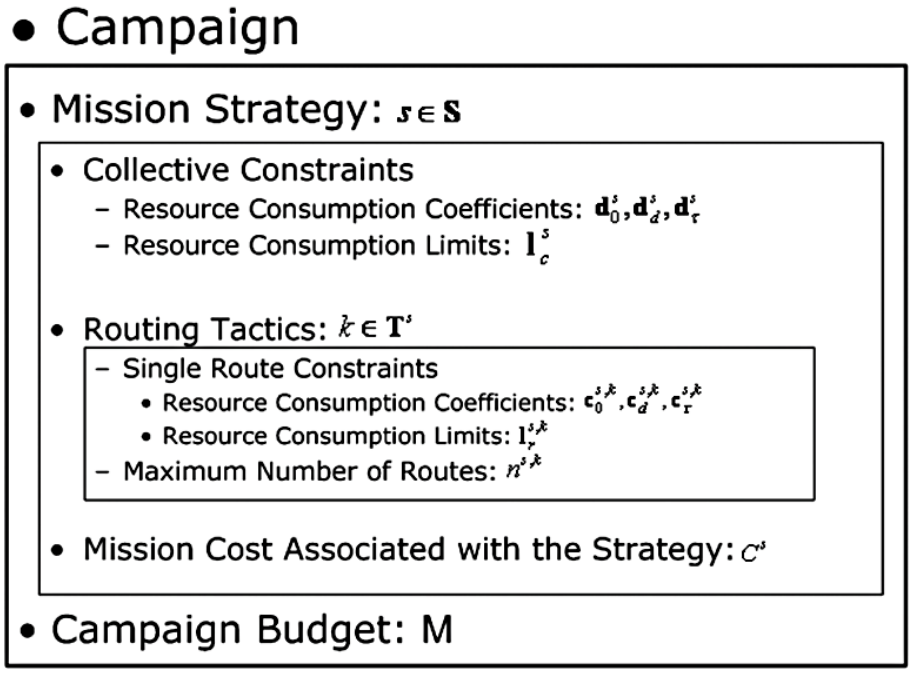

Figure 3.1: Overview of constraints, requirements and decision hierarchy

$$\min \sum_{i \in \mathcal{E}} p_i s_i \tag{3.4}$$

$$\text{s.t. } s_i + \sum_{b \in \mathcal{B}} \sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{T}^s} \sum_{j \in \mathcal{J}^{b,s,k}} a_{ij} x_j^{b,s,k} = 1 \qquad \forall i \in \mathcal{E} \tag{3.5}$$

$$\sum_{k \in \mathcal{T}^s} \sum_{j \in \mathcal{J}^{b,s,k}} h_j^{b,s} x_j^{b,s,k} \leq l_c^s y^{b,s} \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S} \tag{3.6}$$

$$\sum_{j \in \mathcal{J}^{b,s,k}} x_j^{b,s,k} \leq n^{s,k} y^{b,s} \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S}, \forall k \in T^s \tag{3.7}$$

$$\sum_{s \in \mathcal{S}} y^{b,s} \leq 1 \qquad \forall b \in \mathcal{B} \tag{3.8}$$

$$\sum_{b \in \mathcal{B}} \sum_{s \in \mathcal{S}} C^s y^{b,s} \leq M \tag{3.9}$$

$$s_i \in \{0,1\} \qquad \forall i \in \mathcal{E} \tag{3.10}$$

$$x_j^{b,s,k} \in \{0,1\} \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S}, \forall k \in T^s, \forall j \in \mathcal{J}^{b,s,k} \tag{3.11}$$

$$y^{b,s} \in \{0,1\} \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S} \tag{3.12}$$

where s_i are binary variables one for every site $i \in \mathcal{E}$. They represent the skipping of a sites and in fact $s_i = 1$ if site $i \in \mathcal{E}$ is not visited by any route. Binary variables $x_j^{b,s,k}$ symbolize routes and are equal to 1 if the route $j \in \mathcal{J}^{b,s,k}$ is used. Binary variables $y^{b,s}$ describe the association of a strategy with a base and are equal to 1 if strategy $s \in \mathcal{S}$ is assigned to base $b \in \mathcal{B}$. Binary coefficients a_{ij} take value 1 if and only if site $i \in \mathcal{E}$ is visited by route $j \in \mathcal{J}^{b,s,k}$.

For convenience, in our formulation the objective function (3.4) is written as minimization of the profit not collected during the campaign: every time a site is not visited a fee is paid in the objective function. In particular, for our purpose this function is totally equivalent to the maximization of the total profit and in fact a set of sites maximizing the collected profit can then simply be found by complementing each s_i variable. Equations (3.5) represent partitioning constraints, they state that none of the site is mandatory and in fact every site $i \in \mathcal{E}$ can be either visited by a route or avoided assigning to its associated skipping variable s_i the value 1. Constraints (3.5) correspond to condition VII. Collective constraints associated with the strategy employed at every selected base are described by inequalities (3.6), they simply indicate that the sum of the collective resource consumed by all the routes starting from the same base (expressed by the value $h_j^{b,s}$) is limited by the resource available to the associated strategy l_c^s, these constraints are the translation in mathematical language of the condition III. For every tactic $k \in \mathcal{T}^s$ of the strategy $s \in \mathcal{S}$ employed by a selected base $b \in \mathcal{B}$ only a limited number of routes can be selected as stated by constraints (3.7) that model condition IV. Constraints (3.8) formalize that only a strategy can be associated with a selected base as required by condition I. Finally inequalities (3.9) make the budget limitations clear as requested by II. In fact the sum of all costs associated with every strategy $s \in \mathcal{S}$ employed by every selected base $b \in \mathcal{B}$ cannot exceed the global budget M available for the campaign. Conditions VI and V are not explicitly stated in the model because they are implicit in the definition of the set $\mathcal{J}^{b,s,k}$.

In the remainder we discuss the approach we used to solve this problem.

3.4 Branch-and-cut-and-price

The algorithm proposed to solve the GRLPP follows the branch-and-cut-and-price approach (see Ladányi et al. [44]) that is a generalization of the well known LP based branch-and-bound paradigm (see Balas and Toth [45]) in which the linear relaxation of the problem,

usually referred to as the Master Problem (MP), obtained substituting constraints (3.10), (3.11) and (3.12) with

$$0 \leq s_i \leq 1 \qquad \forall i \in \mathcal{E} \tag{3.13}$$

$$0 \leq x_j^{b,s,k} \leq 1 \qquad \forall j \in \mathcal{J}^{b,s,k}, \forall b \in \mathcal{B}, \forall s \in \mathcal{S}, \forall k \in T^s \tag{3.14}$$

$$0 \leq y^{b,s} \leq 1 \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S} \tag{3.15}$$

is solved via column generation (see Desaulniers et al. [55]) to obtain a valid lower bound; additional inequalities are dynamically generated and the column-and-row generation loop is iterated until neither useful columns nor violated cuts are found. If the solution obtained in this way is fractional, a tree search is performed though branching, repeating the bound computation at each node. In this section a description of the main components of the resulting branch-and-cut-and-price algorithm, namely preprocessing, column generation, pricing, cut generation and branching is given.

3.4.1 Preprocessing

In order to strengthen constraints (3.7) we observed that the coefficient $n^{b,s}$ may be reduced computing, for every combination of bases and strategies, the maximum number of routes that can be performed. Let $\mathcal{M}^{b,s}$ be the set of all sites reachable by a route starting from base b and employing strategy s. A site $i \in \mathcal{E}$ belongs to $\mathcal{M}^{b,s}$ if there is a feasible route visiting i starting from base b and using the most resourceful of the routing tactics $k_{max} \in \mathcal{T}^s$ available for mission strategy s. Computing $t_{min}^{b,s}$ and $f_{min}^{b,s}$ as the minimum service time and minimal distance from b among all sites $i \in \mathcal{M}^{b,s}$, the minimum route cost $j_{min}^{b,s}$ can be defined as

$$j_{min}^{b,s} = c_0^{s,k} + c_\tau^{s,k} \cdot t_{min}^{b,s} + 2 \cdot c_d^{s,k} \cdot f_{min}^{b,s}$$

where all c coefficients are the smallest among all available tactics of strategy s. Then the maximum number of route $r_{max}^{b,s}$ starting from $b \in \mathcal{B}$ and employing strategy $s \in \mathcal{S}$ is $r_{max}^{b,s} = \lfloor \frac{l^s}{j_{min}^{b,s}} \rfloor$. At this point inequalities (3.7) can be re-written as follows:

$$\sum_{j \in \mathcal{J}^{b,s,k}} x_j^{b,s,k} \leq \bar{n}^{s,k} y^{b,s}, (\forall b \in \mathcal{B}, s \in \mathcal{S}, k \in T^s), \tag{3.16}$$

where $\bar{n}^{s,k} = \min\{n^{b,s}, r^{b,s}_{max}, |\mathcal{M}^{b,s}|\}$.

3.4.2 Column Generation

Since the number of variables in the MP is exponential in the cardinality of the site set $\mathcal{E}$, thus a column generation approach is used to find the optimal solution of the MP linear relaxation. In fact it would be impossible to solve the problem taking into account all the variables thus, initially, only a small subset of the variables is considered in the MP. In our case such initial Restricted Master Problem (RMP) includes

(a) All columns corresponding to skip variables s_i, in fact there is only a polynomial number of them that is $|\mathcal{E}|$.

(b) All columns associated with $y_{b,s}$. The number of y variables is polynomial and equal to $|\mathcal{B}| \cdot |\mathcal{S}|$.

(c) A subset of $J^{b,s,k}$ made up of $|\mathcal{B}| \cdot |\mathcal{S}| \cdot |\mathcal{T}| \cdot |\mathcal{E}|$ columns, one for each base, strategy, tactic and site, representing the optimal paths serving one customer at a time from every base using every routing tactic available for every strategy.

The RMP is solved once and then the search for columns corresponding to variables $x^{b,s,k}_j \in J^{b,s,k}$ which are not in the RMP but have a negative reduced cost begins. If no such column exists, the solution of the RMP is optimal for the MP linear relaxation as well, and thus yields a valid lower bound to the problem. On the contrary, if any negative reduced cost column is found, it is added to the RMP, and the process is iterated.

In order to find variables with a negative reduced cost the column generation method exploits the dual representation of the problem making use of the values of the dual variables to evaluate the reduced costs of the primal variables. Rewriting constraints (3.6), (3.7), (3.8) and (3.9) as $\geq$ inequalities and denoting by μ, σ, ρ and ν their respective dual vectors, the dual GLRPP problem reads as follows:

$$\max \sum_{i\in\mathcal{E}} \lambda_i - \sum_{b\in\mathcal{B}} \rho^b - M\nu \tag{3.17}$$

$$\text{s.t. } \lambda_i \leq p_i \qquad \forall i \in \mathcal{E} \tag{3.18}$$

$$\sum_{i\in\mathcal{R}_j} \lambda_i - h_j^{b,s}\mu_j^{b,s} - \sigma^{b,s,k} \leq 0 \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S},$$

$$\forall k \in T^s, \forall j \in \mathcal{J}^{b,s,k} \tag{3.19}$$

$$l_c^s \mu^{b,s} + \sum_{k\in\mathcal{T}^s} n^{s,k}\sigma^{b,s,k} - \rho^b - C^s\nu \leq 0 \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S} \tag{3.20}$$

$$\lambda_i \quad \text{unrestricted} \qquad \forall i \in \mathcal{N} \tag{3.21}$$

$$\mu^{b,s} \geq 0 \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S} \tag{3.22}$$

$$\sigma^{b,s,k} \geq 0 \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S}, \forall k \in T^s \tag{3.23}$$

$$\rho^b \geq 0 \qquad \forall b \in \mathcal{B} \tag{3.24}$$

$$\nu \geq 0 \tag{3.25}$$

where λ is the dual vector corresponding to constraints (3.5) in the primal problem. It is worth noting that although λ variables are unrestricted in sign they never take negative values, indeed taking into account the dual objective function (3.17) and the only two constraints (3.18) and (3.19) involving λ variables, it is never convenient for them to assume values lower than zero.

Since to every primal $x_j^{b,s,k}$ corresponds a constraint (3.19) in the dual problem, the pricing problem is equivalent, from a dual point of view, to find the most violated (3.19) constraint. Considering the dual representation of the problem, the reduced cost of the variable $x_j^{b,s,k}$ are as follows:

$$\xi_j^{b,s,k} = -\sum_{i\in\mathcal{R}_j} \lambda_i + h_j^{b,s}\mu^{b,s} + \sigma^{b,s,k}.$$

The coefficient $h_j^{b,s}$ can be expanded taking into account the equation (3.1) and thus the expression of the reduced costs can be rewritten in the following equivalent way:

$$\xi_j^{b,s,k} = \sum_{i \in \mathcal{R}_j} (-\lambda_i + \mu^{b,s} \cdot d_\tau^s \cdot t_i) + TSP_j^b \cdot \mu^{b,s} \cdot d_d^s + \mu^{b,s} \cdot d_0^s + \sigma^{b,s,k}.$$

The problem to be solved in order to generate a new variable to be added to the master problem requires to find a variable $x_j^{b,s,k} \in J^{b,s,k}$ such that its reduced cost $\xi_j^{b,s,k}$ is minimum. It is worth noting that since every variable $x_j^{b,s,k} \in J^{b,s,k}$ represents a feasible route starting from base b and using routing tactic k of mission strategy s it must satisfy conditions V and VI. Constraints (3.5) have repercussion in the generation of the variables in fact, since the selection of a non elementary route would immediately violate the constraint (3.5) of the MP and since no profit is collected by visiting a site more than once, we ensure the feasibility of every generated variable improving, at the same time, the bounds by enforcing each route to be elementary.

Given a base $b \in \mathcal{B}$, a strategy $s \in \mathcal{S}$ and a routing tactic $k \in \mathcal{T}^s$ then the pricing problem can be summarized as follows:

$$\min \xi_j^{b,s,k} = \sum_{i \in \mathcal{R}_j} (-\lambda_i + \mu^{b,s} \cdot d_\tau^s \cdot t_i) + TSP_j^b \cdot \mu^{b,s} \cdot d_d^s + \mu^{b,s} \cdot d_0^s + \sigma^{b,s,k} \tag{3.26}$$

$$\text{s.t. } c_0^{s,k} + TSP^b \cdot c_d^{s,k} + (\sum_{i \in \mathcal{R}_j} t_i \cdot c_\tau^{s,k}) \leq l_r^{s,k} \tag{3.27}$$

where the set of site $\mathcal{R}_j$ is to be determined.

Since dual variables $\mu^{b,s}$ and $\sigma^{b,s,k}$ depend on the employed tactic of the selected strategy at a specific base, at each column generation iteration $|\mathcal{B}| \cdot |\mathcal{S}| \cdot |\mathcal{T}^s|$ pricing subproblems must be solved to ensure the optimality of the solution found. Hence, given a particular base $b \in \mathcal{B}$, a mission strategy $s \in S$ and a routing tactic $k \in \mathcal{T}^s$, the problem of finding the minimum reduced cost column encoding a feasible route that starts and ends at base b and employs routing tactic k of the strategy s turns out to be a well known NP-hard problem (see Dror [57]) called Resource Constrained Elementary Shortest Path Problem (RCESPP, see [59]). Following and extending the approach detailed by Righini and Salani [58] we propose four pricing algorithms to solve the RCESPP: a greedy one, a tabu search one, a heuristic dynamic programming one and an exact dynamic programming one. They are called in sequence, only if the previous pricing algorithm cannot find any column with

negative reduced cost. In the remainder we give a description of the pricing problem and of the four algorithms. For the purpose of better illustrating them, we follow the reverse order with respect to their execution.

3.4.3 Exact Dynamic Programming Algorithm

Let us consider first the case of a single base $b \in \mathcal{B}$, a single mission strategy $s \in \mathcal{S}$ with a single routing tactic $k \in \mathcal{T}^s$, let r and v be the two distinct copies of the base b that is the starting and ending point of the route and let $L = l_r^{sk}$ be the amount of per-route resource provided by the only tactic available. For the exact solution of the RCESPP we use the technique proposed by Righini and Salani [60], which consists of a bi-directional extension of node labels. It associates labels, encoding partial paths, with each site $i \in \mathcal{E}$ of the graph G. Each label is iteratively considered and the corresponding path is extended to adjacent nodes.

Label Structure. A label is defined by the tuple $(\mathcal{W}, q, C, i)$, where C is the cost of the path, i is the last site visited, $\mathcal{W}$ is a set indicating which nodes have been already visited and q represents the amount of resource consumed. Each label identifies a state that represents a path of cost C from a base b to the site i using an amount of resource equal to q and visiting all sites in $\mathcal{W}$. The optimal solution of the problem is the minimum cost path going from r to v consuming an amount of resource $q \leq L$.

Extension. A label $(\emptyset, c_0^{s,k}, 0, r)$ is initially created, it encodes a path associated with the starting base, visiting no site, collecting no profit and consuming the amount of local resource needed to set up the agent. Then, a label $l' = (\mathcal{W}', q', C', i')$ is iteratively selected and extended from site i' to each, not already visited, site $i'' \in (\mathcal{E} \cup \{v\}) \setminus \mathcal{W}')$, creating another label $l'' = (\mathcal{W}'', q'', C'', i'')$ as follows:

$$\begin{aligned} \mathcal{W}'' &= \mathcal{W}' \cup \{i''\} \\ q'' &= q' + c_d^{s,k} \cdot f_{i'i''} + c_\tau^{s,k} \cdot t_{i''} \\ C'' &= C' - \lambda^{i''} + \mu^{b,s} \cdot d_\tau^s \cdot t_{i''} + \mu^{b,s} \cdot d_d^s \cdot f_{i'i''} \end{aligned} \tag{3.28}$$

where $\lambda^r = \lambda^v = 0$. The new state l'' is feasible if $q'' + f_{i''v} \leq L$, with $f_{rv} = f_{vr} = f_{vv} = 0$, in other words a label can be extended from i' to i'' if there is enough resource available to

reach and explore j and going back to the base.

Dominance Test. During the extension of labels, a dominance test is performed to fathom labels that cannot lead to an optimal solution. Let $l' = (\mathcal{W}', q', C', i)$ and $l'' = (\mathcal{W}'', q'', C'', i)$ be two generic labels associated with the same site i. Then the former dominates the latter if the following conditions hold:

$$\mathcal{W}' \subseteq \mathcal{W}'' \tag{3.29}$$

$$q' \leq q'' \tag{3.30}$$

$$C' \leq C'' \tag{3.31}$$

Relations (3.29), (3.30) and (3.31) represent a set of necessary dominance conditions, in fact dropping any of them introduces the possibility for an optimal label to be discarded. Furthermore, as shown in Feillet et al. [59], it is sometimes possible to identify a site $u \in \mathcal{E}$ that cannot be reached by any feasible extension of a given label, because of resource limitations. In this case it is useful to insert u in the set $\mathcal{W}$ of that label: it is easy to check that enlarging set $\mathcal{W}''$ helps satisfying condition (3.29); at the same time, if a site cannot be reached by extending label l' due to resource limitations, it cannot be reached by extending label l'' since resource consumption in l'' is not lower. Therefore, enlarging each set $\mathcal{W}$ allows dynamic programming fathoming a larger number of labels and hence it reduces the computation time.

Decremental state space relaxation. In order to speed up the solution process the dynamic programming algorithm is executed iteratively applying decremental state space relaxation as described in Salani et al. [80]. The idea behind such a relaxation is simple: the state space of the problem is reduced projecting it to a smaller one by discarding the elementary constraints, and then iteratively reintroducing them until a feasible solution for the RCESPP is found. More formally, given a set of sites $\tilde{\mathcal{E}} \subseteq \mathcal{E}$ called critical set the extension rule (3.28) can be replaced with

$$\mathcal{W}' = (\mathcal{W}'' \cup j) \cap \tilde{\mathcal{E}} \tag{3.32}$$

This relaxed problem can be solved more efficiently, since more labels can be compared in the dominance test. In order to identify a good critical node set, $\tilde{\mathcal{E}}$ is initialized to $\emptyset$ then

the state space relaxation of the pricing problem is solved and all the nodes visited more than once in the optimal path are added to the set $\tilde{\mathcal{E}}$. In our algorithm, this procedure is iterated until either an elementary path with a negative reduced cost is found or the optimal value of the pricing subproblem is nonnegative.

Bidirectional dynamic programming. Our algorithm makes use of a bidirectional extension of labels as described in Righini et al [58]. Two kind of labels are associated with every site: forward labels and backward labels. The forward set is initialized with label $(\emptyset, c_0^{s,k}, 0, r)$ while the backward set with label $(\emptyset, 0, 0, v)$. Extension and dominance phases are then composed by two steps, in which forward and backward labels are treaten independently. The updating rules and feasibility tests for backward extension are symmetrical. Since the consumption of the local resource is monotone along the path, the extension of forward and backward labels can be limited and useless duplication of paths can be reduced, imposing that each partial path can use at most half of the available local resource, therefore with bidirectional extension a state s, associated with either a forward or a backward label, is feasible if $q \leq L/2$.

Join. In order to obtain a complete route from r to v a forward and a backward partial path must be joined together. Each forward path $(\mathcal{W}^{fw}, q^{fw}, C^{fw}, i)$ can be joined with a backward path $(\mathcal{W}^{bw}, q^{bw}, C^{bw}, j)$ if the complete path is elementary and the total consumption of resource does not exceed the amount of available resource. These two conditions can be formally stated as:

$$\mathcal{W}^{fw} \cap \mathcal{W}^{bw} = \varnothing$$
$$q^{fw} + f_{ij} \cdot c_d^{s,k} + q^{bw} \leq L$$

If both conditions are satisfied the path from r to v is feasible and the values $\mathcal{W}, q, C$ of the corresponding label can be computed as follows:

$$\mathcal{W} = \mathcal{W}^{fw} \cup \mathcal{W}^{bw}$$
$$q = q^{fw} + f_{ij} \cdot c_d^{s,k} + q^{bw}$$
$$C = C^{fw} + f_{ij} \cdot \mu^{b,s} \cdot d_d^s + \mu^{b,s} \cdot d_0^s + \sigma^{b,s,k} + C^{bw}$$

The join operation is run when no more feasible extensions can be performed, it generates a complete path $(\mathcal{W}, q, C, v)$ representing a feasible solution for the RCESPP. Among all the paths the minimum cost path after join is an optimal solution of the pricing problem.

Aggregated pricing algorithm. Since in our problem the reduced costs associated with the routes to be generated depend on the chosen base and the employed tactic of the selected strategy, in principle it would be necessary to execute the pricing algorithm for each combination of $b \in \mathcal{B}$, $s \in \mathcal{S}$ and $k \in T^s$. Instead, applying a technique called aggregated pricing described by Bettinelli et al. [81], our algorithm is able to perform a single passage for all the bases reducing the total number of iterations needed to $|S| \times |T^s|$. In order to optimize all the bases at the same time the site state space described by a label needs to be enlarged adding different reduced costs $\mathcal{C}_b$ and resource consumption q_b for every base $b \in \mathcal{B}$. When a forward label is extended from a site i to a site j and $q_b^{fw} > L/2$ for a certain $b \in \mathcal{B}$, the path is not feasible for base b: thus q_b^{fw} is set equal to $+\infty$. For what concerns domination rules, whenever two labels l' and l'' satisfy conditions (3.29), (3.30) and (3.31) for a particular base b, resources q_h'' and C_h'' are set to $+\infty$, as label l'' can never yield an optimal path for base b. A forward label is fathomed when, due either to extensions, feasibility checks or dominance, it has $q_b^{fw} > L$ for every $b \in \mathcal{B}$. This technique is then coupled with bidirectional dynamic programming as described above.

3.4.4 Heuristic Dynamic Programming

A heuristic algorithm for solving the RCESPP is obtained with a small modification from the exact dynamic programming algorithm presented in the previous section. The label structure, extension and join rules are the same as in the exact algorithm but dominance conditions are different. In fact in the heuristic algorithm condition (3.29) is discarded and so given two generic label $l' = (\mathcal{W}', q', C', i)$ and $l'' = (\mathcal{W}'', q'', C'', i)$ the first dominates the second if:

$$q' \leq q''$$
$$C' \leq C''$$

This simplified dominance test allows for the fathoming of several labels and reduces the computational time of the algorithm but at the same time it cannot guarantee optimality.

3.4.5 Tabu Search

Being able to generate a good set of different solutions, that is applying the so-called multiple pricing technique, turned out to be useful in column generation methods as shown by Archetti et al. [67]. We therefore implemented the following algorithm based on the combination of constructive heuristics and tabu search for the RCESPP. It starts from an optimal RMP fractional solution, heuristically identifies good RCESPP solutions and tries to improve them.

Route construction. The route construction phase identifies a subset P of routes whose columns are in the RMP and tries to build a feasible MP solution from them.

Column filtering. The RMP columns are sorted by reduced cost using a priority queue mechanism. Only the $initCols$ columns with the minimum reduced costs are kept.

Mission Strategy Selection. Since the number and the type of routes that can be selected depends on both the bases opened and the strategies employed at first the algorithm evaluates all the $y^{b,s}$ variables. They are sorted in descending order with respect to the value they assumed in the last RMP fractional solution. Then $y^{b,s}$ variables are set to 1 in a greedy way following this order; whenever setting to 1 a particular $y^{b,s}$ variable violates constraints (3.8) or (3.9), such a variable is set to 0. Let $\mathcal{Y}$ be the subset of pairs (b, s) with $b \in \mathcal{B}$ and $s \in \mathcal{S}$ having $y^{b,s} = 1$ found in this way. In an intuitive way the set $\mathcal{Y}$ can be perceived as the set of y variables with maximum LP value that could assume value 1 at the same time in the RMP.

Initial Columns Pool. All the $x_j^{b,s,k}$ with $(b, s) \notin \mathcal{Y}$ are discarded. The remaining ones are sorted in descending order with respect to their value in the last RMP fractional solution, and set to 1 in a greedy way following this order. Whenever setting to 1 one of them violates constraints (3.6) or (3.16), such a variable is set to 0. Let P be the set of routes having the corresponding $x_j^{b,s,k}$ variables to 1 after this step. It is worth noting that all variables added to P have reduced cost equal to zero since they were associated with basic columns in the solution of the computed LP relaxation of the RMP.

Solution feasibility. The pool P of columns found in the previous step does not necessary fulfill constraints (3.5) in fact a sites $i \in \mathcal{E}$ may be covered by more than one

column in P. For this reason the algorithm removes sites from routes encoded by variables in the pool until no site is visited more than once. This deletion is done, without route reoptimization, by removing the site from the route in which such a removal would be the cheapest in term of reduced cost increase.

Solution improvement. At the same time, the routes in P may not cover all the sites; therefore the algorithm tries to insert every node not covered in any of the routes in P following the lexicographic order and using a best insertion strategy. Among all possibilities, if any, for visiting a site extending a route in the pool, the combination of route and position inside the route that maximize the decrement in the reduced cost of the selected variable is chosen. Finally before moving to the tabu serach algorithm the relocation of every single site from a route to any position in any of the other routes of the pool P is taken into account. Once again the best relocation strategy is employed. In this case the goal is similar to the best insertion strategy but the value to minimize is given by the difference between the new reduced cost of the column losing a site and the reduced cost of the variable after adding the sites to its encoded route. At the end of this procedure the set P represents the starting solution for Tabu Search. Moreover, we have implemented a small heuristic algorithm that tries to build out of P a complete feasible solution for the GLRPP. The algorithm makes use of an enlarged set P_{ex} made up of the columns in P together with the skip variables associated with sites that are not visited in any of the routes encoded by variables in P. Then it checks if it is possible to set all variables in P_{ex} to 1 without violating any of the constraints in the MP and any of the decisions imposed by the branching rules (see Section 3.4.8). If this is the case, then P_{ex} is a complete feasible solution for the GLRPP and if its value is better than the current upper bound it becomes the new best solution. Otherwise the solution is discarded.

Tabu Search. Our algorithm is inspired by the approach proposed by Archetti et al. [67]. It works by considering in turn every route in P and trying to improve them through local search by iteratively adding and removing sites to the corresponding route. Given a route $j^{b,s,k} \in P$ its neighborhood is defined by two moves: Insertion and Removal.

Insertion This move takes into account all the sites $i \notin \mathcal{R}_j^{b,s,k}$ and tries to add a site into the route $j^{b,s,k}$ using a best insertion policy. In this case the best insertion is represented by the selection of the best site i and feasible position p between two sites in

the route where adding i causes the maximum decrement in the reduced cost of the route $j^{b,s,k}$. The new route $j^{b,s,k}+i$ is obtained by joining the sites v in position $p-1$ with site i and joining i with site z in position $p+1$. The reduced cost of the new route is computed as follows:

$$\xi_{j^{b,s,k}+i} = \xi_{j^{b,s,k}} - \lambda_i + (f_{v,i} + f_{i,z} - f_{v,z})\mu^{b,s}d_d^s + t_i\mu^{b,s}d_\tau^s.$$

Remove In this move all sites served by $j^{b,s,k}$ are taken into account and among them the site $i \in \mathcal{R}_{j^{b,s,k}}$ that minimizes the increment in the reduced cost of $j^{b,s,k}$ is removed from the route. The new route $j^{b,s,k}-i$ is obtained joining the predecessor v of i with successor z of i. Thus the reduced cost of the new route becomes:

$$\xi_{j^{b,s,k}-i} = \xi_{j^{b,s,k}} + \lambda_i + (f_{v,z} - f_{v,i} - f_{i,z})\mu^{b,s}d_d^s - t_i\mu^{b,s}d_\tau^s.$$

For every column in P a number of iterations equal to $maxIter$ are performed. A single iteration starts with an Insertion move. A Removal operation is then performed only if no feasible insertion was made. Any column with negative reduced cost which is found during an iteration is directly stored into the RMP. The tabu search algorithm makes use of two tabu lists: TL_{insert} and TL_{remove}. When an insertion move adds the site i to a column j, i is inserted in the TL_{remove} making i not removable from route j for a number of iterations equal to $minTabu$. In a similar way when a site i is removed from a route j it is inserted into the TL_{insert} and so it is not re-addable to the column j for $minTabu$ iterations of the algorithm.

During preliminary experiments we found that the following parameters setting gives good results: $initCols = 500$, $maxIter = 12$ and $maxTabu = 4$.

3.4.6 Greedy

In the greedy pricing algorithm a single label $(\mathcal{W}, q, C, i)$ is considered and iteratively extended to a single node with a nearest neighbor policy. Given a base b, a mission strategy s and a routing tactic k the set $\mathcal{V} \subseteq \mathcal{E}$ of the sites reachable from i is computed. A site $j \in \mathcal{E}$ belongs to $\mathcal{V}$ if both the following conditions hold:

$$j \notin \mathcal{W}$$
$$q + b_{jv} \cdot c_d^{s,k} + f_{jv} \cdot c_d^{s,k} < L$$

If $\mathcal{V} = \emptyset$ the path is closed going back to the depot and the search is stopped. Otherwise, among the sites in $\mathcal{V}$, we select the one that minimizes the path cost, that is

$$\bar{j} = argmin_{j \in \mathcal{V}}\{-\lambda_j + \mu^{b,s} \cdot d^s_\tau \cdot t_j + b_{ij} \cdot \mu^{b,s} \cdot d^s_d\}$$

the label is extended to node $\bar{j}$ using the same update rules described for the exact pricing algorithm, and we iterate. This greedy algorithm is repeated for every possible combination of base $b \in \mathcal{B}$, mission strategy $s \in \mathcal{S}$ and routing tactic $k \in \mathcal{T}^s$.

3.4.7 Additional Inequalities

In order to improve the lower bound given by the optimal solution of the MP additional cuts are dynamically generated and inserted into the Master Problem. In particular we have taken into account two families of inequalities: a special case of the subset-row inequalities introduced by Jepsen et al. [64] in which only subsets made up of three rows are considered and consistency cuts (see [82]) customized for the specific problem.

Subset-row inequalities

The idea behind these cuts is the following: for every set of three sites there must be at most a single route visiting at least two of them. More formally, let $\mathcal{G} = \{G \subseteq \mathcal{E} : |G| = 3\}$ be the set made of all possible clusters of three sites. For $G \in \mathcal{G}$ let $J(G) \subseteq J$ the set of all routes, regardless of their starting base, mission strategy and routing tactic, that visit at least two of the three sites belonging to G. Then the following are valid inequalities:

$$\sum_{j \in J(G)} x_j \leq 1 \qquad \forall G \in \mathcal{G} \tag{3.33}$$

Although the number of these inequalities is polynomial ($|\mathcal{E}|^3$), in very large instances, involving hundreds of sites, the computational time required for their separation is not negligible. To speed up the separation process only a subset of the site set $E_s \subseteq \mathcal{E}$ is taken into account for the generation of the cluster set $\mathcal{C}$, in particular a site $i \in \mathcal{E}$ belongs to E_s if the value of its associated skip variable s_i is strictly lower than 1 in the last LP iteration. In the worst case $|\mathcal{E}_s| = |\mathcal{E}|$ but when only a subset of sites in $\mathcal{E}$ can be reached from a base, that is usually the case in planetary space exploration, this method can greatly reduce the number of triplets to be taken into account without weakening the

completeness of the separation procedure. Moreover, a further reduction in the size of C can be achieved by taking into account every possible triplet of nodes in $\mathcal{E}$. Let us consider a triplet $\beta = (a, b, c) : a, b, c \in E_s$, if none of the sites in β is reachable, due to local resource limitations, from any other of the sites in β starting from any base and using any strategy and tactic then this triplet is useless and is not inserted in the set $\mathcal{G}$. It is worth noting that, since the outcome of this second method does not depend on the value of any variable, it can be executed in a preprocessing phase leading to a permanent reduction in the $|\mathcal{G}|$.

The introduction of these additional inequalities causes the alteration of the pricing subproblem and therefore it requires the modification of the pricing algorithms to take into account the values of the corresponding dual variables. In particular, called $\mathcal{P}$ the index set of the valid subset row inequalities added to the MP, the state label used in the exact dynamic programming procedure has to be extended with the introduction of a new variable z_p for every additional cut $p \in \mathcal{P}$ to handle its associated negative dual value ψ_p and the extension, dominance and join rules have to be modified.

In details, given label $l' = (\mathcal{W}', q', z'_p, C', i')$, label $l'' = (\mathcal{W}'', q'', z''_p, C'', i'')$ and defining $\bar{\mathcal{W}}'' \subset \mathcal{W}''$ as the set of all sites visited along the path encoded by label l'' without the inclusion of unreachable sites coming from the application of the technique by Feillet et al. [59], the extension of label l' to l'' is done in the following way:

$$
\begin{aligned}
\mathcal{W}'' =& \mathcal{W}' \cup \{i''\} \\
z''_p =& |G_p \cap \bar{\mathcal{W}}''| \; mod \, 2 \qquad \forall p \in \mathcal{P} \\
q'' =& q' + c_d^{s,k} \cdot f_{i'i''} + c_\tau^{s,k} \cdot t_{i''} \\
C'' =& C' - \lambda^{i''} + \mu^{b,s} \cdot d_\tau^s \cdot t_{i''} + \mu^{b,s} \cdot d_d^s \cdot f_{i'i''} - \sum_{p \in \mathcal{P}} \psi_p \lfloor \frac{|G_p \cap \bar{\mathcal{W}}''|}{2} \rfloor
\end{aligned}
$$

where $i' \neq i''$ and G_p is the subset of $\mathcal{E}$ representing the triplet of sites associated with subset-row inequality $p \in \mathcal{P}$. Dominance rules have to be relaxed since the cost of a path is no longer monotonic and so label l' dominates l'' if:

$$
\begin{aligned}
&\mathcal{W}' \subseteq \mathcal{W}'' \\
&q' \leq q'' \\
&C' - \sum_{p \in \mathcal{V}} \psi_p \leq C''
\end{aligned}
$$

with $i' = i''$ and $\mathcal{V} = \{v \in \mathcal{P} : \psi_v \leq 0, z'_v > z''_v\}$. Finally the values of $z_p \, \forall p \in P$ have to be taken into account when two partial paths are joined together to compose a single complete route and in particular we check if $z'_p = 1 \wedge z'_p = 1 \, \forall p \in \mathcal{P}$ the value ψ_p is subtracted from the cost of the route.

Consistency Cuts

Unlike the subset-row inequalities, which focus only on the routing part of the problem, these inequalities deal with the very location-routing nature of the problem enforcing the consistency between every route and its associated base. They come from the observation that $\forall b \in \mathcal{B}, \forall s \in \mathcal{S}, \forall k \in \mathcal{T}^s$ the value of every variable $x_j^{b,s,k}$ with $j \in \mathcal{J}^{b,s,k}$ can never exceed the value of its associated location variable that is $y^{b,s}$. This observation leads to the formalization of following constraints:

$$x_j^{b,s,k} \leq y^{b,s} \qquad \forall b \in \mathcal{B}, \forall s \in \mathcal{S}, \forall k \in \mathcal{T}^s, \forall j \in \mathcal{J}^{b,s,k} \tag{3.34}$$

These additional inequalities are not in the original formulation of the GLRPP in fact they are not necessary to describe a feasible integer solution. On the other hand they are able to cut out a lot of fractional solutions that are feasible with respect to the constraints in the MP. In particular, focusing on constraints (3.7) and their improved version (3.16) it is easy to see that there are solutions, as the one described in Figure 3.2, which are feasible for these constraints but become infeasible if the constraints (3.34) are introduced. It is worth noting that the consistency cuts cannot be used as replacement for the constraints (3.16) since they do not carry any information on the maximum number of routes that can be performed using a particular tactic k of a mission strategy s. Since there is one consistency cut for every variable these inequalities are dynamically generated and added to the problem every time a new column enters into the RMP. The introduction of these constraints has an impact on the pricer subproblem, in fact the values of dual variables associated with constraints (3.34) have to be taken into account during the route generation. The approach followed in order to deal with these additional dual variables requires the modification of the pricing algorithms and in particular of the dominance and join rules described in the Section 3.4.3.

Let $\eta_j \quad \forall b \in \mathcal{B}, \forall s \in \mathcal{S}, \forall k \in \mathcal{T}^s, \forall j \in \mathcal{J}^{b,s,k}$ be the dual variables associated with the consistency cuts. When checking if a label l'' is dominated by a label l', we must take into

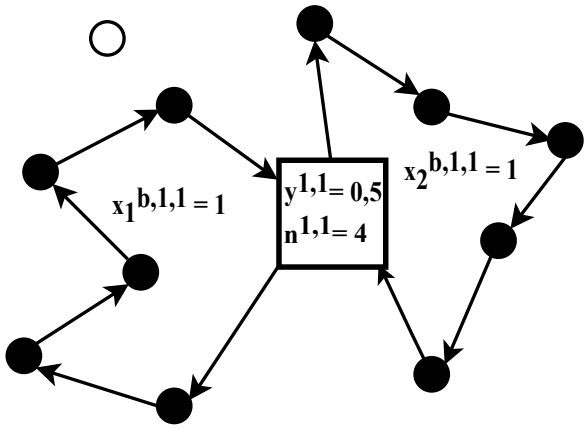

Figure 3.2: Consistency cuts: a solution, feasible for the MP, that would be discarded using the custom consistency cuts

account the maximum value η_{max} among the dual variables associated with consistency cuts involving $x_j^{b,s,k}$ variables that encode a route which is either equal to the partial path described by the label l' or can be built by repeated extension of this partial path. In particular $l' = (\mathcal{W}', q', C', i)$ dominates $l'' = (\mathcal{W}'', q'', C'', i)$ if the following conditions hold:

$$\mathcal{W}' \subseteq \mathcal{W}'' \tag{3.35}$$

$$q' \leq q'' \tag{3.36}$$

$$C' + \eta_{max} \leq C'' \tag{3.37}$$

The computation of η_{max} may be very time expensive and so, in order to minimize the impact of the consistency cuts on the global performance of our algorithm, we have implemented a custom data-structure (whose detailed description is reported in Appendix B) similar, in principle, to a search tree. In the join phase we have to check if the complete path described by the union of l' and l'' represents a route encoded by a variable $x_j^{b,s,k}$ that belongs to a consistency cut, in that case the associated dual variable has to be added to the cost of the complete path. In this case the path generated by $l' \cup l''$ can be discarded, in fact, since every variable $x_j^{b,s,k}$ involved in a consistency cut is in the RMP, it represents a route encoded by a variable already in the problem. Despite the fact that the idea behind these cuts is not entirely new their implementation in a branch-and-cut-and-price algorithm, to the best of our knowledge, has never been attempted before. The introduction

of consistency cuts brings in a structural modification of the general branch-and-cut-and-price framework as presented in Section 1.3. In fact, when consistency cuts are employed, in the pricing stage both variables and inequalities are generated contradicting the original branch-and-cut-and-price scheme in which additional inequalities are only found in the cutting stage. In particular, in our implementation, a consistency cut is generated and inserted in the RMP every time a new variable with negative reduced cost is found. Moreover, it is worth noting that, although in this section we only discussed the application of consistency cuts to the GLRPP, this same method can be followed to introduce such constraints in any column-generation-based algorithm for location and routing problem with a similar structure.

3.4.8 Branching

In oder to recover the integrality when a fractional solution arises at the end of the root node five different branching strategies have been designed. They are considered in the order presented below: branching is executed using the first applicable rule. In the following we indicate as $\bar{y}$, $\bar{x}$ and $\bar{s}$ the values taken by the variables in the optimal solution of MP.

Branching on Strategies. Among all $y^{b,s}$ variables the fractional variable $y^{b',s'}$ whose value $\bar{y}^{b',s'}$ is closest to 0.5 is selected. Then a binary branching is performed creating two nodes: in the former it is imposed that $y^{b',s'} = 1$ and all $y^{b,s} = 0 \quad \forall (b,s) \neq (b',s')$ in the latter child node $y^{b',s'}$ takes value 0. This branching rule does not require to modify the branching subproblem and the associated algorithms, in fact all y variables are already in the RMP and are not dynamically found via column generation.

Branching on Skip Sites. If all $\bar{y}^{b,s}$ values are integer $\forall b \in \mathcal{B}, s \in \mathcal{S}$ a second branching rule is taken into account. Let s_i be the variable whose value is fractional and closest to 1. The branching is performed in the following way: two child nodes are generated and in the first the visit of the site i is forbidden imposing $s_i = 1$ on the contrary in the other node $s_i = 0$ and therefore the site i must be visited.

Branching on Number of Routes. When all s_i are integer $\forall i \in E$ a third branching policy is used. Let $m^{b,s,k}$ the number of routes starting from base $b \in \mathcal{B}$, using routing tactic $k \in \mathcal{T}^s$ of mission strategy $s \in \mathcal{S}$. The value $m^{b,s,k}$ is computing as the sum of the values

taken by all the $j \in \mathcal{J}^{b,s,k}$ in the optimal MP fractional solution that is $\sum_{j \in \mathcal{J}^{b,s,k}} \bar{x}_j^{b,s,k}$. Then computed all $m^{b,s,k}$ for every combination of $b \in \mathcal{B}$, $s \in \mathcal{S}$, $k \in \mathcal{T}^s$ a binary branching is executed on the $m^{b,s,k}$ closest to 0.5. Two nodes are generated, in the first it is imposed to use at least $m^{b,s,k}+1$ variables belonging to $\mathcal{J}^{b,s,k}$ while in the other the maximum number of routes starting from base $b \in \mathcal{B}$, using routing tactics $k \in \mathcal{T}^s$ of mission strategy $s \in \mathcal{S}$ is limited to $n^{b,s,k}$. This result is obtained modifying the left and right hand side of the constraint (3.7) corresponding to the selected b, s and k, in particular in the first case the modified constraint becomes as follows:

$$m^{b,s,k} \leq \sum_{j \in \mathcal{J}^{b,s,k}} x_j^{b,s,k} \leq n^{s,k} y^{b,s}$$

in the other node instead the constraint is

$$\sum_{j \in \mathcal{J}^{b,s,k}} x_j^{b,s,k} \leq m^{b,s,k} y^{b,s}.$$

This branching technique leaves the pricing subproblem unchanged: dual variables $\sigma^{b,s,k}$ still appear as constants in the objective function of the pricing problem, but they are now unrestricted in sign. No modification in the pricing algorithms is required to handle this branching technique.

Branching on Arcs. When the number of routes used in the solution is not fractional branching on an arc value is considered. We choose the site $i \in \mathcal{E}$ that is split among the largest number of routes in the optimal fractional solution of MP and that has at least two open outgoing arcs. Then we forbid half of its outgoing arcs to be used in the first child node, and the other half to be used in the second child node. To handle these branching decisions in the pricing problem it is enough to set travel time of forbidden arcs to $+\infty$.

Branching on Routes Finally when none of the previous rules can be applied branching on variables encoding routes is performed. Among all x variables in the RMP let $x_{\bar{i}}$ be the one whose value, in the optimal LP solution of the MP, is fractional and closest to $0,5$. The branching is performed in the following way: two children nodes are generated. In the former the usage of the route $x_{\bar{i}}$ is forced setting $x_{\bar{i}} = 1$, while in the latter the route $x_{\bar{i}}$ is forbidden imposing $x_{\bar{i}} = 0$.

Branching Tree Exploration Strategy. The iterative application of the four previously described branching rules produces a branching tree that needs to be visited in order to find an optimal integer solution. The exploration of the search tree is performed using a strategy mixing depth first and best bound first search. In particular the first two branching rules assign a higher priority at the first child that, consequently, is always visited before the second one; the remaining two rules assign equal priority to the children, and therefore they are visited in best bound order.

3.4.9 Primal Heuristic Algorithm

Although the variables in the restricted master problem does not describe the complete solution space, an appropriate subset of them may represent a feasible integer solution for the original problem. In order to quickly find good primal solutions a simple greedy algorithm is executed once for every node of the search tree. This procedure consists of three phases, in the first phase to every base the most promising strategy is assigned then for every opened base a set of routes complying with the employed strategy is selected and finally the last phase tries to ensue the feasibility of the solution. In details:

Mission Strategy Selection. In the first phase we build the set $\mathcal{Y}$ of pairs (b, s) representing open bases and their corresponding mission strategies, exactly as described in Subsection 3.4.5. At the end of this phase all the $x_j^{b,s,k}$ with $(b, s) \notin \mathcal{Y}$ are discarded.

Route Selection. In the second phase we iteratively compute a potential profit $\bar{p}_j^{b,s,k} = \sum_{i \in \mathcal{R}_j} p_i$ for all remaining $x_j^{b,s,k}$ variables, and, in a greedy way according to their potential profit, we set to 1 the variable having maximum $\bar{p}_j^{b,s,k}$ which does not yield a violation of constraints (3.5), (3.6) and (3.16), until no more variables can be set to 1 without making the solution infeasible.

Solution feasibility. In the last phase, in order to achieve a feasible solution with respect to constraint (3.5), all skip variables s_i corresponding to sites not covered by any of the routes selected in second phase are set to 1. It is worth noting that this procedure could produce solutions that, in some node of the search tree, do not match branching decisions. In this case the algorithm fails.

3.5 Experimental Analysis

Implementation. The algorithms have been implemented in C++, using SCIP 2.1 [69] as branch-and-cut-and-price framework linked to CPLEX 12.2 as a pure LP solver. SCIP takes care of the management of the column and row pools and it is able to remove and re-insert them as needed. All presolving algorithms and the automatic cut generator embedded in SCIP along with the following general purpose heuristic algorithms: shift-and-propagate, fix-and-infer, int-diving and shifting have been disabled as they were considered either incompatible or useless for our problem. All remaining SCIP parameters were kept at their default values including the branching tree exploration strategies.

The experimental campaign has been performed on a single core of a PC Intel® Core 2 Duo™ CPU T7300 (2.00 GHz) with 4 GB RAM and running Ubuntu 10.04 operating system. A time limit of 1 hour was imposed to each run.

Benchmark Instances. For testing purpose we have generated 216 GLRPP instances taking the case study presented by Ahn et al. [42]. The planetary surface considered in our instances is always included in a 3400$[Km]$ radius that is roughly the equatorial radius of Mars, and in this surface are located, randomly using an uniform distribution, a number of potential bases $|\mathcal{B}|$ between 10 and 100 and a number of sites $|\mathcal{E}|$ from 100 to 1000. Our instances take into account two mission strategies ($\mathcal{S} = \{1, 2\}$): a standard strategy (Std., $s = 1$) and an orbiting depot strategy (Orb.Dpt., $s = 2$). Only one routing tactic, called standard tactic (Std.R.T., $k = 1$), is associated with the standard strategy, while two tactics, namely standard ($k = 1$) and depot-assisted (Dpt.Ass.R.T., $k = 2$) are associated with the orbiting depot strategy. All the parameters associated with these mission strategies and routing tactics are reported in Table 3.5. For each combination of $|\mathcal{B}|$ and $|\mathcal{E}|$, six instances are created, in which the budget is fixed to $\max_{s\in\mathcal{S}}\{C^s\} \cdot |\mathcal{B}| \cdot g/5$ for $g \in \{0.5, 1, 2, 3, 4, 5\}$.

3.5.1 Lower Bound

In order to evaluate the quality, in term of both efficiency and efficacy, of the additional inequalities (see section 3.4.7) we have tested four different configurations of our algorithm:

- No Cuts (NC): no additional inequalities are taken into account in this configuration.
- Subset-row Inequalities (SI): only the subset-row inequalities are used in this configuration.

	Strategy	
	Standard	**Orbiting Depot**
Collective Constraint		
Resource	Exploration Time	Exploration Time
Coefficients	$d_0^1 = 0$ [hr]	$d_0^2 = 0$ [hr]
	$d_d^1 = 0.2$ [hr/km]	$d_d^2 = 0.2$ [hr/km]
	$d_\tau^1 = 3$ [hr/hr]	$d_\tau^2 = 3$ [hr/hr]
Limit	$l_c^1 = 2100$ [hr]	$l_c^1 = 2100$ [hr]
Local Constraint		
Routing Tactics I		
	Standard	Standard
Single Route Constraint		
Resource	Fuel	Fuel
Coefficients	$c_0^{1,1} = 0$ [kg]	$c_0^{2,1} = 0$ [kg]
	$c_d^{1,1} = 1.1$ [kg/km]	$c_d^{2,1} = 1.1$ [kg/km]
	$c_\tau^{1,1} = 5.7$ [kg/hr]	$c_\tau^{2,1} = 5.7$ [kg/hr]
Limit	$l_r^{1,1} = 677$ [kg]	$l_r^{2,1} = 677$ [kg]
Max. Number of Routes	$n^{1,1} = \infty$	$n^{2,1} = \infty$
Routing Tactics II		
		Depot-Assisted
Single Route Constraint		
Resource		Fuel
Coefficients		$c_0^{2,2} = 0$ [kg]
		$c_d^{2,2} = 1.1$ [kg/km]
		$c_\tau^{2,2} = 5.7$ [kg/hr]
Limit		$l_r^{2,2} = 1286$ [kg]
Max. Number of Routes		$n^{2,2} = 4$
Strategy Cost	$C^1 = 54$ [MT]	$C^2 = 64$ [MT]

Table 3.1: Characteristics of Instances.

- Consistency Cuts (CC): in this configuration only the consistency cuts are taken into account.
- All Cuts (AC): both subset-row inequalities and consistency cuts are considered in this configuration.

The algorithm has been executed on all the instance in the test set stopping the computation at the root node, in this way we are able to easily compare the values of the lower bounds and the computational effort required to handle the additional inequalities. In Tables 3.2, 3.3 and 3.4 we present a summary of the comparison between the quality of the dual bound computed with all the four different settings of the cut generator. In particular, the Table 3.2 shows the average, maximum and minimum percentage improvement in the value of the lower bound, going from any configuration to any other. Table 3.3 contains the average, maximum and minimum number of additional inequalities added to the RMP using SI, CC and AC, it is worth noting that every time a new inequality is found the problem is re-optimized. Finally, the computational times, average, maximum and minimum used by every configuration of the cut generator, are reported in Table 3.4.

Our algorithm, thanks to the quickness of the pricing procedures, was able to compute valid lower bounds for every instance in less than 200 seconds. From these results it is clearly visible that the consistency cuts shows the best ratio between improvement and computational effort while the subset row inequalities are only marginally useful providing a maximum of 1.84% increase in the value of the lower bound. It is worth noting that, as reported in Table 3.4, the CC configuration is often faster than the NC, in fact with the addition of the consistency cuts the route node can be solved with less LP iterations and, consequently, it requires to solve less pricing subproblems. Combining both additional inequalities together produces a configuration (AC) that on the one hand is able to find the best lower bound in all instances but one. On the other hand this setting is the most time expensive raising the computational time by more than one and an half time in respect to the NC configuration and requiring almost twice the time spent by the SI setting. Analyzing the efficacy of the cuts, with respect to the structure of the instance, we found out that the consistency cuts are more useful when the problem has a limited budget ($g \in \{0.5, 1, 2\}$) while the subset-row inequalities can give a noticeable contribution in the instance with $g \geq 4$. This behavior does not come unexpected. In fact, when only a limited budget is available the location nature of the problem prevails on the routing part accentuating

the importance of the consistency cuts. In such instances, indeed, to maximize the profit collected it can be useful to partially use several bases assigning, in a fractional way, a strategy to a base ($y^{b,s} \in (0,1)$) in order to reach the majority of the most profitable clients. The consistency cuts explicitly forbid such a behavior and, consequently, give a fundamental contribution in raising the value of the lower bound. On the contrary with a big budget a lot of the potential bases (all if $g = 5$) can be opened ($y^{b,s} = 1$) shifting the focus of the problem from the location to the routing part, in this case the contribution coming form the subset row inequalities can be hard to compensate using only consistency cuts. Moreover, analyzing the structure of the routes used in the feasible solutions we have seen that they are usually very short: almost half of the routes represents one-site route and the others, on average, visit from two to five sites before coming back to the starting base. This observation helps in explaining the weakness of the subset row inequalities, in fact all the single-site routes are not taken into account by these cuts.

Imp. %	**Configurations** (From-To)					
	NC-SI	NC-CC	NC-AC	SI-CC	SI-AC	CC-AC
Avg	0.07	6.45	6.51	6.38	6.44	0.06
Max	1.84	150.49	150.49	150.49	150.49	1.08
Min	0.00	0.00	0.00	-0.46	-0.20	0.00

Table 3.2: Quality of the Lower Bound - Root Node

Time [s]	**Configurations**		
	SI	CC	AC
Avg	1.50	736.69	738.57
Max	14.00	4818.00	4821.00
Min	0.00	37.00	37.00

Table 3.3: Number of Additional Inequalities - Root Node

Among the four configurations proposed two, NC and SI, seem to be pretty useless, in fact there is no point in not using the consistency cuts if they provide better results without raising the computational time. On the other hand the contribution of the subset row inequalities is, with very few exceptions, so small to hardly justify the computational effort paid. Although the best lower bounds are, obviously, found using the AC setting, a hard price is paid in term of computational time: this configuration requires almost three

Time [s]	**Configurations**			
	NC	SI	CC	AC
Avg	1.40	2.80	1.37	5.44
Max	33.22	83.50	27.18	188.53
Min	0.01	0.01	0.01	0.01

Table 3.4: Computational Time - Root Node

times the computational times employed by the CC configuration. We have then observed that the best results for this problem can be found using a mixed configuration in which AC is employed only at the route node and then, if the solution is fractional, we switch to CC for solving the other nodes in the branching tree. Taking into consideration the results presented in this section in the next one we analyze the quality of the primal solutions.

3.5.2 Primal Heuristic Algorithm Evaluation

To evaluate the performances of our primal procedure we have followed a similar way as in the analysis of the lower bound quality. For every configuration of the cut generator taken into consideration in the previous section, all instances were solved using three different primal politics: only with SCIP heuristics (SH) see Achterberg [69] and the online documentation [83], only with our greedy algorithm (GH), see section 3.4.9, and with both (BH). The computation was stopped at the end of the route node, that is after the first run of the procedure presented in Section 3.4.9. This allows for a fair comparison between the result achieved by the SCIP heuristics with the solutions found by our primal heuristic algorithm. Moreover, since all primal heuristics taken into account work only within the space described by the variables in the RMP, we can evaluate if the introduction of additional cuts has any impact on the primal solutions found.

In Table 3.5 a comparison between any couple of primal politics is reported, in particular the table reports the average, maximum and minimum improvement in the quality of the primal bound found at the route node for every configuration of the cut generator. The comparison between SH and GH shows that GH is able to find the best feasible solution on more than 90% of the instances and that, on average, the value of these solutions is better by 3% than the ones found by SCIP. Combining our procedures with the SCIP heuristics does not raise too much the quality of the solutions and in fact the improvement is, on average by less than 1%.

Imp. % (From-To)		**Configurations** NC	SI	CC	AC
SH-GH	Avg	3.99	3.04	2.67	2.58
	Max	228.43	230.57	208.43	208.43
	Min	-0.29	-0.29	-7.06	-7.06
SH-BH	Avg	3.99	3.05	2.76	2.69
	Max	228.43	232.19	208.43	208.43
	Min	0.00	0.00	0.00	0.00
GH-BH	Avg	0.00	0.01	0.09	0.11
	Max	0.29	0.54	7.59	7.59
	Min	0.00	0.00	0.00	0.00

Table 3.5: Primal Heuristics Comparison

It is worth noting that different configurations of the cut generator can lead to different primal solutions at the end of the root node as reported in Table 3.6. This table displays for every primal politic the variation in the quality of the solution found going from any configuration of the cut generator to any other. On average, it can be said that using configurations with stronger cuts leads to better primal solutions but it must also be noted that there are exceptions, in fact a few instances shows a different behavior in which the introduction of additional cuts has a negative impact on the performance of any of the primal heuristics employed.

Primal Politics		**Imp. %** (From Config. - To Config.) NC-SI	NC-CC	NC-AC	SI-CC	SI-AC	CC-AC
SH	Avg	0.90	2.92	3.04	2.10	2.22	0.12
	Max	61.65	211.33	211.33	211.33	211.33	6.19
	Min	0.00	-61.59	-61.59	-61.59	-61.59	0.00
GH	Avg	-0.01	1.07	1.10	1.09	1.12	0.03
	Max	4.65	34.72	34.72	34.72	34.72	5.68
	Min	-4.09	-9.38	-12.39	-9.38	-12.39	-7.34
BH	Avg	-0.01	1.16	1.21	1.17	1.22	0.05
	Max	4.65	34.72	34.72	34.72	34.72	5.68
	Min	-4.09	-9.38	-12.39	-9.38	-12.39	-7.34

Table 3.6: Primal solution quality with different cut generation politics

It it evident from this analysis that our custom algorithm is fairly better than any of the SCIP heuristics. This claim is even more true if we consider that the computational time

spent by a single iteration of our greedy algorithm is, on average, lower than a tenth of a second and it is never higher than half a second and that is, roughly, the same amount of time spent in a single iteration of one of the most time consuming heuristics embedded in SCIP. Anyway, considering the quickness of these algorithms the BH politic assures to find always the better primal solution with only a small additional effort. This can be useful when a fractional solution arises at the end of the root node, in fact improving the primal bound allows for the fathoming of more nodes of the branching tree.

Given the results of this and the previous section we identified the combination of AC cut generator and BH primal politic as the most promising setting in term of quality of the solution achieved at the end of the route node and CC + BH as the most useful setting during the exploration of the possible branching tree. For this reason, in the reminder, we focus our attention on the efficiency of the pricing algorithm and on the analysis of the overall performance of our branch-and-price-and-cut algorithm using only these settings.

3.5.3 Pricing Algorithms

Tables 3.7 and 3.8 reports, respectively, a measure of the efficiency and efficacy of our four pricer algorithms (see Sections 3.4.3, 3.4.4, 3.4.5, and 3.4.6). As is clearly visible in Table 3.7 all pricing algorithms are pretty fast, the average computational time is in the order of magnitude of tenths of seconds while the maximum time for a single iteration is little above half a second. Among those algorithms the tabu search is the most time expensive requiring always at least as much time as the exact dynamic programming procedure. However, the extraordinary quickness of the dynamic programming algorithms is mainly to be credited, as already noted in Section 3.5.1, to the very strong limitations that the problem imposes on the length of the routes; this, combined with the sophistication of our dynamic programming approach, leads to procedures that can be executed very quickly.

		Pricers			
		Greedy	Tabu	Heur. DP	Exact DP
Sec./Iter.	Avg	0.01	0.05	0.02	0.03
	Max	0.21	0.73	0.29	0.46
	Min	0.00	0.00	0.00	0.00

Table 3.7: Pricer Performances: Time Single Iterations

From the efficacy point of view Table 3.8 reports, for a single iteration of every pricing

algorithm, the average and maximum number of variables added to the RMP. The tabu search procedure turns out to be responsible for the introduction in the RMP of the majority of the columns and in particular, analyzing the structure of the solutions, it must be said that the variables found by the tabu serach algorithm compose a large portion of the best solutions the branch-and-cut-and-price algorithm is able to achieve within the time limit. As to the other pricers, the greedy algorithm is mainly used in the first iterations when the dual values show a very high variability and is able to add a lot of good variables that help in stabilizing the value of the dual variables. Finally the dynamic programming procedures are employed in the last iterations and introduce in the MP only few columns which are needed to close the residual dual gap and to prove the optimality of the current primal solution.

		Pricers			
		Greedy	Tabu	Heur. DP	Exact DP
N Sol./Iter.	Avg	0.41	1.28	0.10	0.01
	Max	4.36	19.00	1.00	0.12

Table 3.8: Pricer Performances: Number of Solutions per Iteration

3.5.4 Overall Performance

An overview of the general computational results is presented in two tables. On the former Table 3.9 we reported results aggregated by sites; each row contains average results over instances having same $|\mathcal{E}|$ and same g. In the latter, Table 3.10, the results are aggregated by bases leading to rows showing average results over instances with the same $|\mathcal{B}|$ and g. In both tables the columns have the same meaning: in the first column (Inst.) we indicate instance type in the format $S|\mathcal{E}|.g$ for Table 3.9 or $B|\mathcal{B}|.g$ for Table 3.10; then five columns report the duality gap at the root node (Gap R.), the duality gap at the end of the computation (Gap F.), the number of instances solved to proven optimality (Opt.), the number of branch-and-bound nodes explored (B.B.N.) and the CPU time spent. Moreover, in both table the rows marked with "AVG" contain the total number of instances solved and the average results for instances with respectively the same number of sites and bases. Detailed results for every single instance are reported in Appendix D. As expected the higher is the number of bases and sites involved in an instance the more difficult is its solution. However, bases and sites have a different impact on the performances of our algorithm. In particular the number

of bases seems to have a greater influence on the Master Problem, while increasing the number of sites leads to more difficult pricing subproblems. This is clearly visible looking at the B.B.N. and GapR % columns, they show that on the one hand raising the number of sites does not impact too much the quality of the solution at the end of the root node, but decrease the number of nodes visited in the search three since every node requires to solve harder pricing subproblems. On the other hand, the more the number of bases is increased the larger becomes the gap at the root node, but this seems not to impact the performance of the pricing algorithms, in fact we are able to explore larger and larger branching trees.

Moreover, as reported in Table 3.11 that shows average results aggregated by budget, besides instance size, the budget has a huge and distinct influence on the difficulty of the instances. The trend obtaining varying the budget looks like a parabola where the lowest and the higher budgets leads to easier instances while the hardest ones are associated with budget g equal to either 2 or 3. In particular when $g = 5$ the problems become much easier. This behavior is easily explainable, in fact with the maximum budget all bases can be opened and can employ the most resourceful strategy making the location part of the problem extremely simple. Similar behavior is shown when the budget is so small that only a few bases can be actually opened using the most limited tactics, in this case the routing subproblems become easier requiring a little computational effort to be optimally solved. On the contrary when the budget allows for using almost half of the possible bases, the problems becomes much more difficult. In this case both the location and the routing parts of the problem assume the same importance leading to almost always fractional dual bound and requiring to branch several times before the optimal integer solution is found.

3.6 Conclusions

In this chapter we have discussed a very general logistics problem called the Generalized Logistic and Routing Problem with Profit and its application in the context of planetary surface explorations. We presented a mathematical model, compatible with our general location-routing framework, that introduces the use of skip variables and we devised an exact algorithm, based on the branch-and-price-and-cut paradigm that represents the first attempt to solve the GLRPP to proven optimality.

Within our framework, we have implemented four different pricing procedures that are based on the revision and extension of established techniques, furthermore two additional

Inst.	GapR.%	GapF.%	Opt.	B.B.N.	Time [s]
S100-0.5	17.23	0	6	41572	502.23
S100-1	29.72	2.16	5	45239.83	607.26
S100-2	15.98	0	6	1008.83	6.13
S100-3	9.01	0	6	164.5	1.20
S100-4	0.49	0	6	3.5	0.06
S100-5	0	0	6	1	0.06
AVG	12.07	0.36	35	14664.94	186.16
S200-0.5	12.21	1.84	5	16867.5	650.14
S200-1	26.81	7.97	4	38238.5	1292.89
S200-2	32.37	9.54	3	65065.5	2001.83
S200-3	12.69	0.38	5	28804.33	770.80
S200-4	1.26	0	6	30.5	0.62
S200-5	0	0	6	1	0.11
AVG	14.22	3.29	29	24834.56	786.06
S300-0.5	11.03	2.97	5	19873.33	1009.54
S300-1	19.17	7.33	4	37091	1833.01
S300-2	32.91	9.23	2	39632.67	2402.18
S300-3	10	0.67	5	17102	1114.04
S300-4	2.52	0	6	126.83	7.06
S300-5	0	0	6	1	0.25
AVG	12.61	3.37	28	18971.14	1061.01
S400-0.5	9.67	2.32	5	6075.17	760.39
S400-1	15.66	6.1	3	27795.17	2336.31
S400-2	30.48	13.98	2	19912	2447.49
S400-3	13.4	2.28	2	18188.33	2404.07
S400-4	4.19	0.08	5	5690.83	822.31
S400-5	0.08	0	6	32.67	4.48
AVG	12.25	4.13	23	12949.03	1462.51
S500-0.5	8.51	2.52	4	10087	1244.85
S500-1	13.68	6.39	3	15539.33	1905.51
S500-2	25.65	11.86	3	15146.17	2415.11
S500-3	13.89	1.97	3	13145	2462.40
S500-4	5.23	0	6	2882	694.05
S500-5	0.04	0	6	5.67	1.75
AVG	11.17	3.79	25	9467.53	1453.94
S1000-0.5	4.99	2.03	3	5112.83	1989.35
S1000-1	8.72	5.33	2	4565.83	2468.92
S1000-2	15.47	11.42	0	8787.5	3600.24
S1000-3	18.61	13.44	1	6242.17	3598.76
S1000-4	13.1	3.87	0	5353.5	3600.55
S1000-5	2.77	0.31	0	4638.33	3602.23
AVG	10.61	6.07	6	5783.36	3143.34

Table 3.9: Computational Results - Aggregation over Sites

Inst.	GapR.%	GapF.%	Opt.	B.B.N.	Time [s]
B10-0.5	2.30	0.00	6	28.00	1.65
B10-1	2.99	0.00	6	50.67	3.47
B10-2	3.14	0.01	5	4339.50	600.96
B10-3	2.87	0.01	5	2711.67	600.90
B10-4	0.79	0.01	5	2281.00	600.52
B10-5	0.02	0.01	5	1957.83	600.08
AVG	2.02	0.01	32	1894.78	401.26
B20-0.5	4.29	0.00	6	120.33	7.12
B20-1	5.40	-0.03	6	704.17	80.46
B20-2	6.67	0.05	5	4685.00	767.47
B20-3	5.71	0.00	6	2954.83	727.64
B20-4	2.67	0.00	5	1123.17	609.65
B20-5	0.02	0.02	5	916.00	600.17
AVG	4.13	0.01	33	1750.58	465.42
B30-0.5	7.00	0.00	6	1634.33	202.35
B30-1	10.62	0.21	5	21042.17	1375.53
B30-2	14.03	1.45	3	39971.50	2502.45
B30-3	12.39	0.62	4	17945.83	2057.86
B30-4	4.58	0.23	5	1084.00	644.54
B30-5	0.29	0.05	5	724.83	600.27
AVG	8.15	0.43	28	13733.78	1230.50
B40-0.5	8.78	0.01	5	5242.00	699.72
B40-0.1	11.38	0.75	3	37107.83	2378.63
B40-2	15.34	3.34	1	43429.00	3000.37
B40-3	12.81	2.15	3	17597.50	2160.87
B40-4	4.39	0.40	5	2669.33	890.80
B40-5	0.32	0.13	5	702.83	600.96
AVG	8.84	1.13	22	17791.42	1621.89
B50-0.5	11.72	0.49	4	24910.50	1746.58
B50-1	18.74	5.10	1	44206.00	3005.75
B50-2	26.95	9.20	1	33217.83	3001.55
B50-3	21.12	3.81	1	36761.33	3000.40
B50-4	5.25	0.68	5	2790.50	1177.03
B50-5	0.68	0.01	5	210.83	600.92
AVG	14.08	3.22	17	23682.83	2088.70
B100-0.5	29.55	11.19	1	67652.67	3499.10
B100-1	64.63	29.27	0	65358.83	3600.06
B100-2	86.73	41.97	1	23909.83	3000.20
B100-3	22.70	12.14	3	5675.17	1803.60
B100-4	9.11	2.62	4	4139.17	1202.10
B100-5	1.56	0.10	5	167.33	606.48
AVG	35.71	16.22	14	27817.17	2285.26

Table 3.10: Computational Results - Aggregation over Bases

Inst.	GapR.%	GapF.%	Opt.	B.B.N.	Time [s]
U0.5	10.61	1.95	28	16597.97	1026.08
U1	18.96	5.88	21	28078.28	1740.65
U2	25.48	9.34	16	24925.44	2145.50
U3	12.93	3.12	22	13941.06	1725.21
U4	4.47	0.66	29	2347.86	845.10
U5	0.48	0.05	30	779.94	601.48

Table 3.11: Computational Results - Aggregation over Budget

inequalities are used to improve the lower bound. One of them has been especially designed to address the complex location-routing nature of the problem and is new in the context of column generation based algorithms.

Our algorithm has been tested on a set of 216 containing from 10 to 100 bases and a number of sites ranging from 100 to 1000. Our approach turned out to be successful in tackling GLRPP instances of real size, finding proven lower and upper bounds for any of them, and represents a solid starting point for further researches. Indeed the framework revealed itself flexible enough to easily incorporate new requirements and constraints. Moreover we demonstrate the usefulness and the tractability of the consistency cuts concluding that they are a viable way for improving the dual bound in LRP.

Chapter 4

Drug Distribution in Case of Emergency

4.1 Introduction

Mathematical programming models and algorithms have been successfully used for decades to optimize operations in distribution logistics: typical examples concern freight carriers, mail services and on-demand pick-up and delivery services.

A more recent field of investigation concerns the application of similar techniques to the optimization of logistics operations in health care systems and emergency management. These sectors are characterized by a larger dependency on "human factors", such as the behavior of the customers (which is often unpredictable), fairness in service provision (which is not an issue in industrial logistics) and lack of reliable historical data (because of the uniqueness of the events considered, especially in case of emergency management). In particular in this chapter we discuss the application of operations research to the distribution of drugs in case of emergency. Such situations are always hardly foreseeable and usually almost impossible to avoid. In these situations only a posteriori actions are possible and indeed the best way to handle an health emergency event, such as the release of anthrax spores, is to set up a very efficient response system that, if something happens, is able to provide all the required services in the most effective way. The optimization in this case is such an essential component of a good response system that the U.S Centers for Diseases Control and Prevention in 2007 evaluated "the timeliness of distributing antibiotics to the general public as an effective measure against a release of anthrax" [84].

The starting point for our study is a paper by Shen et al. [43], who presented a stochastic VRP model for large-scale bioterrorism emergency which is then reformulated and solved as a deterministic VRP with a tabu search algorithm. The original problem requires to reach the maximum number of citizens within a specified time limit using an heterogeneous fleet of vehicles. We expanded the problem adding a location component and then including an additional distribution strategy in order to model more realistic situations in which multiple distribution methodologies can be taken into account at the same time and mixed together to maximize the fraction of the population served. In particular, we explore the option of reaching citizens in two ways: by establishing depots, and delivering the drugs at home using a heterogeneous fleet of vehicles, or by establishing distribution centers where the citizens go by their own means to receive treatments or drugs.

In this way a particular combined location-routing problem arises. Even if location-routing is a lively research area [38], to the best of our knowledge such kind of problem has never been addressed before. We present an exact algorithm for such a problem, which is based on column and cut generation and branch-and-bound, and where the pricing sub-problems are solved through advanced dynamic programming techniques.

This chapter is organized as follows: in the next Section 4.2 we describe the problem pointing out all features and requirements. The mathematical model of the problem is given in Section 4.3 and then the solution approach is carefully reported in Section 4.4. In the last two sections 4.5 and 4.6 the results of our test campaign and our conclusions are reported.

4.2 Problem Description

The problem we address in this chapter fits into the following scenario: in a geographical region there has been an epidemic outbreak of a severe human disease like anthrax, human version of swine or avian flu, etc. The region involved is partitioned in zones and for every zone a delivery site is identified. These places represent the locations where the drugs have to be brought, we assumed that once a delivery sites is visited all people in that zone will get the drugs within a very short time. Among all the delivery sites a few, that will be called potential facility locations, are selected, they are potential warehouses where the drugs are stored and represent the potential starting points of the distribution. When a facility location is used it can employ two different strategies to deliver the drugs to the sites:

Distribution Center Strategy (Figure 4.1) Facility locations using this strategy are called distribution centers (DC). They are characterized by a capacity and a distribution range. The former is an estimate of the service potential based on both the drugs supply at that location and its ability in satisfying the requests. The latter instead defines the maximal distance within which a distribution center can provide its services. Locations used as DCs can be basically perceived as self-arranging facilities that are able to provide the drugs needed to a subset of delivery sites within their range and up to their capacity. Several distribution methods can lie behind this strategy. For instance: the use of many small vehicles one for every site; self-service points; this overturns the usual distribution hierarchy making the clients move to get the drugs; the use of non-conventional delivery systems as the existing postal service to distribute drugs door to door.

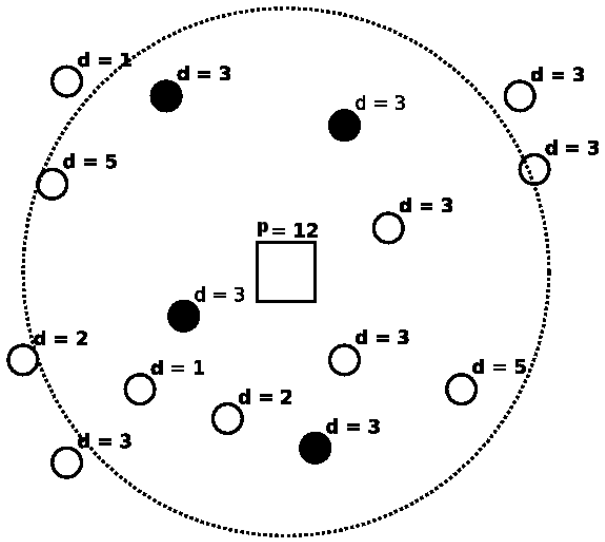

Figure 4.1: Distribution Center Strategy: the dotted circle represents the range of the DC, **p** is the capacity of the DC, **d** is the demand of the sites and the black sites are the sites served.

Routing Strategy (Figure 4.2) This strategy comes from the classical VRP; in fact when a facility location makes use of this strategy it is called depot. With every depot is associated a fleet of heterogeneous vehicles with different capacities. Vehicles are used to distribute the drugs; every vehicle can perform a single route starting from

the depot, visiting a few delivery sites and coming back to the same depot.

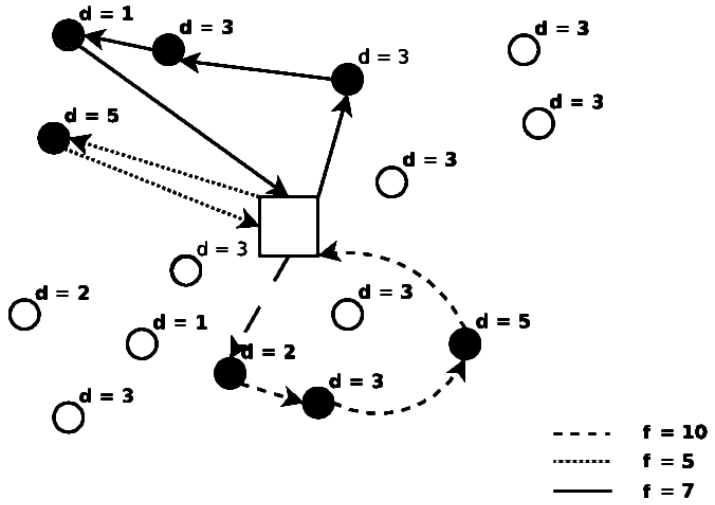

Figure 4.2: Routing Strategy: three vehicles, **f** is the capacity of the vehicle, **d** is the demand of the sites and the black sites are the sites served.

The big picture describing our problem represents a logistic system, employing a mixed distribution strategy, that is able to choose which facility locations are useful and to select for every chosen location the most appropriate distribution strategy (Figure 4.3).

Given the described scenario, we want to design and optimize such a logistic system exploiting the potential of the mixed delivery strategy to bring the necessary drugs to the population respecting all the following conditions:

I No delivery site can be visited more than once by a single route or served multiple times by a single distribution center. Multiple visits by different routes or distribution centers are allowed even if they do not directly improve the number of citizens served.

II No split delivery: when a delivery site is reached its demand is entirely served in a single visit.

III If a facility location is used it can be either a depot or a distribution center, not both at the same time.

IV The number of vehicles available, for every type of vehicles, is limited.

V The total number of distribution centers that can be open is limited due to budget restrictions.

VI Each route begins and ends at the same depot.

VII The sum of the demands of the clients belonging to the same route must not exceed the capacity of the vehicle employed in that route.

VIII Every delivery site in a route must be visited before the deadline. The deadline represents a time limit within which the pharmacological treatments are more effective. It is worth noting that it is not necessary for a vehicle to be back at the depot within the same time limit.

IX The total demands of the delivery sites covered by a single distribution center must not exceed its capacity.

X All delivery sites covered by a distribution center must be within its range.

Among all feasible distribution plans, the one that maximizes the number of citizens reached is considered optimal. The resulting problem does not fit into the description of any classical LRP and corresponds to the third type of problem we identified in the introduction (see Chapter 1) that is the "Active Location Decisions" types. Indeed it requires to select not only which locations have to be used but also, for every location opened, which distribution strategy has to be employed and thus the location decisions may directly influence the value of the objective function of the problem. Since this problem combines location features with different distribution methods we decided to call it, regardless of the specific context, Generalized Location and Distribution Problem (GLDP).

4.3 Mathematical Formulation

The GLDP for drug distribution in case of emergency can be formally described as follows.

A complete graph $G = (\mathcal{N}, \mathcal{A})$ is given where $\mathcal{N}$ is the set of nodes representing the delivery sites and $\mathcal{A} = \{(i, j) : i \in N, j \in N\}$ is the set of the arcs. With every delivery site $i \in N$ two values d_i and t_i are associated; the former, d_i, represents an estimate of the number of people served when the site is visited; the latter, t_i, is the time needed to visit the site. For every arc $(i, j) \in \mathcal{A}$ a value c_{ij} is given; it represents the time needed by a vehicle to go from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$.

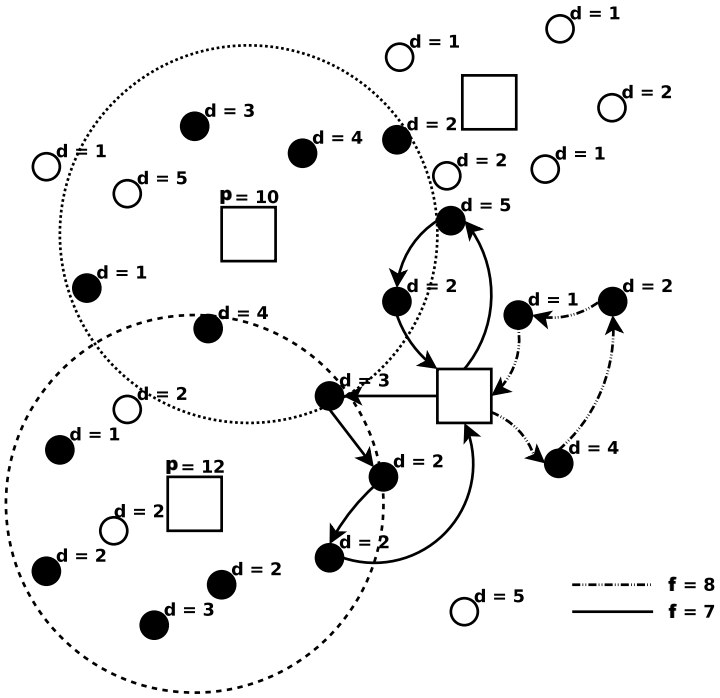

Figure 4.3: Mixed Strategy: combination of distribution centers and depot strategies.

Let a subset of the delivery sites $\mathcal{L} \subseteq \mathcal{N}$ be the set of the potential facility locations; It is worth noting that this definition allows a node to be at the same time a delivery site and a potential facility location. For every location $l \in \mathcal{L}$ two values p_l and r_l, respectively the service capacity and the delivery range of the possible distribution center located in l, are given. The value p_l represents the maximum total demand that DC l can satisfy while r_l is the maximum traveling time, within which the distribution center l is able to provide an effective service. In other words a site $i \in \mathcal{N}$ is reachable by a distribution center $l \in L$ if $c_{li} \leq r_l$. Moreover, we define, for every $l \in L$, $\mathcal{W}_l$ as the set of all feasible clusters of delivery sites associated with DC l. A cluster $w \in \mathcal{W}_l$ is feasible if it represents a subset of sites $\mathcal{R}_w \in \mathcal{N}$ that can be served together by the same distribution center, that is all sites in $\mathcal{R}_w$ are within the range of l, as requested by Condition X, and the sum of the demands of the sites in $\mathcal{R}_w$ must not exceed the capacity of the DC l (Condition IX). More formally $\forall l \in \mathcal{L}$ a cluster w is feasible if the following conditions hold:

$$\begin{aligned} &\mathcal{R}_w \subseteq \mathcal{N} \\ &c_{li} + t_i \leq r_l && \forall i \in \mathcal{R}_w \\ &\sum_{i \in \mathcal{R}_w} d_i \leq p_l \end{aligned}$$

In our problem, due to budget limitations, among all possible locations $l \in \mathcal{L}$ at most B distribution centers can be activated at the same time.

Moreover, since our problem involves a heterogeneous fleet of vehicles let $\mathcal{H}$ be the set of vehicle types and f_h and V_h be respectively the capacity and the maximum number of vehicles available for every type $h \in \mathcal{H}$; consequently the total number of vehicles T composing the fleet is equal to $T = \sum_{h \in \mathcal{H}} V_h$. A vehicle can be allocated to any possible depot location $l \in L$ and starting from there it can perform a single route. Define $\mathcal{K}$ as the set of all feasible routes starting from any possible depot and using any possible vehicle, we identify $\mathcal{K}_{lh} \subseteq \mathcal{K}$ as the set of feasible routes using a vehicle of type $h \in \mathcal{H}$ and associated with depot $l \in \mathcal{L}$. A route is feasible if it starts and ends at the same depot $l \in L$, as requested by Condition VI, if the sum of the demands of the sites visited does not exceed the capacity of the vehicle $h \in H$ used, following condition VII, and if all the sites are visited before the deadline M as stated in condition VIII. With more formality, named $\mathcal{R}_k \subseteq \mathcal{N}$ the set of sites visited along route $k \in \mathcal{K}_{lh}$ and SP_k^- the minimum traveling time

required to visit all sites in R_k starting from the depot l, a route $k \in \mathcal{K}_{lh}$ is feasible if:

$$SP_k^- + \sum_{i \in R_k} t_i \leq M \tag{4.1}$$

$$\sum_{i \in R_k} d_i \leq f_h \tag{4.2}$$

Taken into account the goal of our GLDP that is the maximization of the population receiving the necessary drugs, the problem can be described using the following mathematical formulation:

$$\min \sum_{i \in \mathcal{N}} d_i s_i \tag{4.3}$$

$$\text{s.t. } s_i + \sum_{l \in \mathcal{L}} \sum_{h \in \mathcal{H}} \sum_{k \in \mathcal{K}_{lh}} a_{ik} z_k + \sum_{l \in \mathcal{L}} \sum_{w \in \mathcal{W}_l} b_{iw} y_w \geq 1 \qquad \forall i \in \mathcal{N} \tag{4.4}$$

$$\sum_{l \in \mathcal{L}} \sum_{k \in \mathcal{K}_{lh}} z_k \leq V_h \qquad \forall h \in \mathcal{H} \tag{4.5}$$

$$\sum_{k \in \mathcal{K}_{lh}} z_k + \sum_{w \in \mathcal{W}_l} V_h y_w \leq V_h \qquad \forall l \in \mathcal{L}, \forall h \in \mathcal{H} \tag{4.6}$$

$$\sum_{l \in \mathcal{L}} \sum_{w \in \mathcal{W}_l} y_w \leq B \tag{4.7}$$

$$s_i \in \{0, 1\} \qquad \forall i \in \mathcal{N} \tag{4.8}$$

$$y_w \in \{0, 1\} \qquad \forall l \in \mathcal{L}, \forall w \in \mathcal{W}_l \tag{4.9}$$

$$z_k \in \{0, 1\} \qquad \forall l \in \mathcal{L}, \forall h \in \mathcal{H}, \forall k \in \mathcal{K}_{lh} \tag{4.10}$$

Our model makes use of three different sets of binary variables. For every $i \in \mathcal{N}$ there is a variable s_i representing the skipping of a delivery site: it takes value one if site i is not visited and zero otherwise. With each cluster $w \in \mathcal{W}_l$ is associated a binary variable y_w: it assumes value one if the facility location $l \in \mathcal{L}$ is used as a distribution center serving the sites belonging to the cluster $w \in \mathcal{W}_l$, zero otherwise. Finally a variable z_k for each $k \in \mathcal{K}_{lh}$ represents a route using vehicle type $h \in \mathcal{H}$ and starting from a depot located at the facility location $l \in \mathcal{L}$. It takes value one if this route is employed and zero otherwise. In addition two binary coefficients a_{ik} and b_{iw} are considered. The former, a_{ik}, takes value

one if site $i \in \mathcal{N}$ belongs to route $k \in \mathcal{K}_{lh}$ and zero otherwise; in a similar way the latter, b_{iw}, assumes value one if site $i \in \mathcal{N}$ belongs to cluster $w \in \mathcal{W}_l$ and zero otherwise.

In our problem the objective function (4.3) is written in the reverse form as the minimization of the demands not satisfied; no site has to be compulsorily visited and there are no prizes in serving the sites but we pay a fee proportional to the demand every time we are not able to deliver the drugs where requested. Constraints (4.4) are covering constraint stating that either a site is served by at least a route or distribution center or its associated skip variable must take value one. It is worth noting that although these constraints allow for multiple visits of the same site, visiting a site more than once does not improve the value of the objective function. In addition, by definition a site cannot be visited more than once in a single route or cluster making the multiple visits of a site possible only by means of at least two routes or clusters. However constraints (4.4) are kept in this way in order to allow for the selection, among all solutions of the same value, of the most flexible ones identified as the solutions involving the maximum number of multiple visits. These constraints together with the definitions of sets W_l and K_{lh} formalize Condition I. With constraints (4.5) Condition IV is satisfied: indeed these constraints impose $\forall h \in \mathcal{H}$ an upper bound equal to V_h on the number of vehicles of type h that can be used. Constraints (4.6) say that if a facility location is used as a depot it cannot be used as a distribution center and vice-versa. These constraints come from the disaggregation of a more straightforward version: $\sum_{h \in H} \sum_{k \in \mathcal{K}_{lh}} z_k + \sum_{w \in \mathcal{W}_l} T y_w \leq T \quad \forall l \in L$, that is more intuitive but dominated by constraints (4.6). It is worth noting that these constraints do no force any location to be used. With these constraints, Condition III is formalized. The last constraints (4.7), formalizing what is requested by Condition IX, put an upper limit on the number of distribution centers that can be opened at the same time.

4.4 Branch-and-cut-and-price

We propose an algorithm for the exact solution of the GLDP that is based on the branch-and-cut-and-price paradigm (see Ladányi et al. [44]). This method starts from the linear

relaxation of the problem, usually referred to as Master Problem (MP), obtained substituting constrains (4.8), (4.9) and(4.10) with

$$\begin{aligned} 0 \leq s_i \leq 1 \qquad & \forall i \in N \\ 0 \leq z_k \leq 1 \qquad & \forall l \in \mathcal{L}, \forall h \in \mathcal{H}, \forall k \in \mathcal{K}_{lh} \\ 0 \leq y_w \leq 1 \qquad & \forall l \in \mathcal{L}, \forall w \in W_l. \end{aligned}$$

The derived model has an exponential number of columns and so, in order to obtain valid lower bounds, it is solved via column generation (see Desaulniers et al. [55]). When the MP is solved to optimality we look for additional inequalities that are violated by the current LP solution. This process of column and row generation is iterated until no further useful variable or violated cut is found. If the solution achieved is either not integer or higher than the lower bound then a branching is performed and we start exploring the branching tree obtained recomputing the lower bound at every node. In this section we present the main components of our branch-and-cut-and-price algorithm namely: pricing, cut generation, branching and stabilization.

4.4.1 Column Generation

The MP contains a number of variables (z and y) whose number is exponential in the cardinality of the site set $\mathcal{N}$ and so it is not possible to solve it taking explicitly into account all variables. For this reason a column generation method is applied to find an optimal solution. Such approach starts from a Restricted MP (RMP) made up of only a small subset of variables and then it iteratively evaluates, exploiting the dual representation of the problem, which variables have to be added to the RMP in order to improve the value of the objective function. This process is stopped when there are no other useful variables to introduce into the MP.

In particular, in our case, the initial RMP is composed by:

(a) all columns corresponding to skip variables s_i; in fact there is only a polynomial number of them, that is $|N|$;

(b) a subset of K variables made up of all $|L| \cdot |H| \cdot |N|$ columns representing, for every possible depot location and every type of vehicle, routes serving only one site;

(c) a subset of W with $|L| \cdot |N|$ columns that represents, for every possible distribution center location, a cluster with only one site.

The RMP is solved and then the process of seeking for useful columns not already in the RMP, called pricing, begins. To evaluate if a variable can lead to an improvement in the objective function the value of its associated reduced cost has to be analyzed, the more negative it is the more useful is the variable. If no variables with negative reduced cost are found, the solution of the RMP is optimal for the MP as well, and thus it yields a valid lower bound to the problem.

In order to compute the reduced cost of a variable, we have to exploit the dual form of our problem. The formulation of the dual MP of the GLDP reads as follows:

$$\max \sum_{i \in \mathcal{N}} \lambda_i - \sum_{h \in \mathcal{H}} V_h \mu_h - \sum_{l \in \mathcal{L}} \sum_{h \in \mathcal{H}} v_h \sigma_{lh} - B\rho \tag{4.11}$$

$$\text{s.t. } \lambda_i \leq d_i \qquad \forall i \in \mathcal{N} \tag{4.12}$$

$$\sum_{i \in \mathcal{N}} a_{ik} \lambda_i - \mu_h - \sigma_{lh} \leq 0 \qquad \forall l \in \mathcal{L}, \forall h \in \mathcal{H}, \forall k \in \mathcal{K}_{lh} \tag{4.13}$$

$$\sum_{i \in \mathcal{N}} b_{iw} \lambda_i - \sum_{h \in H} V_h \sigma_{lh} - \rho \leq 0 \qquad \forall l \in \mathcal{L}, \forall w \in W_l \tag{4.14}$$

$$\lambda_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{4.15}$$

$$\mu_h \geq 0 \qquad \forall h \in \mathcal{H} \tag{4.16}$$

$$\sigma_{lh} \geq 0 \qquad \forall l \in \mathcal{L}, \forall h \in \mathcal{H} \tag{4.17}$$

$$\rho \geq 0 \tag{4.18}$$

where dual not-negative vectors λ, μ, σ and ρ correspond to primal constraints (4.4) and to constraints (4.5), (4.6), (4.7) rewritten as $\geq$ inequalities. Dual constraints (4.12), (4.13) and (4.14) correspond to primal variable vectors s, z and y respectively.

The reduced costs, ζ_w, and ξ_k, associated with primal variables y_w and z_k respectively can then be computed, by means of the vectors of dual variables, in the following way:

$$\zeta_w = -\sum_{i \in \mathcal{R}_w} \lambda_i + \sum_{h \in H} V_h \sigma_{lh} + \rho \tag{4.19}$$

$$\xi_k = -\sum_{i \in \mathcal{R}_k} \lambda_i + \mu_h + \sigma_{lh}. \tag{4.20}$$

Since in our problem there are two different kinds of variables, then two different pricing subproblems, associated respectively with y_w and z_k variables have to be solved in order to find the column with the minimum reduced cost to be added to the RMP. It is worth noting that since variables y_w and z_k represent respectively clusters and routes solutions to their associated pricing subproblems have to satisfy Conditions X, IX and Conditions VIII, VII.

Distribution center pricing subproblem. Formally, for a given $l \in L$, the pricing subproblem associated with variables y_w is:

$$\min \zeta_w = -\sum_{i \in \mathcal{R}_w} \lambda_i + \sum_{h \in H} V_h \sigma_{lh} + \rho \tag{4.21}$$

$$\text{s.t. } c_{li} + t_i \leq r_l \qquad \forall i \in \mathcal{R}_w$$

$$\sum_{i \in \mathcal{R}_w} d_i \leq p_l \tag{4.22}$$

where the set $\mathcal{R}_w$ including all sites belonging to the cluster $w \in W_l$ served by the DC located in located in l has to be determined.

This problem together with Condition I, that is implicit in the definition of the set W_l, turns out to be the well studied 0-1 knapsack problem (see Martello and Toth [85]) in which given a set of items, each characterized by a value and a weight and a backpack with a finite capacity, it is required to find the most convenient combination of items whose total weight does not exceed the capacity of the backpack and that maximizes the total value carried. In our problem, for a given location $l \in \mathcal{L}$, the items are the sites $\mathcal{N}(r_l) = \{i \in \mathcal{N} : c_{li} + t_i \leq r_l\}$, while $-\lambda_i$ are the values and d_i represent the weights. Thus a feasible cluster of minimum reduced cost can be computed as the subset of elements belonging to $\mathcal{N}(r_l)$, giving the maximum value and having a total weight that does not exceed p_l. Since, parameters r_l, p_l and dual variables σ_{lh} depend on the facility location where the DC is located, $|L|$ different knapsack pricing problems have to be solved to find the cluster with the minimum reduced cost. In order to optimally solve this pricing subproblem we propose an algorithm based on dynamic programming that runs in pseudo-polynomial time.

Route pricing subproblem. Given a vehicle $h \in H$ and a potential location $l \in \mathcal{L}$ for a depot, the pricing subproblem associated with variables z_k reads:

$$\min \xi_k = -\sum_{i \in \mathcal{R}_k} \lambda_i + \mu_h + \sigma_{lh} \tag{4.23}$$

$$\text{s.t. } SP_k^- + \sum_{i \in \mathcal{R}_k} t_i \leq M$$

$$\sum_{i \in \mathcal{R}_k} d_i \leq V_h \tag{4.24}$$

where the set $\mathcal{R}_k$ of sites visited by the route $k \in K_{lh}$ has to be determined.

Taking into account that the Condition I is implicitly satisfied by the definition of K and that Condition II is never violated if constraints (4.24) are respected, the problem described by this formulation is the Resource Constrained Elementary Shortest Path Problem (RCESPP, see Feillet et al. [59]). Our approach for solving this problem proposes three different algorithms, in particular: two heuristic procedures namely a greedy one and a dynamic programming one and an exact method based on an aggregated bounded bi-directional dynamic programming algorithm with decremental state space relaxation. The value of variables μ_h and σ_{lh} that are used as parameters in this problem depends on $h \in H$ and $l \in L$; for this reason to find the route with the minimum reduced cost we have to, theoretically, solve a total of $|H| \cdot |L|$ problem instances one for every combination of types of vehicles and potential locations.

Pricing algorithms are called in sequence starting from the dynamic programming procedure for the knapsack problem, if it fails then the pricing procedures for the RCESPP are called. An outline of the invoking sequence of the pricing algorithms is shown in Figure 4.4.

In the reminder of this section we give a detailed description of our pricing procedures.

4.4.2 Knapsack Pricing Algorithm

The columns with negative reduced cost that represent clusters can be found solving instances of the NP-hard 0-1 knapsack problem. This problem is one of the most studied in the operations research area and is somehow considered as "the beginning of Integer Linear Programming" (Caprara [86]). A description of the problem and the most common solution approaches can be found in any book about integer linear programming, in particular an extensive discussion of the problem and its variations along with a detailed explanation of

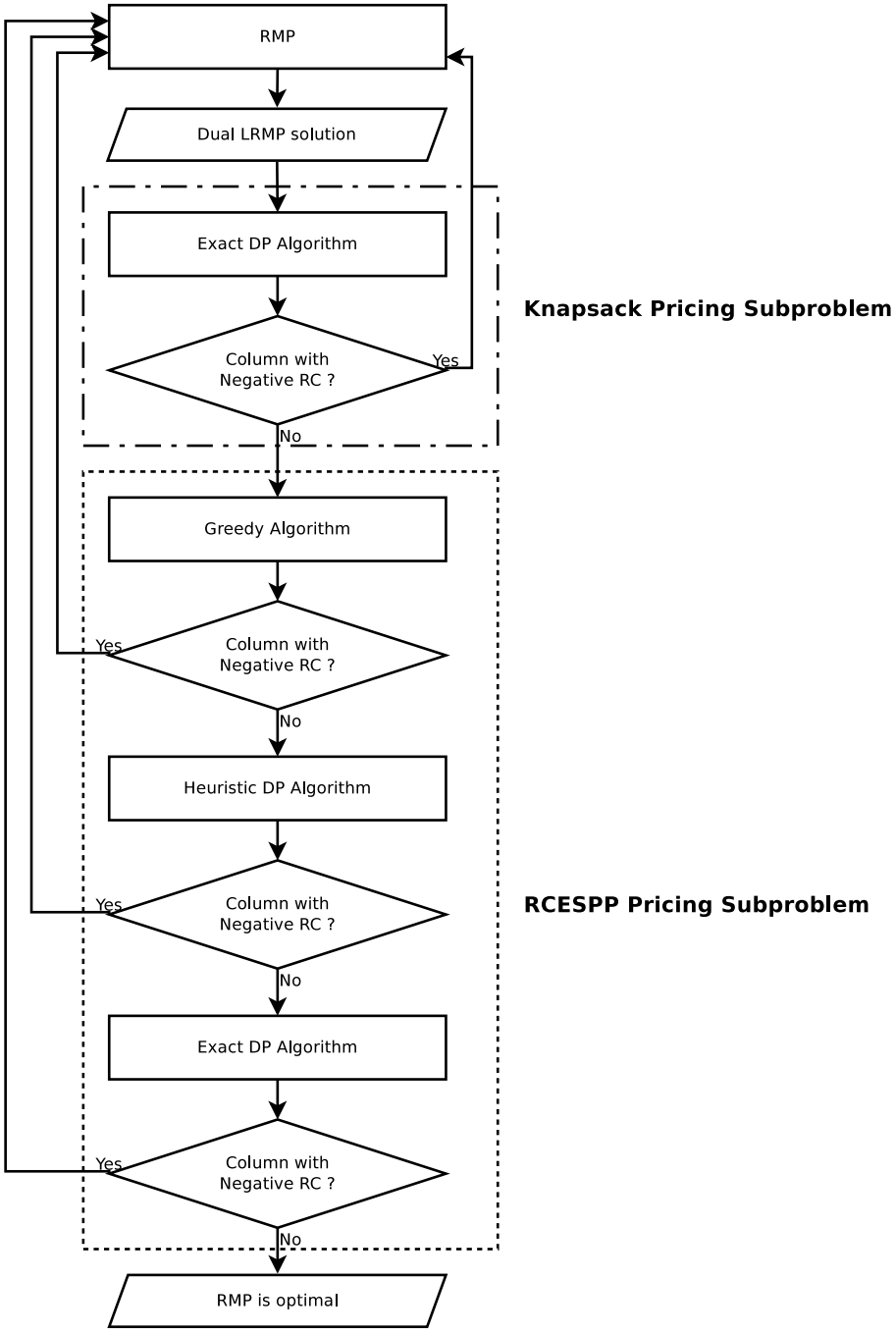

Figure 4.4: The sequence of the pricing algorithms

many exact and heuristic algorithms for their solution can be found in a book by Pisinger et al. [87].

Our approach is based on the classical pseudo-polynomial time dynamic programming algorithm that was already extensively described in the late 50's by Dantzig [88]. This technique achieves an optimal solution to the 0-1 knapsack problem iteratively solving knapsack subproblem of size $u \leq p_l, l \in \mathcal{L}$ considering only the first $i \leq |\mathcal{N}(r_l)|$ items. First, for each $l \in \mathcal{L}$, we define an order on the elements of the set $\mathcal{N}(r_l) = \{i \in \mathcal{N} : c_{li} + t_i \leq r_l\}$. Then, a set of labels $(i, u, \mathcal{S}, v)$ is computed, representing the minimum reduced cost v that can be obtained using at most u units of capacity and considering only the first i elements in the set $\mathcal{N}(r_l)$, and the set of elements $\mathcal{S}$ yielding this reduced cost. The computation of all these labels is done by a dynamic programming algorithm. We create an initial label $(0, 0, \emptyset, 0)$. Then, we iteratively extend labels by increasing values of i, and for each i by increasing values of u. When extending a label $(i, u, \mathcal{S}, v)$, two labels may be created:

- a label $(i + 1, u, \mathcal{S}', v')$, corresponding to the choice of skipping element $i + 1$, with $\mathcal{S}' = \mathcal{S}$, $v' = v$ is generated if $u + d_{i+1} > p_l$.
- a label $(i + 1, u + d_{i+1}, \mathcal{S}'', v'')$, corresponding to the choice of including element $i + 1$, with $\mathcal{S}'' = \mathcal{S} \cup \{i + 1\}$, and $v'' = v - \lambda_{i+1}$ is created if $u + d_{i+1} \leq p_l$.

An optimal solution to the DC pricing problem can then be found in the set $\{(|\mathcal{N}(r_l)|, u, \mathcal{S}, v)\}$. It is worth noting that this algorithm allows to find several solutions in a single run and then it is useful for multiple pricing. Moreover, it allows us to take into account additional features arising after the introduction of strengthening cuts, as explained in the subsequent sections.

4.4.3 RCESPP Pricing Algorithms

In our branch-and-price-and-cut procedure three different algorithms are used to price out columns that represent routes. For the purpose of better illustrating them, we give their detailed description following the reverse order with respect to their execution.

Exact Dynamic Programming Algorithm

Let us consider first the case of a single depot $l \in \mathcal{L}$, a single kind of vehicle $h \in \mathcal{H}$, let q and v be two distinct copies of the base b representing, respectively, the starting and ending

point of the route and let $f = f_h$ be the capacity of the vehicle h. In order to achieve the exact solution of the RCESPP we use the methods presented by Righini and Salani [60], which consists of a bi-directional extension of node labels. It associates labels, which encode partial paths, with every site $i \in \mathcal{N}$ of the graph G. Each label is iteratively considered and the corresponding path is extended to sites not already visited.

Label Structure. Each label is defined by a tuple $(\mathcal{S}, \delta, \tau, C, i)$, where C is the cost of the path, i identifies the last delivery site visited, $\mathcal{S}$ is a set indicating which nodes have been already visited and δ and τ represent resource consumptions. In particular δ indicates the amount of drugs delivered while τ represents the elapsed time from the beginning of the route to the end of the visit of node i. Each label identifies a state that represents a path, visiting all sites in $\mathcal{S}$, of cost C starting from a depot located at l and arriving to the site i delivering, along the way, in τ time, a total amount of drugs equal to δ. The optimal solution of the problem is the minimum cost path going from q to v delivering at most a quantity $\delta \leq f$ of drugs and spending a time $\tau \leq M$ for reaching and serving all nodes in $\mathcal{S}$.

Extension. A label $(\emptyset, 0, 0, 0, q)$ is initially created. Then, at every iteration of the algorithm, a label $l' = (\mathcal{S}', \delta', \tau', C', i')$ is iteratively selected and extended from site i' to each site $i'' \in (\mathcal{N} \cup \{v\}) \setminus \mathcal{S}'$, creating another label $l'' = (\mathcal{S}'', \delta'', \tau'', C'', i'')$; this enforces the elementary of the path as no sites already visited are taken into account for the extension. The following rules are used to carry out the resulting label:

$$\mathcal{S}'' = \mathcal{S}' \cup \{i''\} \tag{4.25}$$

$$\delta'' = \delta' + d_{i''} \tag{4.26}$$

$$\tau'' = \tau'' + c_{i'i''} + t_{i''} \tag{4.27}$$

$$C'' = C' - \frac{\lambda^{i'}}{2} - \frac{\lambda^{i''}}{2}$$

with $\lambda^q = \lambda^v = 0$, $t_q = t_v = 0$, $d_q = d_v = 0$ and $c_{iv} = 0 \, \forall i \in N$. The new label l'' represents a feasible state if $\delta'' \leq f$ and $\tau'' \leq M$, in other words a label can be extended from i to i'' if there are both enough drugs on the vehicle and enough time before the deadline to totally serve the delivery site i''. It is worth noting that, as requested by the conditions detailed in Section 4.2, this extension rule does not allow for split delivery and that the arc identifying

the path returning to the depot after visiting the last served node is not taken into account.

Dominance Test. The exponential number of labels that have to be potentially considered to find the optimal $q - v$ path is a primary source of potential inefficiency. For this reason to reduce the number of labels to analyze, during the extension of labels, a dominance test is performed to fathom labels that cannot lead to an optimal solution. Let $l' = (\mathcal{S}', \delta', \tau', C', i)$ and $l'' = (\mathcal{S}'', \delta'', \tau'', C'', i)$ be two generic labels associated with the same delivery site i. Then the former dominates the latter if the following conditions hold:

$$\mathcal{S}' \subseteq \mathcal{S}'' \tag{4.28}$$

$$\tau' \leq \tau'' \tag{4.29}$$

$$C' \leq C'' \tag{4.30}$$

None of the inequalities (4.28), (4.29) and (4.30) can be left out, in fact they represent a set of necessary dominance conditions and dropping any of them can lead to disregarding an optimal label. It is worth noting that, since the total amount of drugs delivered along a path depends only on the set $\mathcal{S}$ of sites served, to determine if l' dominates l'' we do not have to compare δ' and δ''; in fact, if condition (4.28) is satisfied, the inequality $\delta' \leq \delta''$ is always true. Furthermore, as shown in Feillet et al. [59], it is sometimes possible to identify a site $u \in \mathcal{N}$ that cannot be reached by any feasible extension of a given label, because of resource limitations. In this case it is useful to insert u in the set $\mathcal{S}''$ of that label: it is easy to check that enlarging $\mathcal{S}''$ helps satisfying condition (4.28); at the same time, if a site cannot be reached by extending label l' due to resource limitations, it cannot be reached by extending label l'' either, since resource consumption in l'' is not lower. Therefore, enlarging each set $\mathcal{S}$ allows the dynamic programming algorithm to fathom a larger number of labels consequently reducing the computation time. When the technique by Feillet et al. [59] is employed the value of δ' and δ'' must be explicitly taken into account in the dominance test. In fact, since $\mathcal{S}'$ and $\mathcal{S}''$ may contain unreachable nodes, in order for the label l' to dominate label l'' the following additional conditions must hold:

$$\delta' \leq \delta''. \tag{4.31}$$

Decremental state space relaxation. In order to speed up the solution process the dynamic programming algorithm is executed iteratively applying decremental state space

relaxation as described by Righini and Salani [60]. The idea behind such a relaxation is simple: the state space of the problem is reduced projecting it to a smaller one by discarding the elementary constraints, and then iteratively reintroducing them until a feasible solution for the RCESPP is found. More formally, given a set of delivery sites $\tilde{\mathcal{N}} \subseteq \mathcal{N}$ called critical set, the extension rule (4.25) can be replaced with

$$\mathcal{S}' = (\mathcal{S} \cup j) \cap \tilde{\mathcal{N}} \tag{4.32}$$

This relaxed problem can be solved more efficiently, since more labels can be compared in the dominance test. In order to identify a good critical node set, $\tilde{\mathcal{N}}$ is initially empty then the state space relaxation of the pricing problem is solved and all the nodes visited more than once in the optimal path are added to the set $\tilde{\mathcal{N}}$. In our algorithm, this procedure is iterated until either an elementary path with a negative reduced cost is found or the optimal value of the pricing subproblem is nonnegative.

Bidirectional dynamic programming. Another method that has been proved to be useful by Righini and Salani [58] in speeding up the computation of the optimal $q - v$ path is bidirectional dynamic programming. In this algorithm two kind of labels are associated with every delivery site: forward labels and backward labels. They, respectively, represent paths going from q to its successors and from v to its predecessors. The forward set is initialized with label $(\emptyset, 0, 0, 0, q)$ while the backward set with label $(\emptyset, 0, 0, 0, v)$. Extension and dominance rules are then composed by two steps, in which forward and backward labels are treated independently. The updating rules and feasibility tests for backward extension are almost symmetrical to those for the forward labels with a single noticeable exception. In fact in the backward extension the time spent to go backward from v to i is not taken into account while, in the forward extension, the traveling time from q to i is considered. Since the consumption of both resources, amount of delivered drugs and time, is monotone along the path a bounding technique can be employed together with the bidirectional dynamic programming in order to reduce the generation of backward and forward labels encoding the same paths. In particular, identifying the time as the most scarce and critical resource, the extension of forward and backward labels can be limited and useless duplication of paths can be reduced, imposing that each partial path can use at most half of the available time, that is $\tau \leq M/2$ for both forward and backward labels.

Join. A complete route going from q to v can then be obtained joining together a forward and a backward partial path. Each forward path $(\mathcal{S}^{fw}, \delta^{fw}, \tau^{fw}, C^{fw}, i^{fw})$ can be joined with a backward path $(\mathcal{S}^{bw}, \delta^{bw}, \tau^{bw}, C^{bw}, i^{bw})$ if the complete path is elementary and the total consumption of both resources, amount of drugs delivered and elapsed time, does not exceed their availability. These conditions can be formalized as follows:

$$\mathcal{S}^{fw} \cap \mathcal{S}^{bw} = \varnothing$$
$$\delta^{fw} + \delta^{bw} \leq f$$
$$\tau^{fw} + c_{i^{fw}i^{bw}} + \tau^{bw} \leq M$$

If all conditions are satisfied the path from q to v is feasible and the values $\mathcal{S}, \delta, \tau$ and C of the corresponding label can be computed as follows:

$$\begin{aligned} \mathcal{S} &= \mathcal{S}^{fw} \cup \mathcal{S}^{bw} \\ \delta &= \delta^{fw} + \delta^{bw} \\ \tau &= \tau^{fw} + c_{i^{fw}i^{bw}} + \tau^{bw} \\ C &= C^{fw} - \frac{\lambda^{i^{fw}}}{2} - \frac{\lambda^{i^{bw}}}{2} + C^{bw} \end{aligned}$$

This label $l^* = (\mathcal{S}, \delta, \tau, C, v)$ represents a feasible route that corresponds to a variable in the MP having a reduced cost $\xi_{l^*} = C + \mu_h + \sigma_{lh}$. The join operation is run when no more feasible extensions can be performed and among all the paths the minimum cost path after join is the optimal solution of the pricing problem.

Aggregated pricing algorithm. Since in our problem both the reduced costs and the resource consumption associated with the routes to be generated depend on the location $l \in \mathcal{L}$ of the depot and on the kind $h \in \mathcal{H}$ of the vehicle employed, in principle it would be necessary to execute the pricing algorithm $|\mathcal{L}| \cdot |\mathcal{H}|$ times. Instead, applying a technique called aggregated pricing described by Bettinelli et al. [81], our algorithm is able to perform a single passage for all the potential depots reducing the total number of iterations needed to $|H|$ that is the number of type of vehicles. In order to optimize all the depots at the same time the state space described by a label needs to be enlarged adding different reduced costs $\mathcal{C}_l$ and time consumption τ_l for every depot $l \in \mathcal{L}$, δ does not need to be computed for every possible depot as it only depends on the sites visited. When a forward label is

extended from a site i to a site j and $\tau_l^{fw} > M/2$ for a certain $l \in \mathcal{B}$, the path is not feasible for depot l: thus τ_l^{fw} is set equal to $+\infty$. For what concerns domination rules, whenever two labels e' and e'' satisfy conditions (4.28), (4.29) and (4.30) for a particular depot l, resources τ_e'' and C_e'' are set to $+\infty$, as label e'' can never yield an optimal path for depot l. A forward label is fathomed when, due either to extensions, feasibility checks or dominance, it has $\tau_l^{fw} > M/2$ for every $l \in \mathcal{L}$. This technique can be easily coupled with bidirectional dynamic programming as described above.

Heuristic Dynamic Programming Algorithm

Solving the RCESPP by means of the exact dynamic programming algorithm described in Subsection 4.4.3 can be very time consuming and thus we developed a heuristic dynamic programming algorithm for the same problem that runs in a shorter time. The basic structure of the heuristic algorithm is borrowed from the exact one but we relax dominance rules removing condition (4.28). This modified dominance test allows for fathoming many more labels reducing consequently the computational time required to find a solution for the RCESPP. At the same time, since all conditions (4.28), (4.29) and (4.30) are necessary, with the relaxed dominance test an optimal partial path could be discarded leading to solutions without any optimality guarantee.

Greedy Algorithm

Another non-optimal algorithm used to quickly find columns with negative reduced costs is the greedy one. In this pricing algorithm a route $q - v$ is generated by repeated extensions of a single label. In details, a label $l = (\mathcal{S}, \delta, \tau, C, i)$ is considered and iteratively extended to a single node with a nearest neighbor policy. Given a depot $l \in L$ and a vehicle $h \in H$ the set $\mathcal{R} \subseteq \mathcal{N} \cup \{v\}$ of the reachable sites is computed. A delivery site $j \in \mathcal{N}$ belongs to $\mathcal{R}$ if all the following conditions hold:

$$j \notin \mathcal{S} \tag{4.33}$$

$$\delta + d_j \leq f_h \tag{4.34}$$

$$\tau + c_{ij} + t_j \leq M \tag{4.35}$$

where $c_{ij} = 0$ if $j = v$. If $\mathcal{R} = \emptyset$ the path is closed going back to the depot and the search is stopped. Otherwise, among the sites in $\mathcal{R}$, we select the one that minimizes the path cost,

that is

$$\bar{j} = argmin_{j \in \mathcal{R}}\{-\lambda_j\}$$

with $\lambda_q = \lambda_v = 0$. The label is extended to node $\bar{j}$ using the same update rules described for the exact pricing algorithm, and we iterate. This greedy algorithm is repeated for each combination of possible depot locations $l \in \mathcal{L}$ and kinds of vehicles $h \in \mathcal{H}$.

4.4.4 Additional Inequalities

When none of the pricing algorithms is able to find a column with negative reduced cost the solution of the RMP is optimal even for the MP and thus represents a valid lower bound for our problem. The higher is this bound the faster the algorithm converges to an optimal solution. A method to improve the lower bound is to find violated inequalities and so, when no other useful variables can be found by the pricing algorithms, we look for additional cuts to be added to the MP. In particular we modified and extended two classical cut types in order to work properly in our problem: subset-row and 2-path inequalities and then we propose a new family of inequalities for the problem.

Subset Row Inequalities

These cuts represent an adaptation of a special case of subset-row inequalities for the VRP proposed by Jepsen et al. [64], limited to triplets of sites. The idea behind the original cuts is fairly simple: in every integer feasible solution, for every set of three sites, there must be at most a single route visiting at least two of them. Since, in our problem, the sites can be served either by routes or by distribution centers this idea needs a little modification to be applied. For the GLDP we can say that: in every integer feasible solution, for every set of three sites, there must be at most either a single route or a single distribution center serving at least two of them.

Subset-row inequalities can then be formalized as follows. Let $\mathcal{C} = \{C \subseteq \mathcal{N} : |C| = 3\}$ be the set made of all possible subsets of three sites. For $C \in \mathcal{C}$ let $\mathcal{W}(C) \subseteq \mathcal{W}$ be the subset of all clusters, regardless of the location of their associated DC, containing at least two of the three sites in C; let $\mathcal{K}(C) \subseteq \mathcal{K}$ be the set of all routes, regardless of their starting depot and kind of vehicle used, that visit at least two of the three sites in C. Then the

following inequalities are valid for the GLDP:

$$\sum_{w \in \mathcal{W}(C)} y_w + \sum_{k \in \mathcal{K}(C)} z_k \leq 1 \qquad \forall C \in \mathcal{C} \tag{4.36}$$

The separation of these inequalities can be done by complete enumeration since their number is polynomial in the cardinality of the set of sites ($O(|\mathcal{N}|^3)$). However to speed up the separation process we take into account only a subset $\bar{\mathcal{N}} \subseteq \mathcal{N}$ of the sites for the generation of the clusters set $\mathcal{C}$. In our algorithm a site $i \in \mathcal{N}$ belongs to $\bar{\mathcal{N}}$ if the value of its associated skip variable s_i is strictly lower than 1 in the current LP iteration.

The introduction of these inequalities changes the pricing subproblems and therefore it requires to modify the pricing algorithms to take into account the values of the corresponding dual variables. Let ψ_C be the (non-positive) dual variable associated with the inequality corresponding to the triplet C. The definition of the labels used in both the knapsack pricer and the dynamic programming procedures for the RCESPP needs to be modified, adding a new resource for the cut associated with every additional triplet C.

The labels used in the knapsack pricing algorithm for each $l \in \mathcal{L}$ are $(i, u, \mathcal{S}, \vec{q}, v)$, where $\vec{q}$ is a vector with one component q_C for each triplet C whose subset-row inequality is in the RMP. Each q_C represents the number of elements of C included in $\mathcal{S}$, and so $0 \leq q_C \leq 3$. Our label pushing algorithm is modified as follows. First, we create an initial label $(0, 0, \emptyset, (0 \dots 0), 0)$. Then, we iteratively extend labels as described in Section 4.4.2. When extending a label $(i, u, \mathcal{S}, \vec{q}, v)$, two labels are created:

- a label $(i+1, u, \mathcal{S}', \vec{q'}, v')$, corresponding to the choice of skipping element $i+1$, with $\mathcal{S}' = \mathcal{S}$, $\vec{q'} = \vec{q}$, $v' = v$
- a label $(i+1, u + d_{i+1}, \mathcal{S}'', \vec{q''}, v'')$, corresponding to the choice of including element $i+1$, with $\mathcal{S}'' = \mathcal{S} \cup \{i+1\}$, $q''_C = |C \cap \mathcal{S}''|$, for each $C \in \mathcal{C}$, and $v'' = v - \lambda_{i+1} - \sum_{C \in \bar{\mathcal{C}}} \psi_C$ where

  $$\bar{\mathcal{C}} = \{C \in \mathcal{C} : q''_C = 2 \text{ and } q'_C = 1\}.$$

  This second label is created only if $u + d_{i+1} \leq p_l$.

Since this method can, in principle, generate for each i and u a different label for each $\vec{q}$, when each pair of values i and u is considered we perform the following dominance checks:

a label $\ell' = (i, u, \mathcal{S}', \vec{q'}, v')$ dominates a label $\ell'' = (i, u, \mathcal{S}'', \vec{q''}, v'')$ if

$$v' - \sum_{C \in \tilde{\mathcal{C}}} \psi_C \leq v'', \tag{4.37}$$

where

$$\tilde{\mathcal{C}} = \{C \in \mathcal{C} : C \setminus \{1 \dots i\} \neq \emptyset \text{ and } q'_C = 1 \text{ and } q''_C \in \{0, 2\}\}.$$

The rationale behind rule (4.37) is the following. When $C \setminus \{1..i\} = \emptyset$, the contribution of ψ_C cannot be triggered in ℓ' anymore. When $|C \cap \mathcal{S}'| \geq 2$, then v' already includes the contribution of ψ_C; at the same time when $|C \cap \mathcal{S}'| \leq |C \cap \mathcal{S}''| \leq 1$ any extension of ℓ' triggering the contribution of ψ_C would trigger the same contribution in ℓ''. Finally, if $q''_C = 2$ then at most one element of C follows i in the given order, and therefore if $q'_C = 0$, the contribution of ψ_C will never be triggered in ℓ'; the same applies if $q''_C = 3$ (that is, no element of C follows i in the given order), and $q'_C \leq 1$. Only two cases are left open: $(q'_C = 1, q''_C = 0)$ and $(q'_C = 1, q''_C = 2)$.

We remark that only labels having same i and u values need to be compared during the dominance checks, and that when a particular (i, u) pair is considered, all labels for that pair have already been generated; therefore it is still possible to store labels in a table having one row for each value of i and one column for each value of u, and keep the dynamic programming algorithm very efficient also after the introduction of subset-row cuts.

In the same way, the state label used in the exact dynamic programming procedure for the RCESPP has to be extended with a new vector of variables $\vec{q}$ with one component q_C for each triplet C whose subset-row inequality is in the RMP. Extension, dominance and join rules have to be modified in the following way. Given label $l' = (\mathcal{S}', \delta', \tau', \vec{q}', C', i')$, label $l' = (\mathcal{S}', \delta', \tau', \vec{q}'', C'', i'')$ and defining $\bar{\mathcal{S}}'' \subset \mathcal{S}''$ as the set of all sites visited along the path encoded by label l'' without the inclusion of unreachable sites taken into account when the technique by Feillet et al. [59] is employed, the extension of label l' to l'' is done in the

following way:

$$\begin{aligned}
\mathcal{S}'' =& \mathcal{S}' \cup \{i''\} \\
q''_C =& |C \cap \bar{\mathcal{S}}''| \text{mod} 2 \qquad \forall C \in \mathcal{G} \\
\delta'' =& \delta' + d_{i''} \qquad (4.38) \\
\tau'' =& \tau'' + c_{i'i''} + t_{i''} \qquad (4.39) \\
C'' =& C' - \frac{\lambda^{i'}}{2} - \frac{\lambda^{i''}}{2} - \sum_{C \in \mathcal{G}} \psi_C \lfloor \frac{|C \cap \bar{\mathcal{S}}''|}{2} \rfloor
\end{aligned}$$

where $i' \neq i''$ and $\mathcal{G} \subset \mathcal{C}$ is the set of all triplets whose corresponding subset-row inequality is in the RMP. Dominance rules have to be relaxed since the cost of a path is no longer monotonic and so label l' dominates l'' if:

$$\begin{aligned}
\mathcal{S}' &\subseteq \mathcal{S}'' \\
\delta' &\leq \delta'' \\
\tau' &\leq \tau'' \\
C' - \sum_{C \in \mathcal{V}} \psi_C &\leq C''
\end{aligned}$$

with $i' = i''$ and $V = \{C \in \mathcal{G} : \psi_C \leq 0, q'_C > q''_C\}$. Finally the values of $q_C \, \forall C \in G$ have to be taken into account when two partial paths are joined together to compose a complete route and in particular we check if $q'_C = 1 \wedge q''_C = 1 \, \forall C \in \mathcal{G}$ the value ψ_C is subtracted from the cost of the route.

2-path Inequalities

The classical 2-path inequalities, originally introduced by Kohl et al. [65], were developed for the capacitated VRP with time windows and impose that at least two routes must be used to visit every set $Q \subseteq \mathcal{N}$ of nodes that cannot be served by a single route either due to capacity or time limitations.

in a similar way as done by Desaulniers et al. in [89], we have translated this idea in our problem. Given such a set Q and taking into account that our sites can be skipped and that there are two different distribution systems, namely routes and distribution centers,

we have devised the following valid inequalities for the GLDP:

$$\sum_{i \in Q} s_i + \sum_{w \in \mathcal{W}(Q)} e_w^Q y_w + \sum_{k \in \mathcal{K}(Q)} g_k^Q z_k \geq 2 \qquad \forall Q \subseteq \mathcal{N} \tag{4.40}$$

where $\mathcal{K}(Q)$ and $\mathcal{W}(Q)$ denote respectively the set of routes and the set of clusters serving at least one site belonging to Q. Coefficients e_w^Q and g_k^Q are equal to:

- two if the variables y_w or z_k encode respectively a cluster or a route containing all sites in Q,
- one if a cluster or route serves Q only partially,
- zero otherwise.

In this way we obtained a cut that, if conveniently updated with the newly generated columns, is always valid for our problem. In fact if any of the three variables s, z, and y as well as the two coefficients are not considered in the generation of the cuts this may lead to inequalities that can cause the discarded of the optimal solution. Let consider the case where in the optimal solution a variable z_k representing a route that visits all the nodes in S takes value 1, if, during the solving process, we introduce into the MP a cut like the inequality of type (4.40) with the coefficient g_k^Q always equal to 1, the optimal solution will be dropped as it will never be feasible with respect to this additional inequality.

We observe that, in the traditional VRP, given a set Q that cannot be served by a single route but which is visited by less than two vehicles in a fractional solution it is always possible to find a violated 2-path inequality. This is not always true for the GLDP, as all coefficients e_w^Q and g_k^Q for the columns in the RMP can be equal to two. In order to avoid useless checks we perform three steps: we identify sets Q potentially leading to a violated cut (identification step), we check necessary conditions for a violated cut based on Q to exist (filtering step) and only for the unfiltered Q sets we proceed to actual separation (coefficients setting).

The generation step is carried out as in [65], considering sets Q having

$$\sum_{i \in Q} s_i + \sum_{l \in \mathcal{L}} \sum_{w \in \mathcal{W}_l} a_{iw} y_w + \sum_{l \in \mathcal{L}} \sum_{h \in \mathcal{H}} \sum_{k \in \mathcal{K}_{lh}} b_{ik} z_k < 2.$$

Then we filter each such a set Q by checking the following necessary conditions. Let $\bar{\mathcal{L}} \subseteq \mathcal{L}$ be the set of locations hosting DCs serving some customer in Q in the current fractional

solution, let $\tilde{\mathcal{L}} \subseteq \mathcal{L}$ be the set of locations hosting depots, where routes serving customers in Q start and let $\tilde{\mathcal{H}} \subseteq \mathcal{H}$ be the set of vehicle types used to serve customers in Q. In order for a violated cut to exist, at least one of the following conditions must hold:

- a DC $l \in \bar{\mathcal{L}}$ exists, having not enough capacity to fully serve Q ($\sum_{i \in Q} d_i > p_l$);
- a DC $l \in \bar{\mathcal{L}}$ exists, having sites in Q outside of its range ($\max_{i \in Q}\{c_{li}\} > r_l$);
- a vehicle type $h \in \tilde{\mathcal{H}}$ exists, having not enough capacity to fully serve Q ($\sum_{i \in Q} d_i > f_h$);
- a depot $l \in \tilde{\mathcal{L}}$ exists, from which no route can visit all sites in Q within the deadline M, i.e. such that $HP(Q \cup \{l\}) > T$, where $HP(Q \cup \{l\})$ is the time taken to perform an Hamiltonian Path on $Q \cup \{l\}$.

We check these conditions in this order. As soon as one of them is found to be true, we move to the coefficients setting step; if all of them are false, no violated cut based on Q can be found, and therefore we skip the coefficient setting step and we consider another set Q.

Finally, in the coefficient setting step, all columns are checked to fix the values of coefficients e_w^Q and g_k^Q according to the definitions given above. As outlined before, when all e and g coefficients take value two, the generated inequality does not help in cutting the current fractional solution; however, since it might become useful when further columns are added to the RMP, it is added to the problem in any case.

These additional inequalities introduce a new vector χ of variables in the dual problem that requires the modification of the pricing algorithm in order to take them into account properly.

In details, let $\mathcal{Q}$ be the collection of all the subsets Q whose corresponding 2-path inequality is in the RMP. A new set of dual variables $\gamma_Q \geq 0$ is introduced, one for each $Q \in \mathcal{Q}$, that requires the modification of the pricing algorithm.

We modified the exact dynamic programming algorithm for the RCESPP adding a new resource, one for each $Q \in \mathcal{Q}$, in the labels. Such resource is initialized at 0; after the first visit to a customer in Q, the resource is set to 1, and the reduced cost of the label is decreased by the value of the corresponding dual variable γ_Q. If it occurs that a label encodes a route visiting all sites in Q then the value of the resource is set equal to 2 and the value of dual variable γ_Q is subtracted again from the reduced cost of the label. In the joining phase if the resource is equal to 1 for both forward and backward paths we evaluate

the complete path in order to find out whether it serves all sites in Q, if this is the case then the dual variable associated with the corresponding cuts is subtracted from the reduced cost of the complete path. Finally dominance rules are modified with an additional condition saying that no label having one of these resources at 1 or 2 can dominate a label having the corresponding resource at 0.

The knapsack algorithm instead was modified as follows: the labels become $(i, u, \mathcal{S}, \vec{q}, \vec{o}, v)$, where $\vec{o}$ is a vector having one component o_Q for each set $Q \in \mathcal{Q}$ whose 2-path inequality is in the RMP, represented the number of elements of Q included in $\mathcal{S}$. Our label pushing algorithm is modified as follows. First, we create an initial label $(0, 0, \emptyset, (0 \dots 0), (0 \dots 0), 0)$. Then, we iteratively extend labels in order of increasing values of i and u, and when extending a label $(i, u, \mathcal{S}, \vec{q}, \vec{o}, v)$, two labels are created:

- a label $(i+1, u, \mathcal{S}', \vec{q'}, \vec{o'}, v')$ corresponding to the choice of skipping element $i+1$, with $\vec{o'} = \vec{o}$ and $v' = v$,
- a label $(i+1, u+d_{i+1}, \mathcal{S}'', \vec{q''}, \vec{o''}, v'')$, corresponding to the choice of including element $i+1$, with $o''_Q = |Q \cap \mathcal{S}''|$, for each set Q,

$$v'' = v - \lambda_{i+1} - \sum_{C \in \bar{\mathcal{C}}} \psi_C - \sum_{Q \in \bar{\mathcal{Q}}'} \gamma_Q - \sum_{Q \in \bar{\mathcal{Q}}''} \gamma_Q$$

where

$$\bar{\mathcal{Q}}' = \{Q \in \mathcal{Q} : o''_Q > 1 \text{ and } o_Q = 0\}$$

and

$$\bar{\mathcal{Q}}'' = \{Q \in \mathcal{Q} : o''_Q = |Q| \text{ and } o_Q < |Q|\}.$$

We remark that now the contribution of each γ_Q can be added to the reduced cost of a label up to two times: when the first element of Q is visited (since the $e_w^{\mathcal{Q}}$ coefficient in the corresponding column changes from 0 to 1), and when all elements of Q are visited (since $e_w^{\mathcal{Q}}$ changes from 1 to 2).

Resources $\mathcal{S}$ and $\vec{q}$ are updated as before. After this modification to the pricing algorithm,

a label $\ell' = (i, u, \mathcal{S}', \vec{q'}, \vec{o'}, v')$ can dominate a label $\ell'' = (i, u, \mathcal{S}'', \vec{q''}, \vec{o''}, v'')$ if

$$v' - \sum_{C \in \tilde{\mathcal{C}}} \psi_C \leq v'' - \sum_{Q \in \tilde{\mathcal{Q}}'} \gamma_Q - \sum_{Q \in \tilde{\mathcal{Q}}''} \gamma_Q, . \tag{4.41}$$

where

$$\tilde{\mathcal{Q}}' = \{Q \in \mathcal{Q} : Q \setminus \{1 \dots i\} \neq \emptyset \text{ and } o''_Q = 0 \text{ and } o'_Q > 0\}$$

and

$$\tilde{\mathcal{Q}}'' = \{Q \in \mathcal{Q} : |Q \cap \{1 \dots i\}| = o''_Q \text{ and } o''_Q > o'_Q\}.$$

In this case, a contribution γ_Q may be gained by ℓ'', but not by ℓ' in two cases, modeled by the last two terms of expression (4.41). First, ℓ'' may include for the first time an element of Q, while ℓ' cannot; this can happen only if (a) elements of Q can still be included in $\mathcal{S}''$ (b) no element of Q was already included in $\mathcal{S}''$ (c) at least an element of Q was already included in $\mathcal{S}'$. Second, ℓ'' may include all the elements of Q, while ℓ' cannot; this can happen only if (a) all those elements $1 \dots i$ belonging to Q were selected in $\mathcal{S}''$, while some of them were not selected in $\mathcal{S}'$. Even if the dominance criterion is slightly loosened, we experimentally observed that the DC pricing algorithm remains very efficient, also after the introduction of 2-path inequalities.

Consistency Cuts

We identified constraints (4.6) as the most likely cause of a possible poor linear relaxation of the MP since they may promote the use, in a fractional solution, of a facility location as both depot and distribution center. In fact, for every location $l \in L$ and every kind of vehicle $h \in \mathcal{H}$, may be convenient to either set several z_k to 1 assigning only a fractional value to the corresponding y_w or to assign a fractional value to many y_w and z_k related to the same location in order to partially cover as many sites as possible (see Figure 4.5). For this reason we devise new constraints that deal with this issue, that is:

$$z_k + \sum_{w \in W_l} y_w \leq 1 \qquad \forall h \in \mathcal{H}, \forall l \in \mathcal{L}, \forall k \in K_{lh}. \tag{4.42}$$

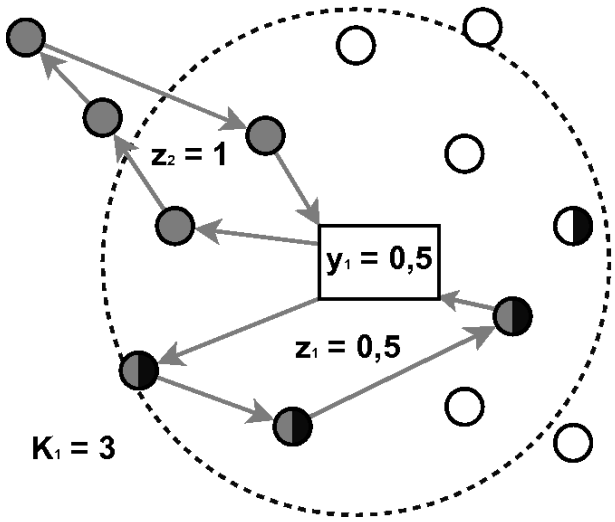

Figure 4.5: A partial solution using a location as both distribution center and depot without violating constraints (4.6)

They impose that if a route starting from a location l is used then all y_w, $\forall w \in W_l$ must be equal to zero. It is easy to see that although these constraints are able to cut solutions feasible for the constraints in the MP and seem just an improvement of the constraints (4.6) they actually cannot substitute them as there are solutions, as shown in Figure 4.6, which are feasible with respect to (4.42) that at the same time are infeasible for constraints (4.6).

Since there is an exponential number, equal to the number of the variables z_k, of constraints (4.6) it is not possible to add them all to the MP, instead in our algorithm a dynamic generation of these constraints is adopted. In particular every time a variable z_k encoding a route starting from the depot l is introduced in the MP its corresponding constraints involving all $y_w \forall w \in W_l$ is generated and added to the MP. Moreover every time a variable y_w encoding a cluster of the DC l is generated, it is added to every constraint of type (4.42) whose corresponding z_k variable encode a route starting from l. The introduction of these constraints trigger a modification of the pricing subproblems and consequently an alteration of the pricing algorithms. Let $\mathcal{P}$ and ν be respectively the set of all inequalities of type (4.42) that are in the MP and their associated non-negative dual vector of variables as the constraints (4.42) were rewritten as $\geq$ inequalities. Let $\mathcal{P}(l)$ be the subset of $\mathcal{P}$ made up of all the constraints which involve variables z_k and y_w encoding a route and clusters using

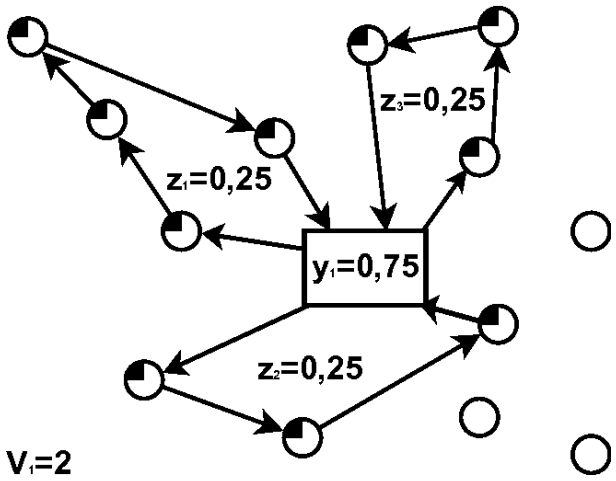

Figure 4.6: A solution feasible with respect to (4.42) but infeasible for constraints (4.6)

facility location l. Then, on the one hand in the knapsack pricer no algorithmic modification is required, indeed the new dual variables appear as a constant, whose value is equal to $\sum_{p\in\mathcal{P}(l)} \nu_p$, in the objective function of the pricing subproblem (4.21). On the other hand adjustments of the dominance and joining rules of the exact dynamic programming algorithm for the RCESPP are required to tackle the dual value ν. In particular, since the introduction of these new constraints break the monotonicity of the costs along the path, when comparing two label $l' = (\mathcal{S}', \delta', \tau', C', i)$ and $l' = (\mathcal{S}'', \delta'', \tau'', C'', i)$ the former dominates the latter if:

$$\mathcal{S}' \subseteq \mathcal{S}'' \tag{4.43}$$

$$\delta' \leq \delta'' \tag{4.44}$$

$$\tau' \leq \tau'' \tag{4.45}$$

$$C' + \nu_{max} \leq C'' \tag{4.46}$$

where the value ν_{max} is computed as the maximum ν among all the dual variables corresponding to constraints whose associated z_k variable encodes a route that is either equal to the path identified by l' or can be obtained extending the partial path represented by l'. During the joining phase we have to check if the complete $q - v$ path produced linking together a forward and a backward path represents a route that is already encoded by a variable z_k in the MP, in this case the complete path is discarded as useless since already in

the MP. In order to minimize the computational effort require to find ν_{max} we use a custom data structure whose organization is detailed in Appendix B.

These new cuts cause a modification in the standard structure of the branch-and-cut-and-price algorithm. Usually the generation of columns and rows is performed in two different stages: columns are generated by pricing algorithms while additional rows are found by cutting procedures. Instead, with the introduction of the consistency cuts our algorithm generates, every time a pricing procedure finds a new variable encoding a route, a column and a row and adds them to the MP at the same time.

4.4.5 Branching

When a fractional solution arises at the end of the root node we use six different branching policies to recover integrality. In particular, they are considered in the order presented thereafter and branching is executed using the first applicable rule. In the following we indicate as $\bar{y}$, $\bar{z}$ and $\bar{s}$ the values taken by the variables in the optimal LP relaxation of MP.

Branching on Locations. At first we try to force the exclusive use of every open location $l \in \mathcal{L}$ either as depot or a distribution center. In details, let $\mathcal{K}(l)$ and $\mathcal{W}(l)$ be the sets of the variables encoding respectively routes or clusters whose associated depot and distribution center is located in l and let $K(l) = \sum_{k \in \mathcal{K}(l)} \bar{z}_k$ and $W(l) = \sum_{w \in \mathcal{W}(l)} \bar{y}_w$. If $K(l) > 0 \wedge W(l) > 0$ then we branch as follows: two children nodes are generated; in the first we force l to be used as distribution center imposing $K(l) = 0$; in the other l will be used as depot since we impose $W(l) = 0$.

Branching on Skip Sites. When none of the conditions $K(l) > 0 \wedge W(l) > 0 \, \forall l \in \mathcal{L}$ are satisfied then a second branching rule is taken into account. Let $s_{\bar{i}}$ be the variable whose value is fractional and closest to 1. Branching is performed in the following way: two children nodes are generated and in the first the visit of the site $\bar{i}$ is forbidden imposing $s_{\bar{i}} = 1$ on the contrary in the other node $s_{\bar{i}} = 0$ and therefore the site $\bar{i}$ must be visited.

Branching on Clusters. If all skip variables are integer then a third branching rule is considered. We evaluate the sum of the variables representing clusters serving site i as

$$D_i = \sum_{l \in \mathcal{L}} \sum_{w \in W_l : i \in \mathcal{R}_w} \bar{y}_w.$$

If this value is not integer then we identify the set of distribution centers giving a fractional contribution to D_i that is

$$\mathcal{L}(D_i) = \{l \in \mathcal{L} : 0 < \sum_{w \in W_l : i \in \mathcal{R}_w} \bar{y}_w < 1\}.$$

Then branching is performed: two children for every $l \in \mathcal{L}(D_i)$ are generated imposing to serve site i only from the DC l in the first node and forcing not to serve the site from the DC in l in the other node. To tackle this branching rule in the knapsack pricing algorithm we proceeded in the following way, let $\mathcal{F} \subseteq \mathcal{N}$ be the set of sites that are forced to be served by a distribution center in l, then, all sites either forbidden or forced are removed from $\mathcal{N}(r_l)$ that is the set of available sites for distribution centers located in l. After that, if there are sites whose visit is forced the label $B[0,0]$ used in the exact dynamic programming as starting origin is initialized with $\mathcal{W}(0,0) = \mathcal{F}$ and $v(0,0) = -\sum_{i \in \mathcal{F}} \lambda_i$ while the maximal size of the knapsack is reduced and becomes $p_l = p_l - \sum_{i \in \mathcal{F}} d_i$ and the algorithm is executed as previously described.

Branching on the Number of Vehicles. When all D_i are integer $\forall i \in N$ a fourth branching policy is used. Let m_h be the number of routes using a vehicle of type $h \in \mathcal{H}$. The value m_h is computed as $\sum_{l \in \mathcal{L}} \sum_{k \in \mathcal{K}_{lh}} \bar{z}_k \in$ that is the sum of the values of all variables encoding routes using a vehicle h regardless of the originating depot. After computing all m_h for every type of vehicle a binary branching is executed on the m_h whose fractional value is closest to 0.5. Two nodes are generated, in the first it is imposed to use at least $m_h + 1$ variables belonging to $\cup_{l \in \mathcal{L}} \mathcal{K}_{lh}$ while in the other the maximum number of routes using vehicle $h \in \mathcal{H}$ is limited to m_h. This result is obtained modifying the left and right hand side of the constraint (4.5) corresponding to the selected h, in particular in the first case the modified constraint becomes as follows

$$m_h \leq \sum_{l \in \mathcal{L}} \sum_{k \in \mathcal{K}_{lh}} z_k \leq V_h$$

in the other node instead the constraint is

$$\sum_{l \in \mathcal{L}} \sum_{k \in \mathcal{K}_{lh}} z_k \leq m_h.$$

This branching technique leaves the pricing subproblem unchanged: dual variables μ_h still appear as constants in the objective function of the pricing problem, but they are now unrestricted in sign. No modifications in the pricing algorithms are required to handle this branching technique.

Branching on Arcs. When the number of vehicles used is not fractional branching on arcs is considered. We choose the site $i \in \mathcal{N}$ that is split among the largest number of routes in the optimal fractional solution of MP and that has at least two open outgoing arcs. Then we forbid half of its outgoing arcs to be used in the first child node, and the other half to be used in the second child node. To handle these branching decisions in the pricing problem it is enough to set travel time of forbidden arcs to $+\infty$.

Branching on Variables Finally when none of the previous rules can be applied branching on variables encoding either routes or clusters is performed. Among all y and z variables in the RMP let $\Upsilon_{\bar{i}}$ be the one whose value, in the optimal LP solution of the MP, is fractional and closest to $0,5$. The branching is performed in the following way: two children nodes are generated. In the former the usage of the variable $\Upsilon_{\bar{i}}$ is forced setting $\Upsilon_{\bar{i}} = 1$, while in the latter the variable $\Upsilon_{\bar{i}}$ is forbidden imposing $\Upsilon_{\bar{i}} = 0$.

4.4.6 Stabilization

In order to mitigate the possible convergence problems of the column generation algorithm, we implemented a stabilization technique whose general description is reported in Appendix A. This stabilization method heavily relies on the computation of an optimal solution to the Lagrangian relaxation of the RMP, however this is not a difficult task. In fact, given

dual vectors λ, μ, σ and ρ a Lagrangean relaxation of our problem reads:

$$
\begin{aligned}
\min\ & \sum_{i\in\mathcal{N}}(d_i-\lambda_i)s_i+\sum_{i\in\mathcal{N}}\lambda_i-\sum_{l\in\mathcal{L}}\sum_{h\in\mathcal{H}}V_h\sigma_{lh}+\\
&+\sum_{l\in\mathcal{L}}\sum_{h\in\mathcal{H}}\sum_{k\in\mathcal{K}_{lh}}(\sum_{i\in\mathcal{N}}a_{ik}\lambda_i+\sigma_{lh})z_k+\\
&+\sum_{l\in\mathcal{L}}\sum_{w\in\mathcal{W}_l}(\sum_{i\in\mathcal{N}}b_{iw}\lambda_i+\sum_{h\in\mathcal{H}}V_h\sigma_{lh})y_w\\
\text{s.t.}\ & \sum_{l\in\mathcal{L}}\sum_{k\in\mathcal{K}_{lh}}z_k\le V_h \qquad \forall h\in\mathcal{H}\\
& \sum_{l\in\mathcal{L}}\sum_{w\in\mathcal{W}_l}y_w\le B\\
& 0\le s_i\le 1 \qquad \forall i\in\mathcal{N}\\
& 0\le z_k\le 1 \qquad \forall l\in\mathcal{L},\forall h\in\mathcal{H},\forall k\in\mathcal{K}_{lh}\\
& 0\le y_w\le 1 \qquad \forall l\in\mathcal{L},\forall w\in\mathcal{W}_l
\end{aligned}
$$

The objective function comes from the relaxation of constraints (4.4) and (4.6) of the MP while original constraints (4.5) and (4.7) appear unchanged in the Lagrangean problem. This problems has the integrality property and its optimal solution is easily found by inspection analyzing the coefficients of the variables s_i, z_k and y_w. In particular, on the one hand there are no limits on the skip variables so every s_i whose coefficient is negative is set to 1, on the other hand the maximum number of both x_k and y_k is bounded and so for every kind of vehicle $h \in \mathcal{H}$ the V_h variables z_k with the most negative coefficients in the objective function are set to 1, in the same way among all y_w only the B with the most negative coefficients in the objective function are set to 1 all the other variables take value equal to zero.

4.4.7 Primal Heuristic Algorithm

In order to find feasible solutions during the computation we have developed a primal heuristic that looks for feasible solution among the solution space described by the variables that are in the RMP.

The algorithm goes through five different phases: location selection, distribution center allocation, allocation of vehicles, improvement and feasibility and it is executed once for every node of the branching tree.

In details indicating as $\bar{y}$, $\bar{z}$ and $\bar{s}$ the values taken by the variables in the optimal LP relaxation of RMP and defining $\mathcal{I}$ as the set of the variables composing the heuristic solution, the algorithm runs as follows.

Location Selection. In the first phase every location $l \in \mathcal{L}$ is marked as either potential distribution center or depot. The algorithm computes the use of every potential distribution center location u_l^{dc} and every potential depot u_l^{dep} as $u_l^{dc} = \sum_{w \in \mathcal{W}_l} \bar{y}_w$ and $u_l^{dc} = \sum_{h \in \mathcal{H}} \sum_{k \in \mathcal{K}_{lh}} \bar{z}_k$ then every location is marked as potential distribution center if $u_l^{dc} \geq u_l^{dep}$ and as potential depot otherwise.

Distribution Center Allocation. In this phase only variables in the RMP that encode clusters are taken into account. Let $\mathcal{L}(dc)$ be the set of location marked as potential distribution centers that is $\mathcal{L}(dc) = \{l \in \mathcal{L} : u_l^{dc} \geq u_l^{dep}\}$. All locations in $\mathcal{L}(dc)$ are sorted in descending order by their usage value u_l^{dc} then, following this order, for every variable y_w with $w \in W_l$, associated to the location l taken currently into account, its covering potential cp_w^{dc} is computed as the sum of the demands of the sites covered by the cluster described by y_w that are not already served by any variable in $\mathcal{I}$, more formally

$$cp_w^{dc} = \sum_{i \in R_w : i \notin \{\cup_{j \in \mathcal{I}} R_j\}} d_i.$$

The variable y_w whose corresponding cp_w^{dc} is maximum is selected and added to the solution set $\mathcal{I}$, then the algorithm moves to the next location in $\mathcal{L}(dc)$ and this process is iterated until either there are no more locations in $\mathcal{L}(dc)$ or B variables are added to the set $\mathcal{I}$. If no cluster $w \in W_l$ associate with a potential location l are selected the location is removed from $\mathcal{L}(dc)$ and marked as potential depot location.

Vehicle Allocation. In this phase the variables encoding routes are taken into account and allocated to the potential depot location. Vehicle by vehicle going in descending order by capacity, the algorithm computes the covering potential cp_k^{ro} of every route $z_k \in K_{lh} \forall l \notin \mathcal{L}(dc)$ using a vehicle $h \in \mathcal{H}$ and that does not start from a distribution center as follows:

$$cp_k^{dc} = \sum_{i \notin R_k : i \notin \{\cup_{j \in \mathcal{I}} R_j\}} d_i.$$

The variable z_k whose cp_k^{ro} is the maximum among all variables using vehicle h is selected and added to $\mathcal{I}$. This process is repeated until either there are no more variables associated with vehicle h or V_h variables are selected. Then the algorithm moves to the next vehicle. This phase terminates when all the types of vehicles are taken into account. It is worth noting that at this point the value of the solution can be computed as the sum of the demand of the sites not visited by any variable in $\mathcal{I}$.

Improvement. A simple local search procedure is applied during this phase aiming at improving the quality of the solution found. In details, the algorithm evaluates, for every variable y_w or z_k in $\mathcal{I}$, if it is replaceable by another variable, not in the solution set, that improves the value of the solution. This is done removing the variable y_w or z_k from $\mathcal{I}$ and recomputing the covering potential for all the variables not in $\mathcal{I}$ that are compatible with the removed one and selecting the y'_w or z'_k that is associated with the maximum potential. Needless to say, the variable y'_w or z'_k can be the exactly same variable previously removed. This procedure is iterated, following different politics for selecting the variables in $\mathcal{I}$, until the solution value cannot be further improved exchanging any variables in $\mathcal{I}$ with any other ones.

Feasibility. The last phase of the algorithm tries to ensure the feasibility of the solution. The algorithm considers all variables in $\mathcal{I}$ and computes the set of the sites served $\mathcal{S}(\mathcal{I}) = \{i \in R_j : j \in \mathcal{I}\}$ then adds to $\mathcal{I}$ all skip variables s_i associated with sites not in $\mathcal{S}(\mathcal{I})$. At this point the set of variables $\mathcal{I}$ represents a complete solution for the GLDP. This procedure always generates feasible solutions when executed at the root node but it may produce, due to branching decisions, infeasible solutions in some node of the search three.

4.5 Experimental Analysis

Implementation. The algorithms have been implemented in C++, using SCIP 2.1 [69] as branch-and-cut-and-price framework linked to CPLEX 12.2 as pure LP solver. SCIP is used as a branch-cut-and-price framework and it takes care of the management of the column and row pools and it is able to remove and re-insert them as needed. All presolving algorithms and the automatic cut generator embedded in SCIP along with the following general purpose heuristic algorithms: shift-and-propagate, fix-and-infer, int-diving, DINS, RENS, RINS, undercover, mutation, crossover and shifting have been disabled as they were

considered either incompatible or useless for our problem. All remaining SCIP parameters were kept at their default values including the branching tree exploration strategies.

The experimental campaign has been performed on a single core of a PC Intel® Core 2 Duo™ CPU T7300 (2.00 GHz) with 4 GB RAM and running Ubuntu 10.04 operating system. A time limit of 1 hour was imposed to each run.

Benchmark Instances. In order to properly test our approach we generated 440 instances starting from the approach detailed by Shen et al. [43]. We took into account three different problem size: 10, 20 and 50 delivery sites and we set the number of potential facility locations to 2, 3 and 5 respectively. For each problem size we randomly generated, following a uniform distribution, five different coordinate settings for every delivery site and every potential facility location. Moreover for every delivery site the demands were generated using the same uniform distribution. We built different scenarios for every setting modifying the availability of the resources in the following way: on the one hand we computed the total supply D as the sum of all site demands and for every setting we generated scenarios with 70, 80, 90 and 100% of the total supply. On the other hand the deadline M for every instance was set to 40, 50, 60, 80, 100% of the base route length that is computed as the average length of all the edges in the graph times the average number of served nodes per vehicle, that is $\frac{|\mathcal{N}|}{T}$. Additional values for D (120, 160 and 200%) and for M (120%) are taken into account in small instances. For every combination of D and M, every problem size and every setting one instance was generated. In every instance three different types (V_1, V_2 and V_3) of vehicles were considered whose capacity was set, respectively, to 60, 80, and 120% of the average delivery capacity computed as $\frac{D}{T}$. In instances with 10 sites $T = 4, V_1 = 2, V_2 = 1, V_3 = 1$, in the instances with 20 sites $T = 5, V_1 = 2, V_2 = 2, V_3 = 1$ and in instances with 50 sites $T = 11, V_1 = 5, V_4 = 1, V_3 = 1$.

4.5.1 Stabilization Impact

The evaluation of the usefulness of the stabilization method has been done executing all instances in our test set with six different values of parameter α: 0.00 (value that disables the stabilization technique), 0.02, 0.05, 0.01, 0.2, 0.5. The computation was carried out using only exact pricers, none of the cuts and was stopped at the root node. The results of the stabilized algorithm compared with the non stabilized one are reported in tables 4.1 and 4.2. They show, for every class of instances (10, 20 and 50 nodes) the maximum, average and

minimum percentage variation in the number of pricing iterations and in computational time. As it is clearly visible in the tables the results are characterized by a very high variability, despite that some trend may be identified. On small instances, with 10 nodes, the stabilization method is able to reduce, on average, the number of pricing iterations by 30%. However, this achievement fades away with the increase of the instance size and in fact only about 7% of pricing iterations are saved in instances with 20 nodes and going up to the largest instances the stabilization method starts requiring on average more pricing iterations than the non-stabilized one. Nevertheless, it is often possible to find a value of α which reduces the number of iterations needed. The issue is that this value is not constant but varies for every instance.

		Pricing Iterations % variation				
Inst.		$\alpha = 0.02$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.5$
10	Avg	-30.61	-30.97	-29.90	-31.11	-33.19
	Max	57.89	57.89	57.89	57.89	57.89
	Min	-166.67	-166.67	-166.67	-166.67	-166.67
20	Avg	-5.84	-7.70	-6.90	-9.50	-9.94
	Max	47.83	48.44	46.67	48.23	47.76
	Min	-208.33	-208.33	-208.33	-208.33	-171.43
50	Avg	23.70	21.94	24.21	25.96	28.82
	Max	61.07	61.07	63.57	65.54	70.52
	Min	-26.46	-28.77	-16.05	-11.84	-5.95
Tot	Avg	-10.94	-12.06	-10.80	-11.75	-12.22

Table 4.1: Impact of the stabilization method on number of pricing iterations

From the computational time point of view the introduction of a stabilization method slows down the solution process by, on average, about 19%. In fact even when the number of pricing iterations is reduced the computational time required to optimally solve the relaxed MP is very often greater than without stabilization. Two aspects are responsible for this increment, on the one hand every pricing iteration of the stabilized algorithm may require, when mispriced columns are generated, more than one call to the pricing algorithm on the other hand stabilized dual variables may lead to more difficult pricing subproblems requiring more time to be solved. Numerically, the stabilized version of the algorithm makes about 2% more calls to the pricing algorithm than the non-stabilized one. As to the pricing subproblems we have analyzed the number of labels extended during the exact dynamic programming procedure (whose average variation is reported in table 4.3) for the

RCESPP and we found out the stabilization method can either lead to easier or harder pricing subproblems. In particular on instances with 20 nodes it always requires much more labels to solve the dynamic programming procedure while on the other instances the stabilization makes the pricing subproblems easier, anyway this variation does not compensate for the increment in the number of pricing iteration on instances with 50 nodes and makes the overall algorithm slower than without the stabilization method.

		Computational Time % variation				
Inst.		$\alpha = 0.02$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.5$
10	Avg	23.81	23.57	24.29	25.00	25.83
	Max	100.00	100.00	100.00	100.00	100.00
	Min	-100.00	-100.00	-100.00	-100.00	-100.00
20	Avg	29.87	-4.80	6.14	11.35	23.33
	Max	78.85	41.59	43.79	61.71	81.59
	Min	-153.85	-76.47	-53.85	-66.67	-66.67
50	Avg	21.44	-5.07	-4.45	30.69	43.09
	Max	90.76	71.26	69.98	88.48	93.23
	Min	-163.65	-135.95	-187.78	-156.98	-144.67
Tot	Avg	25.07	8.65	12.38	22.23	29.00

Table 4.2: Impact of the stabilization method on the computational time

Observing the high variability of the impact of the stabilization technique and the difficulty in finding a good value for α, we decided to undertake the subsequent tests disabling the stabilization method.

		Labels DP % variation				
Inst.		$\alpha = 0.02$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.5$
10	Avg	-16.72	-15.88	-14.21	-12.74	-9.00
20	Avg	15.28	17.07	19.34	22.17	29.53
50	Avg	-38.66	-38.68	-37.44	-5.88	13.39

Table 4.3: Impact of the stabilization method on the RCESPP pricer

4.5.2 Cut Comparison

We have tested our approach on all instances using different combinations of the cuts presented in Section 4.4.4 in particular five different configurations have been tested:

NC : no cuts are used in this configuration.

CC : only consistency cuts are taken into account.

SC : only subset-row inequalities are considered.

2C : 2-path inequalities are the only cuts in this configuration.

AC : all cuts (consistency, subset-row and 2-path) are used.

The percentage variations, aggregated over every instance size, in the number of the branch and bound nodes explored (B.B.N.), in the value of the lower bound at the root node (L.B.) and in the computational time (Time) are reported in Tables 4.4 and 4.5. The first thing that attracts the attention looking at the figures is the very little increment given by any of the cuts to the lower bound. In fact, the lower bound is influenced by the cuts only on instances with 10 nodes and by at most 1.42% using the AC configuration. However, the introduction of cuts marks a more solid difference in the number of nodes explored and in the computational time, in particular on the largest instances. In this case all cuts but the subset-row inequalities are able to reduce both the number of the nodes explored and the computational time. Configurations CC and AC turned out to bring the greatest improvement reaching a reduction by more than 30% in both values. It must be said that on the one hand, since a consistency cut is added every time a variable z_k is generated, there are a lot of them in the MP: the average number of consistency cuts added in instances with 50 nodes is about 500. On the other hand only a handful, less than 10, of 2-path and subset-row inequalities are introduced in the MP in instances of the same size. Subset-row inequalities are the less efficient cuts in this problem and indeed introduce almost as much delay as the consistency cuts but are not able to reduce the number of nodes explored on any instance with more than 10 nodes. The 2-path inequalities instead does not introduce too much additional computational effort and can reduce the number of nodes explored in instances with 10 and 50 nodes. Anyway, all separation procedures are pretty fast, 2-path inequalities are separated in less than one tenth of a second while the exact subset-row separator runs, on average, in less than a second. No additional computational effort is required to introduce consistency cuts since they are instantaneously created and added when columns are generated.

In Table 4.6 the number of instances optimally solved with every configuration of cuts is reported. The maximum number of instances is solved used the CC configuration, AC and

	AC			CC		
Inst.	B.B.N.	L.B.	Time	B.B.N.	L.B.	Time
D10	-7.59	1.42	34.88	-1.90	1.20	16.28
D20	5.33	0.00	33.15	16.77	0.00	21.00
D50	-46.86	0.00	-36.56	-21.58	0.00	-33.38

Table 4.4: Configuration AC and CC - Variation % compared with NC

	2C			SC		
Inst.	B.B.N.	L.B.	Time	B.B.N.	L.B.s	Time
D10	-1.58	0.07	7.75	-7.28	0.19	31.01
D20	4.08	0.00	8.94	1.34	0.00	17.67
D50	-7.73	0.00	-11.29	12.97	0.00	19.99

Table 4.5: Configuration 2C and SC - Variation % compared with NC

2C follows with respectively a total of 434 and 433. Using only the subset-row inequalities does not increment the number of instances solved compared with the NC configuration. In the next section we give a detailed analysis of the results achieved with CC configuration.

Insts.	NC	AC	CC	2C	SC
D10	210	210	210	210	210
D20	130	130	130	130	130
D50	90	94	95	93	90
Total	430	434	435	433	430

Table 4.6: Number of instances solved for every configuration of cuts

4.5.3 General Performance

An overview of the computational results that our approach is able to achieve with the best combination of stabilization and cuts found in the previous sections is reported in tables 4.7 and 4.8 while detailed results for every singe instance can be found in Appendix E. Table 4.7 shows, aggregating the values for every class of instances, the average and maximum number of nodes explored in the branch and bound tree (B.B.N), the gap at the root node and at the end of the computation (Gap R%, Gap F%) and the computational time in seconds (Time[s]). In table 4.8 the percentage of instances solved (Solved), solved at root node (S. Root) and in which the lower bound at rood node is equal to the value of the

optimal solution (Opt LB) are reported.

Our algorithm is able to optimally solve 435 instances of the 440 available, in particular all instances with 10 and 20 nodes are closed in less than 20 seconds. The residual gap considering only the five instances with 50 nodes not solved to optimality ranges from 2.22% to a maximum of 6.38%.

Inst.		B.B.N	Gap R%	Gap F%	Time[s]
D10	Avg	1.48	1.27	0.00	0.01
	Max	19.00	63.17	0.00	0.05
D20	Avg	28.81	1.43	0.00	0.98
	Max	305.00	28.57	0.00	17.27
D50	Avg	80.27	0.21	0.21	387.62
	Max	1046.00	6.38	6.38	3600.19

Table 4.7: General Performance Evaluation - Search Tree and Residual Gaps

More than half of the instances are solved at the root node, when branching is required the size of the tree explored before an optimal solution is reached does not exceed 100 nodes on average.

The last column of the Table 4.8 shows an interesting result, it contains the percentage of instances in which the value of lower bound at the root does not further increase during branching since it is equal to the value of the optimal solution. This value is about 95% and is a certification of the strength of our mathematical formulation. Moreover it may also help in explaining the poor contribution given by the additional cuts, indeed if the possibly fractional value of the RMP is very close (or even equal) to the value of the optimal integer solution very few useful cuts can be added at the root node. Taking that into account this problem seems to be harder from the primal point of view than from the dual. Indeed although the dual bound is almost always optimal the optimal integer solution of the same value is not as easy to find and may require to branch several times before a feasible integer solution whose value is equal to the lower bound is found.

4.6 Conclusions

In this chapter we have presented a new problem, with a possible application in the distribution of drugs in case of emergency, that goes beyond the definition of location and routing and indeed we have called it Generalized Location and Distribution Problem (GLDP). It

Inst.	Solved %	S. Root %	Opt LB
D10	100.00	83.81	92.38
D20	100.00	36.15	93.42
D50	95.00	48.00	100.00
AVG	98.33	55.99	94.92

Table 4.8: General Performance Evaluation - Number of Instances Solved and LB Improvement

not only represents an extension and a generalization of the classical multi-depot heterogeneous fleet VRP with profits but also introduces a new degree of freedom identified by the availability of multiple distribution models that can be combined together on the same level in order to build a more complex distribution system.

We have devised a mathematical formulation that is compatible with our general framework and we presented a branch-and-price-and-cut procedure to optimally solve it. This algorithm includes multiple pricers, three different families of cuts, a primal heuristic procedure and several branching strategies. Two of the cuts presented are a generalization of cuts that can be found in the literature for which new separation procedures have been devised while the third cut is a new introduction in the context of column generation based algorithm. Moreover, in this chapter we described one of the first attempts to include the stabilization method by Pessoa et al. [90] in a problem whose complexity level is definitely higher than in the toy problems originally considered by the authors.

The approach has been tested on a set of 440 instances showing good results and reaching the optimal solution in almost all the instances. When the optimality is not proven we are however able to provide a valid lower bound and a feasible primal solution. Moreover the structure of the solutions suggests that our algorithm is able to quickly find the best lower bound while the most part of the computational time is spent for finding a feasible integer solution of the same value.

Chapter 5

Conclusions

In this thesis we have analyzed the nature of the location and routing problems, we identified three different families of problems based on the relevance of the location decisions on the structure of the problem and we proposed an algorithmic approach to solve them in a uniform way. Our investigation started from routing problems and gradually introduced location features involving multiple depots and combining different distribution strategies. From a modeling point of view the scenario is clear: the higher is the degree of freedom in the problem the more complex is its mathematical formulation. Variables are required to handle location decisions and additional constraints are introduced to model new limitations and to link together the two aspects of a location and routing problem. Despite that, from a practical point of view the additional complexity, introduced when location and routing features are combined together, can be handled by modern operations research techniques with an affordable additional computational effort. In particular with regard to the three families of problems analyzed, the introduction of multiple locations can be efficiently tackled in both the MP and the pricing subproblem employing, respectively, the consistency cuts, which strengthen the relationship between routing and location decisions in the MP, and the aggregated pricing method that allows for considering all depots at the same time. When multiple distribution strategies are combined together on the same network, as in the "Active Location Decisions" type of problems, the increased complexity of the problem is mainly reflected in the MP; in fact, to each distribution strategy corresponds a different pricing subproblem that can be solved independently and that generates variables whose interaction, with other variables coming from other pricing subproblems, is only visible in the MP. Moreover we observed that when multiple pricing subproblems

are involved the pricing phase can be quicker than in the case with only a single pricing subproblem. In fact, a careful managing of the execution order of the pricing algorithms in which easier subproblems are solved first and then, only when no other columns can be found, the more complex subproblems are considered, can substantially reduce the number of calls to the most time expensive pricing procedures. Overall, efficient multiple-pricing techniques, combined with the introduction of the consistency cuts, represent a viable way for solving reasonable size instances of problems with profits involving heterogeneous fleets of vehicles, multiple-locations and multiple distribution strategies.

Still, our approach is able to deal with all three families of problems regardless of their specific requirements and limitations. The branch-and-cut-and-price algorithm we presented is based on the revision, extension and improvement of several state of the art methods and brings in a few innovative aspects. In particular, pricing algorithms have to be extensively adapted in order to handle the nature of the problems addressed, new cut generation procedures needed to be devised and their reflection on the pricing subproblems had to be carefully taken into account to ensure the optimality of the approach. Primal heuristic algorithms, branching strategies and support data-structures were developed to give efficiency and completeness to the framework. During the experimental phase all these components were carefully tested and analyzed. One of the cuts proposed turned out to be particularly interesting. Indeed, it gives a robust contribution on speeding up the computation and it is characterized by a structure that is general enough to be easily adapted to many other LRP. The implementation of these cut generates a shift in the general branch-and-cut-and-price framework turning the nature of the pricing procedure from the usual generation of columns to the simultaneous generation of rows and columns.

On the whole, the experimental analysis demonstrates that our approach can be usefully applied to all problems and represents a viable way for handling elaborate problems involving both location and routing aspects and characterized by a multiplicity of non-trivial additional requirements. This achievement is further enhanced when considering the uniform implementation we proposed. In fact, all ideas and concepts presented in this thesis needed to be translated into computer codes in order to be properly executed and tested. In doing that we took advantage of the non-commercial framework for Mixed Integer Programming (MIP) SCIP [69]. Exploiting the great flexibility of the SCIP framework we were able to implement our approach on all the three different problems with only a few modifications from its general outline.

Within our framework we implemented a stabilization method proposed in 2008 by Pessoa et al. [90]. It is characterized by interesting properties and showed promising results when applied on simple problems by the authors. We gave a complete and detailed description of this method proving its properties. However, when we applied this stabilization strategy to complex location and routing problems its behavior turned out to be very variable showing a substantial reduction in computational time in some instances but a tangible increment in other ones making difficult to draw general conclusions.

Finally, new research streams are promoted by this thesis such as the study of how supplementary column variables, which do not represent combinatorial entities, can be used to formulate additional features and requirements that cannot be easily modeled in the pricing subproblems and what is their impact in the master problem. The introduction of a general location and routing problem involving two distinct distribution strategies mixed on the same network calls for further researches in this direction to address more complex and realistic logistic problems involving multiple delivery methods. Moreover, it appears clear from the experimental campaign that researches aimed at finding optimized and fast algorithms for the resource constrained elementary shortest path problem are essential, since in many practical cases they are the foundation of the exact pricing procedures used in several branch-and-cut-and-price algorithms for LRP.

As a concluding remark we would like to stress the fact that, considering recent advances in LRP and the results of this thesis, while hierarchical methods remain appealing to deal with large scale instances, they are likely to be outperformed by integrated ones when instances of reasonable size are considered.

Appendix A

A Stabilization Technique for Column Generation

The column generation is one of the most used technique for solving very large linear programming problems that would be intractable if all their variables were taken into account at the same time (see Barnhart et al. [56]). Although this method is very useful it may require, as described by Desrosiers et al. [91], a very long computational time to converge. In fact, column generation algorithms often show an instable behavior that is clearly represented by two critical issues: on the one hand, in the first iterations the algorithm is usually affected by dual degeneracy that can provoke a very high variability in the values of the dual variables leading to the generation of many useless columns; on the other hand, when the majority of the useful columns have been added to the RMP, the algorithms, due to primal degeneracy, may require several iterations to generate the last useful columns and to prove the optimality of the LP solution found; such a behavior is often called tailing-off effect (see Gilmore and Gomory [92]). To put a limit to this instability since the early seventies several stabilization methods have been devised, see Desrosiers and Lübbecke [93] for an overview of the most common stabilization strategies and Clautiaux et al. [94] as a recent successful implementation of effective stabilization method in a complex problem such as the cutting stock problem.

When huge problems, solved via column generation, show high levels of symmetry in the columns or are not tightly bounded they are likely to be affected by degeneracy and in this case the performance of the column generation algorithm may drop since several, apparently useless, iterations in which the value of the optimal solution does not change

are performed in order to leave the degenerate basis. For this reason we try to mitigate possible convergence problems using a stabilization technique that is based on the methods presented by Uchoa et al. [68].

The idea behind this stabilization technique is to keep during the column generation process a set of "good" dual variables to be used as stabilization center, when new dual variables that are perceived as better than the old ones are found, the stabilization center is repositioned. In details, let π_{be}, π_{rm} and π_{st} be, respectively, the best set of dual variables found (representing the stabilization center), the set of dual variables coming from the optimal solution of the RMP and the actual vector of dual variables used in the solution of the stabilized pricing subproblems; also let $Lag(\pi)$ be the optimal solution value of the Lagrangian relaxation of the MP computed using π as Lagrangian multiplier vector.

The algorithm starts initializing the π_{be} at a suitable value, then the RMP is solved obtaining an optimal LP solution of value Z_{RM} and, exploiting the dual representation of this solution, the vector of dual variables π_{rm}. Instead of considering directly π_{rm}, in similar way as proposed by Wentges [95], we compute a new vector of dual variable π_{st} as the linear combination of π_{be} and π_{rm} and we use it to solve the pricing subproblems. In particular the value of this vector is

$$\pi_{st} = \alpha \cdot \pi_{be} + (1 - \alpha) \cdot \pi_{rm} \tag{A.1}$$

where $\alpha \in [0, 1)$ is a user defined coefficient balancing the contributions of the best and the current dual variables. The pricing algorithms are executed using π_{st} and the optimal solution of the pricing subproblems gives as outcome a column j. Considering the solution of the pricing procedures, the stabilization algorithm evaluates if π_{st} makes a better stability center than π_{be}. This is done comparing the value of the Lagrangian relaxation of the RMP using π_{st} with the value achieved using π_{be}, if $Lag(\pi_{st}) > Lag(\pi_{be})$ then the stabilization center is relocated in π_{st}, that is $\pi_{be} = \pi_{st}$. Then, the reduced cost of j with respect to the current duals π_{rm} ($\xi_j(\pi_{rm})$) is assessed, two different cases may arise:

If $\xi_j(\pi_{rm}) < 0$ the column found is added to the RMP.

If $\xi_j(\pi_{rm}) \geq 0$ the column j is *mispriced* meaning that although its reduced cost computed using π_{st} may be negative, its real reduced cost is non-negative and thus adding j to the RMP is useless. However, finding a mispriced column is not a waste of time as proved by Theorem A.0.1, $Lag(\pi_{st}) \geq Lag(\pi_{be}) + \alpha(Z_{RM} - Lag(\pi_{be}))$ and consequently

the dual gap is reduced at least by a factor of $\frac{1}{1-\alpha}$. Therefore a mispriced column causes the relocation of the stabilization center on π_{st} and leads to a reduction of the gap between Z_{RM} and $Lag(\pi_{be})$, possibly improving the convergence. In this case, since no column was added, in the next iteration the re-computation of Z_{RM} can be skipped.

This process is iterated until $Z_{RM} - Lag(\pi_{be}) < \epsilon$. A pseudo-code outlining this stabilization method is reported in Figure A.1.

input : a parameter $0 \leq \alpha < 1$ and a small enough value ϵ
initialization: $\pi_{best} \leftarrow 0$;
while $Z_{RM} - Lag(\pi_{be}) < \epsilon$ **do**
  solve RMP obtaining Z_{RM} and π_{rm};
  $\pi_{st} \leftarrow \alpha \cdot \pi_{be} + (1-\alpha) \cdot \pi_{rm}$;
  solve pricing subproblems using π_{st} obtaining column j;
  if $Lag(\pi_{st}) > Lag(\pi_{be})$ **then**
    $\pi_{be} \leftarrow \pi_{st}$;
  end
  if $\xi_j(\pi_{rm}) < 0$ **then**
    Add column j to the RMP;
  end
end

Figure A.1: Stabilization Column Generation Method

Theorem A.0.1. *If the solution of the pricing problem with dual vector π_{st} does not give a column with negative reduced cost with respect to π_{rm}, then $\exists \epsilon > 0$ such that*

$$Lag(\pi_{st}) \geq Lag(\pi_{be}) + \epsilon(Z_{RM} - Lag(\pi_{be}))$$

Proof. We prove this theorem by contradiction. Suppose that $Lag(\pi_{st}) < Lag(\pi_{be}) + \epsilon(Z_{RM} - Lag(\pi_{be}))$ then

$$Lag(\pi_{st}) - \epsilon Z_{RM} < (1-\epsilon) Lag(\pi_{be}).$$

Since $Lag(\pi_{st}) = Lag(\alpha\pi_{be} + (1-\alpha)\pi_{rm}) \geq \alpha Lag(\pi_{be}) + (1-\alpha) Lag(\pi_{rm})$ from the concavity of the Lagrangian dual function (see Figure A.2) and considering that $Lag(\pi_{rm}) = Z_{RM}$

then

$$\alpha Lag(\pi_{be}) + Z_{RM} - \alpha Z_{RM} - \epsilon Z_{RM} < (1-\epsilon) Lag(\pi_{be})$$

and so

$$(1-\alpha-\epsilon) Z_{RM} < (1-\alpha-\epsilon) Lag(\pi_{be}) \tag{A.2}$$

that if $0 < \epsilon \leq 1-\alpha$ is false, since $Z_{RM} \geq Lag(\pi), \forall \pi$, by Lagrangian duality. □

Corollary 1. *If* $1-2\alpha > 0$ *then*

$$Lag(\pi_{st}) \geq Lag(\pi_{be}) + \alpha(Z_{RM} - Lag(\pi_{be}))$$

This can be easily proved substituting ϵ with α in the previous proof.

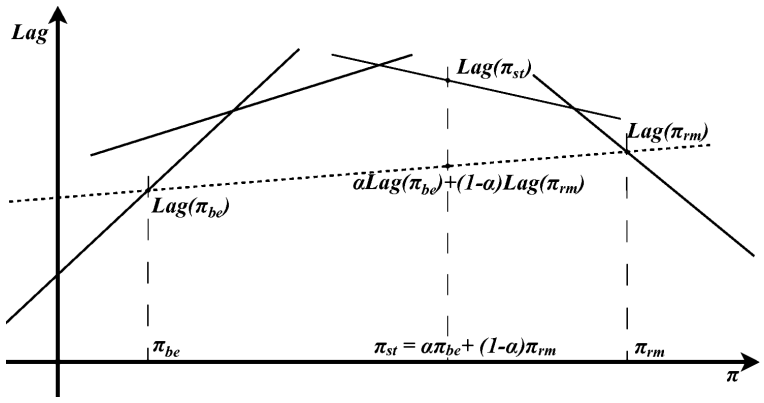

Figure A.2: Lagrangian function

Appendix B

A Route Search Tree

The data-structure presented in this appendix is used to efficiently handle the dual variables coming from the consistency cuts as described in sections 3.4.7 and 4.4.4. Since there are as many consistency cuts as route variables in the MP, checking their dual values by pure enumeration is not a viable option. However we can exploit three observations. First, consistency cuts whose corresponding route variable is not in the RMP are never active, and therefore the corresponding dual variable is 0. Second, given a partial path p only consistency cuts corresponding to route variables in the RMP which are compatible with p need to be checked. By compatible we mean that such a route either equals p or can be obtained by extending p. Third, among all routes compatible with path p, only the one corresponding to the cut having maximum dual value π_{max} is important in the computation of the reduced costs. Hence we devise a tree-like data-structure that allows us, once properly initialized, to find π_{max} in $O(l_p)$ time, where l_p is the length of partial path p.

Each leaf of this data structure T represents a route whose variable is in the RMP, and branches represent partial paths; routes sharing initial partial paths share also the same branches of the tree. Moreover, in each node t of the tree a value is stored, representing the value of the maximum dual variable corresponding to a route whose corresponding leaf in T belongs to the subtree rooted in t. An example of this data-structure is reported in Figure B.1, this tree is obtained with the six variables z listed in the picture.

Four operations need to be performed on T: initialization, route insertion, dual update and find. They are described hereafter.

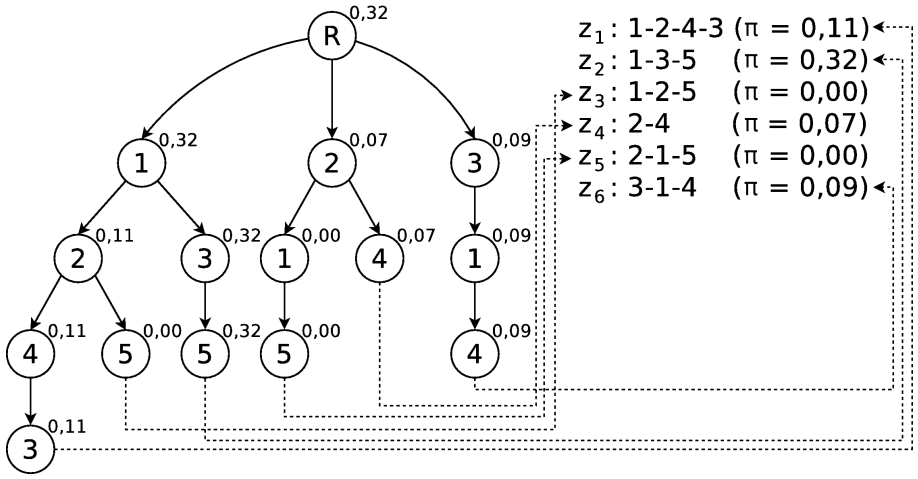

Figure B.1: An example of the route search tree

Initialization. At first the tree T is initialized with only one dummy node R that represents the starting point of every route. This operation can be done in $O(1)$.

Route Insertion. Every time a variable is added in the RMP its corresponding route r is inserted in T in the following way: T and r are traversed simultaneously starting respectively from R and from the depot. The first node of the route is visited and then the corresponding node in T, if it is available among the children of R, is visited and the process is iterated, considering only the subtree rooted in the last visited node of T, until either the end of the route is reached or there is no node in the first level of the current subtree corresponding to the last node visited in the route r. When the route diverges from T, that is when there is no branch in T that is equivalent to the partial route traversed, a new branch is created and all remaining nodes in r are added along that branch. The computational complexity of this operation is $O(N \log N)$ where N is the number of clients in the problem. An example of insertion of a new route in the tree is reported in figure B.2.

Dual update. Every time the RMP is solved new dual values are generated and the search tree needs to be updated as follows. The value of the dual variable of each consistency constraint is stored in the leaf of T corresponding to its route; a value 0 is stored in the inner nodes. Then the update proceeds level by level. The value stored in each node is compared with that of its father: if the value stored in the father is lower, then the father

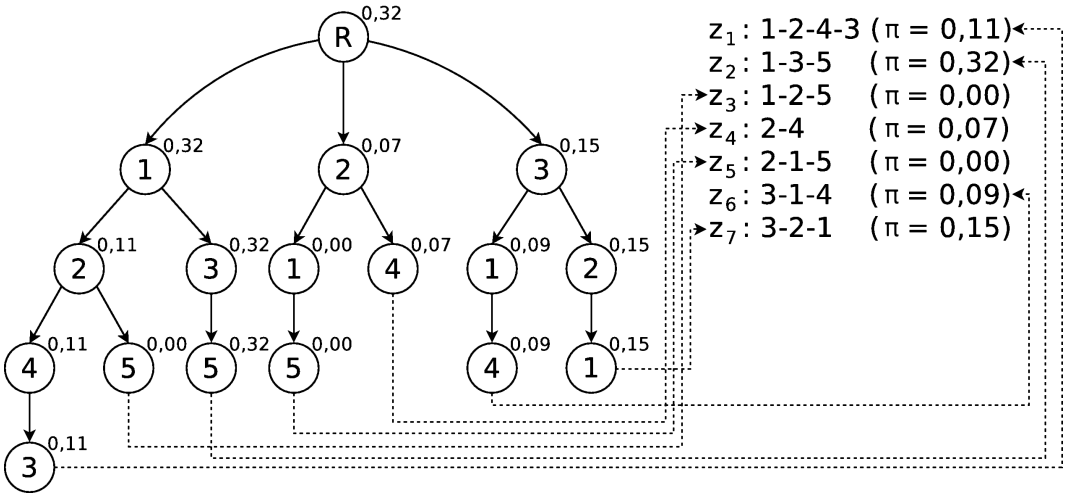

Figure B.2: The route search tree after the insertion of the route corresponding to variable z_7

is updated with the value of the child. The update process has a computational complexity $O(M \times N)$ where M is the number of variables in the RMP and N is the number of clients in the problem.

Find. If the tree is properly updated the value of π_{max} can be found in $O(l_p)$ visiting the branch of the tree corresponding to the partial path encoded by p, when such a branch does not exist $\pi_{max} = 0$.

Appendix C

Detailed Results: Waste Collection

The detailed results for the Waste Collection problem (see Chapter 2) are reported in this Appendix. In particular the tables contains the results on all instances taking into account three different values for L_{min}. The column headers used have the following mining:

Inst. is the instance name in the format *_number of bins_instance number*.

B.B.N. is the number of nodes explored in the branch and bound tree.

L.B.R. is the lower bound achieved at the end of the root node.

U.B.R. is the upper bound achieved at the end of the root node.

L.B.F. is the final lower bound obtained at the end of the computation.

Best is the best feasible solution found during the computation.

Gap% is the residual dual gap at the end of the computation.

N.Cuts is the total number of cuts added to the MP.

Time[s] is the computational time in seconds.

Inst.	B.B.N.	L.B.R.	U.B.R.	F.B.R.	Best	Gap%	N.Cuts	Time[s]
15_01	1	2688	2688	2688	2688	0	0	0.01
15_02	1	4887	4887	4887	4887	0	0	0.01
15_03	31	3331.88	inf	3721	3721	0	0	5.49
15_04	3	6643	6846	6756	6756	0	2	0.01
15_05	1	5395	5395	5395	5395	0	0	0.01
15_06	39	2535.33	2673	2673	2673	0	5	0.1
15_07	5	1568.05	2094	1796	1796	0	0	0.02
15_08	21	3762.67	inf	4134	4134	0	13	0.09
15_09	111	2477.4	2700	2633	2633	0	9	0.31
15_10	1	5961	5961	5961	5961	0	0	0.01
15_11	1	5747	5747	5747	5747	0	0	0.01
15_12	3	3071.33	3935	3347	3347	0	10	0.04
15_13	1	2276	2276	2276	2276	0	0	0.01
15_14	1	4241	4241	4241	4241	0	0	0.01
15_15	3	5187	5272	5272	5272	0	3	0.02
25_01	79	7008.36	inf	7065	7065	0	16	2.67
25_02	1	741	741	741	741	0	0	0.01
25_03	69	3164.17	inf	3365	3365	0	10	520.18
25_04	3	844.8	923	923	923	0	1	0.02
25_05	3	2097.12	2208	2136	2136	0	0	0.03
25_06	3	5794.4	inf	5817	5817	0	10	2.1
25_07	1	5614	5614	5614	5614	0	7	0.73
25_08	3	6395	6534	6534	6534	0	2	0.01
25_09	3	8788	inf	8851	8851	0	10	0.2
25_10	135	8001	8247	8142	8142	0	28	0.56
25_11	330	4360.13	inf	4689	4689	0	15	147.34
25_12	1	4063	4063	4063	4063	0	2	0.02
25_13	3	7906.67	8248	7958	7958	0	6	0.08
25_14	1	6167	6167	6167	6167	0	0	0.01
25_15	3	2834.34	inf	3201	3201	0	11	318.87
50_01	1	12367	12367	12367	12367	0	0	15.09
50_02	3	13287.5	13294	13294	13294	0	3	1.11
50_03	97	13372.86	inf	13410	13410	0	44	62.45
50_04	1500	17680.29	inf	17816.17	17823	0.04	128	3600
50_05	3	23862.5	24020	23927	23927	0	5	0.4
50_06	1	12882	12882	12882	12882	0	0	0.24
50_07	91	13983.53	14528	14047	14047	0	32	155.33
50_08	88	2514.86	2642	2626	2626	0	5	342.3
50_09	5930	14415.69	inf	14395	14395	0	1145	1836.51
50_10	51	5973.76	6044	6018	6018	0	6	22.57
50_11	3	15639.33	15758	15658	15658	0	6	1.13
50_12	3	12242	12633	12297	12297	0	1	28.31
50_13	3898	10402	inf	10650.55	10771	1.13	55	3600
50_14	9503	17273.44	inf	17316.64	17325	0.05	374	3600
50_15	47	14755.3	16142	14789	14789	0	23	17.93

Table C.1: Solution for instances with $L_{min} = 0.60 \cdot W$ (15-50 nodes)

Inst.	B.B.N.	L.B.R.	U.B.R.	F.B.R.	Best	Gap%	N.Cuts	Time[s]
75_01	7351	23104.55	inf	23157.24	23193	0.15	671	3600
75_02	1	17219	17219	17219	17219	0	3	13.46
75_03	1	38713	38713	38713	38713	0	9	0.08
75_04	7606	20973.42	inf	21048.84	21240	0.91	408	3600
75_05	604	6239.12	7948	6449	6449	0	29	688.19
75_06	1	14293	14293	14293	14293	0	0	2
75_07	9633	24613.9	inf	24654	24654	0	824	2333.99
75_08	1815	21250.84	inf	21345	21523	0.83	234	3600
75_09	395	21254.71	inf	21367	21411	0.21	95	3600
75_10	1	9039	9039	9039	9039	0	0	16.07
75_11	1798	23654.57	inf	23709	23709	0	527	426.06
75_12	5566	21087.45	inf	21158.33	21215	0.27	562	3600
75_13	1	18269	18269	18269	18269	0	0	1.57
75_14	3	27803.5	27807	27807	27807	0	6	0.72
75_15	3099	25582.17	inf	25613.59	25628	0.06	288	3600
100_01	59	18640.42	19547	18703	18721	0.1	23	3600
100_02	81	29016.5	29068	29020	29020	0	7	35.39
100_03	3	24737	24741	24741	24741	0	1	4.46
100_04	3	118.15	143	143	143	0	0	61.41
100_05	6682	37526.33	inf	37626	37663	0.1	410	3600
100_06	182	24450	inf	24545.61	24864	1.3	82	3600
100_07	10	26010.94	inf	24631	24631	0	21	464.42
100_08	1	29278	29278	29278	29278	0	6	5.56
100_09	54	5857.26	inf	5940	5940	0	4	440.95
100_10	109	2103.16	2381	2179.49	2381	9.25	23	3600
100_11	146776	50281.5	50784	50346.5	50390	0.09	1232	3600
100_12	566	32013.96	inf	32095.08	32469	1.17	137	3600
100_13	80	6669.91	inf	6696.69	6727	0.45	38	3600
100_14	124	20424.59	inf	20504	20504	0	22	591.57
100_15	289	32008.8	inf	32031.09	32462	1.35	86	3600

Table C.2: Solution for instances with $L_{min} = 0.60 \cdot W$ (75-100 nodes)

Inst.	B.B.N.	L.B.R.	U.B.R.	F.B.R.	Best	Gap%	N.Cuts	Time[s]
15_01	1	2688	2688	2688	2688	0	0	0
15_02	1	4887	4887	4887	4887	0	0	0.01
15_03	3	3340.59	inf	3820	3820	0	1	0.95
15_05	1	5926	5926	5926	5926	0	1	0
15_06	27	2535.33	2673	2673	2673	0	6	0.08
15_07	17	1815.38	inf	2537	2537	0	10	0.2
15_08	3	3748.67	inf	4314	4314	0	10	0.05
15_09	29	2625.76	2751	2700	2700	0	9	0.1
15_10	1	5961	5961	5961	5961	0	0	0.01
15_11	1	5747	5747	5747	5747	0	0	0.01
15_12	3	3071.33	3935	3378	3378	0	10	0.06
15_13	1	2276	2276	2276	2276	0	0	0.02
15_14	1	4331	4331	4331	4331	0	0	0.01
15_15	3	5189	5480	5284	5284	0	4	0.08
25_01	61	7162.67	7260	7219	7219	0	8	3.63
25_02	3	787.08	1016	952	952	0	0	0.04
25_03	1017	3176.06	inf	3401	3401	0	21	3600
25_04	3	844.8	inf	1146	1146	0	0	0.02
25_05	3	2212.4	2582	2393	2393	0	2	0.19
25_06	3	5976	inf	6225	6225	0	15	3.19
25_07	1	5614	5614	5614	5614	0	7	0.72
25_08	3	6508	6869	6797	6797	0	2	0.04
25_09	3	8788	inf	8967	8967	0	12	0.24
25_10	9	8001	8413	8159	8159	0	10	0.49
25_11	1320	4355.15	inf	4961	4961	0	6	513.36
25_12	1	4082	4082	4082	4082	0	2	0.04
25_13	3	7906.67	7958	7958	7958	0	6	0.06
25_14	1	6169	6169	6169	6169	0	0	0.02
50_01	1	12367	12367	12367	12367	0	0	14.64
50_02	3	13291	13391	13365	13365	0	3	1.36
50_03	3	13381	13483	13410	13410	0	18	18.6
50_04	882	17745.44	inf	17806.5	17825	0.1	115	3600
50_06	3	13631	14112	13657	13657	0	0	0.61
50_07	195	13989.67	inf	14047	14047	0	48	234.73
50_08	73	2650.74	3168	2879	2879	0	10	133.76
50_09	4368	13932.33	inf	14329	14329	0	509	432.42
50_10	35	5973.76	6044	6018	6018	0	4	15.9
50_11	3	15639.33	16254	15658	15658	0	6	14.08
50_12	3	12610	inf	12769	12769	0	2	231.36
50_13	952	10404.14	inf	10646.04	10762	1.09	43	3600
50_14	5756	17275.86	inf	17325	17325	0	360	2333.63
50_15	673	14771.14	inf	14879	14879	0	89	2274.44

Table C.3: Solution for instances with $L_{min} = 0.80 \cdot W$ (15-50 nodes)

Inst.	B.B.N.	L.B.R.	U.B.R.	F.B.R.	Best	Gap%	N.Cuts	Time[s]
75_01	6358	23115.24	inf	23156.26	23219	0.27	516	3600
75_02	1	17219	17219	17219	17219	0	3	3.74
75_03	1	38713	38713	38713	38713	0	9	0.14
75_04	4566	20989.25	inf	21046.35	21237	0.91	266	3600
75_05	214	6239.12	7782	6453	6453	0	21	85.24
75_06	1	14294	14294	14294	14294	0	0	2.2
75_07	16325	24592	inf	24643.21	24654	0.04	1357	3600
75_08	1597	21248.25	inf	21392.91	21403	0.05	163	3600
75_09	197	21252.86	inf	21377.5	21408	0.14	101	3600
75_10	1	9048	9048	9048	9048	0	0	32.54
75_11	809	23648.85	inf	23709	23709	0	228	237.43
75_12	987	21096.88	inf	21152.28	21250	0.46	130	3600
75_13	1	18404	18404	18404	18404	0	0	5.48
75_14	3	27803.5	27807	27807	27807	0	6	0.7
75_15	3193	25570.54	inf	25609.69	25628	0.07	268	3600
100_01	33	18646.7	inf	18701.5	18756	0.29	26	3600
100_02	81	29016.5	30041	29020	29020	0	9	41.21
100_03	3	24737	24741	24741	24741	0	0	2.88
100_04	3	118.15	143	143	143	0	0	143.22
100_05	7391	37530	38454	37689.7	37743	0.14	404	3600
100_06	186	24447.85	inf	24542.67	24752	0.85	78	3600
100_07	154	26047.89	inf	26047.89	inf	inf	84	3600
100_08	877	29309	31544	29480	29480	0	90	928.03
100_09	3	5857.26	6258	5939	5939	0	1	28.58
100_10	42	2101.36	2353	2229	2229	0	20	2800
100_12	459	32024.02	inf	32153.6	32599	1.39	151	3600
100_13	88	6673.12	6946	6697.84	6732	0.51	56	3600
100_14	273	20424.59	23380	20548	20548	0	78	2011.78
100_15	414	32008.75	33979	32045.93	32270	0.7	136	3600

Table C.4: Solution for instances with $L_{min} = 0.80 \cdot W$ (75-100 nodes)

Inst.	B.B.N.	L.B.R.	U.B.R.	F.B.R.	Best	Gap%	N.Cuts	Time[s]
15_01	1	2961	2961	2961	2961	0	0	0.01
15_02	1	5335	5335	5335	5335	0	0	0.02
15_05	1	6300	6300	6300	6300	0	1	0.01
15_06	3	2919.45	inf	3618	3618	0	0	0.02
15_07	3	1999.84	3212	2966	2966	0	3	0.10
15_08	3	3748	4887	4344	4344	0	6	0.04
15_09	3	2625.76	2795	2756	2756	0	2	0.03
15_12	3	3086	3935	3378	3378	0	14	0.06
15_13	1	2276	2276	2276	2276	0	0	0.00
15_14	1	4699	4699	4699	4699	0	0	0.03
25_01	19	7192.04	7957	7578	7578	0	10	2.61
25_02	3	894.05	1192	1192	1192	0	0	0.07
25_04	3	844.8	inf	1191	1191	0	0	0.02
25_05	3	2212.4	2582	2565	2565	0	1	0.14
25_07	1	5614	5614	5614	5614	0	7	0.72
25_08	3	6597	7068	7011	7011	0	3	0.07
25_10	121	8339	inf	8863	8863	0	37	3.15
25_12	3	4123.36	4597	4242	4242	0	9	0.18
25_13	3	8087.25	8127	8112	8112	0	5	0.27
25_14	1	6293	6293	6293	6293	0	0	0.07
50_01	1	13065	13065	13065	13065	0	0	163.56
50_02	3	13329	13429	13404	13404	0	2	0.62
50_03	7	13467.72	inf	13604	13604	0	19	52.73
50_04	63	17849.06	18975	17901	17901	0	33	826.28
50_06	3	14280	15096	14329	14329	0	0	0.67
50_07	1	14173	14173	14173	14173	0	0	12.47
50_08	3	2774.36	inf	3140	3140	0	1	18.38
50_10	3	6028.29	6365	6161	6161	0	3	3.68
50_11	2605	15800.67	inf	15936	16013	0.48	310	3600.00
50_12	8	13166.5	inf	13575	13575	0	6	1755.69
50_13	621	10404.82	inf	10857.35	11301	4.09	46	6702.70
50_14	1380	17303.6	inf	17594	17594	0	184	1239.10
50_15	47	14940.13	inf	15421	15421	0	39	1642.62
75_01	63	23482.82	inf	23495	23495	0	30	308.20
75_02	3	17376.67	17472	17447	17447	0	5	106.21
75_03	1	38713	38713	38713	38713	0	9	0.13
75_04	4722	20996.93	inf	21054.43	21223	0.8	281	3601.47
75_05	13	6360.14	inf	6453	6453	0	9	68.05
75_06	1	14945	14945	14945	14945	0	0	6.89
75_07	11554	24671.09	inf	24796.87	24827	0.12	1019	3603.59
75_10	1	9277	9277	9277	9277	0	1	126.31
75_11	904	23668.81	25626	23709	23709	0	275	369.33
75_12	324	21183.28	inf	21227.36	21364	0.64	185	3603.94
75_13	1	19690	19690	19690	19690	0	0	28.01
75_14	343	27803.5	inf	27819	27819	0	47	23.22
75_15	308	25603.55	inf	25718	25718	0	82	356.45
100_01	8	19266.47	inf	19298.34	19432	0.69	27	3600.00
100_02	3308	29086.5	29509	29119.5	29162	0.15	224	3600.00
100_03	149	24887	24958	24896	24896	0	4	3583.70
100_04	3	131.72	177	177	177	0	0	347.76
100_05	6928	37607.67	inf	37840.33	37907	0.18	301	3600.00
100_06	251	24498.77	inf	24701.43	24824	0.5	114	3600.00
100_07	2	23136.38	inf	23136.38	23889	3.25	14	3600.00
100_08	45	29309	33644	29488	29488	0	30	44.99
100_09	3	5857.26	6247	6011	6011	0	1	39.24
100_10	3	2334.29	inf	2722	2722	0	7	755.57
100_12	331	32086.74	inf	32229.78	33265	3.21	134	3600.00
100_13	3	6729.73	inf	6989	6989	0	8	886.49
100_14	43	23781.76	24843	23781.76	24843	4.46	69	3600.00

Table C.5: Solution for instances with $L_{min} = 0.99 \cdot W$

Appendix D

Detailed Results: Planetary Space Exploration

In this Appendix are reported the tables containing detailed results for every single instances used as test set for the algorithm presented in the Chapter 3 using the AC configuration of the cut generator. The column used have the following mining:

Inst. is the instance name in the format *bases_sites_budget*.

B.B.N. is the number of nodes explored in the branch and bound tree.

L.B.R. is the lower bound achieved at the end of the root node.

U.B.R. is the upper bound achieved at the end of the root node.

L.B.F. is the final lower bound obtained at the end of the computation.

Best is the best feasible solution found during the computation.

Gap% is the residual dual gap at the end of the computation.

N.Cuts is the total number of cuts added to the MP.

Time[s] is the computational time in seconds.

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
10_100_0.5	11	190.00	195.00	195.00	195.00	0.00	39	0.05
10_100_1	21	182.83	188.00	188.00	188.00	0.00	39	0.07
10_100_2	29	169.50	179.00	175.00	175.00	0.00	39	0.09
10_100_3	23	162.00	174.00	165.00	165.00	0.00	39	0.08
10_100_4	3	157.50	158.00	158.00	158.00	0.00	39	0.05
10_100_5	1	157.00	157.00	157.00	157.00	0.00	37	0.01
10_200_0.5	11	364.85	377.00	374.00	374.00	0.00	105	0.15
10_200_1	19	348.00	361.00	360.00	360.00	0.00	110	0.18
10_200_2	23	329.50	338.00	338.00	338.00	0.00	107	0.23
10_200_3	35	321.20	328.00	326.00	326.00	0.00	97	0.33
10_200_4	13	318.10	319.00	319.00	319.00	0.00	96	0.17
10_200_5	1	318.00	318.00	318.00	318.00	0.00	89	0.03
10_300_0.5	13	554.74	572.00	570.00	570.00	0.00	163	0.24
10_300_1	47	531.56	556.00	553.00	553.00	0.00	171	0.68
10_300_2	59	503.32	524.00	519.00	519.00	0.00	154	0.98
10_300_3	9	484.03	490.00	487.00	487.00	0.00	148	0.32
10_300_4	5	473.40	475.00	475.00	475.00	0.00	125	0.12
10_300_5	1	469.00	469.00	469.00	469.00	0.00	125	0.06
10_400_0.5	11	757.47	772.00	770.00	770.00	0.00	174	0.34
10_400_1	25	737.05	754.00	751.00	751.00	0.00	172	0.69
10_400_2	19	704.96	717.00	714.00	714.00	0.00	170	0.78
10_400_3	29	679.08	689.00	688.00	688.00	0.00	168	1.16
10_400_4	21	662.33	671.00	668.00	668.00	0.00	155	0.99
10_400_5	1	655.00	655.00	655.00	655.00	0.00	136	0.07
10_500_0.5	17	970.71	987.00	986.00	986.00	0.00	272	0.66
10_500_1	43	943.23	968.00	963.00	963.00	0.00	299	1.96
10_500_2	65	902.31	923.00	920.00	920.00	0.00	280	3.58
10_500_3	51	871.26	893.00	884.00	884.00	0.00	270	3.51
10_500_4	19	849.25	858.00	857.00	857.00	0.00	248	1.77
10_500_5	1	844.00	844.00	844.00	844.00	0.00	196	0.11
10_1000_0.5	105	1975.70	1998.00	1994.00	1994.00	0.00	690	8.45
10_1000_1	149	1929.89	1966.00	1956.00	1956.00	0.00	738	17.24
10_1000_2	25842	1852.78	1900.00	1886.00	1887.00	0.05	869	3600.08
10_1000_3	16123	1798.35	1843.00	1828.00	1829.00	0.05	855	3600.00
10_1000_4	13625	1759.83	1786.00	1775.00	1776.00	0.06	776	3600.05
10_1000_5	11742	1733.32	1735.00	1734.00	1735.00	0.06	565	3600.17

Table D.1: Solution of GRLPP using AC Configuration (10 bases)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
20_100_0.5	67	161.90	178.00	173.00	173.00	0.00	64	0.26
20_100_1	28	149.13	165.00	155.38	155.00	0.00	64	0.17
20_100_2	51	134.25	149.00	141.00	141.00	0.00	63	0.21
20_100_3	45	126.00	138.00	130.00	130.00	0.00	62	0.22
20_100_4	1	126.00	126.00	126.00	126.00	0.00	57	0.02
20_100_5	1	126.00	126.00	126.00	126.00	0.00	57	0.02
20_200_0.5	45	375.48	386.00	386.00	386.00	0.00	187	0.53
20_200_1	137	347.88	363.00	359.00	359.00	0.00	186	1.37
20_200_2	547	302.86	322.00	317.00	317.00	0.00	189	6.54
20_200_3	420	268.92	288.00	285.00	285.00	0.00	186	5.74
20_200_4	155	258.77	276.00	264.00	264.00	0.00	180	2.39
20_200_5	1	258.00	258.00	258.00	258.00	0.00	153	0.09
20_300_0.5	43	532.07	552.00	549.00	549.00	0.00	294	0.95
20_300_1	69	498.21	520.00	512.00	512.00	0.00	282	2.06
20_300_2	366	445.22	466.00	456.00	456.00	0.00	289	12.04
20_300_3	292	403.77	418.00	413.00	413.00	0.00	290	11.97
20_300_4	71	380.33	385.00	385.00	385.00	0.00	265	3.48
20_300_5	1	378.00	378.00	378.00	378.00	0.00	238	0.13
20_400_0.5	129	716.52	741.00	739.00	739.00	0.00	417	3.20
20_400_1	123	674.52	709.00	690.72	691.00	0.00	424	5.84
20_400_2	5861	614.20	659.00	642.00	642.00	0.00	540	283.84
20_400_3	324	573.10	596.00	585.00	585.00	0.00	445	23.15
20_400_4	39	552.00	559.00	556.00	556.00	0.00	379	4.15
20_400_5	1	552.00	552.00	552.00	552.00	0.00	303	0.14
20_500_0.5	217	912.39	947.00	938.00	938.00	0.00	488	6.03
20_500_1	1953	867.71	907.00	896.00	896.00	0.00	541	77.85
20_500_2	10659	784.43	834.00	820.00	820.00	0.00	570	702.07
20_500_3	8532	725.72	766.00	754.00	754.00	0.00	550	733.66
20_500_4	436	676.13	702.00	693.00	693.00	0.00	465	47.54
20_500_5	1	647.00	647.00	647.00	647.00	0.00	364	0.43
20_1000_0.5	221	1927.63	1967.00	1956.00	1956.00	0.00	1501	31.71
20_1000_1	1915	1848.39	1911.00	1888.00	1888.00	0.00	2122	395.45
20_1000_2	10626	1718.70	1795.00	1760.51	1766.00	0.31	2422	3600.14
20_1000_3	8116	1612.13	1686.00	1658.00	1658.00	0.00	2230	3588.15
20_1000_4	6037	1529.14	1576.00	1556.65	1557.00	0.02	1653	3600.31
20_1000_5	5491	1475.87	1478.00	1476.12	1478.00	0.13	1110	3600.18

Table D.2: Solution of GRLPP using AC Configuration (20 bases)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
30_100_0.5	423	146.20	165.00	161.00	161.00	0.00	129	1.84
30_100_1	893	116.50	142.00	135.00	135.00	0.00	129	4.76
30_100_2	4795	87.96	112.00	103.00	103.00	0.00	128	25.54
30_100_3	283	77.00	96.00	83.00	83.00	0.00	127	1.89
30_100_4	14	77.00	79.00	77.00	77.00	0.00	127	0.15
30_100_5	1	77.00	77.00	77.00	77.00	0.00	123	0.02
30_200_0.5	135	328.47	348.00	343.00	343.00	0.00	208	1.64
30_200_1	4835	295.33	318.00	314.00	314.00	0.00	210	48.25
30_200_2	62625	247.32	277.00	269.00	269.00	0.00	212	1204.17
30_200_3	3900	225.00	244.00	235.00	235.00	0.00	209	58.07
30_200_4	1	218.00	218.00	218.00	218.00	0.00	188	0.08
30_200_5	1	218.00	218.00	218.00	218.00	0.00	184	0.05
30_300_0.5	1219	518.46	553.00	547.00	547.00	0.00	441	18.86
30_300_1	33842	468.72	514.00	502.00	502.00	0.00	459	835.96
30_300_2	92873	392.96	440.00	421.78	427.00	1.24	472	3600.03
30_300_3	39220	344.91	386.00	368.00	368.00	0.00	447	1849.78
30_300_4	673	326.29	350.00	333.00	333.00	0.00	401	37.03
30_300_5	1	326.00	326.00	326.00	326.00	0.00	337	0.17
30_400_0.5	2158	689.94	746.00	738.00	738.00	0.00	924	74.57
30_400_1	68747	625.06	690.00	683.00	683.00	0.00	1056	3211.07
30_400_2	43163	523.64	607.00	561.89	582.00	3.58	953	3600.02
30_400_3	35782	456.70	515.00	487.75	496.00	1.69	974	3600.03
30_400_4	903	421.00	456.00	435.00	435.00	0.00	759	110.71
30_400_5	1	413.00	413.00	413.00	413.00	0.00	504	0.35
30_500_0.5	473	856.12	899.00	893.00	893.00	0.00	836	22.54
30_500_1	8535	778.06	845.00	831.00	831.00	0.00	972	553.05
30_500_2	30306	678.09	747.00	719.00	719.00	0.00	988	2984.84
30_500_3	23799	607.15	667.00	639.00	639.00	0.00	960	3236.98
30_500_4	771	565.25	590.00	580.00	580.00	0.00	723	119.18
30_500_5	1	563.00	563.00	563.00	563.00	0.00	522	0.41
30_1000_0.5	5398	1816.09	1878.00	1860.00	1860.00	0.00	4096	1094.69
30_1000_1	9401	1698.33	1792.00	1742.40	1764.00	1.24	4186	3600.12
30_1000_2	6067	1513.70	1617.00	1541.21	1601.00	3.88	2869	3600.08
30_1000_3	4691	1373.95	1466.00	1397.95	1426.00	2.01	2641	3600.38
30_1000_4	4142	1268.71	1331.00	1291.81	1310.00	1.41	2490	3600.10
30_1000_5	4344	1215.51	1237.00	1216.60	1220.00	0.28	1576	3600.62

Table D.3: Solution of GRLPP using AC Configuration (30 bases)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
40_100_0.5	277	143.23	161.00	158.00	158.00	0.00	159	1.76
40_100_1	629	117.17	131.00	127.00	127.00	0.00	160	4.67
40_100_2	267	87.43	97.00	93.00	93.00	0.00	155	1.98
40_100_3	497	74.93	79.00	78.00	78.00	0.00	153	3.44
40_100_4	1	74.00	74.00	74.00	74.00	0.00	145	0.02
40_100_5	1	74.00	74.00	74.00	74.00	0.00	145	0.05
40_200_0.5	4075	302.57	337.00	330.00	330.00	0.00	423	53.54
40_200_1	31379	251.05	288.00	279.00	279.00	0.00	440	507.50
40_200_2	132167	182.31	230.00	208.75	213.00	2.04	453	3600.01
40_200_3	32332	157.33	193.00	171.00	171.00	0.00	429	948.38
40_200_4	12	155.00	156.00	155.00	155.00	0.00	336	0.72
40_200_5	1	155.00	155.00	155.00	155.00	0.00	310	0.14
40_300_0.5	3329	476.94	519.00	515.00	515.00	0.00	578	89.20
40_300_1	86847	413.26	472.00	456.00	456.00	0.00	592	2959.28
40_300_2	66573	333.67	381.00	358.46	364.00	1.54	562	3600.00
40_300_3	19463	291.23	312.00	309.00	309.00	0.00	558	1213.27
40_300_4	7	285.00	294.00	285.00	285.00	0.00	437	0.95
40_300_5	1	285.00	285.00	285.00	285.00	0.00	418	0.35
40_400_0.5	5147	650.87	718.00	700.00	700.00	0.00	1165	213.98
40_400_1	47799	565.14	637.00	613.12	622.00	1.45	1234	3600.05
40_400_2	32516	441.91	521.00	475.49	495.00	4.10	1042	3600.05
40_400_3	28447	364.74	430.00	390.18	403.00	3.29	1046	3600.02
40_400_4	8553	336.00	361.00	347.00	347.00	0.00	864	1073.42
40_400_5	1	336.00	336.00	336.00	336.00	0.00	623	0.31
40_500_0.5	4565	843.10	894.00	878.00	878.00	0.00	1162	239.69
40_500_1	47555	753.46	818.00	791.69	803.00	1.43	1194	3600.02
40_500_2	23839	621.75	698.00	645.83	683.00	5.76	1158	3600.05
40_500_3	20486	538.90	609.00	561.50	581.00	3.47	1101	3600.03
40_500_4	3545	492.33	529.00	507.00	507.00	0.00	970	669.01
40_500_5	23	485.50	486.00	486.00	486.00	0.00	686	4.19
40_1000_0.5	14059	1785.37	1852.00	1836.04	1837.00	0.05	4679	3600.12
40_1000_1	8438	1639.73	1742.00	1684.30	1712.00	1.64	3740	3600.26
40_1000_2	5212	1397.90	1546.00	1425.59	1520.00	6.62	3272	3600.14
40_1000_3	4360	1233.08	1365.00	1256.48	1334.00	6.17	2705	3600.12
40_1000_4	3898	1124.70	1211.00	1144.59	1172.00	2.40	2390	3600.65
40_1000_5	4190	1088.50	1108.00	1088.81	1097.00	0.75	1707	3600.71

Table D.4: Solution of GRLPP using AC Configuration (40 bases)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
50_100_0.5	1893	121.85	142.00	141.00	141.00	0.00	240	15.86
50_100_1	3193	93.52	117.00	107.00	107.00	0.00	242	33.92
50_100_2	903	61.58	75.00	68.00	68.00	0.00	227	8.75
50_100_3	138	57.00	61.00	57.00	57.00	0.00	223	1.51
50_100_4	1	57.00	57.00	57.00	57.00	0.00	202	0.03
50_100_5	1	57.00	57.00	57.00	57.00	0.00	198	0.04
50_200_0.5	14673	303.00	347.00	335.00	335.00	0.00	475	244.96
50_200_1	128893	234.65	291.00	260.66	275.00	5.50	483	3600.02
50_200_2	103427	141.34	208.00	170.79	191.00	11.83	478	3600.02
50_200_3	136043	109.00	145.00	126.12	129.00	2.29	466	3600.02
50_200_4	1	109.00	109.00	109.00	109.00	0.00	387	0.15
50_200_5	1	109.00	109.00	109.00	109.00	0.00	385	0.10
50_300_0.5	73932	477.67	530.00	523.00	523.00	0.00	797	2347.97
50_300_1	68155	383.30	456.00	417.03	449.00	7.67	785	3600.00
50_300_2	48526	256.47	331.00	283.63	317.00	11.76	759	3600.05
50_300_3	43586	192.75	248.00	212.40	221.00	4.05	727	3600.01
50_300_4	4	190.00	196.00	190.00	190.00	0.00	558	0.50
50_300_5	1	190.00	190.00	190.00	190.00	0.00	558	0.21
50_400_0.5	9787	604.09	668.00	650.00	650.00	0.00	1425	670.25
50_400_1	34656	496.10	585.00	532.62	554.00	4.01	1347	3600.08
50_400_2	24904	355.53	447.00	377.50	420.00	11.26	1277	3600.10
50_400_3	22376	277.83	333.00	295.62	309.00	4.53	1237	3600.01
50_400_4	1249	253.00	269.00	257.00	257.00	0.00	1017	144.58
50_400_5	1	253.00	253.00	253.00	253.00	0.00	746	0.51
50_500_0.5	40902	833.55	930.00	893.01	905.00	1.34	2047	3600.06
50_500_1	24621	702.27	820.00	744.72	797.00	7.02	1709	3600.03
50_500_2	17727	525.28	648.00	549.29	611.00	11.24	1520	3600.13
50_500_3	15253	409.65	504.00	431.14	448.00	3.91	1407	3600.03
50_500_4	12515	350.35	389.00	364.00	364.00	0.00	1521	3315.95
50_500_5	7	347.50	348.00	348.00	348.00	0.00	974	2.38
50_1000_0.5	8276	1778.05	1887.00	1828.79	1858.00	1.60	5614	3600.35
50_1000_1	5718	1585.52	1739.00	1624.13	1728.00	6.40	4227	3600.44
50_1000_2	3820	1298.53	1488.00	1323.12	1444.00	9.14	3719	3600.25
50_1000_3	3172	1087.32	1252.00	1107.20	1197.00	8.11	3253	3600.83
50_1000_4	2973	943.32	1047.00	954.14	993.00	4.07	3074	3600.95
50_1000_5	1254	886.17	921.00	886.67	887.00	0.04	2104	3602.30

Table D.5: Solution of GRLPP using AC Configuration (50 bases)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
100_100_0.5	246761	82.56	123.00	101.00	101.00	0.00	386	2993.63
100_100_1	266675	37.85	78.00	50.34	57.00	13.23	368	3600.00
100_100_2	8	26.00	31.00	26.00	26.00	0.00	347	0.24
100_100_3	1	26.00	26.00	26.00	26.00	0.00	330	0.09
100_100_4	1	26.00	26.00	26.00	26.00	0.00	328	0.11
100_100_5	1	26.00	26.00	26.00	26.00	0.00	327	0.20
100_200_0.5	82266	211.37	286.00	242.27	269.00	11.03	971	3600.02
100_200_1	64168	95.47	197.00	115.22	164.00	42.33	942	3600.01
100_200_2	91604	36.00	72.00	36.97	53.00	43.34	910	3600.01
100_200_3	96	36.00	37.00	36.00	36.00	0.00	817	12.26
100_200_4	1	36.00	36.00	36.00	36.00	0.00	736	0.21
100_200_5	1	36.00	36.00	36.00	36.00	0.00	722	0.23
100_300_0.5	40704	335.61	446.00	362.35	427.00	17.84	1540	3600.05
100_300_1	33586	199.74	326.00	217.14	296.00	36.32	1467	3600.07
100_300_2	29399	93.38	218.00	99.41	140.00	40.83	1539	3600.01
100_300_3	42	93.00	100.00	93.00	93.00	0.00	1168	8.93
100_300_4	1	93.00	93.00	93.00	93.00	0.00	1066	0.26
100_300_5	1	93.00	93.00	93.00	93.00	0.00	1077	0.55
100_400_0.5	19219	471.46	583.00	495.11	564.00	13.92	2818	3600.00
100_400_1	15421	303.70	442.00	314.23	412.00	31.11	2486	3600.16
100_400_2	13009	116.16	249.00	125.49	207.00	64.95	2454	3600.14
100_400_3	22172	108.50	135.00	108.50	113.00	4.15	2692	3600.04
100_400_4	23380	108.50	109.00	108.50	109.00	0.46	2503	3600.02
100_400_5	191	108.50	109.00	109.00	109.00	0.00	1508	25.51
100_500_0.5	14348	626.90	771.00	658.20	749.00	13.80	3028	3600.11
100_500_1	10529	426.34	601.00	437.19	568.00	29.92	2948	3600.14
100_500_2	8281	186.45	372.00	197.21	304.00	54.15	2803	3600.00
100_500_3	10749	136.00	176.00	136.00	142.00	4.41	2983	3600.22
100_500_4	6	136.00	141.00	136.00	136.00	0.00	1791	10.82
100_500_5	1	136.00	136.00	136.00	136.00	0.00	1811	2.97
100_1000_0.5	2618	1526.38	1732.00	1550.40	1714.00	10.55	7488	3600.77
100_1000_1	1774	1165.24	1464.00	1171.93	1438.00	22.70	7002	3600.01
100_1000_2	1158	649.51	999.00	658.51	978.00	48.52	6319	3600.78
100_1000_3	991	362.06	623.00	371.27	610.00	64.30	5764	3600.09
100_1000_4	1446	252.50	380.00	253.30	292.00	15.28	5682	3601.21
100_1000_5	809	252.50	275.00	252.50	254.00	0.59	3868	3609.41

Table D.6: Solution of GRLPP using AC Configuration (100 bases)

Appendix E

Detailed Results: Drugs Distribution

The detailed results for the Drug Distribution in Case of Emergency (see Chapter 4) are reported in this Appendix. In particular the tables are derived form the tests on all instances using CC cuts configuration and setting $\alpha = 0$. The column used have the following mining:

Inst. is the instance name in the format *delivery sites_M_D_settings*.

B.B.N. is the number of nodes explored in the branch and bound tree.

L.B.R. is the lower bound achieved at the end of the root node.

U.B.R. is the upper bound achieved at the end of the root node.

L.B.F. is the final lower bound obtained at the end of the computation.

Best is the best feasible solution found during the computation.

Gap% is the residual dual gap at the end of the computation.

N.Cuts is the total number of cuts added to the MP.

Time[s] is the computational time in seconds.

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
10_04_07_1	9	14.00	16.00	16.00	16.00	0.00	29	0.02
10_04_07_2	3	23.00	43.00	23.00	23.00	0.00	290	0.01
10_04_07_3	7	16.50	19.00	19.00	19.00	0.00	72	0.02
10_04_07_4	1	11.00	11.00	11.00	11.00	0.00	49	0.01
10_04_07_5	1	30.00	30.00	30.00	30.00	0.00	2798	0
10_04_08_1	1	7.00	7.00	7.00	7.00	0.00	279	0.02
10_04_08_2	3	23.00	34.00	23.00	23.00	0.00	86	0.01
10_04_08_3	1	5.00	5.00	5.00	5.00	0.00	164	0
10_04_08_4	1	11.00	11.00	11.00	11.00	0.00	91	0.01
10_04_08_5	1	30.00	30.00	30.00	30.00	0.00	85	0
10_04_09_1	1	7.00	7.00	7.00	7.00	0.00	79	0.01
10_04_09_2	1	23.00	23.00	23.00	23.00	0.00	64	0
10_04_09_3	1	5.00	5.00	5.00	5.00	0.00	54	0
10_04_09_4	1	11.00	11.00	11.00	11.00	0.00	32	0.01
10_04_09_5	1	30.00	30.00	30.00	30.00	0.00	215	0.01
10_04_10_1	3	6.00	7.00	7.00	7.00	0.00	2719	0.01
10_04_10_2	1	23.00	23.00	23.00	23.00	0.00	440	0
10_04_10_3	1	5.00	5.00	5.00	5.00	0.00	62	0
10_04_10_4	1	11.00	11.00	11.00	11.00	0.00	296	0.01
10_04_10_5	1	30.00	30.00	30.00	30.00	0.00	77	0
10_05_07_1	1	12.00	12.00	12.00	12.00	0.00	65	0
10_05_07_2	2	23.00	43.00	23.00	23.00	0.00	227	0.01
10_05_07_3	5	14.29	16.00	16.00	16.00	0.00	341	0.02
10_05_07_4	5	9.90	11.00	11.00	11.00	0.00	86	0.02
10_05_07_5	5	10.50	11.00	11.00	11.00	0.00	2702	0.02
10_05_08_1	3	4.29	7.00	7.00	7.00	0.00	94	0.01
10_05_08_2	1	14.00	14.00	14.00	14.00	0.00	63	0
10_05_08_3	1	5.00	5.00	5.00	5.00	0.00	2712	0.01
10_05_08_4	1	0.00	0.00	0.00	0.00	0.00	211	0
10_05_08_5	1	5.00	5.00	5.00	5.00	0.00	62	0.01
10_05_09_1	1	0.00	0.00	0.00	0.00	0.00	82	0
10_05_09_2	1	14.00	14.00	14.00	14.00	0.00	43	0
10_05_09_3	1	5.00	5.00	5.00	5.00	0.00	215	0.01
10_05_09_4	1	0.00	0.00	0.00	0.00	0.00	198	0.01
10_05_09_5	1	5.00	5.00	5.00	5.00	0.00	279	0
10_05_10_1	1	0.00	0.00	0.00	0.00	0.00	183	0
10_05_10_2	1	14.00	14.00	14.00	14.00	0.00	8594	0
10_05_10_3	1	5.00	5.00	5.00	5.00	0.00	379	0
10_05_10_4	2	0.00	6.00	0.00	0.00	0.00	267	0.01
10_05_10_5	1	5.00	5.00	5.00	5.00	0.00	81	0.01
10_06_07_1	1	12.00	12.00	12.00	12.00	0.00	43	0
10_06_07_2	1	14.00	14.00	14.00	14.00	0.00	181	0.01
10_06_07_3	3	10.50	11.00	11.00	11.00	0.00	65	0.02
10_06_07_4	19	9.17	11.00	11.00	11.00	0.00	59	0.05
10_06_07_5	7	6.29	8.00	8.00	8.00	0.00	88	0.03
10_06_08_1	1	0.00	0.00	0.00	0.00	0.00	450	0.02
10_06_08_2	1	0.00	0.00	0.00	0.00	0.00	101	0.01
10_06_08_3	1	0.00	0.00	0.00	0.00	0.00	56	0.02
10_06_08_4	1	0.00	0.00	0.00	0.00	0.00	319	0
10_06_08_5	1	0.00	0.00	0.00	0.00	0.00	61	0.01
10_06_09_1	1	0.00	0.00	0.00	0.00	0.00	280	0
10_06_09_2	1	0.00	0.00	0.00	0.00	0.00	2563	0.01
10_06_09_3	4	0.00	5.00	0.00	0.00	0.00	2704	0.02
10_06_09_4	1	0.00	0.00	0.00	0.00	0.00	58	0
10_06_09_5	1	0.00	0.00	0.00	0.00	0.00	2178	0.02

Table E.1: Solution for instances with 10 nodes (1 of 3)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
10_06_10_1	2	0.00	7.00	0.00	0.00	0.00	271	0.02
10_06_10_2	1	0.00	0.00	0.00	0.00	0.00	211	0
10_06_10_3	1	0.00	0.00	0.00	0.00	0.00	86	0.01
10_06_10_4	1	0.00	0.00	0.00	0.00	0.00	2318	0
10_06_10_5	1	0.00	0.00	0.00	0.00	0.00	89	0
10_08_07_1	1	12.00	12.00	12.00	12.00	0.00	4895	0.01
10_08_07_2	3	13.00	18.00	13.00	13.00	0.00	355	0.02
10_08_07_3	1	6.00	6.00	6.00	6.00	0.00	70	0
10_08_07_4	1	8.00	8.00	8.00	8.00	0.00	343	0.01
10_08_07_5	3	5.50	6.00	6.00	6.00	0.00	2769	0
10_08_08_1	1	0.00	0.00	0.00	0.00	0.00	53	0
10_08_08_2	1	0.00	0.00	0.00	0.00	0.00	164	0.01
10_08_08_3	1	0.00	0.00	0.00	0.00	0.00	66	0
10_08_08_4	1	0.00	0.00	0.00	0.00	0.00	2784	0
10_08_08_5	1	0.00	0.00	0.00	0.00	0.00	85	0.01
10_08_09_1	1	0.00	0.00	0.00	0.00	0.00	53	0.01
10_08_09_2	1	0.00	0.00	0.00	0.00	0.00	264	0.02
10_08_09_3	1	0.00	0.00	0.00	0.00	0.00	96	0.01
10_08_09_4	1	0.00	0.00	0.00	0.00	0.00	2977	0
10_08_09_5	1	0.00	0.00	0.00	0.00	0.00	69	0.01
10_08_10_1	1	0.00	0.00	0.00	0.00	0.00	313	0.02
10_08_10_2	1	0.00	0.00	0.00	0.00	0.00	84	0.01
10_08_10_3	4	0.00	5.00	0.00	0.00	0.00	245	0.02
10_08_10_4	1	0.00	0.00	0.00	0.00	0.00	278	0.01
10_08_10_5	1	0.00	0.00	0.00	0.00	0.00	61	0
10_08_12_1	1	0.00	0.00	0.00	0.00	0.00	66	0
10_08_12_2	2	0.00	9.00	0.00	0.00	0.00	2729	0
10_08_12_3	1	0.00	0.00	0.00	0.00	0.00	184	0.01
10_08_12_4	1	0.00	0.00	0.00	0.00	0.00	21	0.01
10_08_12_5	1	0.00	0.00	0.00	0.00	0.00	57	0
10_08_16_1	1	0.00	0.00	0.00	0.00	0.00	88	0.02
10_08_16_2	1	0.00	0.00	0.00	0.00	0.00	301	0.01
10_08_16_3	1	0.00	0.00	0.00	0.00	0.00	270	0
10_08_16_4	1	0.00	0.00	0.00	0.00	0.00	56	0.01
10_08_16_5	1	0.00	0.00	0.00	0.00	0.00	76	0.01
10_08_20_1	1	0.00	0.00	0.00	0.00	0.00	204	0.01
10_08_20_2	1	0.00	0.00	0.00	0.00	0.00	57	0.01
10_08_20_3	1	0.00	0.00	0.00	0.00	0.00	391	0
10_08_20_4	1	0.00	0.00	0.00	0.00	0.00	327	0
10_08_20_5	1	0.00	0.00	0.00	0.00	0.00	56	0.01
10_10_07_1	1	12.00	12.00	12.00	12.00	0.00	82	0
10_10_07_2	1	13.00	13.00	13.00	13.00	0.00	2670	0.01
10_10_07_3	1	6.00	6.00	6.00	6.00	0.00	50	0.01
10_10_07_4	1	8.00	8.00	8.00	8.00	0.00	55	0
10_10_07_5	3	5.50	8.00	6.00	6.00	0.00	185	0.01
10_10_08_1	1	0.00	0.00	0.00	0.00	0.00	2746	0.01
10_10_08_2	1	0.00	0.00	0.00	0.00	0.00	2860	0.01
10_10_08_3	1	0.00	0.00	0.00	0.00	0.00	86	0
10_10_08_4	1	0.00	0.00	0.00	0.00	0.00	226	0.01
10_10_08_5	1	0.00	0.00	0.00	0.00	0.00	63	0.01
10_10_09_1	1	0.00	0.00	0.00	0.00	0.00	2382	0
10_10_09_2	1	0.00	0.00	0.00	0.00	0.00	63	0.01
10_10_09_3	1	0.00	0.00	0.00	0.00	0.00	71	0
10_10_09_4	1	0.00	0.00	0.00	0.00	0.00	314	0
10_10_09_5	1	0.00	0.00	0.00	0.00	0.00	317	0.01

Table E.2: Solution for instances with 10 nodes (2 of 3)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
10_10_10_1	1	0.00	0.00	0.00	0.00	0.00	2210	0.01
10_10_10_2	1	0.00	0.00	0.00	0.00	0.00	2446	0.01
10_10_10_3	1	0.00	0.00	0.00	0.00	0.00	62	0.01
10_10_10_4	1	0.00	0.00	0.00	0.00	0.00	1473	0
10_10_10_5	1	0.00	0.00	0.00	0.00	0.00	388	0.02
10_10_12_1	1	0.00	0.00	0.00	0.00	0.00	221	0.01
10_10_12_2	1	0.00	0.00	0.00	0.00	0.00	142	0.01
10_10_12_3	1	0.00	0.00	0.00	0.00	0.00	69	0.01
10_10_12_4	1	0.00	0.00	0.00	0.00	0.00	60	0.01
10_10_12_5	1	0.00	0.00	0.00	0.00	0.00	195	0.01
10_10_16_1	1	0.00	0.00	0.00	0.00	0.00	265	0.01
10_10_16_2	1	0.00	0.00	0.00	0.00	0.00	1944	0
10_10_16_3	1	0.00	0.00	0.00	0.00	0.00	61	0.01
10_10_16_4	1	0.00	0.00	0.00	0.00	0.00	72	0.01
10_10_16_5	1	0.00	0.00	0.00	0.00	0.00	238	0
10_10_20_1	1	0.00	0.00	0.00	0.00	0.00	257	0.01
10_10_20_2	1	0.00	0.00	0.00	0.00	0.00	67	0.01
10_10_20_3	1	0.00	0.00	0.00	0.00	0.00	421	0
10_10_20_4	1	0.00	0.00	0.00	0.00	0.00	2384	0
10_10_20_5	1	0.00	0.00	0.00	0.00	0.00	2859	0
10_12_07_1	1	12.00	12.00	12.00	12.00	0.00	188	0
10_12_07_2	1	13.00	13.00	13.00	13.00	0.00	2216	0
10_12_07_3	1	6.00	6.00	6.00	6.00	0.00	2214	0.01
10_12_07_4	1	8.00	8.00	8.00	8.00	0.00	5248	0.01
10_12_07_5	3	5.50	6.00	6.00	6.00	0.00	66	0.02
10_12_08_1	1	0.00	0.00	0.00	0.00	0.00	69	0.01
10_12_08_2	1	0.00	0.00	0.00	0.00	0.00	45	0
10_12_08_3	1	0.00	0.00	0.00	0.00	0.00	251	0.01
10_12_08_4	1	0.00	0.00	0.00	0.00	0.00	63	0
10_12_08_5	1	0.00	0.00	0.00	0.00	0.00	57	0.01
10_12_09_1	1	0.00	0.00	0.00	0.00	0.00	53	0.01
10_12_09_2	1	0.00	0.00	0.00	0.00	0.00	2929	0.01
10_12_09_3	2	0.00	6.00	0.00	0.00	0.00	90	0.01
10_12_09_4	1	0.00	0.00	0.00	0.00	0.00	89	0
10_12_09_5	1	0.00	0.00	0.00	0.00	0.00	41	0.01
10_12_10_1	1	0.00	0.00	0.00	0.00	0.00	210	0.01
10_12_10_2	1	0.00	0.00	0.00	0.00	0.00	76	0.01
10_12_10_3	2	0.00	5.00	0.00	0.00	0.00	57	0.01
10_12_10_4	1	0.00	0.00	0.00	0.00	0.00	1669	0
10_12_10_5	1	0.00	0.00	0.00	0.00	0.00	319	0.01
10_12_12_1	1	0.00	0.00	0.00	0.00	0.00	2859	0
10_12_12_2	1	0.00	0.00	0.00	0.00	0.00	2643	0
10_12_12_3	1	0.00	0.00	0.00	0.00	0.00	82	0
10_12_12_4	1	0.00	0.00	0.00	0.00	0.00	55	0
10_12_12_5	1	0.00	0.00	0.00	0.00	0.00	2691	0
10_12_16_1	1	0.00	0.00	0.00	0.00	0.00	59	0
10_12_16_2	1	0.00	0.00	0.00	0.00	0.00	335	0.01
10_12_16_3	1	0.00	0.00	0.00	0.00	0.00	2868	0
10_12_16_4	1	0.00	0.00	0.00	0.00	0.00	2616	0.01
10_12_16_5	1	0.00	0.00	0.00	0.00	0.00	76	0.01
10_12_20_1	1	0.00	0.00	0.00	0.00	0.00	2485	0
10_12_20_2	1	0.00	0.00	0.00	0.00	0.00	63	0.01
10_12_20_3	1	0.00	0.00	0.00	0.00	0.00	4465	0
10_12_20_4	1	0.00	0.00	0.00	0.00	0.00	88	0.02
10_12_20_5	1	0.00	0.00	0.00	0.00	0.00	60	0.01

Table E.3: Solution for instances with 10 nodes (3 of 3)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
20_04_07_1	12	38.00	40.00	38.00	38.00	0.00	242	0.24
20_04_07_2	1	31.00	31.00	31.00	31.00	0.00	96	0.04
20_04_07_3	1	32.00	32.00	32.00	32.00	0.00	90	0.1
20_04_07_4	145	31.50	41.00	32.00	32.00	0.00	88	2.62
20_04_07_5	2	35.00	36.00	35.00	35.00	0.00	468	0.07
20_04_08_1	10	8.00	14.00	8.00	8.00	0.00	52	0.31
20_04_08_2	46	8.20	17.00	9.00	9.00	0.00	63	1.67
20_04_08_3	198	8.00	24.00	10.00	10.00	0.00	332	5.76
20_04_08_4	294	7.00	14.00	9.00	9.00	0.00	303	4.81
20_04_08_5	52	11.00	14.00	11.00	11.00	0.00	338	1.12
20_04_09_1	10	0.00	10.00	0.00	0.00	0.00	1855	0.09
20_04_09_2	5	0.00	31.00	0.00	0.00	0.00	66	0.16
20_04_09_3	2	0.00	30.00	0.00	0.00	0.00	73	0.09
20_04_09_4	2	0.00	10.00	0.00	0.00	0.00	294	0.09
20_04_09_5	2	0.00	12.00	0.00	0.00	0.00	2478	0.07
20_04_10_1	2	0.00	10.00	0.00	0.00	0.00	4418	0.04
20_04_10_2	7	0.00	25.00	0.00	0.00	0.00	21	0.06
20_04_10_3	10	0.00	13.00	0.00	0.00	0.00	1718	0.12
20_04_10_4	7	0.00	17.00	0.00	0.00	0.00	72	0.1
20_04_10_5	12	0.00	12.00	0.00	0.00	0.00	75	0.17
20_04_12_1	1	0.00	0.00	0.00	0.00	0.00	154	0.02
20_04_12_2	7	0.00	12.00	0.00	0.00	0.00	98	0.08
20_04_12_3	10	0.00	7.00	0.00	0.00	0.00	65	0.17
20_04_12_4	1	0.00	0.00	0.00	0.00	0.00	73	0.03
20_04_12_5	1	0.00	0.00	0.00	0.00	0.00	336	0.02
20_04_16_1	1	0.00	0.00	0.00	0.00	0.00	94	0.01
20_04_16_2	2	0.00	10.00	0.00	0.00	0.00	68	0.05
20_04_16_3	1	0.00	0.00	0.00	0.00	0.00	54	0.02
20_04_16_4	1	0.00	0.00	0.00	0.00	0.00	58	0.05
20_04_16_5	8	0.00	17.00	0.00	0.00	0.00	162	0.05
20_04_20_1	1	0.00	0.00	0.00	0.00	0.00	338	0.01
20_04_20_2	1	0.00	0.00	0.00	0.00	0.00	341	0.01
20_04_20_3	1	0.00	0.00	0.00	0.00	0.00	57	0.02
20_04_20_4	2	0.00	6.00	0.00	0.00	0.00	2052	0.03
20_04_20_5	1	0.00	0.00	0.00	0.00	0.00	53	0.01
20_05_07_1	1	38.00	38.00	38.00	38.00	0.00	317	0.06
20_05_07_2	2	31.00	47.00	31.00	31.00	0.00	76	0.1
20_05_07_3	39	32.00	34.00	32.00	32.00	0.00	2456	0.78
20_05_07_4	299	31.50	33.00	32.00	32.00	0.00	80	6.98
20_05_07_5	1	35.00	35.00	35.00	35.00	0.00	223	0.04
20_05_08_1	3	8.00	19.00	8.00	8.00	0.00	2627	0.21
20_05_08_2	89	7.00	16.00	7.00	7.00	0.00	2570	2.8
20_05_08_3	93	8.00	13.00	8.00	8.00	0.00	56	1.93
20_05_08_4	63	7.00	9.00	9.00	9.00	0.00	120	1.79
20_05_08_5	182	11.00	14.00	11.00	11.00	0.00	2520	4.1
20_05_09_1	4	0.00	14.00	0.00	0.00	0.00	62	0.07
20_05_09_2	10	0.00	15.00	0.00	0.00	0.00	74	0.17
20_05_09_3	10	0.00	31.00	0.00	0.00	0.00	180	0.13
20_05_09_4	5	0.00	18.00	0.00	0.00	0.00	75	0.1
20_05_09_5	4	0.00	12.00	0.00	0.00	0.00	276	0.05

Table E.4: Solution for instances with 20 nodes (1 of 3)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
20_05_10_1	1	0.00	0.00	0.00	0.00	0.00	57	0.02
20_05_10_2	1	0.00	0.00	0.00	0.00	0.00	2982	0.02
20_05_10_3	8	0.00	12.00	0.00	0.00	0.00	3523	0.09
20_05_10_4	5	0.00	5.00	0.00	0.00	0.00	205	0.09
20_05_10_5	1	0.00	0.00	0.00	0.00	0.00	165	0.01
20_05_12_1	7	0.00	11.00	0.00	0.00	0.00	97	0.09
20_05_12_2	1	0.00	0.00	0.00	0.00	0.00	46	0.02
20_05_12_3	2	0.00	13.00	0.00	0.00	0.00	70	0.03
20_05_12_4	1	0.00	0.00	0.00	0.00	0.00	296	0.02
20_05_12_5	3	0.00	11.00	0.00	0.00	0.00	2430	0.06
20_05_16_1	1	0.00	0.00	0.00	0.00	0.00	2665	0.02
20_05_16_2	1	0.00	0.00	0.00	0.00	0.00	57	0.01
20_05_16_3	3	0.00	8.00	0.00	0.00	0.00	67	0.04
20_05_16_4	4	0.00	5.00	0.00	0.00	0.00	2312	0.06
20_05_16_5	2	0.00	9.00	0.00	0.00	0.00	1708	0.03
20_05_20_1	1	0.00	0.00	0.00	0.00	0.00	50	0.04
20_05_20_2	1	0.00	0.00	0.00	0.00	0.00	279	0.02
20_05_20_3	1	0.00	0.00	0.00	0.00	0.00	386	0.01
20_05_20_4	1	0.00	0.00	0.00	0.00	0.00	2741	0.02
20_05_20_5	2	0.00	9.00	0.00	0.00	0.00	55	0.02
20_06_07_1	4	38.00	42.00	38.00	38.00	0.00	1434	0.2
20_06_07_2	2	31.00	35.00	31.00	31.00	0.00	2264	0.1
20_06_07_3	1	32.00	32.00	32.00	32.00	0.00	185	0.04
20_06_07_4	299	31.50	32.00	32.00	32.00	0.00	282	6.69
20_06_07_5	1	35.00	35.00	35.00	35.00	0.00	99	0.05
20_06_08_1	140	8.00	22.00	8.00	8.00	0.00	66	4.55
20_06_08_2	1	7.00	7.00	7.00	7.00	0.00	2951	0.08
20_06_08_3	4	8.00	17.00	8.00	8.00	0.00	42	0.17
20_06_08_4	111	7.00	14.00	9.00	9.00	0.00	51	3.63
20_06_08_5	84	11.00	21.00	11.00	11.00	0.00	40	2.93
20_06_09_1	1	0.00	0.00	0.00	0.00	0.00	190	0.02
20_06_09_2	4	0.00	20.00	0.00	0.00	0.00	56	0.08
20_06_09_3	7	0.00	12.00	0.00	0.00	0.00	2138	0.09
20_06_09_4	9	0.00	28.00	0.00	0.00	0.00	45	0.14
20_06_09_5	2	0.00	12.00	0.00	0.00	0.00	55	0.04
20_06_10_1	1	0.00	0.00	0.00	0.00	0.00	62	0.03
20_06_10_2	5	0.00	19.00	0.00	0.00	0.00	2720	0.07
20_06_10_3	1	0.00	0.00	0.00	0.00	0.00	81	0.02
20_06_10_4	1	0.00	0.00	0.00	0.00	0.00	98	0.02
20_06_10_5	1	0.00	0.00	0.00	0.00	0.00	54	0.01

Table E.5: Solution for instances with 20 nodes (2 of 3)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
20_08_07_1	1	38.00	38.00	38.00	38.00	0.00	162	0.1
20_08_07_2	1	31.00	31.00	31.00	31.00	0.00	54	0.05
20_08_07_3	3	32.00	33.00	32.00	32.00	0.00	264	0.19
20_08_07_4	305	31.50	37.00	32.00	32.00	0.00	259	8.46
20_08_07_5	3	35.00	36.00	35.00	35.00	0.00	99	0.14
20_08_08_1	1	8.00	8.00	8.00	8.00	0.00	220	0.26
20_08_08_2	1	7.00	7.00	7.00	7.00	0.00	76	0.23
20_08_08_3	2	8.00	13.00	8.00	8.00	0.00	60	0.32
20_08_08_4	85	7.00	14.00	9.00	9.00	0.00	2147	3.59
20_08_08_5	139	11.00	12.00	11.00	11.00	0.00	192	8.79
20_08_09_1	1	0.00	0.00	0.00	0.00	0.00	274	0.03
20_08_09_2	5	0.00	12.00	0.00	0.00	0.00	70	0.07
20_08_09_3	12	0.00	12.00	0.00	0.00	0.00	59	0.14
20_08_09_4	2	0.00	5.00	0.00	0.00	0.00	2694	0.05
20_08_09_5	7	0.00	5.00	0.00	0.00	0.00	162	0.1
20_08_10_1	6	0.00	21.00	0.00	0.00	0.00	63	0.08
20_08_10_2	1	0.00	0.00	0.00	0.00	0.00	156	0.02
20_08_10_3	5	0.00	16.00	0.00	0.00	0.00	59	0.07
20_08_10_4	1	0.00	0.00	0.00	0.00	0.00	219	0.02
20_08_10_5	7	0.00	7.00	0.00	0.00	0.00	42	0.08
20_10_07_1	1	38.00	38.00	38.00	38.00	0.00	289	0.24
20_10_07_2	1	31.00	31.00	31.00	31.00	0.00	80	0.12
20_10_07_3	1	32.00	32.00	32.00	32.00	0.00	3579	0.08
20_10_07_4	305	31.50	36.00	32.00	32.00	0.00	90	10
20_10_07_5	1	35.00	35.00	35.00	35.00	0.00	80	0.06
20_10_08_1	2	8.00	24.00	8.00	8.00	0.00	92	1.11
20_10_08_2	1	7.00	7.00	7.00	7.00	0.00	63	0.34
20_10_08_3	2	8.00	12.00	8.00	8.00	0.00	69	1.04
20_10_08_4	217	7.00	10.00	9.00	9.00	0.00	269	16.83
20_10_08_5	168	11.00	12.00	11.00	11.00	0.00	2890	17.27
20_10_09_1	1	0.00	0.00	0.00	0.00	0.00	119	0.02
20_10_09_2	11	0.00	6.00	0.00	0.00	0.00	2721	0.15
20_10_09_3	1	0.00	0.00	0.00	0.00	0.00	341	0.02
20_10_09_4	6	0.00	21.00	0.00	0.00	0.00	21	0.1
20_10_09_5	7	0.00	13.00	0.00	0.00	0.00	72	0.1
20_10_10_1	5	0.00	5.00	0.00	0.00	0.00	311	0.07
20_10_10_2	11	0.00	5.00	0.00	0.00	0.00	186	0.12
20_10_10_3	6	0.00	10.00	0.00	0.00	0.00	68	0.07
20_10_10_4	9	0.00	12.00	0.00	0.00	0.00	6220	0.09
20_10_10_5	11	0.00	14.00	0.00	0.00	0.00	90	0.08

Table E.6: Solution for instances with 20 nodes (3 of 3)

Inst.	B.B.N.	L.B.R.	U.B.R.	L.B.R.	Best	Gap%	N.Cuts	Time[s]
50_04_07_1	1	147.00	147.00	147.00	147.00	0.00	67	0.42
50_04_07_2	10	159.00	160.00	159.00	159.00	0.00	183	7.83
50_04_07_3	1	149.00	149.00	149.00	149.00	0.00	55	0.64
50_04_07_4	1	155.00	155.00	155.00	155.00	0.00	27	0.96
50_04_07_5	1	155.00	155.00	155.00	155.00	0.00	207	2.12
50_04_08_1	1	94.00	94.00	94.00	94.00	0.00	180	0.94
50_04_08_2	1	100.00	100.00	100.00	100.00	0.00	71	5.88
50_04_08_3	1	98.00	98.00	98.00	98.00	0.00	2786	5.46
50_04_08_4	2	104.00	114.00	104.00	104.00	0.00	3287	10.13
50_04_08_5	3	104.00	116.00	104.00	104.00	0.00	2588	17.09
50_04_09_1	104	45.00	64.00	45.00	45.00	0.00	205	255.22
50_04_09_2	602	51.00	90.00	51.00	51.00	0.00	58	1947.73
50_04_09_3	246	47.00	86.00	47.00	47.00	0.00	180	721.75
50_04_09_4	134	53.00	83.00	53.00	53.00	0.00	46	481.12
50_04_09_5	27	53.00	78.00	53.00	53.00	0.00	2708	98.02
50_04_10_1	26	0.00	31.00	0.00	0.00	0.00	89	5.08
50_04_10_2	61	0.00	30.00	0.00	0.00	0.00	125	12.55
50_04_10_3	106	0.00	30.00	0.00	0.00	0.00	2719	66.5
50_04_10_4	58	0.00	56.00	0.00	0.00	0.00	268	14.97
50_04_10_5	52	0.00	57.00	0.00	0.00	0.00	2562	7.99
50_05_07_1	1	147.00	147.00	147.00	147.00	0.00	2715	0.75
50_05_07_2	2	159.00	161.00	159.00	159.00	0.00	72	9.72
50_05_07_3	1	149.00	149.00	149.00	149.00	0.00	2539	1.94
50_05_07_4	1	155.00	155.00	155.00	155.00	0.00	98	7.3
50_05_07_5	1	155.00	155.00	155.00	155.00	0.00	80	7.61
50_05_08_1	1	94.00	94.00	94.00	94.00	0.00	68	2.75
50_05_08_2	1	100.00	100.00	100.00	100.00	0.00	315	24.9
50_05_08_3	1	98.00	98.00	98.00	98.00	0.00	78	27.58
50_05_08_4	1	104.00	104.00	104.00	104.00	0.00	89	23.33
50_05_08_5	10	104.00	105.00	104.00	104.00	0.00	179	72
50_05_09_1	38	45.00	51.00	45.00	45.00	0.00	315	137.45
50_05_09_2	42	51.00	55.00	51.00	51.00	0.00	474	409.74
50_05_09_3	353	47.00	52.00	47.00	49.00	4.26	66	3600.00
50_05_09_4	1	53.00	53.00	53.00	53.00	0.00	89	67.7
50_05_09_5	287	53.00	61.00	53.00	53.00	0.00	3840	3000.39
50_05_10_1	25	0.00	40.00	0.00	0.00	0.00	53	4.56
50_05_10_2	61	0.00	28.00	0.00	0.00	0.00	70	8.73
50_05_10_3	56	0.00	29.00	0.00	0.00	0.00	158	15.88
50_05_10_4	64	0.00	43.00	0.00	0.00	0.00	72	4.49
50_05_10_5	398	0.00	36.00	0.00	0.00	0.00	60	120.15
50_06_07_1	1	147.00	147.00	147.00	147.00	0.00	88	2.8
50_06_07_2	5	159.00	162.00	159.00	159.00	0.00	2799	35.17
50_06_07_3	1	149.00	149.00	149.00	149.00	0.00	2722	1.49
50_06_07_4	3	155.00	157.00	155.00	155.00	0.00	65	16.29
50_06_07_5	1	155.00	155.00	155.00	155.00	0.00	105	1.87
50_06_08_1	1	94.00	94.00	94.00	94.00	0.00	63	12.06
50_06_08_2	1	100.00	100.00	100.00	100.00	0.00	73	4.89
50_06_08_3	2	98.00	99.00	98.00	98.00	0.00	57	20.02
50_06_08_4	1	104.00	104.00	104.00	104.00	0.00	69	34.92
50_06_08_5	2	104.00	106.00	104.00	104.00	0.00	133	106.79

Table E.7: Solution for instances with 50 nodes (1 of 2)

Inst.	B.B.N.	L.B.R.	U.B.R.	F.B.R.	Best	Gap%	N.Cuts	Time[s]
50_06_09_1	28	45.00	55.00	45.00	45.00	0.00	54	345.04
50_06_09_2	109	51.00	57.00	51.00	51.00	0.00	173	1402.51
50_06_09_3	418	47.00	59.00	47.00	50.00	6.38	52	3600.00
50_06_09_4	39	53.00	67.00	53.00	53.00	0.00	347	877.23
50_06_09_5	1	53.00	53.00	53.00	53.00	0.00	2150	163.6
50_06_10_1	54	0.00	32.00	0.00	0.00	0.00	65	7
50_06_10_2	36	0.00	19.00	0.00	0.00	0.00	312	5.43
50_06_10_3	36	0.00	28.00	0.00	0.00	0.00	381	2.84
50_06_10_4	184	0.00	21.00	0.00	0.00	0.00	1792	44.73
50_06_10_5	499	0.00	13.00	0.00	0.00	0.00	58	256.44
50_08_07_1	1	147.00	147.00	147.00	147.00	0.00	4836	7.59
50_08_07_2	1	159.00	159.00	159.00	159.00	0.00	1694	22.77
50_08_07_3	1	149.00	149.00	149.00	149.00	0.00	181	11.1
50_08_07_4	1	155.00	155.00	155.00	155.00	0.00	89	17.43
50_08_07_5	5	155.00	164.00	155.00	155.00	0.00	85	31.07
50_08_08_1	1	94.00	94.00	94.00	94.00	0.00	1582	81.84
50_08_08_2	1	100.00	100.00	100.00	100.00	0.00	2622	147.24
50_08_08_3	2	98.00	99.00	98.00	98.00	0.00	2288	104.03
50_08_08_4	1	104.00	104.00	104.00	104.00	0.00	47	95.32
50_08_08_5	1	104.00	104.00	104.00	104.00	0.00	63	28.46
50_08_09_1	1	45.00	45.00	45.00	45.00	0.00	90	139.19
50_08_09_2	1	51.00	51.00	51.00	51.00	0.00	376	229.24
50_08_09_3	1	47.00	47.00	47.00	47.00	0.00	70	718.29
50_08_09_4	1	53.00	53.00	53.00	53.00	0.00	202	750.48
50_08_09_5	1	53.00	53.00	53.00	53.00	0.00	1689	266.49
50_08_10_1	106	0.00	34.00	0.00	0.00	0.00	4863	19.72
50_08_10_2	216	0.00	63.00	0.00	0.00	0.00	60	51.65
50_08_10_3	121	0.00	22.00	0.00	0.00	0.00	53	30.15
50_08_10_4	70	0.00	25.00	0.00	0.00	0.00	51	18.1
50_08_10_5	1046	0.00	47.00	0.00	0.00	0.00	87	1709.52
50_10_07_1	1	147.00	147.00	147.00	147.00	0.00	2157	4.22
50_10_07_2	1	159.00	159.00	159.00	159.00	0.00	82	75.93
50_10_07_3	1	149.00	149.00	149.00	149.00	0.00	7758	16.06
50_10_07_4	1	155.00	155.00	155.00	155.00	0.00	94	19.96
50_10_07_5	1	155.00	155.00	155.00	155.00	0.00	443	26.45
50_10_08_1	1	94.00	94.00	94.00	94.00	0.00	106	50.79
50_10_08_2	4	100.00	114.00	100.00	100.00	0.00	6138	406.79
50_10_08_3	1	98.00	98.00	98.00	98.00	0.00	226	58.48
50_10_08_4	1	104.00	104.00	104.00	104.00	0.00	42	121.05
50_10_08_5	1	104.00	104.00	104.00	104.00	0.00	2536	135.07
50_10_09_1	116	45.00	50.00	45.00	46.00	2.22	2211	3600.00
50_10_09_2	1	51.00	51.00	51.00	51.00	0.00	59	691.23
50_10_09_3	28	47.00	61.00	47.00	49.00	4.26	303	3600.00
50_10_09_4	1	53.00	53.00	53.00	53.00	0.00	2371	1076.3
50_10_09_5	68	53.00	59.00	53.00	55.00	3.77	351	3600.00
50_10_10_1	918	0.00	24.00	0.00	0.00	0.00	85	977.97
50_10_10_2	444	0.00	28.00	0.00	0.00	0.00	283	187.97
50_10_10_3	394	0.00	33.00	0.00	0.00	0.00	2846	532.46
50_10_10_4	1	0.00	0.00	0.00	0.00	0.00	292	0.35
50_10_10_5	259	0.00	20.00	0.00	0.00	0.00	21	764.87

Table E.8: Solution for instances with 50 nodes (2 of 2)

Bibliography

[1] A. Weber. *Über den Standort der Industrien - Theory of the Location of Industries*. University of Chicago Press, 1909. English translation by C.J. Friedrich (1929).

[2] R. Love, J. Morris, and G. Wesolowsky. *Facility Location: Models and Methods*. North-Holland, Amsterdam, 1988.

[3] R.Z. Farahani and M. Hekmatfar, editors. *Facility Location: Concepts, Models, Algorithms and Case Studies*. Physica-Verlag, Heidelberg, Germany, 2009.

[4] R.L. Church and A.T. Murray. *Business Site Selection, Location Analysis and GIS*. Wiley, New York, 2009.

[5] R. Zanjirani Farahani, M. Steadie Seifi, and N. Asgari. Multiple criteria facility location problems: a survey. *Applied Mathematical Modelling*, 34(t):1689–1709, 2010.

[6] M.T. Melo, S. Nickel, and F. Saldanha da Gama. Facility location and supply chain management – a review. *European Journal of Operational Research*, 196:401–412, 2009.

[7] J.D. Gough and W.O. McCarthy. *The ambulance facility location problem - a survey of methods and a simple application*. Lincoln College. Agricultural Economics Research Unit., 1975.

[8] H. Yaman. *Concentrator Location in Telecommunications Networks*. Springer, 2004.

[9] K. Takano and M. Arai. A genetic algorithm for the hub-and-spoke problem applied to containerized cargo transport. *Journal of Marine Science and Technology*, 2008.

[10] J. Reese. Solution methods for the p-median problem: An annotated bibliography. *Networks*, 48(3):125–142, October 2006.

[11] Z. Drezner. The p-centre problem-heuristic and optimal algorithms. *The Journal of the Operational Research Society*, 35(8):741–748, August 1984.

[12] E. M. Loiola, N. M. Maia de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007.

[13] N. Mladenović, J. Brimberg, P. Hansen, and J. A. Moreno-Pérez. The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, 179(3):927–939, 2007.

[14] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing.*, 33, March 2004.

[15] A. Klose and S. Görtz. A branch-and-price algorithm for the capacitated facility location problem. *European Journal of Operational Research*, 179(3):1109–1125, 2007.

[16] A. Ceselli and G. Righini. A branch-and-price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, 2005.

[17] N. L. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory 1736-1936.* Clarendon Press, 1984.

[18] A. Schrijver. *Handbooks in Operations Research and Management Science*, chapter On the history of combinatorial optimization (until 1960). Elsevier, 2005.

[19] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2007.

[20] G. Gutin and A.P. Punnen, editors. *The Traveling Salesman Problem and Its Variations.* Springer, 2007.

[21] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[22] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem.* SIAM, 2001.

[23] B.L. Golden, S. Raghavan, and E.A. Wasil, editors. *The Vehicle Routing Problem: Latest Advances and New Challenges.* Springer, 2008.

[24] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 49(2):188–205, 2005.

[25] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[26] C. R. D. Serna and J. P. Bonrostro. Minmax vehicle routing problems: Application to school transport in the province of burgos. *Lecture Notes in Economics and Mathematical Systems*, 505(3):297–317, 2001.

[27] J. F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573–586, 2006.

[28] M. Rajmohan and P. Shahabudeen. Heuristic for solving routing problem in supply chain management. *International Journal of Applied Decision Sciences*, 1(2):153–178, 2008.

[29] L. Santos, J. Coutinho-Rodrigues, and J. R. Current. Implementing a multi-vehicle multi-route spatial decision support system for efficient trash collection in portugal. *Transportation Research Part A: Policy and Practice*, 42(6):922–934, 2008.

[30] M. W. P. Savelsbergh, V. Hemmelmayr, K.F. Doerner, and R.F. Hartl. Delivery strategies for blood products supplies. *OR Spektrum*, 31:707–725, 2009.

[31] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

[32] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39:119–139, 2005.

[33] C. Y. Liong, I. Wan Rosmanira, O. Khairuddin, and Z. Mourad. Vehicle routing problem: models and solutions. *Journal of Quality Measurement and Analysis*, 4:205–218, 2008.

[34] R. Baldacci, E. Bartolini, A. Mingozzi, and R. Roberti. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7:229–268, 2010.

[35] A. Ceselli, G. Righini, and M. Salani. A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, 43(1):56–69, 2009.

[36] F.E. Maranzana. On the location of supply points to minimise transport costs. *Operational Research Quarterly*, page 261–270, 1964.

[37] S. Salhi and G. K. Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

[38] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177:649–672, 2007.

[39] A. W. Chan and D. W. Hearn. A rectilinear distance round-trip location problem. *Transportation Science*, 11(2):107–123, 1977.

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

[41] Rita Macedo, Cláudio Alves, and José M. Valério de Carvalho. Exact algorithms for vehicle routing problems with different service constraints. In *CTW*, pages 215–218, 2009.

[42] J. Ahn, O. de Weck, and J. Hoffman. An optimization framework for global planetary surface exploration campaigns. *Journal of the British Interplanetary Society*, 61(12):487–498, Dec 2008.

[43] Z. Shen, F. Ordonez, and M. Dessouky. A two-stage vehicle routing model for large-scale bioterrorism emergencies. *Networks*, 54(4):255–269, 2009.

[44] L. Ladányi, T. K. Ralphs, and L. E. Trotter. *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, volume 2241 of *Lecture Notes in Computer Science*, chapter Branch, Cut, and Price: Sequential and Parallel, pages 223–260. Springer, 2001.

[45] E. Balas and P. Toth. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, volume 361, chapter Branch and Bound Methods. Wiley, 1985.

[46] J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of Applied Optimization*, pages 65–77, January 2002.

[47] E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42, 1996.

[48] United Nations Department of Economic and Social Affairs Population Division. *World Population Prospects, the 2000 Revision*, chapter 5th - Waste Management, pages 155–197. United Nations publication, 2000.

[49] U.S. Census Bureau. World popclock projection. Available online: `http://www.census.gov/population/popclockworld.html`.

[50] U.S. Census Bureau. International data base - world population: 1950-2050. Available online: `http://www.census.gov/population/international/data/idb/worldpopgraph.php`.

[51] A. Cohen and B. Tagg. Solid waste management services - operating budget analyst notes. Available online: `www.toronto.ca/budget2011/pdf/op11_an_swm.pdf`, January 2011.

[52] United Nations Department of Economic and Social Affairs Division for Sustainable Development. *Trends in Sustainable Development: Chemicals, Mining, Transport and Waste Management*, chapter 4th - Waste Management, pages 26–35. United Nations publication, April 2010.

[53] François Clautiaux, Cláudio Alves, and José M. Valério de Carvalho. A survey of dual-feasible and superadditive functions. *Annals of Operations Research*, 179(1):317–342, September 2010.

[54] J. Desrosiers and M. E. Lübbecke. Branch-price-and-cut algorithms. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[55] J. Desrosiers G Desaulniers and M.M. Solomon. *Column Generation*. Springer, 2005.

[56] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for huge integer programs. *Operations Research*, 46:316–329, 1998.

[57] M. Dror. Note on the complexity of the shortest path models for column generation in vrptw. *Operation Research*, 42(5):977–978, 1994.

[58] G. Righini and M. Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3:255–273, 2006.

[59] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–22, October 2004.

[60] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, May 2008.

[61] A.V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for a*: Efficient point-to-point shortest path algorithms. In *SIAM Workshop on Algorithms Engineering and Experimentation (ALENEX 06)*, 2006.

[62] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[63] R. Baldacci, M. Battarra, and D. Vigo. Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs. *Networks*, 54(4), December 2009.

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

[65] N. Kohl, J. Desrosiers, O. Madsen, M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.

[66] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-gomory rank-1 cuts used in a dantzig-wolfe decomposition of the vehicle routing problem with time windows. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, pages 397–419. Springer, 2008.

[67] C. Archetti, M. Bouchard, and G. Desaulniers. Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Science*, March 2011.

[68] A. Pessoa, E. Uchoa, and M. P. de Aragao. A new stabilization method for column generation. Available online: `http://www.gerad.ca/colloques/ColumnGeneration2008/slides/EduardoU.pdf`. Presented at Column Generation 2008, June 17-20, 2008, Aussois, France.

[69] T. Achterberg. Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, July 2009. `http://mpc.zib.de/index.php/MPC/article/view/4`.

[70] T. Berthold. Heuristics of the branch-cut-and-price-framework scip. Technical report, ZIB, Berlin, October 2007.

[71] NASA. Viking mission to mars. `http://nssdc.gsfc.nasa.gov/planetary/viking.html`. (Date access June 2011).

[72] NASA. Discovery program. `http://discovery.nasa.gov/index.cfml`. (Date access June 2011).

[73] NASA/JPL. Mars pathfinder. `http://mpfwww.jpl.nasa.gov/MPF/default.html`. (Date access June 2011).

[74] F. Forget, F. Costard, and P. Lognonné. *Planet Mars: Story of Another World*, chapter Exploring Mars, pages 183–219. Springer Praxis Books, 2008.

[75] A.F. Chicarro. Mars express mission: Overview and scientific observations. In *5th International Conference on Mars*, Pasadena, California, 1999.

[76] M. Coradini. The exomars program. Technical report, ESA, 2009.

[77] R. Greeley and P. E. Thomas. Mars landing site catalog: The electronic version. `http://cmex.ihmc.us/marstools/mars_cat/Mars_Cat.html`. (Date access June 2011).

[78] H. Yamakawa, H. Ogawa, Y. Kasaba, H. Hayakawa, T. Mukai, and M. Adachi. Current status of the bepicolombo/mmo spacecraft design. *Advances in Space Research*, 33(12):2133–2141, 2004.

[79] J. Ahn. *The Generalized Location Routing Problem with Profits for Planetary Exploration and Terrestrial Applications*. PhD thesis, Massachusetts Institute of Technology, 2008.

[80] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers and Operations Research*, 36:1191–1203, 2009.

[81] A. Bettinelli, A. Ceselli, and G. Righini. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 19(5):723–740, August 2011.

[82] R. O. Roundy J. A. Muckstadt. *Handbooks in Operations Research and Management Science*, volume 4, chapter Analysis of multistage production systems, pages 59–131. Elsevier, 1993.

[83] Zuse Institute Berlin (ZIB). SCIP online documentation. Available online: `http://scip.zib.de/doc/html/index.html`.

[84] CDC's Division of Strategic National Stockpile. Emergency medkit evaluation study summary - background, key results, and next steps. Available online: `http://www.bt.cdc.gov/cri/pdf/medkit-evaluation-summary-2007updated.pdf`, November 2007.

[85] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. Wiley, New York, November 1990.

[86] A. Caprara. Knapsack problems: A book review. *4OR*, 2(4):317–320, December 2004.

[87] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, Germany, 2004.

[88] G. B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 1957.

[89] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42:387–404, 2008.

[90] A. Pessoa, E. Uchoa, and M. Poggi de Aragao. A new stabilization method for column generation. Column Generation Workshop, 2008.

[91] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229 – 237, 1999.

[92] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem—part ii. *Operations Research*, 11(6):863–888, 1963.

[93] J. Desrosiers and M. E. Lübbecke. Selected topics in column generation. *Operations Research*, 53(6), November 2005.

[94] F. Clautiaux, C. Alves, J.M. Valério de Carvalho, and J. Rietz. New stabilization procedures for the cutting stock problem. *INFORMS Journal on Computing*, 2011.

[95] P. Wentges. Weighted dantzig-wolfe decomposition of linear mixed-integer programming. *International Transactions in Operational Research*, 4(2):151–162, November 1997.