Pontifícia Universidade Católica
do Rio de Janeiro

André Mazal Krauss

A Branch and Price Algorithm for a Static Ambulance Routing Problem

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor :	Prof. Thibaut Victor Gaston Vidal
Co-advisor:	Prof. Vincent Gérard Yannick Guigues

Rio de Janeiro
January 2023

André Mazal Krauss

A Branch and Price Algorithm for a Static Ambulance Routing Problem

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Thibaut Victor Gaston Vidal
Advisor
Departamento de Informática – PUC-Rio

Prof. Vincent Gérard Yannick Guigues
Co-advisor
Escola de Matemática Aplicada – FGV

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

Prof. Anton Kleywegt
Georgia Tech –

Rio de Janeiro, January 9th, 2023

André Mazal Krauss

Bachelor's in Computer Science at PUC-Rio (2019)

Bibliographic data

Krauss,André Mazal

A Branch and Price Algorithm for a Static Ambulance Routing Problem / André Mazal Krauss; advisor: Thibaut Victor Gaston Vidal; co-advisor: Vincent Gérard Yannick Guigues. – 2023.

52 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Informática – Teses. 2. Transportes. 3. Roteamento. 4. Serviços de Emergência Médica. 5. Geração de Colunas. 6. Branch and Price. I. Vidal, Thibaut Victor Gaston. II. Gérard Yannick Guigues, Vincent. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Acknowledgments

I'd like to acknowledge my advisors Thibaut Vidal and Vincent Guigues for their continued support and guidance.

I'd like to acknowledge the University PUC-Rio and its Department of Informatics (DI) for all the support given to me since I started my studies in 2014.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

This research project was enabled in part by computing infrastructure and technical support provided by the institutions Compute Canada and Digital Research Alliance of Canada (alliance.can.ca).

We acknowledge the development team responsible for SCIP 7.0 for making this project possible

I'd like to thank Victor Hugo do Nascimento for all the work done on the instance generation process.

Abstract

Krauss,André Mazal; Vidal, Thibaut Victor Gaston (Advisor); Gérard Yannick Guigues, Vincent (Co-Advisor). **A Branch and Price Algorithm for a Static Ambulance Routing Problem**. Rio de Janeiro, 2023. 52p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Emergency Medical Service (EMS) systems provide life-saving support to people in emergency situations via first aid treatment and emergency transport to medical facilities. Such systems must strive to make the best use of their limited resources; they have thus been studied in the context of static and dynamic vehicle routing problems. In this work, we study a static ambulance routing problem aiming to minimize the weighted sum of patients' waiting time while considering ambulance compatibility, patients' priorities, ambulance redirection, and ambulance reassignment. We implement an exact Branch-and-Price algorithm over a Set Partitioning Formulation, study the results of this algorithm, and compare them to previously studied online heuristics using data from Rio de Janeiro's public SAMU system. The results obtained allow us to assess the value of perfect information in such systems, providing a comparative baseline for subsequent developments of online algorithms.

Keywords

Transportation; Routing; Emergency Medical Services; Column Generation; Branch and Price.

Resumo

Krauss,André Mazal; Vidal, Thibaut Victor Gaston; Gérard Yannick Guigues, Vincent. **Um algoritmo Branch and Price para um problema estático de roteamento de ambulâncias**. Rio de Janeiro, 2023. 52p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Serviços Médicos de Emergência (SME) proveem ajuda essencial a pessoas em situações de emergência, através de atendimento com primeiros socorros e transporte para unidades de saúde. Sistemas SME devem utilizar da melhor maneira possível seus recursos limitados de atendimento. Esse desafio já foi amplamente estudado por pesquisadores, na forma de problemas de roteamento de veículos, tanto estáticos quanto dinâmicos. No presente trabalho, estudamos um problema estático de roteamento de ambulâncias, cujo objetivo é minimizar o tempo ponderado de espera dos pacientes. O problema considera também o tempo acumulado de espera, restrições de compatibilidade de ambulâncias a serviços, seleção de pacientes, redirecionamento de ambulâncias e redistribuição de ambulâncias. Implementamos um algoritmo exato usando Branch and Price e uma formulação do problema como uma Partição de Conjuntos, usando código aberto. Estudamos os resultados obtidos com esse algoritmo e os comparamos com métodos heurísticos online estudados anteriormente. Para tal, utilizamos dados obtidos do SAMU da cidade do Rio de Janeiro. Os resultados possibilitam a avaliação do valor de informação perfeita nesse contexto e proveem resultados comparativos para embasar o futuro desenvolvimento de algoritmos online.

Palavras-chave

Transportes; Roteamento; Serviços de Emergência Médica; Geração de Colunas; Branch and Price.

Table of contents

List of figures

List of tables

List of algorithms

1
Introduction

The term Emergency Medical Service (EMS) refers to the array of services provided by public and private operators to assist people in need of urgent medical attention. Such systems typically employ fleets of ambulances, but other vehicle types, such as helicopters or boats, may also be used. Even though each system is bound to have its particular requirements and limitations, a typical EMS system will have the objective and/or the legal obligation of servicing a percentage of calls within a pre-determined response time, which varies according to the severity of the emergency. [Reuter-Oppermann et al., 2017]. At the same time, operators are limited by their budgets, crew, and vehicle capabilities. This naturally gives rise to many optimization problems at the strategic, tactical, and operational levels, and there is a rich literature on the subject.

In the present work, we'll focus on a static vehicle routing problem related to the routing of ambulances in an EMS system. In this problem, we take the viewpoint of an operator aiming at minimizing the total weighted waiting time across a set of emergency requests. Each request‘s weight is assumed proportional to its severity, and all requests are known beforehand. The operator has at its disposal a fleet of different vehicles. Each vehicle may or may not be compatible with each request. The vehicles are allowed to service requests up to a predefined time horizon. If a request isn't serviced by the end of the horizon, a penalty is incurred. Even though foreknowledge of all requests is assumed, vehicles must respect non-anticipativity, that is, they cannot move towards a request before the request's release date. This problem is service-oriented in regard to its objective and is constrained by time and resource limitations. We propose a Branch-and-Price (B&P) algorithm over a Set Partitioning formulation and present experimental results based on data from the public EMS system of the city of Rio de Janeiro.

Despite only developing a method for a static problem, the present work is still useful for comparing online algorithms against the best offline value attainable, assessing expected service levels given known conditions, and possibly imbued as a subproblem in an eventual online algorithm. We aim to demonstrate that a Set Partitioning approach like ours is suited to the problem at hand.

This work is organized as follows: In Section 2 we present the problem statement, notation, and a set partitioning formulation for it. Section 3 is a Literature Review about other works in the literature related to EMS systems, Vehicle Routing Problems (VRP) and similar B&P approaches. In Section 4 we detail the algorithm used for solving the presented problem, and Section 5 shows experimental results.

2
Problem Statement

We will now define the deterministic static vehicle routing problem. The problem can be modelled as a VRP with heterogeneous fleet, heterogeneous demand, compatibility constraints, prize-collecting / visit selection and release dates with minimization of weighted lateness as the objective. See [Vidal et al., 2020] for a summary of these attributes. Subsequently a set-partitioning formulation of the problem will be presented.

2.1
Definition

The problem is defined over a complete digraph $G = (V, E)$, where

$$V = P \cup D \cup W \cup \{0_k | k = 1..m\}$$

is the nodes set and E is a set of directed weighted edges. P is the set $\{1, 2, ..., n\}$ of request nodes and $D = \{n + 1, n + 2, ..., 2n\}$ are destination nodes; each pickup request i is associated with a drop-off at destination $i + n$. Since request-destination pairings are given as input, the choice of optimal hospitals is not modelled here. W is a set of waiting locations. Each edge $e = (i, j)$ has weight t_{ij}, understood as the travel time between i and j. To each request i there is a known release time τ_i, which is the first admissible time the request can be serviced, and there is a lateness weight w_i, proportionate to the severity of the emergency. The demands should be met by a fleet of emergency vehicles $K = \{1, 2, ..., k\}$, each one with a starting position 0_k and time of availability μ_k . To represent compatibility constraints between requests and vehicles, we use a compatibility matrix I, where entry I_i^k is equal to 1 if the request i may be serviced by the vehicle k, and is equal to 0 otherwise. If request i isn't serviced during the planning horizon, a penalty ρ_i is incurred. A request i requires s_i on-site time to be serviced. Finally, time T marks the end of the planning horizon.

Notice that, in order to service a single request i, a vehicle must spend time on different steps. Moving to the request's location takes time t_{vi} if the vehicle was previously at location v; the vehicle spends s_i on site and must move to a hospital at location $i + n$ spending $t_{i,i+n}$ time. Each route should respect non-anticipativity and vehicle compatibility constraints, starting at starting position 0_k and moving to compatible request-destination pairs and optional waiting stations. Additionally, if we allow for redirection, a vehicle moving towards a waiting station may be rerouted to a request before arriving at the waiting station, in which case an intermediate point is dynamically computed and used for calculating the arrival time at said request.

[Guigues et al., 2022] considers two additional possibilities when servicing a request. The first one are requests where there is no need for the patient to be transported to a hospital; the second one are requests after which the ambulance must be cleaned at a specialized cleaning station. Even though we're not modelling them directly, both of these possibilities may be incorporated into the present model by manipulating service times and the distance matrix. Additionally, it is more usual for EMS systems to use a tiered classification of vehicle and request types in order to determine priorities and compatibility. This can be easily translated to the weights and compatibility matrix used here.

The objective in this problem is to plan routes for all vehicles in a way that minimizes the sum of weighted lateness among all requests plus the sum of all non-service penalties, while respecting route constraints. A route is an ordered sequence of vertices, ie. initial positions, requests, waiting stations and destinations. A route is always associated with a particular vehicle. We remark that the objective function in this model is service-oriented instead of the more usual cost-oriented objectives, and that it is equivalent to minimizing the average, over all requests, of the weighted waiting time plus penalty. In some circumstances, it may be mandatory to service all requests. Whenever this is the case, we may emulate this result by precomputing a sufficiently large time horizon combined with sufficiently large non-service penalties.

2.2 Set Partitioning Formulation

A Set Partitioning formulation of the problem is expressed below. Ω is the set of all feasible routes, and $\Omega_k \subset \Omega$ is the set of all feasible routes of vehicle k. λ_σ is a binary variable indicating if route σ is used in the solution. $a_{\sigma i}$ is an integer coefficient equal to the number of times the request i appears in the route σ. For a route $\sigma \in \Omega$, c_σ is the total penalty associated with it (and therefore, with the variable λ_σ). This is calculated as the weighted penalty of each request in the route, that is:

$$c_\sigma = \sum_{i \in \sigma \cap P} w_i \cdot (h_{\sigma i} - \tau_i) \tag{2-1}$$

Where $h_{\sigma i} \geq \tau_i$ is the instant the ambulance arrives at request i. $h_{\sigma i}$ is dependent on the route's structure, and is computed while it is constructed.

All that said, the MIP formulation is as follows:

Objective

$$\min \sum_{\sigma \in \Omega} c_\sigma \lambda_\sigma + \sum_{i \in P} \rho_i y_i \tag{2-2}$$

Subject to:

$$\sum_{\sigma \in \Omega^k} \lambda_\sigma \leq 1 \qquad \forall k \in K \tag{2-3a}$$

$$\sum_{\sigma \in \Omega} a_{\sigma i} \lambda_\sigma + y_i = 1 \qquad \forall i \in P \tag{2-3b}$$

$$\lambda_\sigma \in \{0,1\} \qquad \sigma \in \Omega \tag{2-3c}$$

$$y_i \in \{0,1\} \qquad \forall i \in P \tag{2-3d}$$

The first summation in (2-2) represents the weighted lateness obtained in all serviced requests. The second summation in (2-2) represents the penalty incurred for any requests that aren't serviced. Constraint (2-3a) ensures that each vehicle is used for no more than one route. Constraint (2-3b) ensures that each request is serviced by a route, unless its associated penalty is incurred. Constraint (2-3c) and (2-3d) express that the variables are binary.

This formulation uses a number of variables exponential on the number of vertices, since there is one for each feasible route. This makes it interesting to combine the formulation with a Column Generation procedure, as explained in the methodology section.

3 Literature Review

We start our literature review with a revision on EMS systems and different types of optimization problems arising therefrom. Next, we briefly bring up general VRP problems that relate to our own. Lastly, we review previous works using a Branch and Price(B&P) methodology.

Emergency Medical Services (EMS) Though the term EMS may apply to a wide range of services and types of vehicles, the typical EMS system consists of a fleet of ambulances, deployed with the objective of reaching medical emergencies in a given area and providing adequate medical assistance and/or transport to a hospital. Since EMS systems have a direct impact on people's lives and welfare, and also entail big investments and operational costs from the public and private sectors, it is unsurprising that the theme has been widely studied by researchers in the operational research(OR) field. Furthermore, OR problems arising from the EMS field are varied, ranging from planning to operational levels, and may have to account for high levels of unpredictability in the form of stochastic demand. For a general overview and discussion on EMS systems and their related OR problems, we refer to [Reuter-Oppermann et al., 2017] and [Aringhieri et al., 2017]. Our work aims to further research in the EMS field and contribute towards better management of ambulance fleets. Even though our work solves a static routing problem, we proceed by reviewing the more general literature on the subject.

Location and coverage problems Looking at previous studies, most early works were dedicated to planning problems, aiming to distribute ambulances amongst predetermined positions such that the best quality of service is provided or costs are minimized. Several authors have employed Markov-based Hypercube models to estimate a system's performance given these positions. [Larson, 1974] propose a hypercube model enabling them to analytically compute mean patient wait times, workload imbalances between vehicles, and other performance measures from a set of vehicle positions. [De Souza et al., 2015] extends the usual hyper-cube model to consider request priorities and request queues, in a way that makes it possible to estimate mean wait times for each priority class. They present a case study from the city of Ribeirão Preto, Brazil. [Yoon and Albert, 2018] uses a hypercube model to study the impact of implementing a cutoff priority queue in an EMS system; that is, a vehicle busyness threshold above which low-priority requests are queued or discarded entirely. Another popular methodology for location problems is coverage-based models. In these models, the service region is represented by a set of demand points that must be 'covered' by the ambulances in their designated waiting stations. That is, the waiting station must be close enough to the demand point to be able to arrive at it within a predetermined time. These models are similar to the well-known Set Covering Problem. [Church and Davis, 1992] uses linear programming to solve a Maximal Coverage Location problem, in which the objective is to maximize the number of demand points within a S distance of an ambulance, using a fixed number of ambulances. This, however, does not consider that ambulances may already be busy when a call arrives. [Gendreau, 1997] alleviate this by proposing a double coverage model, solved by tabu search. Other authors such as [Daskin, 1983] have considered an estimated probability q that an ambulance is busy, and use constraints requiring demand points to be serviced with high probability. Lastly, [Gendreau et al., 2001] have proposed computing one optimal positioning for each possible number of available ambulances, and transitioning between these configurations dynamically, while limiting repositioning movements. [Jagtenberg et al., 2017] incorporate coverage criteria for dispatching rules within a Markov Decision Process simulation strategy. We refer to [Brotcorne et al., 2003] for a review on coverage models.

Dynamic Dispatching and Relocation models The present work, however, concerns itself with evaluating operational decisions arising from the routine operation of EMS systems. It becomes necessary to model ambulances and requests individually, and, in fact, the problems studied will generally resemble other problems from the more general literature on dynamic VRPs. In these problems, current system status and open requests are available, but future information is unavailable or is represented by stochastic parameters. Amongst these problems, one of the most fundamental is the ambulance dispatch problem, which aims to select the best ambulance to send to one or more incoming requests. Several systems will simply dispatch the closest available ambulance to an incoming request. This policy is commonly referred to as the closest-available or closest-idle policy [Aringhieri et al., 2017, p.15]. While the closest-available policy is simple to implement and understand, more sophisticated approaches have been studied to try and improve overall performance. For instance, [Andersson and Värbrand, 2007] use a heuristic approach to choose the most appropriate ambulance while also preserving maximum *preparedness*, a value intended to reflect how well the system, in a given moment, is prepared to service one more request. [Bandara et al., 2014] propose a heuristic approach that considers request priorities and maximizes expected patient survivability. [Talarico et al., 2015] use MIP models to minimize the latest completion time weighted by request priority. [Guigues et al., 2022] propose a stochastic two-stage approach with a rolling horizon for dispatching ambulances while also considering how dispatch decisions impact service in a set of future scenarios, allowing for redirections and optimizing hospital choice.

Another related class of real-time problems is ambulance relocation problems, in which the optimal redistribution of ambulances across bases is searched for, considering the current system state and expectation of future requests. [Berman, 1981] evaluate the incorporation of relocation decisions in a hypercube model. [Gendreau et al., 2006] provide a dynamic relocation strategy maximizing expected covered demand, and solve it as an integer linear program. [Andersson and Värbrand, 2007] also propose an integer linear problem to solve a dynamic relocation model which aims to rebalance the distribution of ambulances to raise *preparedness* above predefined levels across different regions. [Guigues et al., 2022], besides their dispatch model, also define a similar model for solving ambulance reassignment decisions. Dispatch and relocation models may be combined in an integrated approach to enable the study of system operations in full.

We take this opportunity to remind the reader that the present work does not directly tackles a dynamic VRP, instead, we choose to provide an exact method for optimally solving a static problem. An optimal solution for this static problem may then be used for assessing the quality of solutions found in the dynamic setting, or may even be used as part of a rolling horizon approach, or hybrid learning and optimization algorithm as in [Parmentier, 2022]. See [Ichoua et al., 2000] for summary of rolling horizon approaches, and the aforementioned [Guigues et al., 2022] for an example on an EMS problem.

Similarities to general static VRPs Some works from the more general static VRP literature are also worth briefly discussing here, as the present work shares several traits with other VRPs. We refer to [Vidal et al., 2020] for a general survey on VRP variants. Its objective function accumulates when servicing consecutive requests, similar to the Time Dependent Traveling Salesman Problem (TDTSP) studied in [Abeledo et al., 2013], the Cumulative Capacitated Vehicle Routing Problem studied in [Ngueveu et al., 2010] and [Damião et al., 2021] or the Traveling Repairman Problem studied in [Afrati et al., 1986]. [Abeledo et al., 2013] states that the TDTSP is harder to solve than its non-cumulative TSP counterpart, suggesting that our problem may also be more difficult than a non-cumulative version of it. Our problem features visit selection / "prize-collecting" like [Stenger et al., 2013], [Vidal et al., 2016] and [Bulhões et al., 2018]. It features an heterogenous fleet of vehicles like [Homsi et al., 2020] and prioritized requests like [Smith et al., 2009], in addition to compatibility constraints between requests and vehicles like [I-Ming et al., 1998] and [Homsi et al., 2020]. The release dates present in our problem can be understood as a time window without an upper bound. The Vehicle Routing Problem with Time Windows (VRPTW) has been surveyed in [Bräysy and Gendreau, 2005].

Regarding our problem, specified in Section 2. It is static and deterministic since all requests are known beforehand and no stochastic information is considered. I has a heterogeneous fleet since vehicles are sorted into different types, and heterogenous demand since requests are sorted by type, and also have an associated weight. It features prize-collecting / visit selection because requests may not be serviced, in which case a penalty is incurred. We refer to [Vidal et al., 2020] for more detailed explanations on these definitions.

Solution methods for VRPs Solution techniques for VRPs can in general be divided into exact or heuristic/meta-heuristic approaches. The first group is able to find provedly optimal solutions, and usually employ Mixed Integer Linear Programming (MILP), while the second group trades certified optimality for smaller runtimes, and have more diverse methodologies. We'll discuss exclusively the first group since we use an exact MILP in the present work. Our algorithm is built on the combination of a Set Partitioning MILP formulation, a pricing procedure to produce new variables, and branch-and-bound for finding integer solutions. Numerous VRP-related works have employed this combination. [Desrosiers et al., 1984] apply it to a variation of the VRPTW with the objetive of minimizing vehicle usage costs and travel costs; experimental results based on school bus transportation problems are presented. [Desrochers et al., 1992] has applied it to the VRPTW itself and tested against benchmark instances. [Bulhões et al., 2018] apply the methodology to a VRP with service levels and visit selection, while [Homsi et al., 2020] use it in a ship routing problem featuring visit selection, vehicle to cargo compatibility constraints and vehicle-dependent starting locations.

[Pessoa et al., 2020] have developed a generic solver for VRPs using Branch-Cut-and-Price called *VRPSolver*, and [Damião et al., 2021] have implemented the Cumulative Capacitated Vehicle Routing Problem within this package. Even though implementing the present problem with the *VRPSolver* package was a promising starting point in our research, we could not model the cumulative aspect of our objective function within this framework in its current state. The approach used by [Damião et al., 2021] would not work because it relies on the fact that customers have the same weight; furthermore, it depends on reasonable estimates of the maximum number of customers serviced in a single route.

B&P methods for VRPs often have an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) as the pricing subproblem, though the ESPPRC, proven to be NP-Hard in [Dror, 1994], is often relaxed to a Shortest Path Problem with Resource Constraints(SPPRC), where routes need not be elementary. Such problems are usually solved with so called "labelling" algorithms using dynamic programming, which are often adaptations of the Bellman-Ford algorithm. This is the case in [Desrosiers et al., 1984]. ESPPRC and SPPRC have also been studied in isolation, precisely because of their importance for B&P methods, as in [Irnich and Villeneuve, 2006], [Righini and Salani, 2008] and [Martinelli et al., 2014]. We refer the reader to [Desrosiers and Lübbecke, 2005], [Lübbecke and Desrosiers, 2005] and [Barnhart et al., 1998] for more general information on column generation techniques.

4
Methodology

In the present chapter we discuss the combination of Column Generation (CG) and branching that constitutes our solving methodology. This combination has been selected for the present work since it allows us to handle the complexities of routing in this context, in particular non-antecipativity and a cumulative route cost function, in a separate pricing subproblem. Other sorts of MIP formulations, such as flow formulations, would need to handle those in the MIP itself. This combination has already been successfully explored in the VRP literature, as discussed in Section 3, and is usually referred to as Branch and Price (B&P). A B&P methodology.

First, consider the SP formulation. As previously stated, solving it directly would mean solving a LP with a number of variables exponential on the number of requests, and this would be intractable for realistic purposes. However, not all variables are in fact needed; in an optimal solution, most σ variables would have value 0, since there are many more feasible routes than there are requests, and a route services multiple requests at once. Knowing that, we may instead solve the LP with a subset $\overline{\Omega}$ of these variables; this smaller LP is called the restricted master problem (RMP) [Desrosiers and Lübbecke, 2005]. Solving the RMP will result in a linearly relaxed solution, from which dual values for each constraint are obtained. This information may in turn be used to search for variables, not already in $\overline{\Omega}$, with negative reduced costs. From LP theory, if these variables exist, they may be added to $\overline{\Omega}$ and will allow for an improvement of the objective value upon re-optimization of the RMP. If they do not exist, then the relaxed solution is proved optimal. Finding these variables, or proving they do not exist, is done in a pricing sub-problem, which has the dual values and all of the graph information of the problem as input. In fact, since each variable corresponds to a feasible route, the pricing subproblem is routing problem over the graph. This solution approach is known as Column Generation (CG) because each new variable corresponds to a new column in the LP matrix.

Once the CG is unable to find new routes, we have obtained an optimal relaxed solution to the RMP. This solution possibly contains fractional variables. If such variables exist, a branch-and-bound procedure is started in which the

LP search space is split into two distinct cases or child nodes. In the first one, the sum of some subset of variables is constrained to 0; in the second one, to 1. The optimal integer solution must fall in one of these cases, thus the algorithm is reapplied recursively to each node and the best solution among them is chosen. Though this may appear excessively costly due to the exponential nature of exploring the search space in this fashion, a known lower bound of the solution may be used to eliminate many nodes without exploring them fully. Thus, the approach is called branch-and-bound.

The previous paragraphs have tried to conceptually explain the B&P methodology, but an extended discussion of it, contemplating its actual implementation, is not the subject of this work. Most of the B&P implementation is in fact handled by the SCIP framework (see [Gamrath et al., 2020]). Instead, we focus only on the problem-specific components which we have implemented. First, we discuss the structure of the negative reduced cost routes in this problem. Then, the pricing algorithm used to construct these routes. Next we present the branching rules used, and lastly some remarks are made about integrating these steps into a complete B&P algorithm.

4.1 Negative Reduced Cost Routes, Labels, and Label Expansions

In order to efficiently search for these new columns in the solution space let us consider the reduced cost of a route σ for a given basis. Let α_k and β_i be the dual variables associated with Constraints (2-3a) and (2-3b) respectively. The reduced cost $\bar{c}_\sigma$ of variable λ_σ associated with route σ of vehicle k can be expressed as:

$$\bar{c}_\sigma = c_\sigma - \alpha_k - \sum_{i \in P} a_{\sigma i} \beta_i \tag{4-1}$$

From this equation a direct relationship between $\bar{c}_\sigma$ and σ can be inferred. In fact, one can easily compute $\bar{c}_\sigma$ by iterating over every request $i \in \sigma$. One should simply sum up each $-\beta_i$, use the arrival times at requests to compute c_σ as per Equation 2-1 and lastly subtract α_k. Also important is the fact that adding a new request j to the end of a route only changes its reduced cost in a known manner; it suffices to subtract β_j and then add in the lateness incurred in this new request. This enables us to construct longer routes by repeatedly expanding smaller ones. We'll now define labels for representing routes and define expansion rules between labels, which will later be used in the pricing algorithm.

We start by defining a label $\mathcal{L}$. A label $\mathcal{L} = (v(\mathcal{L}), \bar{c}(\mathcal{L}), t(\mathcal{L}))$ contains respectively the last vertex in the route, the reduced cost up to and including $v(\mathcal{L})$, and the arrival time at $v(\mathcal{L})$. An initial label $\mathcal{L}_k$ for vehicle k will be constructed simply as

$$\mathcal{L}_k = \left(0^k, -\alpha_k, \mu_k\right) \tag{4-2}$$

Two cases are possible when expanding a label $\mathcal{L}$. The first is extending $\mathcal{L}$ to a request vertex, and the second is expanding to a waiting station vertex. Expanding to initial positions is not needed since they must appear only at the beginning of routes, and expanding to destinations is done implicitly when expanding to requests. For simplicity, labels whose $v(\mathcal{L})$ are requests are called *request labels* and labels whose $v(\mathcal{L})$ are waiting stations are called *waiting station labels*. That said, let us consider the three possible expansion cases: request label to waiting station vertex, request label to request vertex, and waiting label to request vertex. For expansions, initial labels are treated exactly as request labels.

Expanding a request label $\mathcal{L}_i = (i, \bar{c}, t)$ of vehicle k to another request j is permitted under the conditions (let $f = t + s_i + t_{i,i+n}$ be the time when vehicle is ready to leave destination vertex $i + n$):

- $I_j^k = 1$, ie. vehicle k is compatible with request j
- $f \geq \tau_j$, ie. expansion does not violate non-antecipativity
- $f + t_{i+n,j} \leq T$, ie. j is reached within the time horizon T

And the expanded label $\mathcal{L}_j$ is:

$$\mathcal{L}_j = (j, \bar{c} + w_j \cdot (f + t_{i+n,j} - \tau_j) - \beta_j, f + t_{i+n,j}) \tag{4-3}$$

Additionally, the option to consider vehicle redirection from a waiting station to a request has been implemented, and it affects the above request-to-request expansion. The idea is to allow vehicles to change their course from a waiting station to an incoming request, before arriving at said waiting station. It may thus arrive sooner at the request. In order to do this, we must also consider requests happening between f and $f + t_{i+n,w}$, where w is the appropriate waiting station dependant on rerouting rules. Thus, we replace the second condition above to $f + t_{i+n,w} \geq \tau_j$. Whenever redirection is performed, additional movement computations are performed on-the-fly, computing geodesic distances from $i+n$ to an intermediate point and then further to j.

Expanding the same request label to a waiting station w is permitted under the condition that $f + t_{i+n,w} \leq T$, ie. w is reached within the time horizon T. The expanded label $\mathcal{L}_w$ is:

$$\mathcal{L}_w = (w, \bar{c}, f + t_{i+n,w}) \tag{4-4}$$

Lastly, expanding a waiting station label $\mathcal{L}_w = (w, \bar{c}, t)$ to a request i is allowed under condition $t + t_{w,i} \leq T$, and the new label i:

$$\mathcal{L}_i = (i, \bar{c} + w_i \cdot (max(t, \tau_i) + t_{w,i} - \tau_i) - \beta_i, max(t, \tau_i) + t_{w,i}) \tag{4-5}$$

Expansions between waiting stations are not allowed in this problem setting. Thus the three expansions above are enough to cover all possible expansion cases. In each case, route feasibility is guaranteed by the necessary preconditions. In particular, non-antecipativity is guaranteed by only allowing waiting when expanding out of waiting station labels; notice the *max* term in Equation (4-5).

Also of note is that the request to waiting station expansion in (4-4) assumes that the waiting station w is chosen before-hand. The default behaviour is that the waiting station closest to $t(\sigma) + n$ is chosen. This policy is named *Closest Waiting Station*. Another possibility often used in practice is that ambulances always return to their "home base". We refer to this policy as *Fixed Waiting Station*. Additionally, it is simple to explore different waiting station choices in the context of this pricing algorithm. One can simply perform one expansion for each feasible waiting station. We name this policy *Free Waiting Station* and remark that it will be later used to assess the possible impact of ambulance reassignment decisions.

Lastly, let it be noted that every label is associated with a feasible route, unlike other dynamic pricing procedures which have additional criteria for closing routes from existing labels. Here, every route is implicitly closed.

4.2 Label Expanding Algorithm

Having defined the labels and expansions rules, we now define an algorithm able to find the routes with negative reduced costs defined above, if any exist, given the problem data, a vehicle k and the dual values α and β. If no viable routes with negative reduced costs exist, the algorithm is able to detect this and report it. We use a simple label expanding algorithm, adapted from the Bellman-Ford algorithm, in which, starting from small routes, successive expansions are made to construct larger routes, until no more expansions are possible. Multiple labels are kept per vertex, however, dominance checks are performed to discard unnecessary ones.

A simple dominance rule would be that label $l_1 = (i, \bar{c}_1, t_1)$ dominates $l_2 = (i, \bar{c}_2, t_2)$ if $\bar{c}_1 \leq \bar{c}_2$ and $t_1 \leq t_2$. This is the rule used for waiting station labels. However, because of the non-antecipativity constraint, it is possible that, with request labels, l_2 would still generate routes better than l_1's for some expansions. Thus, we've chosen to use a partial enumeration procedure to complement the simpler rule. We test possible expansions to requests arriving between times t_1 and t_2, and if any of l_2's expansions beat l_1's then l_2 isn't dominated; othewise, it is. Though this procedure is costly, we've found that it manages to significantly reduce computation times while maintaining routes that would be ignored by the simpler dominance rule.

Once no new expansions are possible, the n request labels with the best negative reduced cost are selected. If no labels with negative reduced cost exists, the algorithm returns *fail*. Waiting station labels are not returned since, by construction, there must exist an equivalent request label with the same reduced cost and coverage. Reconstructing routes from labels is done by iterating backward, given that each non-initial label stores the label from which it originated. The pseudo-code for this procedure is presented in Algorithm 1.

This algorithm is exact in the sense that it always finds the best routes if any exist. This is understood more easily by considering the same algorithm without dominance checks. If this were the case, every feasible route possible would be constructed, including of course the n best that would be returned. It suffices then to realize that dominance checks do not eliminate the best labels, because they can't be dominated by any other labels. Likewise, none of their partial routes can be dominated either, because it would then be possible to

Algorithm 1: Label expansion Pricing Algorithm

```
Data: vector $P$ of requests, vehicle $k$ with starting position $0^k$,
      number $n$ of desired routes, Dual values $\alpha$ and $\beta$
Result: $n$ best routes found or FAIL
1  Initialize $L[|P| + |W|][]$ a vector of binary search trees of labels ;
2  $success \leftarrow True$ ;
3  while $success$ do
4      $success \leftarrow False$ ;
5      for $i$ in $P \cup W$ do
6          $label \leftarrow$ next unexpanded label in $L[i]$ ;
7          if not $label$ continue ;
8          for $j$ in $P \cup W$ st. $j \neq i$ do
9              $l' \leftarrow LabelExpansion(l, j, \beta_j)$ ;
10             if expansion successful and $l'$ isn't dominated by any label
                 in $L[j]$ then
11                 insert $l'$ in $L[j]$ ;
12                 remove any dominated labels from $L[j]$ ;
13                 $success \leftarrow True$ ;
14 Select $n$ best request labels. If any of them have a negative reduced
     cost, reconstruct their associated routes and return them
```

construct routes that dominate the best routes. However, it should be noted that this algorithm may generate routes with cycles. These repetitions will be eliminated with the convergence of the B&P tree towards the optimal solution, and they do not cause the pricing algorithm to loop infinitely because routes are limited by the time horizon.

A heuristic version of the same algorithm has also been implemented, in which, at each expansion step for request i, only the label with the largest negative reduced cost is expanded. The goal is to have a faster running version of the pricing algorithm, giving up on exactness. A similar approach is used on [Homsi et al., 2020]. The usage of this heuristic pricing in the final B&P is explained in Section 4.4.

4.3
Branch and Price

Given that integer solutions for the problem are desired, a branching procedure is required. We use a total of three branching rules. When branching is required, the algorithm selects which rule to apply based on 'most-fractional' rule. That is, the rule whose sum of variables is closest to 0.5 is selected. Then, new constraints are created in the MIP that restrict the sum to either 0 or 1. The first one of these rules performs branching by detecting fractional usage of vehicles and constraining a vehicle to be either fully used or unused. The second rule detects fractional usage of edges and constrains an edge to be either fully used or not used. The third rule simply restricts a single y variable to 0 or 1. We explain these rules in further detail as follows. All branching rules may cause LP infeasibility at certain nodes. Whenever this happens, a Farkas Pricing method is used to construct new routes that will restore feasibility at that node [Gamrath, 2010, p. 49]. The Farkas pricing method is identical to the usual pricing method used, however it uses Farkas values for each constraint, instead of the usual dual values.

4.3.1 Branching on Vehicles

Branching on vehicles may be applied when, for any $k \in K$, $\sum_{\sigma \in \Omega^k} \lambda_\sigma$ is fractional. If this rule is applied, two child nodes are created; the first one has branching constraint $\sum_{\sigma \in \bar{\Omega}^k} \lambda_\sigma = 0$, the second one $\sum_{\sigma \in \Omega^k} \lambda_\sigma = 1$. There are some precautions that need to be taken when using this branching rule. First, when creating the new child nodes, every existing λ variable associated with σ must be added to the respective branching constraint. Secondly, each future priced variable associated with σ must likewise be added to the constraint. Lastly, the dual values of these new constraints must be considered during pricing. Fortunately, this can be done without changing the pricing algorithm itself, as it suffices to tweak each α_k value used by adding to it the dual value of the respective branching constraint.

4.3.2 Branching on Edges

The second branching rule branches on implicit request-request edges, that is, a pair of requests serviced sequentially. Let $e^\sigma_{ij} = 1$ if requests i and j are serviced sequentially in route σ; that is, i directly follows j or vice-versa, while ignoring waiting stations and redirections. If this is not the case, $e^\sigma_{ij} = 0$. Now consider Equation (4-6). μ_{ij} represents the total usage of edge $(i-j)$ in a given solution. μ_{ij} may be fractional only if at least one route σ that uses $(i-j)$ has fractional λ_σ. Whenever μ_{ij} is fractional for any $i, j \in P$, we may choose to apply this branching rule, creating two child nodes. To the first one the constraint $\mu_{ij} = 0$ is added, and $\mu_{ij} = 1$ to the second one. This means usage of edge $(i-j)$ is forbidden or required respectively. Once again, some precautions must be taken. Already existing variables must be considered when creating the new constraints, and so do any variables created futurely. Additionally, once again, the dual values of branching constraints must be considered when pricing is performed. In order to do this, let us consider constraints $\mu_{ij} = 0$ and $\mu_{ij} = 1$ separately. When $\mu_{ij} = 0$, edge $(i - j)$ is forbidden and all new priced routes must not use it, otherwise, they'd be unused at the current MIP node. Thus, we may simply ignore edge $(i - j)$ in the pricing algorithm. As a result, newly priced variables will not be associated with the branching constraint and are unaffected by its dual value. If $\mu_{ij} = 1$, whenever the pricing algorithm uses an edge $(i - j)$ that has an edge constraint associated with it, the constraint's dual value is subtracted from the reduced cost obtained.

$$\mu_{ij} = \sum_{\sigma \in \overline{\Omega}} e^{\sigma}_{ij} \lambda_{\sigma} \tag{4-6}$$

4.3.3 Branching on y variables

If fractional y variables are obtained in a relaxation, we may simply create two branches with y fixed to 0 or 1. Branching does not affect the duals of the linear program, however, it does affect the pricing algorithm. When pricing routes at a child node, any request i whose variable y_i has been fixed to 1 is ignored since any route servicing i is guaranteed to not be used in the solution due to Constraint (2-3b). Constraining a y variable to 0 has no effect on the pricing algorithm.

4.4 Complete Algorithm

Finally, we present a complete description of the B&P algorithm. This closely follows the typical B&P algorithm, however, we have to account for some problem specifics. First, though this isn't strictly necessary, we have chosen to construct initial columns using routes obtained from a heuristic. This helps ensure that the MILP model is feasible at the root node and accelerates the algorithm. Secondly, both pricing approaches are used ie. the usual reduced cost pricing and Farkas pricing.

Since each vehicle imposes a different pricing subproblem, the pricing algorithm must be run separately for each one. One could simply run the pricing algorithm for every vehicle every time pricing is performed. However, we determined experimentally that it is better to repeatedly price the last vehicle for which routes have been found, and reoptimize the MILP. This is justified because reoptimization is cheap compared to pricing, and enables us to use dual values that are more up-to-date. Once the pricing algorithm fails to find new columns for the current vehicle, the next vehicle is priced, looping back to the first if needed. Pricing fails entirely when no vehicle yields new routes. This is repeated at every node in the B&P tree, however, at the root node, we start by using the heuristic pricing algorithm, and switch to the usual exact pricing only once the heuristic version failed for all requests. Restricting the usage of heuristic pricing to the root node has been experimentally determined to be better than applying it at every B&P node.

Additionally, notice the usage of $NbDesiredRoutes$ to determine the maximum number of columns added each time the pricing algorithm is run. This meta-parameter does not affect the objective value of solutions, but may affect the speed at which solutions are found. A value too small will mean good columns are wasted and convergence will happen slowly; too big and an excessive amount of superfluous columns will be added, slowing the algorithm down. We have used a value of 10 in our experiments.

Algorithm 2 presents a simplified version of the B&P algorithm focusing on the components that were implemented by us in the scope of this project. Other crucial components such as construction of the branching tree, node selection, managing LP solvers, tree pruning and variable aging/cleanup are handled by the SCIP Optimization suite version 7.0.3, and have thus been omitted here. For more details on SCIP's implementation, we refer to [Gamrath et al., 2020] and [Achterberg, 2009].

4.5
Variations in routing rules

Lastly, we summarize the rules variations mentioned throughout this section. Each one of the bullet points below represent independent options, that may combined in any way.

1. *Closest Waiting Station* vs *Fixed Waiting Station* vs *Free Waiting Station* - These options refer to the way in which ambulances are reassigned after finishing their service of a request. If *Closest Waiting Station* is used, the algorithm must send ambulances back to the closest waiting station. If *Fixed Waiting Station* is used, individual ambulances always

Algorithm 2: Complete B&P

```
Data: complete problem data
1  Construct initial solution using the best myopic heuristic
2  Construct Set Partioning MILP model using initial solution
3  $LastSuccesfullyPricedVehicle \leftarrow 0$
4  while Open Nodes Exist do
5      Select an open tree node $node$
6      Solve a linear relaxation of the MILP problem in $node$.
7      if feasible solution found then
8          Set $\alpha, \beta$ to the dual values of solution found
9      else
10         Set $\alpha, \beta$ to the Farkas values of the linear relaxation
11     for $i$ = 0..|K| do
12         $k \leftarrow (LastSuccesfullyPricedVehicle + i)\%|K|$
13         $routes \leftarrow PricingAlgorithm(k, NbDesiredRoutes, \alpha, \beta)$
14         if Pricing sucessfull then
15             create a new column for each route in $routes$
16             $LastSuccesfullyPricedVehicle \leftarrow k$
17             break
18     if new routes were added then
19         continue with next iteration
20     else if solution found is not integer then
21         select most fractional branching rule and apply it
```

return to the same waiting station, referred to as their "home-base". If Free Waiting Station is used, the waiting station resulting in the fastest arrival to the next request is selected.

2. *Redirection* - If redirection is allowed, an ambulance is allowed to change its course midway from a waiting station to an incoming request. If redirection isn't allowed, the ambulance must arrive at the waiting station first before being deployed again. This option changes some criteria for route expansion and, subsequently, may use a redirect value computed on demand, based on geodesic distances from an intermediate point.

5
Experimental Results

For the following results, we consider the implementation made available at GitHub (https://github.com/amk1710/StaticAmbulanceVRP). It was made using C++, double precision floating point arithmetic, and compiled using GCC 9.3.0; it uses the SCIP optimization suite version 7.0.3 with the linear optimizer SoPlex 5.0.2. SCIP is an open-source, non-comercial solver for MIP problems developed at the Zuse Institut Berlin, Berlin, Germany (see [Gamrath et al., 2020] and [Achterberg, 2009]). We ran our tests on the Graham cluster located at the University of Waterloo, Ontario, Canada. Graham is part of Canada's national Advanced Research Computing infrastructure and is currently maintained by the Digital Research Alliance of Canada. Graham is an heterogeneous cluster with different types of CPU cores, which means we cannot precisely determine in which CPU each run of our algorithm has been processed. However, differences in performance between different cores are expected to be small. See the appendix for a list of the CPU cores used.

The instances used here have been generated using data from the SAMU public system of Rio de Janeiro, Brazil. This is the same system underlying the case study in [Guigues et al., 2022]. This system has three types of ambulances, called basic, intermediate and advanced; and requests are sorted into 3 types with the same names. A request may be serviced by any ambulance with an equivalent type or better. That is, basic requests may be serviced by any ambulance, intermediate requests may be serviced by intermediate or advanced ambulances, and advanced requests must be serviced by advanced ambulances. These compatibility constraints may be trivially translated into the matricial format used in our formulation. A request i's given priority w_i is derived from its type. The values are 1, 2 and 4 for basic, intermediate and advanced requests respectively. Requests have been sampled from Poisson distributions calibrated using real call data from the city. The service time s_i is set to 40 minutes for all requests, where 20 minutes are assumed to be used on pick-up procedures on location, and 20 minutes on drop-off procedures at a hospital. Instances have been generated with a combination of 4 time horizons and 3 differently sized study regions within the city. 10 instances have been generated for each combination, totalling 120 instances. The size of the time horizons are 2, 4, 6, and 8 hours, centered around Friday 19:00; that is, the 2 hour time horizon is set to Friday from 18:00 to 20:00, while the 8 hour time horizon is set to Friday from 15:00 to 23:00. These intervals have been selected because they contain the busiest periods for this particular system. The study regions considered are built using 11, 38 or 76 sub-regions of the city, out of a total of 76. This is done in such a way that smaller study regions are fully contained within larger study regions. The study regions will be referred to as Small (S), Medium (M) and Large (L) respectively. The smallest instance has 9 requests, while the larger one has 157. Requests are always directed to the nearest hospital. Real locations of hospitals and ambulance bases are used, totalling 10 hospital and 34 ambulance bases, however, only hospitals and bases inside of each study region are considered. As a result, small, medium and large sized regions use 12, 25 and 34 bases respectively. At each ambulance base, one ambulance is initially stationed. Ambulances are distributed as equally as possible between the three types. Travel times are calculated using geodesic distances and an ambulance speed of 60 km/h.

The sub-regions used to construct the study regions have been obtained by intersecting the city's boundaries with a 10X10 grid of rectangles contained within its bounding box, and then discarding the rectangles without any overlap. For each resulting sub-region a λ value has been calculated for each combination of week-day, half-hour period and request priority. An overview of the resulting values is shown in Figure 5.1.

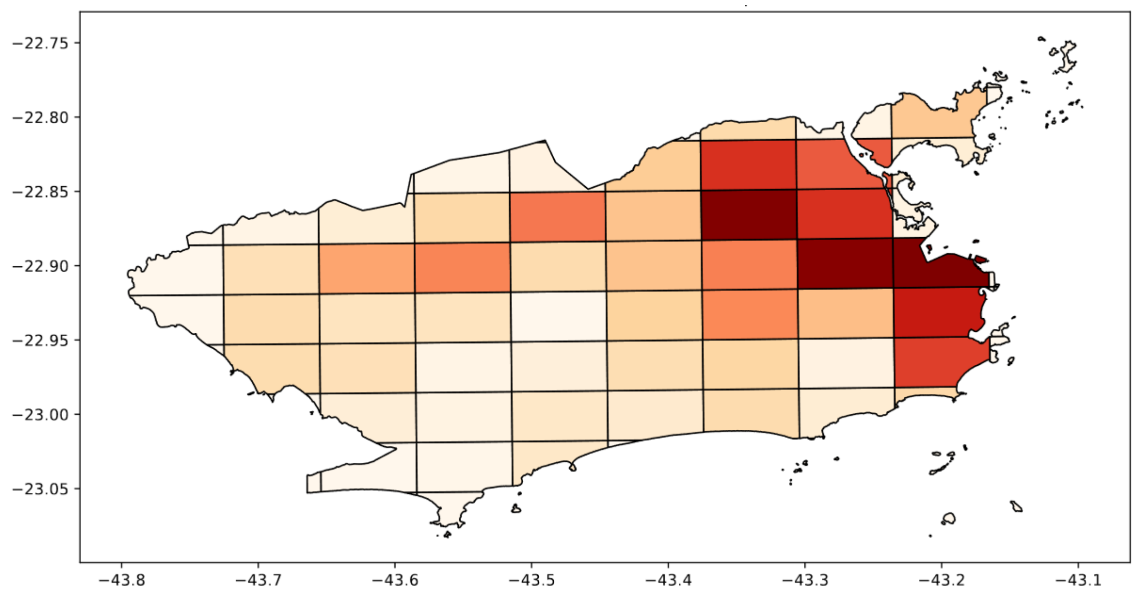


Figure 5.1: Heatmap of mean λ Poisson values for the 76 subregions in the study. Darker tones indicate larger values

We remind the reader that the objective of the present work is limited to developing and evaluating a solution methodology for a static ambulance routing problem. We'd like to stress that the current experiments are designed to validate our methodology and to assess its suitability for future extension to more robust studies and applications considering dynamic requests. The results presented here should not be interpreted as representative of the real conditions of the SAMU system of the city of Rio de Janeiro, and they should not, under any circumstances, be taken as validation for policy changes in the real world.

5.1 Convergence and Performance Considerations

First, we'd like to present experiments showing how well the method is able to find solutions for the proposed problem. That is, how often does the method find optimal solutions, and how long does the solving process take. Out of the 120 RJ instances described above, 115 have been successfully solved within the 2 hours time limit set, when considering the default MIP run. The 5 unsolved instances all belong to the t8_L group. For more details, refer to Table 7.1 in the appendix; unsolved instances have been marked with an '*'. For every table in this section, the column *Obj Value* and *Penalty* show the mean total objective cost and mean objective cost from non-service penalties respectively. *Nb Routes* shows the mean number of routes in solutions. *Not Serviced* or *Not S.* shows the mean number of requests left unattended. *RT* shows the mean response time over all serviced requests, in minutes, and *Weighted RT* or *WRT* shows the mean response time weighted by request priority. Table 5.1 shows aggregated results for the 115 solved instances, grouped by time horizon and region size; Group t8_L has been marked with an '*' as a reminder that some of its instances were unsolved and thus excluded from this comparison. Column *Runtime(s)* shows the mean runtime in seconds spent running the method. *Nb Vars* is the mean number of variables created for instances in the group, while *Nb Nodes* is the mean number of nodes. Lastly, *Reqs/Routes* shows the mean ratio of requests to routes. Correlations are easily found between increases in the size of the time horizon and of the study region to increases in runtime and number of variables. This is further explored in Figure 5.2. It shows runtime results for the RJ instances on the y axis, against other metrics on the x axis. Each dot represents an instance, with blue and red dots representing solved and unsolved instances respectively.

Overall, the method has successfully obtained results for most instances within our time horizon and region size specifications. Larger time horizons and regions have proven more challenging to solve; smaller instances are solved within fractions of a second, while some larger ones were unsolved after 2 hours. It is worth noting that, across all instances, the runtime of the pricing algorithm is responsible for 99.7% of total runtime.

Table 5.1: Results for default MIP by time horizon and region size

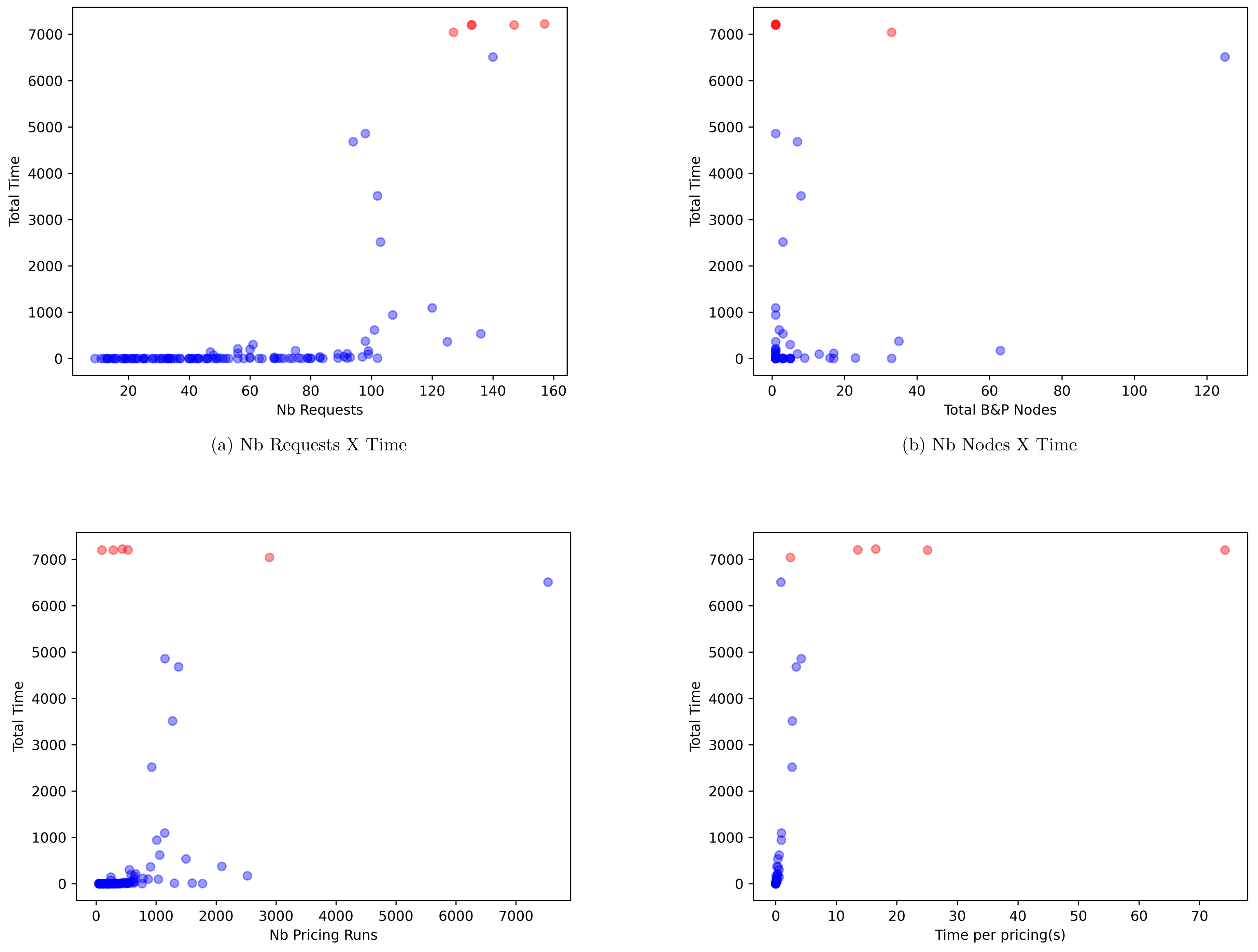
	Nb Reqs	Nb Routes	Obj. V	Penalty	Not S.	RT(m)	WRT	Runtime(s)	Nb Vars	Nb Nodes	Reqs/Routes
t2_S	13.90	9.60	500.92	336.05	0.90	4.13	11.86	0.02	88.70	1.00	1.36
t4_S	25.00	10.70	1293.35	840.06	1.00	5.93	18.13	0.09	363.50	1.00	2.24
t6_S	38.70	12.00	1061.68	288.01	0.20	6.42	19.99	22.49	1062.20	1.60	3.21
t8_S	52.60	12.00	1396.76	480.02	0.40	5.77	17.43	85.39	2670.80	3.00	4.35
t2_M	19.30	15.10	235.31	60.01	0.50	3.35	9.08	0.02	117.20	1.00	1.24
t4_M	46.20	21.80	820.58	360.02	0.60	3.91	9.97	0.44	757.80	3.80	2.09
t6_M	73.70	23.70	1025.51	180.01	0.20	4.27	11.47	36.15	2495.40	7.80	3.10
t8_M	95.00	24.30	1913.12	912.03	0.70	4.10	10.54	1672.23	5862.80	8.00	3.89
t2_L	33.20	23.00	814.64	324.05	1.10	5.35	14.78	0.05	254.60	1.40	1.39
t4_L	66.50	29.80	1245.40	360.03	0.80	5.37	13.31	1.08	1142.10	4.40	2.20
t6_L	89.10	30.20	1722.61	540.02	0.50	5.33	13.27	39.55	2736.00	5.50	2.93
t8_L*	132.50	32.80	4797.46	2352.08	1.70	6.40	18.46	4530.24	6507.20	16.80	3.99

Table 5.2: Aggregated results for 109 instances solved to optimality with variations

	Obj. Value	Penalty	RT(m)	Weighted RT
Online Closest Avail	2171.95	1102.82	6.27	19.31
Offline Mip	1202.74	465.55	4.99	13.31
Mip w fixed station	1429.29	603.14	5.36	14.92
Mip w Opt Reass	846.19	440.72	2.67	7.32
Mip w Redirection	1201.85	465.55	4.98	13.30
Mip w Opt Reass and Red	789.77	412.78	2.43	6.81

5.2 Comparing rules variations and online heuristics

So far we have shown results only for the default MIP procedure. We now compare how online heuristics and other MIP variants perform. See Section 4.5 for a description of variations. In Table 5.2 aggregated results for 109 RJ instances and six solving variations are presented. Eleven instances have not been optimally solved for every variation and have been excluded. In Table 5.3 the same results are presented, but segregated by size of the time horizon. For assessing the value of perfect information on dispatch decisions, one should compare the Offline MIP, which obtains an optimal solution using complete information over all requests, to the Online Closest Available heuristic, in which the closest available vehicle is always dispatched to a request [Aringhieri et al., 2017, p.15]. This comparison indeed shows an improvement of 31% on the mean weighted wait time and 20% on the mean wait time. In turn, measuring the improvement obtained in the reassignment decisions should be done by comparing how the variations to the reassignment behaviour affects the objective metrics. The simple Offline MIP, which uses the closest waiting station rule, has a weighted response time about 10% smaller than the fixed waiting station rule. A larger improvement is obtained when incorporating total freedom in reassignment decisions, that is, when using the Free Waiting Station variation. This yields a large improvement of 45% on the weighted RT when compared to the default MIP, and 51% when compared to the fixed waiting station rule. While incorporating rerouting from waiting stations to requests into the simple offline approach only produced a negligible improvement, using both rerouting and Free Waiting Station simultaneously yielded the best overall result.



(a) Nb Requests X Time

(b) Nb Nodes X Time

(c) Nb Pricing Runs X Time

(d) Time per Pricing X Time

Figure 5.2: Comparison of different measures against computation time for default MIP runs

Method	Obj. Value	Penalty	Nb Routes	Not Serviced	RT(m)	Weighted RT
2 Hours						
Online Closest Avail	738.85	392.05	15.57	1.23	5.16	15.67
Offline Mip	516.96	240.03	15.90	0.83	4.51	12.51
Mip w fixed station	581.03	284.04	15.93	0.97	4.73	13.42
Mip w Opt Reass	317.08	176.02	14.67	0.70	2.42	6.37
Mip w Redirection	514.13	240.03	15.77	0.83	4.45	12.38
Mip w Opt Reass and Red	295.99	172.02	14.83	0.67	2.10	5.60
4 Hours						
Online Closest Avail	1850.41	1016.07	20.10	1.47	5.99	18.18
Offline Mip	1119.78	520.04	20.77	0.80	4.98	13.07
Mip w fixed station	1309.75	648.05	20.37	0.93	5.24	14.42
Mip w Opt Reass	866.75	536.04	17.60	0.83	2.67	7.21
Mip w Redirection	1119.02	520.04	20.70	0.80	4.96	13.05
Mip w Opt Reass and Red	770.90	464.03	17.30	0.70	2.42	6.69
6 Hours						
Online Closest Avail	2705.36	1365.58	21.76	1.17	6.41	19.75
Offline Mip	1264.24	347.60	22.31	0.31	5.06	13.51
Mip w fixed station	1568.01	521.40	22.17	0.45	5.58	15.43
Mip w Opt Reass	799.96	297.94	19.07	0.28	2.70	7.40
Mip w Redirection	1264.72	347.60	22.17	0.31	5.07	13.52
Mip w Opt Reass and Red	763.14	297.94	18.93	0.28	2.44	6.86
8 Hours						
Online Closest Avail	2668.38	1176.04	18.20	0.90	6.34	20.17
Offline Mip	1692.02	720.03	18.45	0.60	4.72	13.14
Mip w fixed station	2000.48	912.03	18.30	0.70	5.04	14.71
Mip w Opt Reass	1326.29	720.03	16.40	0.60	2.81	8.19
Mip w Redirection	1690.12	720.03	18.45	0.60	4.69	13.11
Mip w Opt Reass and Red	1255.41	672.02	16.30	0.50	2.67	7.88

Table 5.3: Aggregated results for 109 instances with different variations, grouped by time horizon

5.3
Sensitivity tests on the number of vehicles used

We now present sensitivity tests varying the number of vehicles available in the system. For this test, instance groups t2_L and t6_M have been selected, in order to showcase the two principal considerations from this test. When varying the number of vehicles, vehicles are distributed sequentially across all bases in the study region. The instances are otherwise unchanged. Figure 5.3 shows a boxplot of the mean objective value with a varying number of vehicles for instance set t2_L. As expected, the value decreases with an increase in the number of vehicles. Still, one can see that the improvement provided by additional vehicles diminishes when more vehicles are available, and that this improvement is quite small after 30 vehicles. This is justified by the fact that the instance set has only 33 requests per instance on average. The second consideration to this test is the fact that using fewer vehicles has a strong impact on the method's runtime, as fewer vehicles means a more challenging instance with more requests per route on average. This is shown in Figure 5.4, where red dots indicate that some instances in the group weren't solved to optimality within two hours.

Figure 5.3: Solution cost with varying number of vehicles for instance set t2_L

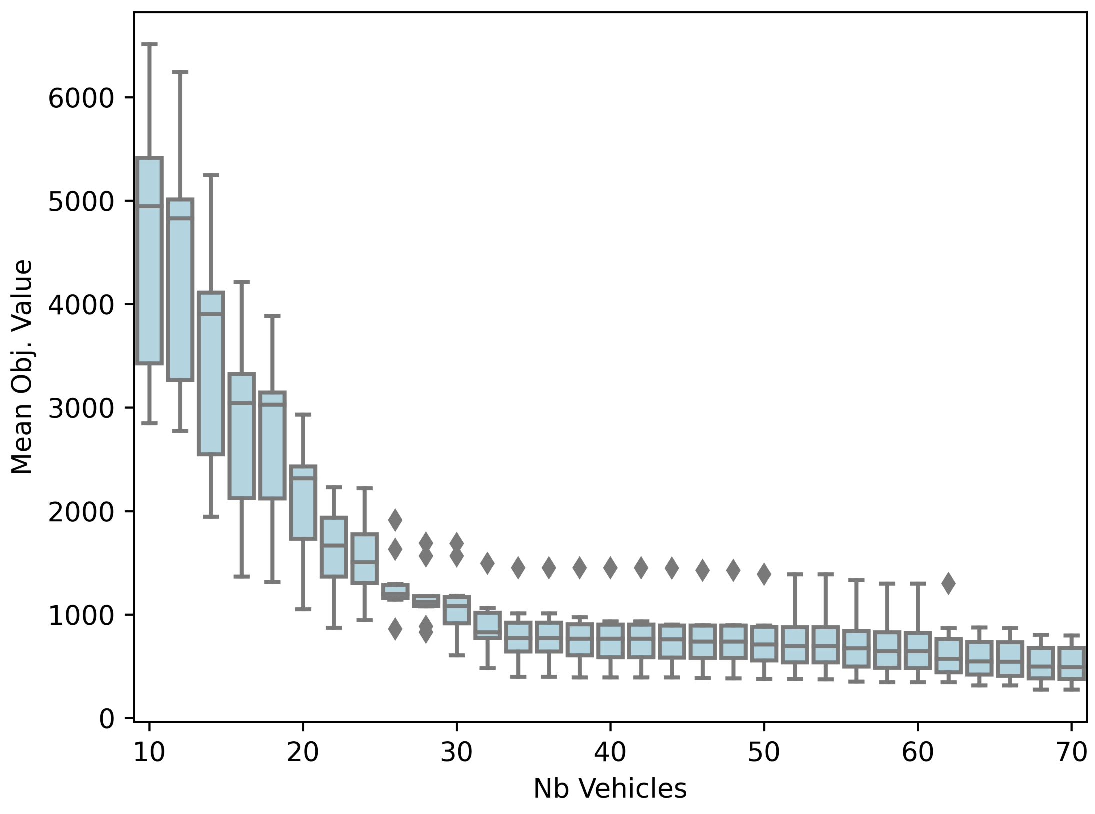
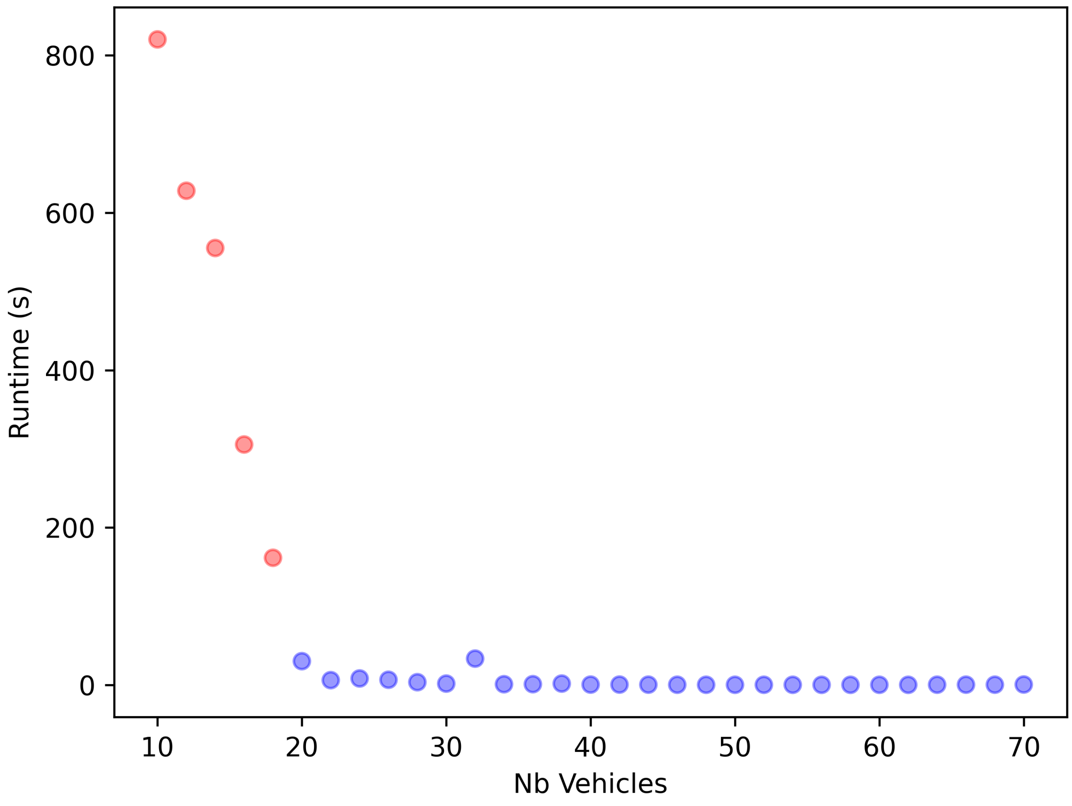


Figure 5.4: Runtime with varying number of vehicles for instance set t6_M



6
Conclusing Remarks

In this work, we have considered a static ambulance routing problem and developed a Set Partioning-based B&P methodology to obtain exact solutions for it. We tested our method on a number of differently sized instances constructed from data obtained from the city of Rio de Janeiro, Brazil. Our tests show that the method can successfully solve smaller instances in runtimes sufficiently small to enable its usage in real-time applications. Larger instances proved more challenging, but most could still be solved optimally within two hours. Stress situations, when the number of requests is excessively larger than the number of ambulances, have proved challenging even with medium-sized instances.

There are multiple possible venues of future research. It would be interesting to further enhance the method with speed-up techniques, particularly in the pricing algorithm, such as with 2 and 3-cycle eliminations ([Irnich and Villeneuve, 2006]), ng-routes and Decremental State-space relaxation as in [Righini and Salani, 2008], [Desaulniers et al., 2002] and [Martinelli et al., 2014]), and possibly some subproblem aggregation strategy for exploiting similarities between pricing runs with different vehicles and dual values. Another important research venue is developing an online algorithm using this offline approach as a subprocedure. This is possible, for example, by using a rolling horizon as in [Guigues et al., 2022], or using a hybrid learning and optimization algorithm as in [Parmentier, 2022]. Lastly, one could study the incorporation of resources with a cumulative property into the generic solver developed by [Pessoa et al., 2020], which would hopefully enable it to solve this problem as well as other TDTSP-like VRPs.

Bibliography

[Abeledo et al., 2013] Abeledo, H., Fukasawa, R., Pessoa, A., and Uchoa, E. (2013). The time dependent traveling salesman problem: Polyhedra and algorithm. *Mathematical Programming Computation*, 5(1):27–55.

[Achterberg, 2009] Achterberg, T. (2009). *Constraint Integer Programming.* PhD thesis.

[Afrati et al., 1986] Afrati, F., Cosmadakis, S., Papadimitriou, C. H., Papageorgiou, G., and Papakostantinou, N. (1986). The complexity ofthe travelling repairman problem. *Informatique théorique et applications*, 20(1):79–87.

[Andersson and Värbrand, 2007] Andersson, T. and Värbrand, P. (2007). Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society*, 58(2):195–201.

[Aringhieri et al., 2017] Aringhieri, R., Bruni, M. E., Khodaparasti, S., and van Essen, J. T. (2017). Emergency medical services and beyond: Addressing new challenges through a wide literature review. *Computers and Operations Research*, 78:349–368.

[Bandara et al., 2014] Bandara, D., Mayorga, M. E., and Mclay, L. A. (2014). Priority dispatching strategies for EMS systems. *Journal of the Operational Research Society*, 65(4):572–587.

[Barnhart et al., 1998] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-Price: Column Generation Solving Huge Integer Programs. *Operations Research*, 46(3):316–329.

[Berman, 1981] Berman, O. (1981). Repositioning of distinguishable urban service units on networks. *Computers and Operations Research*, 8(2):105–118.

[Bräysy and Gendreau, 2005] Bräysy, O. and Gendreau, M. (2005). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118.

[Brotcorne et al., 2003] Brotcorne, L., Laporte, G., and Semet, F. (2003). Ambulance location and relocation models. *European Journal of Operational Research*, 147(3):451–463.

[Bulhões et al., 2018] Bulhões, T., Hà, M. H., Martinelli, R., and Vidal, T. (2018). The vehicle routing problem with service level constraints. *European Journal of Operational Research*, 265(2):544–558.

[Church and Davis, 1992] Church, R. L. and Davis, R. R. (1992). The fixed charge maximal covering location problem. *Papers in Regional Science*, 71(3):199–215.

[Damião et al., 2021] Damião, C. M., Silva, J. M. P., and Uchoa, E. (2021). A branch-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *4or*.

[Daskin, 1983] Daskin, M. S. (1983). A Maximum Expected Covering Location Model. *Transportation Science*, 17(1):48–70.

[De Souza et al., 2015] De Souza, R. M., Morabito, R., Chiyoshi, F. Y., and Iannoni, A. P. (2015). Incorporating priorities for waiting customers in the hypercube queuing model with application to an emergency medical service system in Brazil. *European Journal of Operational Research*, 242(1):274–285.

[Desaulniers et al., 2002] Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2002). 4 ACCELERATING STRATEGIES IN COLUMN GENERATION METHODS FOR VEHICLE ROUTING AND CREW SCHEDULING PROBLEMS.

[Desrochers et al., 1992] Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows.

[Desrosiers and Lübbecke, 2005] Desrosiers, J. and Lübbecke, M. (2005). *A Primer in Column Generation*, chapter 1, pages 1–32. Springer New York, New York, 1 edition.

[Desrosiers et al., 1984] Desrosiers, J., Soumis, F., and Desrochers, M. (1984). Routing with time windows by column generation. *Networks*, 14(4):545–565.

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

[Gamrath, 2010] Gamrath, G. (2010). *Generic branch-cut-and-price.* PhD thesis, Technische Universit¨at Berlin.

[Gamrath et al., 2020] Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Le Bodic, P., Maher, S. J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D., and Witzig, J. (2020). The SCIP Optimization Suite 7.0. Technical report, Optimization Online.

[Gendreau, 1997] Gendreau, M. (1997). Solving an ambulance location model by tabu search. *Location Science*, 5(2):75–88.

[Gendreau et al., 2001] Gendreau, M., Laporte, G., and Semet, F. (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing*, 27(12):1641–1653.

[Gendreau et al., 2006] Gendreau, M., Laporte, G., and Semet, F. (2006). The maximal expected coverage relocation problem for emergency vehicles. *Journal of the Operational Research Society*, 57(1):22–28.

[Guigues et al., 2022] Guigues, V., Kleywegt, A., and Nascimento, V. H. (2022). Operation of an ambulance fleet under uncertainty.

[Homsi et al., 2020] Homsi, G., Martinelli, R., Vidal, T., and Fagerholt, K. (2020). Industrial and tramp ship routing problems: Closing the gap for real-scale instances. *European Journal of Operational Research*, 283(3):972–990.

[I-Ming et al., 1998] I-Ming, C., Golden, B. L., and Wasil, E. A. (1998). A New Algorithm for the Site-Dependent Vehicle Routing Problem. In Woodruff, D. L., editor, *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*, pages 301–312. Springer US, Boston, MA.

[Ichoua et al., 2000] Ichoua, S., Gendreau, M., and Potvin, J. Y. (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science*, 34(4):426–438.

[Irnich and Villeneuve, 2006] Irnich, S. and Villeneuve, D. (2006). The shortest-path problem with resource constraints and k-cycle elimination for k $\geq$ 3. *INFORMS Journal on Computing*, 18(3):391–406.

[Jagtenberg et al., 2017] Jagtenberg, C. J., Bhulai, S., and van der Mei, R. D. (2017). Dynamic ambulance dispatching: is the closest-idle policy always optimal? *Health Care Management Science*, 20(4):517–531.

[Larson, 1974] Larson, R. C. (1974). A hypercube queuing model for facility location and redistricting in urban emergency services. *Computers and Operations Research*, 1(1):67–95.

[Lübbecke and Desrosiers, 2005] Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, 53(6):1007–1023.

[Martinelli et al., 2014] Martinelli, R., Pecin, D., and Poggi, M. (2014). Efficient elementary and restricted non-elementary route pricing. *European Journal of Operational Research*, 239(1):102–111.

[Ngueveu et al., 2010] Ngueveu, S. U., Prins, C., and Wolfler Calvo, R. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers and Operations Research*, 37(11):1877–1885.

[Parmentier, 2022] Parmentier, A. (2022). Learning to approximate industrial problems by operations research classic problems. *Operations Research*, 70(1):606–623.

[Pessoa et al., 2020] Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183(1-2):483–523.

[Reuter-Oppermann et al., 2017] Reuter-Oppermann, M., van den Berg, P. L., and Vile, J. L. (2017). Logistics for Emergency Medical Service systems. *Health Systems*, 6(3):187–208.

[Righini and Salani, 2008] Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170.

[Smith et al., 2009] Smith, S. L., Pavone, M., Bullo, F., and Frazzoli, E. (2009). Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization*, 48(5):3224–3245.

[Stenger et al., 2013] Stenger, A., Schneider, M., and Goeke, D. (2013). The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. *EURO Journal on Transportation and Logistics*, 2(1-2):57–87.

[Talarico et al., 2015] Talarico, L., Meisel, F., and Sörensen, K. (2015). Ambulance routing for disaster response with patient groups. *Computers and Operations Research*, 56:120–133.

[Vidal et al., 2020] Vidal, T., Laporte, G., and Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 286(2):401–416.

[Vidal et al., 2016] Vidal, T., Maculan, N., Ochi, L. S., and Penna, P. H. V. (2016). Large neighborhoods with implicit customer selection for vehicle routing problems with profits. *Transportation Science*, 50(2):720–734.

[Yoon and Albert, 2018] Yoon, S. and Albert, L. A. (2018). An expected coverage model with a cutoff priority queue. *Health Care Management Science*, 21(4):517–533.

7
Appendix

7.1
Nomenclature Reference Table

Set of Request vertices	P
Set of Destination vertices	D
Set of Waiting Station vertices	W
Travel time between vertices i and j	d_{ij}
Release time of request i	τ_i
Service time of request i	s_i
Required cleaning time of request i	l_i
Lateness weight of request i	w_i
Non-service penalty of request i	ρ_i
Set of vehicles	K
Starting position of vehicle k	0_k
Time when vehicle k is first available	AV_k
Compatibility boolean between request i and vehicle k	I_i^k
Time horizon	T

7.2

Full table of results of MIP for RJ instances

Table 7.1: Results for 120 RJ instances

Instance	Obj. Value	Runtime(s)	Nb Nodes	Nb Vars
RJ_t2_S-0-12	245.15	0.02	1	85
RJ_t2_S-1-16	706.71	0.02	1	92
RJ_t2_S-2-13	181.03	0.01	1	53
RJ_t2_S-3-18	408.02	0.02	1	147
RJ_t2_S-4-11	218.22	0.01	1	64
RJ_t2_S-5-14	884.27	0.01	1	80
RJ_t2_S-6-13	103.88	0.02	1	71
RJ_t2_S-7-9	54.83	0.01	1	34

Continued on next page

Table 7.1: Results for 120 RJ instances

Instance	Obj. Value	Runtime(s)	Nb Nodes	Nb Vars
RJ_t2_S-8-18	2044.87	0.02	1	107
RJ_t2_S-9-15	162.19	0.02	1	154
RJ_t2_M-0-21	476.21	0.02	1	84
RJ_t2_M-1-19	217.65	0.02	1	121
RJ_t2_M-2-20	224.46	0.03	1	177
RJ_t2_M-3-22	214.24	0.02	1	117
RJ_t2_M-4-16	213.11	0.01	1	112
RJ_t2_M-5-15	90.65	0.02	1	112
RJ_t2_M-6-19	301.28	0.02	1	179
RJ_t2_M-7-25	144.39	0.03	1	121
RJ_t2_M-8-23	243.61	0.03	1	121
RJ_t2_M-9-13	227.52	0.01	1	28
RJ_t2_L-0-30	927.79	0.07	5	154
RJ_t2_L-1-33	398.65	0.04	1	196
RJ_t2_L-2-31	899.39	0.04	1	319
RJ_t2_L-3-25	705.50	0.02	1	106
RJ_t2_L-4-28	601.93	0.03	1	178
RJ_t2_L-5-36	673.64	0.06	1	238
RJ_t2_L-6-37	633.79	0.08	1	419
RJ_t2_L-7-41	1010.33	0.05	1	342
RJ_t2_L-8-40	844.74	0.07	1	377
RJ_t2_L-9-31	1450.59	0.03	1	217
RJ_t4_S-0-28	701.00	0.06	1	373
RJ_t4_S-1-32	2265.17	0.16	1	452
RJ_t4_S-2-22	1586.63	0.02	1	223
RJ_t4_S-3-23	293.68	0.06	1	377
RJ_t4_S-4-29	536.21	0.08	1	443
RJ_t4_S-5-25	1186.73	0.06	1	407
RJ_t4_S-6-25	1727.29	0.15	1	442
RJ_t4_S-7-21	1144.96	0.02	1	149
RJ_t4_S-8-19	771.44	0.07	1	369
RJ_t4_S-9-26	2720.42	0.19	1	400
RJ_t4_M-0-51	527.95	0.54	5	934
RJ_t4_M-1-46	417.78	0.18	1	617

Continued on next page

Table 7.1: Results for 120 RJ instances

Instance	Obj. Value	Runtime(s)	Nb Nodes	Nb Vars
RJ_t4_M-2-40	1156.42	0.10	1	331
RJ_t4_M-3-43	401.03	0.29	5	809
RJ_t4_M-4-37	665.51	0.11	1	535
RJ_t4_M-5-52	877.43	0.36	3	891
RJ_t4_M-6-45	559.71	0.74	17	766
RJ_t4_M-7-53	607.24	1.42	1	1325
RJ_t4_M-8-49	1310.54	0.23	1	725
RJ_t4_M-9-46	1682.19	0.43	3	645
RJ_t4_L-0-63	826.59	1.26	1	1217
RJ_t4_L-1-56	1507.74	0.24	1	721
RJ_t4_L-2-64	1339.38	0.64	1	971
RJ_t4_L-3-69	790.23	2.43	33	1352
RJ_t4_L-4-80	1908.04	1.78	1	1515
RJ_t4_L-5-58	976.42	0.36	1	820
RJ_t4_L-6-77	1161.88	1.14	1	1478
RJ_t4_L-7-84	2339.88	1.95	1	1677
RJ_t4_L-8-68	1074.96	0.89	3	1122
RJ_t4_L-9-46	528.90	0.12	1	548
RJ_t6_S-0-48	1435.06	72.47	1	1673
RJ_t6_S-1-34	1683.86	0.09	1	474
RJ_t6_S-2-42	517.24	1.40	3	1488
RJ_t6_S-3-35	1667.95	0.10	1	529
RJ_t6_S-4-33	380.65	0.26	1	707
RJ_t6_S-5-33	1381.74	5.25	1	1093
RJ_t6_S-6-34	823.89	0.78	1	956
RJ_t6_S-7-47	1898.71	143.22	1	1623
RJ_t6_S-8-40	361.13	0.35	1	830
RJ_t6_S-9-41	466.61	0.99	5	1249
RJ_t6_M-0-71	683.56	3.51	5	1777
RJ_t6_M-1-92	2914.35	104.33	1	4140
RJ_t6_M-2-60	432.27	26.65	1	3352
RJ_t6_M-3-68	647.87	2.08	1	1480
RJ_t6_M-4-68	799.18	28.63	1	2711
RJ_t6_M-5-74	1279.05	2.57	1	1812

Continued on next page

Table 7.1: Results for 120 RJ instances

Instance	Obj. Value	Runtime(s)	Nb Nodes	Nb Vars
RJ_t6_M-6-70	631.77	7.49	1	2374
RJ_t6_M-7-80	632.08	6.42	3	2304
RJ_t6_M-8-79	909.74	8.49	1	2320
RJ_t6_M-9-75	1325.26	171.32	63	2684
RJ_t6_L-0-91	2270.62	41.13	1	3532
RJ_t6_L-1-83	1310.82	13.97	1	2389
RJ_t6_L-2-99	1259.29	163.79	1	3711
RJ_t6_L-3-97	1606.87	37.88	1	3566
RJ_t6_L-4-102	1217.76	7.92	1	2661
RJ_t6_L-5-76	924.34	11.34	23	2535
RJ_t6_L-6-73	2137.93	3.34	3	1724
RJ_t6_L-7-92	2635.33	7.53	1	2384
RJ_t6_L-8-89	1237.87	9.87	16	2144
RJ_t6_L-9-89	2625.23	98.75	7	2714
RJ_t8_S-0-56	1185.87	211.35	1	5487
RJ_t8_S-1-60	1240.52	197.47	1	4539
RJ_t8_S-2-43	413.29	3.62	1	1390
RJ_t8_S-3-43	2293.39	0.61	1	1218
RJ_t8_S-4-49	1161.06	2.27	1	1229
RJ_t8_S-5-48	482.02	2.12	1	1474
RJ_t8_S-6-50	513.00	11.24	1	2074
RJ_t8_S-7-61	4732.70	301.02	5	3462
RJ_t8_S-8-60	944.12	11.50	1	2280
RJ_t8_S-9-56	1001.64	112.69	17	3555
RJ_t8_M-0-99	2847.99	95.11	13	3860
RJ_t8_M-1-101	1340.42	616.83	2	7956
RJ_t8_M-2-93	1401.99	26.47	1	2769
RJ_t8_M-3-98	3369.44	373.03	35	4950
RJ_t8_M-4-103	991.87	2514.97	3	6887
RJ_t8_M-5-102	4752.75	3511.36	8	7793
RJ_t8_M-6-94	866.12	4679.74	7	8808
RJ_t8_M-7-83	1437.80	38.34	1	4196
RJ_t8_M-8-79	606.95	10.67	9	2071
RJ_t8_M-9-98	1515.90	4855.79	1	9338

Continued on next page

Table 7.1: Results for 120 RJ instances

Instance	Obj. Value	Runtime(s)	Nb Nodes	Nb Vars
RJ_t8_L-0-107	1733.42	940.10	1	7314
RJ_t8_L-1-157*	8411.99	7221.95	1	4096
RJ_t8_L-2-140	4124.83	6508.09	125	10454
RJ_t8_L-3-147*	9627.78	7198.67	1	810
RJ_t8_L-4-133*	4831.55	7203.93	1	5050
RJ_t8_L-5-125	3099.14	363.20	1	5932
RJ_t8_L-6-120	3790.21	1092.60	1	8210
RJ_t8_L-7-136	3289.61	534.62	3	10115
RJ_t8_L-8-133*	6583.33	7198.43	1	2648
RJ_t8_L-9-127*	2482.76	7040.82	33	10443

7.3 Graham CPU Characteristics

The following list shows the different types of CPUs available on the Graham cluster, used to run the experiments, as of 11/2022. This information was obtained from Graham Cluster's Webpage (`https://docs.alliancecan.ca/wiki/Graham`) on 21/11/2022.

- Intel E5-2683 v4 Broadwell @ 2.1GHz
- Intel E7-4850 v4 Broadwell @ 2.1GHz
- Intel Xeon Gold 5120 Skylake @ 2.2GHz
- Intel Xeon Gold 6248 Cascade Lake @ 2.5GHz
- Intel Xeon Silver 4110 Skylake @ 2.10GHz
- Intel Xeon Gold 6238 Cascade Lake @ 2.10GHz