UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

Eduardo dos Santos Teixeira

Formulations and a heuristic approach for logistics problems with d-relaxed priority rule

São José do Rio Preto
2024

Eduardo dos Santos Teixeira

Formulations and a heuristic approach for logistics problems with d-relaxed priority rule

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Matemática, junto ao Programa de Pós-Graduação em Matemática, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: CAPES

Orientador: Prof. Dr. Silvio Alexandre de Araujo

São José do Rio Preto
2024

T266f

Teixeira, Eduardo dos Santos

Formulations and a heuristic approach for logistics problems with d-relaxed priority rule / Eduardo dos Santos Teixeira. -- São José do Rio Preto, 2024

117 p. : il., tabs.

Tese (doutorado) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientador: Silvio Alexandre de Araujo

1. Programação Inteira Mista. 2. Problema do Caixeiro Viajante. 3. Problema de Roteamento de Veículos. 4. Prioridades. 5. Heurística. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Eduardo dos Santos Teixeira

Formulations and a heuristic approach for logistics problems with d-relaxed priority rule

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Matemática, junto ao Programa de Pós-Graduação em Matemática, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: CAPES

Comissão Examinadora

Prof. Dr. Silvio Alexandre de Araujo
Orientador

Profa. Dra. Maria do Socorro Nogueira Rangel
UNESP - Câmpus de São José do Rio Preto

Prof. Dr. Pedro Augusto Munari Junior
UFSCAR - Câmpus de São Carlos

Profa. Dra. Maria José Pinto
IEAv/CTA - São José dos Campos

Prof. Dr. Alf Kimms
Universität Duisburg-Essen - Duisburgo, Alemanha

São José do Rio Preto
06 de março de 2024

À minha família,
dedico

AGRADECIMENTOS

A presente tese de doutorado é fruto de uma trajetória de 12 anos de Universidade, os quais são, por sua vez, a sequência de outros tantos anos de estudos e vivências que se deram antes. Dessa forma, correndo o risco de, injustamente, me esquecer de algumas pessoas que foram parte importante desta conquista, gostaria de registrar aqui os meus sinceros agradecimentos a todos aqueles que estiveram próximos, de qualquer forma, durante o desenvolvimento deste trabalho.

Agradeço, em primeiro lugar, a Deus, inteligência maior e causa primária de todas as coisas, pela oportunidade de chegar até aqui, pela coragem e pelo ânimo de seguir adiante com o trabalho, mesmos nos dias de cansaço, aflição ou ansiedade que insistiram em acontecer.

Agradeço, também, a toda a minha família, que sempre me apoiou em minha trajetória acadêmica e sempre me ensinou que a educação é uma chave com o poder de abrir grandes portas na vida. Através de palavras e exemplos valiosos, essa lição me motivou a buscar subir os degraus da vida acadêmica e a trabalhar pela conquista deste doutorado. Em particular, agradeço aos meus pais, Rita e Matias, ao meu irmão Augusto, e aos meus avós Alice a Alcides.

Agradeço à Ieda, minha namorada, por todo o apoio, toda a comprensão e todo o incentivo durante a realização deste trabalho, bem como de todos os outros planos e projetos que se realizaram durante o desenvolvimento desta tese.

Agradeço ao Programa de Pós Graduação em Matemática (PPGMAT) e aos professores dos Departamentos de Matemática (DMAT) e de Matemática Aplicada (DMAP) do Ibilce - em especial ao meu orientador, professor Sílvio, cujo trabalho foi fundamental para que cada peça deste estudo fosse confeccionada e colocada no seu devido lugar. Sem a sua coordenação e os seus apontamentos, com certeza esta tese perderia em organização, clareza e conteúdo.

Agradeço, também, aos demais funcionários do Ibilce, sem os quais nenhuma atividade da universidade poderia se desenvolver de forma satisfatória. Em parti-

cular, deixo o meu agradecimento ao Thiago, que me ajudou durante esses anos com o acesso aos recursos de infra-estrutura utilizados nos testes computacionais realizados.

Quero agradecer também a todos os membros da banca de defesa deste trabalho, os quais aceitaram o convite para avaliar essa tese e colaborar com a sua elaboração final.

Dentre uma lista vasta de outras pessoas - amigos, colegas e parceiros - que estiveram ao meu lado durante a realização deste curso, parcial ou totalmente, ou que me ajudaram de alguma forma a chegar até aqui, deixo aqui o meu agradecimento à Letícia, à Gi, à Ju, à Ana, à Mariele, ao Luis, ao Felipe, ao Décio e a tantos outros que me apoiaram nessa trajetória.

O tópico central desta tese foi proposto e inicialmente desenvolvido como fruto de um trabalho em parceria com as seguintes pessoas - Drielly, Natália e Marcos - às quais deixo um agradecimento em particular.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

O homem que julga infalível a sua razão
está bem perto do erro,
Allan Kardec

RESUMO

Esta tese considera extensões de dois problemas clássicos de pesquisa operacional no campo da logística em que regras de prioridade são incorporadas às versões clássicas, dando origem ao Problema do Caixeiro Viajante Clusterizado com Regra de Prioridade d-Relaxada (CTSP-d) e ao Problema de Roteamento de Veículos Clusterizado com Regra de Prioridade d-Relaxada (CluVRP-d), que corresponde a um caso particular do Problema de Roteamento de Veículos com Regras de Prioridade Relaxadas (VRP-RPR). Em ambos os problemas, a ordem em que os vértices são visitados é relevante porque é necessário considerar, além da distância percorrida, algum tipo de prioridade de visita entre os vértices, com base em diferentes situações reais. Neste estudo, formulações propostas na literatura para o CTSP-d são melhoradas utilizando-se desigualdades válidas, bem como novas formulações baseadas em variáveis de precedência são propostas. Outra contribuição desta tese é expandir a literatura sobre o VRP-RPR propondo o CluVRP-d, uma versão do problema onde todos os vértices devem ser visitados exatamente uma vez e os veículos possuem limitações de capacidade. Duas novas formulações matemáticas são propostas para o problema, considerando os casos *Local* e *Global Timing*, e uma abordagem heurística é apresentada para tratar de instâncias maiores do caso *Local Timing*. Por fim, resultados computacionais, baseados em dados da literatura, são apresentados para demonstrar a competitividade das formulações propostas quando resolvidas por um pacote de otimização e, no caso do CluVRP-d, comparamos a formulação do caso *Local Timing* com os resultados obtidos através da abordagem heurística.

Palavras-chave: Programação Inteira Mista, Problema do Caixeiro Viajante, Problema de Roteamento de Veículos, Prioridades, Heurística.

ABSTRACT

This thesis considers extensions of two classical operations research problems in the logistics field where priority rules are embedded to the classical versions, defining the Clustered Traveling Salesman Problem with d-Relaxed Priority Rule (CTSP-d) and the Clustered Vehicle Routing Problem with d-Relaxed Priority Rule (CluVRP-d), that corresponds to a particular case of the Vehicle Routing Problem with Relaxed Priority Rules (VRP-RPR). In both problems, the order in which the vertices are visited is relevant because it is necessary to consider, in addition to the distance traveled, some kind of visiting priorities among the vertices, based on different real situations. In this study, formulations for the CTSP-d are improved using valid inequalities as well as new formulations based on precedence variables are proposed. Another contribution of this thesis is to extend the literature on VRP-RPR by proposing the CluVRP-d, a version of the problem where all vertices must be visited exactly once and the vehicles have capacity limitations. Two new mathematical formulations are proposed considering local and global timing cases, and a heuristic approach is presented to tackle bigger instances for the local timing case. Finally, computational results, based on data from the literature, are presented to demonstrate the competitiveness of the proposed formulations running the models on an optimization package and, in the CluVRP-d case, we compare the Local Timing case formulation to the results obtained by the heuristic approach.

Keywords: Mixed Integer Programming, Traveling Salesman Problem, Vehicle Routing Problem, Priorities, Heuristic.

List of Figures

List of Tables

List of Abbreviations

CTSP-d	Clustered Traveling Salesman Problem with d-Relaxed Priority Rule
CluVRP-d	Clustered Vehicle Routing Problem with d-Relaxed Priority Rule
MILP	Mixed-integer linear programming
TSP	Traveling Salesman Problem
VRP-RPR	Vehicle Routing Problem with Relaxed Priority Rules
VRP	Vehicle Routing Problem

Contents

1 Introduction

In recent decades, logistics related problems have been studied in numerous papers, with different mathematical modeling and solution approaches. A central problem in this context is the Traveling Salesman Problem (TSP), also called the Traveling Salesperson Problem - one of the most important problems in combinatorial optimization and computer science (Dantzig et al., 1954). In its classical version, the TSP may be described in the following way: given a set of cities, and knowing the distance between each of these cities, we need to find the shortest possible route such that each city is visited only once, including the city of origin where the route also ends. In other words, the objective is to find the round trip with the minimum possible length. This is known to be an NP-hard problem (Garey and Johnson, 1979) and so, to solve it to optimality, finding and proving that a solution is the best one is not always possible in practice. In such cases, the objective of the problem may be to build the shortest possible tour within computational or solution-time restrictions. An important generalization of this problem is the Vehicle Routing Problem (VRP), where instead of considering only one traveling salesperson visiting a set of cities, the VRP is defined by a fleet of vehicles, and so the problem is to find the optimal routes for the fleet of vehicles, delivering goods or services to a set of customers while minimizing the total cost or distance traveled.

A common and useful way to model the TSP and the VRP is by using an oriented graph, where all the depot from where the vehicles leave and all the customers that must be visited are represented by vertices, and the possibility of traveling among two areas is represented by an arc. In the classical version of the problems we consider only one starting depot and that is possible to travel from any area (depot or customer) to any different area, what leads to a complete graph. In this approach, the problems can be seen as the challenge of choosing the best set of arcs that satisfy all constraints. In this thesis we will consider this modeling resource for the mathematical formulations proposed for all problems, and so we may use the words vertex and arcs to describe the problems from here on.

On the classical TSP and VRP definitions, the order in which the customers are visited has no restrictions. In some real situations, just this condition may not be

sufficient to represent the problem, as there are cases in which some kind of visiting priorities among the customers must be considered - in general, customers with the highest priorities need to be served before lower priority ones. According to Doan et al. (2021) this problem has several applications where different priority groups are needed, such as humanitarian relief operations (Panchamgam, 2011), unmanned aerial vehicles (Shetty et al., 2008), distribution of commercial products when the priority is determined by the customers' storage level and technician scheduling problems faced by internet service providers (Hà et al., 2020). A deeper discussion on variants of the TSP and VRP involving priorities is developed in Chapter 2.

The focus of the present thesis is to study a prioritization rule called "d-relaxed priority rule" in the contexts of the TSP and the VRP separately. To apply the rule, first we need to assign priorities to each vertex of the problem in descending order, that is, vertices that receive priority 0 have the highest priority level, followed by vertices with priority 1, priority 2, and so on. The d-relaxed priority rule is used to manage the operational flexibility when visiting the vertex by allowing the vehicles to visit a whole set of similar priority level sets in an interchangeable manner but forbidding visits to lower priority sets until the higher priority ones have been totally served. By setting $d = 0$, all the priorities must be strictly verified, which usually leads to costly solutions, while higher values of d allows the solution to ignore partially (or even totally) the priorities, leading to smaller/cheaper routes. We will better describe the details of the rule in Chapter 3 for the TSP and Chapter 4 for the VRP.

Two important and still not deeply explored logistics problems with priorities were described in Panchamgam (2011), called Hierarchical Traveling Salesman Problem with d-relaxed priority rule (HTSP-d) and the Vehicle Routing Problem with Relaxed Priority Rules (VRP-RPR). Doan et al. (2021) study VRP-RPR in the context of two important prioritization rules, namely, the d-relaxed priority and the Order of Demand Fulfilment (ODF), where the second rule is used to ensure that at the end of the planning horizon, all the higher priority vertices need to be visited before the vehicles can visit any lower priority one. The d-relaxed priority rule was also studied as a prioritization rule for the TSP (Panchamgam, 2011; Hà et al., 2020; Teixeira and de Araujo, 2024). In this context, since all the vertices need to be visited, we do not need to consider the ODF rule, and so the problem reduces to a TSP with d-relaxed priority rule. This problem is called HTSP-d or the Clustered Traveling Salesman problem with d-relaxed priority rule (CTSP-d). In this thesis we propose a generalization of the CTSP-d as a problem with a set of vehicles, defining the Clustered Vehicle Routing Problem with d-relaxed Priority Rule (CluVRP-d), which is a particular case of the VRP-RPR.

The research in this thesis is inspired by the agricultural pest control presented in Teixeira et al., 2021, where the authors describe an ant control problem in forestry that must be handle considering priorities issues. The problem can be described in the following way: given a set of forest areas, a working group must visit regularly each one of these areas to develop a pest control service, combating an active infestation. Depending on the severity of the infestation, it may impair the forest production in different levels, and so the working group must visit all areas considering the individual urgency levels, which are grouped into priority visiting sets based on technical analysts criteria. This problem is identified with the CTSP-d problem, and the multi working group version of the problem can be seen as the CluVRP-d.

The remainder of the text is organized in the following way: in Chapter 2 we present a literature review on the topic logistics problems with priority constraints, discussing different problems proposed involving the TSP and the VRP in this context. Chapter 3 presents formulations for the CTSP-d based in the classical Miller et al. (1960) formulation and in precedence variables, showing different MILP models for the problem and related results. In Chapter 4 are introduced formulations for the CluVRP-d considering the two ways the d-relaxed priority rule can be applied in the VRP - namely, the Local Timing Model and the Global Timing Model. Valid inequalities are presented for the models, and a heuristic approach for the Local Timing case is proposed. Chapters 5 and 6 present, respectively, the computational results obtained running the formulations proposed for the CTSP-d in a commercial MILP solver, and the computational results obtained for the CluVRP-d running both the formulations using a MILP solver and the heuristic proposed for the problem. Finally, in Chapter 7 are the conclusions and future research proposals. It is worth mentioning that a version of the contents of Chapters 3 and 5 was published on the paper Teixeira and de Araujo (2024).

2 Literature review

This literature review presents a brief overview of the problems and papers related to our research. The review is divided into applications of prioritization rules or related topics to the TSP and the VRP, separately. This is not supposed to be an exhaustive review about TSP/VRP variants with priority criteria, but a list of the recent works related at some level to them.

2.1 Clustered Traveling Salesman Problem

An important problem that has been widely studied in the literature is the so-called Clustered Traveling Salesman Problem (CTSP) introduced by Chisman (1975). In this problem, the traveling salesman not only must visit each vertex exactly once, but the vertices are partitioned into disjoint subsets, called clusters, where all vertices included in a cluster must be visited contiguously. In the CTSP, there is no order relationship between the vertices in each cluster, as well as no order in which the clusters must be visited.

In Laporte et al. (1997) and Potvin and Guertin (1998) the authors consider the CTSP with a prespecified order on the clusters and introduced the Clustered Traveling Salesman Problem with a Prespecified Order (CTSP-PO). In this problem, there are the sets of vertices with associated priorities defined by V_0, V_1, ..., $V_{P_{max}}$, but no d parameter. Since this problem has a prespecified order on the clusters visiting that must be strictly followed, any feasible solution for the CTSP-PO must start at the origin vertex 0 followed by some vertex in V_0, which is the highest priority set. Vertices in V_0 must be visited contiguously, and once the last one is left, the next vertex must be in V_1, which is the next highest priority set still not visited. This process repeats until all vertices in $V_{P_{max}}$ have been visited. We see a hard priority fulfillment on this behavior, that may be relaxed in some practical situations. A natural application of the CTSP-PO occurs when vertices requiring the same level of urgency can be clustered together, and higher priority clusters should be visited before others. Several authors have developed solution approaches to the CTSP (see, for example, Mestria (2018)). The CTSP-d is defined in a similar but more general way than the CTSP-PO, since

there is flexibility in the prespecified order the clusters must be served, which is defined by the d parameter of the rule. The CTSP-PO correspond to the CTSP-d when we take $d = 0$ and there is no flexibility allowed on the clusters visiting order.

Another related problem is the Traveling Salesman Problem with Sequence Priorities (TSPwSP), introduced by Schmitz and Niemann (2009). The TSPwSP is defined by assigning priorities to each vertex and measuring the violations of these priorities using a penalty function. The defined priorities stand for the order in which each vertex asked for the salesman visit, which in practice may represent a certain service that must be provided to a set of clients, adding a "first-come-first-served" format to the problem. Since the objectives of minimizing the total cost of the tour and minimizing the violations on the service order line are conflicting in general, the authors proposed a bicriterion approach to manage the trade-off between them. When comparing the TSPwSP and the CTSP-d, there are clearly some important differences. First, the TSPwSP defines a bi-objective model, in which the priorities are assigned to the vertex, to define the "first-come-first-served" policy intended by the authors while, in the CTSP-d, the only objective is minimizing the total distance of the tour while the priorities are considered as constraints over the whole clusters of vertices. Moreover, the TSPwSP can ignore part of the priorities, since the model only deals with allowing the vertices to be more or less far from the "ideal positions" they should be.

Similarly, in the Travelling Salesman Problem with Priority Prizes (TSPPP), described in Pureza et al. (2018), instead of adding a penalty objective function to the TSP, the authors propose a model based on the Quadratic Assignment Problem, in which they include a prize according to the priorities in the model, which may be collected by the salesman by meeting these order visiting prizes. In the Multicommodity Traveling Salesman Problem (MTSP), described by da Silva et al. (2019), when connecting two vertices (customers), product types and quantities being transported are considered in the objective function and priorities are defined based on the size of demands or on the value/risk of the products that will be delivered. Once again, when compared to the CTSP-d, both problems (TSPPP and MTSP) are not imposing priorities that must be respected in some sense, but just adding conditions on preferable positions some vertices should assume in the tour, and so the solutions obtained via TSPPP or MTSP may not be the ideal for situations where some vertices need to be visited before others. More specifically, when dealing with humanitarian logistics, in planning relief operations after a natural disaster, the objective is clearly not to obtain financial benefits but to save people's lives.

There are other important TSP variants, such as the Generalized Traveling Salesman Problem (GTSP), discussed in Srivastava et al. (1969) and modeled in Laporte

and Nobert (1983), Laporte et al. (1987) and Pop (2007), in which the vertices are partitioned into clusters, as in the CTSP but, in the GTSP, the objective is to build a tour in which the salesman visits only one vertex in each cluster. There are exact and heuristic methods to solve this problem (see, for example, Pintea et al. (2017) and Cosma et al. (2021)), and there is a clustered version of this problem, namely the Clustered Generalized Traveling Salesman Problem (CGTSP - see Baniasadi et al. (2020)). An extension of the GTSP is the Family Traveling Salesman Problem (FTSP), presented in Bernardino and Paias (2018), where the vertices are divided into clusters, called families in the context of the problem, and the objective is to determine a minimum cost route in which the traveling salesman visits a given number of cities in each family. Another relevant way the vertices in the TSP can be ordered is the Traveling Salesman Problem with Time Windows (TSPTW) (Dumas et al., 1995), that does not consider priorities among the vertices explicitly but the time windows in which each customer must be visited can represent the priority among the vertices. In practice, the TSPTW imposes ranges of positions each vertex may figure in a solution, but this is not sufficient to define the d-relaxed priority rule. It is also worth mentioning the TSP variant so-called Crew Scheduling and Routing Problem (CSRP) (Maya Duque et al., 2016; Moreno et al., 2019, 2020) that consists of finding a single route (TSP) to visit nodes that need to be repaired after a disaster and the order of visits is important and implied by the need of relief paths reaching the damaged areas as soon as possible.

The Precedence Constrained Traveling Salesman Problem (PCTSP), also called the Sequential Ordering Problem (Escudero, 1988), is another relevant TSP variant, where there are precedence relationships between part of the vertices of the graph of the problem. It introduces a new set of constraints in the problem, imposing that the traveling salesman visit some defined vertices before (but not necessarily immediately before) other ones. The PCTSP can be extended into the Precedence Constrained Generalized Traveling Salesman Problem (PCGTSP), which brings together both the conditions of GTSP and PCTSP, defining a problem where the vertices are divided into clusters, and the traveling salesman must visit each cluster in a single vertex, respecting the precedence conditions imposed (Salman et al., 2020; Khachai et al., 2023). Some recent optimal solutions and best bounds for the PCTSP can be find in Cire and van Hoeve (2013) and Gouveia and Ruthmair (2015), and some recent advances in formulations for the problem are presented in Letchford and Salazar González (2016) and Gouveia et al. (2018). In Chapter 3 we discuss how the CTSP-d can be seen as a particular case of the PCTSP and how to adapt some classical formulations for it. This can be an important step to allow the use of other models and efficient solution methods for the PCTSP as modeling/solving resources for the CTSP-d.

In this study, new formulations for the CTSP-d are presented that, as discussed above, are different from TSP with prize collection models or models imposing fixed or desirable positions for each vertex. They can be seen as a natural generalization of the CTSP-PO for problems which consider trade-offs between tour costs and the hardness of the priority assigned to the clusters. Although the consideration of priorities is a relevant topic in the routing literature, to the best of our knowledge, the d-relaxed priority rule has not been totally explored. As mentioned before, this rule was proposed by Panchamgam (2011) in the context of retail operations and humanitarian logistics, and Hà et al. (2020) proposed a reformulation of the original model, leading to a more precise representation of the priority rule and better computational results. Teixeira and de Araujo (2024) presented new formulations for the problem based in the work of Hà et al. (2020), using the classical Miller et al. (1960) approach to model the problem, as well as precedence variables models. Doan et al. (2023) extended the work of Hà et al. (2020) by proposing alternative formulations and an iterated local search.

2.2 Vehicle Routing Problem

Considering the VRP, the vast literature has many different ways of treating the existence of priorities on the solutions, or any other kind of vertex ordering requirements that use some criteria to distinguish transportation service between customers such as: priority assignment, weighting factor or penalty objective. On the other hand, the VRP with d-relaxed priority rule was first described in Panchamgam (2011) in the context of disaster relief operations as a way to balance the trade-off between the priorities fulfillment and the costs involved in building the routes in the VRP. To the best of our knowledge, the only paper dealing with the VRP-RPR is Doan et al. (2021), where the authors solved it heuristically. The goal of this literature review is, in this sense, to show some of the strategies used to treat priorities in the VRP literature, comparing it with the d-relaxed priority rule, and to analyze how these approaches differ from each other. To do so, we first present other variants of the VRP with priorities in Subsection 2.2.1; next, we consider VPR with clustering approaches in Subsection 2.2.2 and, finally, some application that consider priorities in Subsection 2.2.3.

2.2.1 VRPs with priorities

Several extensions of the VRP are related to priorities and have been studied in the literature, such as: the VRP with service levels (VRP-SL) (Bulhões et al., 2018), the Inventory Routing Problem (IRP) (Archetti and Ljubić, 2022) and the VRP with Profits (VRPPs) (Archetti et al., 2014). The VRP-SL takes the viewpoint of a third

party logistics provider and because of limited fleet size and vehicle capacity, it is necessary to select a subset of deliveries and determine cost-efficient vehicle routes in such a way that the overall activity is profitable and that the agreements with the partners are respected. In the VRPPs, the set of customers to serve is not given and needs to be decided according to its profit, as well as, how to cluster the customers to be served in different routes. Thus, the routes can be measured both in terms of cost and in terms of profit. According to Doan et al. (2021), when demand is replaced by profit, the VRP-RPR can be reduced to some important problems, such as the Team Orienteering Problem (TOP) (Chao et al., 1996; Boussier et al., 2007), that is a special case of the VRP, where there are a set of potential customers to be visited and a fleet of capacitated vehicles to be assigned to them. Due to some practical limitations, it is considered that not all the customers can be visited during the tours built, and so the objective is to construct vehicle routes that maximize the reward obtained by visiting each vertex.

Other relevant related extensions are: the Dynamic VRP With Time Windows and Multiple Priorities (Yang et al., 2015) and the Multi-objective Vehicle Routing Problem with Multiple Prioritized Time Windows (Beheshti et al., 2015). In the former, there is a fleet of vehicles that is used to serve customers with time windows and different priority levels. Customers with higher priority should have a higher quality of service. Additionally, part of customers are revealed dynamically during the execution of the routes. The latter article, in addition to the minimization of the total traveling cost, aims to maximize customer satisfaction regarding prioritized time windows.

2.2.2 VRP with clustering

Some VRP variants consider that the customers are also divided into clusters and are served according to the cluster to which they belong. These problems are related to those with priorities because in several applications the cluster can be defined according to the priority of each customer and, in fact, we will use this strategy in Chapters 3 and 4. The Generalized Vehicle Routing Problem (GVRP) (Ghiani and Improta, 2000; Hà et al., 2014), similarly to the GTSP, considers that each cluster contains a number of vertices, and a tour includes exactly one vertex from each cluster and satisfies capacity constraints. The Selective Vehicle Routing Problem (SVRP) (Sabo et al., 2020) is a generalization of the GVRP where customers may belong to one or more clusters. The Clustered Vehicle Routing Problem (CluVRP), studied in Battarra et al. (2014) and Pop et al. (2018), is the multi-vehicle case of the CTSP, where once a vehicle visits one customer in a cluster it must visit all the remaining customers of that cluster before leaving it. In Expósito-Izquierdo et al. (2016) the authors consider the capacitated

version on the CluVRP. Yahiaoui et al. (2019) present the Clustered Team Orienteering Problem (CluTOP) where a profit is associated with each cluster and is gained only if all of its customers are served considering a set of available vehicles with a limited travel time.

It is also important to highlight some papers that consider clustering in a more general way, meaning assigning similar objects to groups. These problems have been considered in a broad range of applications and, in several cases, the size of the clusters is limited; consequently these capacitated clustering problems have received increasing attention (Stefanello et al., 2015; Martínez-Gavara et al., 2017; Chaves et al., 2018; Zhou et al., 2019; Gnägi and Baumann, 2021).

2.2.3 Applications

One of the main sources of application of VRP with priorities rules are humanitarian logistics (Campbell et al., 2008) including emergency response (Bretschneider and Kimms, 2011; Jacobson et al., 2012). For a complete analysis about the differences between commercial and humanitarian logistics the reader can refer to Beamon and Balcik (2008), Gralla and Goentzel (2018) and Wassenhove (2006). The literature on logistics for disaster relief operations deals with the existence of priorities in different ways, and this is a very relevant topic, since in this kind of application, bad planning may intensify the suffering of the affected people, exposing them to injuries, diseases or even the risk of death. Barbarosoglu et al. (2002) present two models to deal with tactical and operational planning in the context of routing a fleet of helicopters to work on disaster relief operations. Both the proposed models are mixed integer and the authors describe a hierarchical multi-criteria approach to relate the two stages of the planning. The objective is to minimize the total tour duration for the helicopters, and no explicit priority constraints are defined in any of the models. A problem similar to the VRP is described by Bish (2011), who models the so-called "Bus Evacuation Problem" (BEP), in which the vertices are divided in three subsets: yard vertices, where the buses are initially located and do not return during their tours, demand vertices, where the buses need to visit to do the evacuation work, and shelter vertices, the places where people are taken after being rescued. The problem is similar to a VRP with multi-depot and split delivery, and the authors consider two distinct objectives: to minimize the maximum tour cost over the buses, and to minimize the total cost of the routing problem. The solution procedure proposed is a two-phase heuristic to build and improve a solution using ideas similar to the ones commonly used in the context of the VRP. Briskorn et al. (2020) consider an integrated problem that consists in unblocking roads in order to make demand locations accessible and delivering relief

goods in order to satisfy demand considering strict deadlines. Other applications in the disaster relief context including emergency response can be found in Altay and Green (2006), Sheu (2007), Sheu (2010), Chiu and Zheng (2007), Özdamar and Demir (2012), Simpson and Hancock (2009) and Kimms and Maiwald (2018).

3 Formulations for the CTSP-d

In this chapter we will present formulations for the CTSP-d based in the classical Miller et al. (1960) formulation and in precedence variables. Before presenting the formulations we will formally define the d-relaxed priority rule considering the following sets and parameter.

• *Sets:*

$V = \{0, 1, 2, ..., n-1\}$: Set of the n vertices of the problem, where the vertex 0 represents the start city of the tour, and the other vertices represent the cities that must be visited;

$A \subseteq \{(i, j) : i, j \in V, i \neq j\}$: Set of the arcs of the problem, representing the possibility of the traveling salesman to travel from vertex i to vertex j. For the sake of simplicity, in this thesis, we assume a complete graph, meaning that the set A contains arcs linking all the different vertices in the problem.

$P = \{0, 1, 2, ..., P_{max}\}$: Set of the different priority levels of the vertices in the problem;

V_p: Set of vertices from $V\backslash\{0\}$ with priority level $p \in P$. The sets V_p define a partition in $V\backslash\{0\}$, i.e., the sets are disjoints and the union of them all is equal to $V\backslash\{0\}$, and are ordered in descending order with respect to the priority level they represent, that is, set V_0 include the vertices with the highest priority level, followed by V_1 up to $V_{P_{max}}$.

• *Parameter:*

d: Parameter used to define the hierarchy relaxation level over the priority sets.

For the d-relaxed priority rule, the first vertex after the origin may be in any set among V_0, ..., V_d, and the vertices on these sets may be visited in an interchangeable way, disregarding the different priority levels. Vertices in sets V_{d+1}, ..., $V_{P_{max}}$ stay forbidden until all vertices in V_0 are visited, and once that happens, vertices in V_{d+1} may start to be visited. This pattern repeats for all the remaining sets, that is, vertices in V_q may only be visited after all vertices in V_p were visited, for all $p, q \in P$, $q > p + d$. The rule is illustrated in the example below.

Suppose a problem has eight priority sets, namely V_0, V_1, ..., V_7, and d is set to 1. The only sets that may be visited right after the origin are V_0 and V_1, and the d-relaxed

priority rule imposes that V_2 may only be visited after all the vertices in V_0 have been visited. It is worth emphasizing the cumulative aspect of the rule - vertices in set V_3 may only be visited after all vertices in V_0 and V_1 have been totally visited, vertices in set V_4 can only be accessed after all vertices in V_0, V_1 and V_2 have been totally visited, and so on.

There are several formulations for the TSP based on the combination of an assignment problem with binary variables and a set of subtour elimination constraints (Öncan et al., 2009; Bektaş and Gouveia, 2014), which will be the basis for the formulations presented in this chapter. Since all the formulations are based on a common set of variable, constraints and parameters, all these common elements are defined first before presenting the formulations.

- *Parameter:*

  c_{ij}: Cost associated with the travel from vertex i to vertex j.

- *Variable:*

  $x_{ij} = 1$ when arc $(i, j) \in A$ is included in the tour, or 0 otherwise.

Considering that we will present different formulations for the CTSP with d-relaxed priority rule, a general formulation for the TSP is given considering a generic subtour elimination constraint, as well as a generic d-relaxed priority rule constraint:

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{3.1}$$

$$\text{s. t.} \quad \sum_{i\in V:(i,j)\in A} x_{ij} = 1, \quad \forall j \in V \tag{3.2}$$

$$\sum_{j\in V:(i,j)\in A} x_{ij} = 1, \quad \forall i \in V \tag{3.3}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A \tag{3.4}$$

$$\text{solution does not contain subtours} \tag{3.5}$$

$$\text{solution satisfies the } d\text{-relaxed priority rule} \tag{3.6}$$

In this formulation, the objective function (3.1) represents the goal of minimizing traveling costs in the problem, given by the sum of the costs of the arcs selected for the tour. Constraints (3.2) and (3.3) define the flow in the solution, imposing that every vertex must be visited and left exactly once and constraints (3.4) define the domain of the variables x_{ij}. The other two sets of constraints, namely (3.5) and (3.6), are generic constraints representing the final aspects that a formulation for the CTSP-d must satisfy. Constraints (3.5) represent the subtour elimination constraint set, and so expressions (3.2)-(3.5) define a general model for the TSP, while constraints (3.6) are the d-relaxed priority rule constraints, which turns the classic TSP into the CTSP-d.

3.1 Miller-Tucker-Zemlin based formulations

A simple way to obtain a compact formulation for the CTSP-d is adopting constraints based on the classic Miller-Tucker-Zemlin (MTZ) subtour elimination constraints (Miller et al., 1960). An important feature of this approach is it having a small number of variables and constraints, being an $\mathcal{O}(n^2)$ formulation for variables and constraints. It is the basis for the formulations proposed in Panchamgam (2011) and Hà et al. (2020) for the CTSP-d, and also in Doan et al. (2021) for the VRP with d-relaxed priority rule. In its original form, the MTZ constraints are as in (3.7)

$$u_i - u_j + nx_{ij} \leqslant n - 1, \quad \forall (i,j) \in A, j \neq 0, \tag{3.7}$$

where $u_j \in \mathbb{R}^+$, $j \in V$ is a set of continuous variables that represents the order of each vertex j in the tour. Despite the fact that u_j are defined as real variables, constraints (3.7) ensure that their values will be integer values for any feasible solution.

Since the variables u_j represent the order of the vertices in the solutions, they can be used to define a d-relaxed priority rule constraint by doing

$$u_i + 1 \leqslant u_j, \quad \forall i \in V_p, j \in V_q : p, q \in P, q > p + d. \tag{3.8}$$

- Hence, the first formulation of the CTSP-d, denoted by MTZ1, is defined by (3.1)-(3.4) together with constraints (3.7), that guarantees subtour elimination, and the d-relaxed priority rule constraints (3.8).

A way to strengthen this formulation is to add valid inequalities. In Hà et al. (2020), the authors present four sets of constraints, some of them proposed by Panchamgam (2011), that can be used for this purpose, which are given in (3.9)-(3.12).

$$\sum_{i \in V_p} \sum_{\substack{j \in V_q: \\ (i,j) \in A}} x_{ji} = 0, \quad \forall p, q \in P, q > p + d \tag{3.9}$$

$$\sum_{j \in V_p} x_{0j} = 0, \quad \forall p \in P, p > d \tag{3.10}$$

$$\sum_{i \in V_p} x_{i0} = 0, \quad \forall p \in P, p < P_{max} - d - 1 \tag{3.11}$$

$$\sum_{i \in V_p} \sum_{\substack{j \in V_q: \\ (i,j) \in A}} x_{ij} \leqslant 1, \quad \forall p, q \in P, q > p + d \tag{3.12}$$

Since the vertices in a set V_q only can be visited after all the vertices in V_p have

already been visited for all $q > p + d$, there will be no trips from vertices in V_q back to vertices in V_p, which is expressed in constraints (3.9). Moreover, a similar relationship can be observed for the origin vertex, which is not included in any priority set. This is ensured by constraints (3.10) and (3.11). The constraints in (3.12) do complement this relationship in the following way: when a vertex in V_p is left, the next vertex visited can belong to a V_q set such that $q > p + d$, but this can only occur once at most. It is worth emphasizing the constraints (3.9)-(3.11) are used to fix the values of sets of variables to zero, and this can be done in different ways. In this thesis we will consider the constraints as described in the text to maintain the same approach described in Panchamgam (2011) and Hà et al. (2020).

Formulation MTZ1 was described in Hà et al. (2020) as well as another weaker formulation for the CTSP-d based on the Miller et al. (1960) constraint sets and the authors showed that adding (3.9)-(3.12) to MTZ-based formulations is an effective way to improve the formulation of the CTSP-d.

- The formulation given by (3.1)-(3.4), (3.7)-(3.8) and (3.9)-(3.12) will be denoted by H2020.

3.1.1 Proposed valid inequalities for the MTZ1 formulation

In this section, new constraint sets are proposed that may be included or reformulated to strengthen the MTZ1 formulation.

Variable bounds and strengthening constraints (3.7)

The first improvement proposed is to change the original MTZ formulation to its lifted version, proposed by Desrochers and Laporte (1991), where the constraints set (3.7) is reformulated as (3.13).

$$u_i - u_j + nx_{ij} + (n-2)x_{ji} \leqslant n - 1, \quad \forall (i,j) \in A : j \neq 0. \tag{3.13}$$

In Proposition 1, the constant bounds used in inequality (3.13) are analyzed and shown that they can be strengthened using the variable bounds the CTSP-d induce over u_j. The first step to prove this result is to deduce these new bounds, which are themselves valid inequalities to the problem too. The d-relaxed priority rule imposes that for a fixed value p, all the vertices in priority set V_p must be visited before the vertices in V_q start to be visited, for all $q > p + d$. In this sense, the priority rule defines upper and lower bounds for the position that each vertex may appear in a sequence, depending on the value of parameter d: in the worst case, a vertex in V_p will figure in a solution

as soon as it is allowed by the rule and the last vertex in V_p will be right before the first vertex in V_q, $q > p + d$. This condition can be represented using the cardinality of the priority sets, which defines the two following constraints sets:

$$u_j \geqslant \sum_{r=0}^{p-d-1} |V_r| + 1, \quad \forall j \in V_p, d + 1 \leqslant p \leqslant P_{max} \tag{3.14}$$

$$u_j \leqslant \sum_{r=0}^{p+d} |V_r|, \quad \forall j \in V_p, 0 \leqslant p \leqslant P_{max} - d - 1 \tag{3.15}$$

Constraints (3.14) and (3.15) were not defined for all possible values of p since they would reduce to the original bounds of the variables, that is, they would turn into $u_i \geqslant 1$ or $u_i \leqslant n - 1$ in the non-included cases. But they can still be used to combine constraints (3.13), (3.14) and (3.15) to deduce tighter bounds to (3.13), as described in Proposition 1.

Proposition 1. *Constraints*

$$u_i - u_j + (M_{ij} + 1)x_{ij} + (M_{ij} - 1)x_{ji} \leqslant M_{ij}, \quad \forall (i, j) \in A, i, j \neq 0 \tag{3.16}$$

are valid for the CTSP-d, where

$$M_{ji} = M_{ij} = \sum_{r=\max\{0,p-d\}}^{\min\{P_{max},q+d\}} |V_r| - 1, \quad \forall i \in V_p, j \in V_q, p = 0, ..., P_{max}, q = p, ..., P_{max}, \tag{3.17}$$

and these are the best bounds for this constraints set.

Proof. Inequalities (3.16) are a generalization of the lifted MTZ inequalities described in (3.13) and, for any feasible solution, the right-hand side of (3.16) is an upper bound for the difference $u_i - u_j$. In this sense, we need to prove the values in (3.17) represent upper bounds for this difference.

First, notice that if M_{ij} is an upper bound to $u_i - u_j > 0$, so it is an upper bound to $u_j - u_i$ too since the last difference will be a negative value. It implies that $M_{ij} = M_{ji}$, $\forall (i, j) \in A$, $i, j \neq 0$. Also, the bounds described in (3.14) and (3.15) can be extended to all the V_p sets in the following way: assume that, when $p - d - 1 < 0$, the summation does not represent anything feasible so it can be ignored and expression (3.14) may be written as:

$$u_j \geqslant \sum_{r=0}^{p-d-1} |V_r| + 1, \quad \forall j \in V_p, 0 \leqslant p \leqslant P_{max}.$$

Moreover, all the u_j variables must satisfy the initial TSP upper bound given by $u_j \leqslant n - 1$. In inequality (3.15), this is not included, since this is already a constraint

in the MTZ formulations. A simple way to denote the general case is:

$$u_j \leqslant \sum_{r=0}^{\min\{p+d,P_{max}\}} |V_r|, \quad \forall j \in V_p, 0 \leqslant p \leqslant P_{max}.$$

Now, let $i \in V_p$ and $j \in V_q$ such that $q \geqslant p$. The deduced bounds imply

$$u_j - u_i \leqslant \sum_{r=0}^{\min\{P_{max},q+d\}} |V_r| - \left(\sum_{r=0}^{p-d-1} |V_r| + 1 \right) = \sum_{r=\max\{0,p-d\}}^{\min\{P_{max},q+d\}} |V_r| - 1,$$

$$\forall i \in V_p, j \in V_q, p = 0, ..., P_{max}, q = p, ..., P_{max}, \tag{3.18}$$

where the right hand side of (3.18) can be obtained by noticing that when $p-d-1 < 0$, the bound for u_i is just 1, and so:

$$u_j - u_i \leqslant \sum_{r=0}^{\min\{P_{max},q+d\}} |V_r| - 1, \forall i \in V_p, j \in V_q, p = 0, ..., P_{max}, q = p, ..., P_{max}. \quad \text{(I)}$$

On the other hand, when $p - d - 1 \geqslant 0$, the terms V_r common to both the bounds of u_j and u_i will become 0 in the difference, and the result will be:

$$u_j - u_i \leqslant \sum_{r=p-d}^{\min\{P_{max},q+d\}} |V_r| - 1, \forall i \in V_p, j \in V_q, p = 0, ..., P_{max}, q = p, ..., P_{max}. \quad \text{(II)}$$

Combining the conclusions in (I) and (II), the bound is given by (3.18). It implies that the lowest value M_{ji} may assume is the right hand side of the inequality. Extending this to M_{ij}:

$$M_{ji} = M_{ij} = \sum_{r=\max\{0,p-d\}}^{\min\{P_{max},q+d\}} |V_r| - 1, \quad \forall i \in V_p, j \in V_q, p = 0, ..., P_{max}, q = p, ..., P_{max}. \tag{3.19}$$

Noticing this relationship is valid for any $i, j \in V$, $i \neq j$, $i, j > 0$, it gives a new stronger bound to the lifted MTZ inequalities for the CTSP-d.

To see that this is the best possible bound for this constraints set, suppose that a smaller bound $B_{ji} \in \mathbb{R}$ exists satisfying the condition $B_{ji} < M_{ji}$ for some arc $(j, i) \in A$, and so, without loss of generality, $u_j \in V_q$ and $u_i \in V_p$ such that $u_j > u_i$. The definition of the problem allows a solution to be built such that:

$$u_j = \sum_{r=0}^{q+d} |V_r| \quad \text{and} \quad u_i = \sum_{r=0}^{p-d-1} |V_r| + 1.$$

In this case, the solution obtained satisfies $u_j - u_i = M_{ji}$. But we are supposing the difference $u_j - u_i \leqslant B_{ij} < M_{ji}$, which is a contradiction. It proves that the smallest values the parameters M_{ij} may assume in inequality (3.16) are the ones given in (3.19), which concludes the proof. ■

Beyond the constraints sets proposed in (3.14) and (3.15), there are other known inequalities that may be used to deal with u_j variables bounds. In the paper by Desrochers and Laporte (1991) there are three other constraints sets presented to lift the lower and upper bounds of the u_j variables, namely:

$$u_i \geqslant 1 + \sum_{(j,i)\in A: j\neq 0} x_{ji}, \quad \forall i \in V, i \neq 0 \tag{3.20}$$

$$u_i \leqslant n - (n-2)x_{0i} - \sum_{(i,j)\in A: j\neq 0} x_{ij}, \quad \forall i \in V : (0,i) \in A \tag{3.21}$$

$$u_i \geqslant (n-2)x_{i0} + \sum_{(j,i)\in A: j\neq 0} x_{ji}, \quad \forall i \in V : (i,0) \in A \tag{3.22}$$

The constraints in (3.20) stand for the fact that $u_i \geqslant 1$, $\forall i \in V$, $i \neq 0$, which is valid for both TSP and CTSP-d. In this case, however, constraints (3.14) grant greater bounds for u_j such that $j \in V_p$, $p \geqslant d+1$, and so constraints (3.20) can be restricted to the first d sets, which leads to the constraint set (3.23).

$$u_i \geqslant 1 + \sum_{(j,i)\in A: j\neq 0} x_{ji}, \quad \forall i \in V_p, 0 \leqslant p \leqslant d. \tag{3.23}$$

Constraints (3.21) and (3.22) refer to the vertices that come right before or after the origin in the tour. Considering the constraint sets (3.9)-(3.11) are part of the new model, these constraints can be added to the CTSP-d formulation in its original formulation, since values such that $x_{i0} = 0$ or $x_{0i} = 0$ are already fixed to be zero. Finally, the value of u_0 can be fixed in the following way:

$$u_0 = 0. \tag{3.24}$$

Strengthening constraints (3.8)

In a similar way, the x_{ij} can be added to constraints set (3.8). In fact, the condition of the d-relaxed priority rule imposes that vertices in V_p sets with lower values of p must be visited before the vertices in a V_q set, where $q > p + d$ and, in the worst case, $u_i - u_j = -1$, for some $i \in V_p$, $j \in V_q$ where $q > p + d$, since variables u_k are integer for all feasible solutions. The fact that allows x_{ij} to be added to the constraint set is that if $x_{ij} = 0$, then $u_j - u_i \geqslant 2$, that is, vertex j will be at least 2 positions after vertex i

in the solution. Noticing that x_{ji} will never happen in (3.8), since vertices in V_q must succeed vertices in V_p, the new constraint set is

$$u_i + 2 - x_{ij} \leqslant u_j, \quad \forall i \in V_p, j \in V_q : p, q \in P, q > p + d. \tag{3.25}$$

A more formal way to obtain this conclusion is by applying lifting to constraints (3.8), which are presented in Proposition 2.

Proposition 2. *Constraints* (3.25) *are valid for the CTSP-d.*

Proof. Constraints (3.25) are a lifted version of the original inequalities described in (3.8). The lifting process is applied to constraints (3.8) by adding new real parameters α, β and γ to the original constraints set, which leads to the following:

$$u_i + 1 + \alpha + \beta x_{ij} + \gamma x_{ji} \leqslant u_j, \quad \forall i \in V_p, j \in V_q : p, q \in P, q > p + d.$$

The case where $\alpha = \beta = \gamma = 0$ is the original set (3.8) of inequalities. We need to find the largest possible values for the parameters such that the new constraints set remains valid for the problem. As already discussed, we do not need to consider the γ parameter, since x_{ji} will always be zero for $i \in V_p$, $j \in V_q$ when $q > p + d$, and so the final problem is to find the largest values for α and β in expression (3.26)

$$u_i + 1 + \alpha + \beta x_{ij} \leqslant u_j, \quad \forall i \in V_p, j \in V_q : p, q \in P, q > p + d, \tag{3.26}$$

In the case $x_{ij} = 0$:

$$u_i + 1 + \alpha \leqslant u_j, \quad \forall i \in V_p, j \in V_q : p, q \in P, q > p + d.$$

But $u_j > u_i$ by hypothesis, and moreover $u_j - u_i > 1$, since $x_{ij} \neq 1$, which implies that the largest value α can assume is $\alpha = 1$. Considering the case $x_{ij} = 1$:

$$u_i + 1 + \alpha + \beta \leqslant u_j \Rightarrow 1 + \alpha + \beta \leqslant 1 \Rightarrow \beta \leqslant -\alpha, \quad \forall i \in V_p, j \in V_q : p, q \in P, q > p + d.$$

Applying the value $\alpha = 1$, the greatest value β can assume is $\beta = -1$. Replacing the values $\alpha = 1$ and $\beta = -1$ in (3.26) gives the constraints set (3.25). ■

Other inequalities

Finally, two other valid inequalities sets are proposed to be included in the new MTZ-based formulation that are not directly related to the d-relaxed priority rule, but to general TSP properties. The first one follows from the fact the variables u_j, $j \neq 0$,

will always map onto a permutation of the values 1, 2, ..., $n-1$, and so constraint (3.27) must always be satisfied

$$\sum_{j=0}^{n-1} u_j = \sum_{j=0}^{n-1} j = \frac{n(n-1)}{2}. \tag{3.27}$$

The other relationship is that each arc on the graph can only be traversed in one direction in any feasible solution, and so the following two-vertices clique constraint set can be added:

$$x_{ij} + x_{ji} \leqslant 1, \quad \forall (i,j) \in A. \tag{3.28}$$

- The formulation MTZ2 assembles constraint sets (3.1)-(3.4), (3.9)-(3.12), (3.14)-(3.15), (3.16) applying the bounds presented in (3.17) and expressions (3.21)-(3.25) and (3.27)-(3.28). It is worth noticing that, like the MTZ1, this is a model with $\mathcal{O}(n^2)$ variables and constraint sets.

3.2 Precedence variable-based formulations

Besides the models proposed for the CTSP-d problem based on the classic MTZ formulation, there are other ways to formulate the problem and, in this section, we propose three new formulations based on *precedence variables*, which are more general than the ordering variables used in the MTZ formulations and can be used to build stronger models. In the whole section, the models will be formulated using y_{ij} variables as described below:

$y_{ij} = 1$ if vertex i precedes (not necessarily immediately) vertex j in the tour, and 0 otherwise.

As discussed in Sarin et al. (2005), the y_{ij} variables defined in this way allow *precedence constraints* just by setting a value of one to the variables matching the precedence condition of the problem, that is, adding the following set of constraints to the model:

$$y_{ij} = 1, \quad \forall j \in V \backslash \{0\}, \quad \forall i \in SPC_j, \tag{3.29}$$

where SPC_j is the set of vertices that must precede j in the tour. In the case of the CTSP-d problem, where the precedence relationships are defined through the V_p priority sets, the equations set (3.29) can be formulated in the following way:

$$y_{ij} = 1, \quad \forall i \in V_p, j \in V_q : p, q \in P, q > p + d. \tag{3.30}$$

The formulations proposed in this section are based on the formulations presented in the

review by Öncan et al. (2009) and in the papers in which each one of the formulations was originally proposed. All these formulations are proven to be stronger than the MTZ formulation for the classic TSP. Moreover, the proposed formulations may have valid inequalities included or some other kinds of reformulation techniques, which will be described here.

Finally, we highlight that there are many other formulations for the TSP, and it is not our intention to exhaust the subject, but to demonstrate different ways to formulate the CTSP-d, and present how the problem can be identified with the Precedence Constrained Traveling Salesman Problem (PCTSP). In addition to the formulations we adapted, it is an important research subject to study different ways to model the problem and how different formulations can be used to reach the best results in computational experiments. Moreover, it helps to recognize the CTSP-d as a particular or general case of other known variants of the TSP, which can lead to efficient solution approaches to the problem. In this context, some recent papers which may be interesting are Godinho et al. (2014), where the authors present formulations for the Time-Dependent Traveling Salesman Problem, for which the asymmetrical TSP is a particular case, and then show how to develop an stronger formulation using precedence variables, Balma et al. (2018), where the authors work with strong multi-commodity flow formulations for the classical asymmetrical TSP and for the precedence-constrained case, and Gouveia et al. (2018), in which combinations and projections of flow models are used to improve the PCTSP.

Formulation based on Gouveia and Pires (1999)

The first formulation that is adapted to model the CTSP-d was based on precedence constraints proposed by Gouveia and Pires (1999), where the relationship between the variables x_{ij} and y_{ij} are defined by the constraints:

$$x_{ij} - y_{ij} \leqslant 0, \quad \forall i, j \in V : i, j > 0, i \neq j \tag{3.31}$$

$$x_{ij} + y_{ji} \leqslant 1, \quad \forall i, j \in V : i, j > 0, i \neq j \tag{3.32}$$

$$x_{ji} + x_{ij} + y_{ki} - y_{kj} \leqslant 1, \quad \forall i, j, k \in V : i, j, k \neq 0, i \neq j \neq k \tag{3.33}$$

$$x_{kj} + x_{ik} + x_{ij} + y_{ki} - y_{kj} \leqslant 1, \quad \forall i, j, k \in V : i, j, k > 0, i \neq j \neq k \tag{3.34}$$

Constraints (3.31) represent the fact that if j is the successor of i in the tour, then i precedes j in the tour, and conversely, if i does not precede j in the tour, then j cannot be the successor of i. Similarly, constraints (3.32) ensure that if j is the successor of i, then j does not precede i, as well as if j precedes i, so we know that j is not the successor of i in the tour.

Constraints (3.33) and (3.34) are lifted versions of the constraints:

$$x_{ij} + y_{ki} - y_{kj} \leqslant 1, \quad \forall i, j, k \in V : i, j, k > 0, i \neq j \neq k, \tag{3.35}$$

which ensure the following condition: when the arc (i, j) is included in the tour, that is, when j is the successor of i in the tour, then $y_{ki} \leqslant y_{kj}$, for all possible $k \neq 0$, which means that every vertex k that precedes i in the tour must necessarily precede j. The purpose of using (3.33) and (3.34) instead of (3.35) is to build a stronger formulation (Öncan et al., 2009). One disadvantage of doing this is the fact that the formulation is bigger. It is important to notice that the combination of the expressions (3.1)-(3.4), (3.30), (3.31)-(3.32) and one of the constraints (3.33), (3.34) or (3.35) defines a formulation for the CTSP-d.

- In this work we will call GP1 the formulation defined by (3.1)-(3.4), (3.30), and (3.31)-(3.34), which has $\mathcal{O}(n^2)$ variables and $\mathcal{O}(n^3)$ constraints.

Formulation based in Sarin et al. (2005)

The next formulation to model the CTSP-d is based on the formulation proposed by Sarin et al. (2005), where the variables x_{ij} and y_{ij} are related by the following constraints:

$$y_{ij} \geqslant x_{ij}, \quad \forall i, j \in V : i, j > 0, i \neq j, \tag{3.36}$$

$$y_{ij} + y_{ji} = 1, \quad \forall i, j \in V : i, j > 0, i \neq j \tag{3.37}$$

$$y_{ij} + x_{ji} + y_{jk} + y_{ki} \leqslant 2, \quad \forall i, j, k \in V : i, j, k > 0, i \neq j \neq k, \tag{3.38}$$

$$x_{0j} + x_{j0} \leqslant 1, \quad \forall j \in V, j > 0 \tag{3.39}$$

The constraints set (3.36) ensures that if a vertex j is the successor of vertex i in a tour, then the associated variable y_{ij} is equal to 1, and at the same time it imposes that if j does not succeed i in the tour, then $x_{ij} = 0$. Another basic relationship between a pair of distinct vertices i and j in the graph is that j must be before or after i, condition that is guaranteed by constraints set (3.37). The set in (3.38) is a lifted version of

$$y_{ij} + y_{jk} + y_{ki} \leqslant 2, \quad \forall i, j, k \in V : i, j, k > 0, i \neq j \neq k \tag{3.40}$$

and, together with (3.36) and (3.37), is sufficient to completely define the y_{ij} variables main condition, that is, it ensures $y_{ij} = 1$ if, and only if, j is successor of i in the tour. The constraints set (3.39) is not necessary to define the formulation, but they are included as valid inequalities to strengthen the formulation.

- In summary, an alternative formulation for the CTSP-d, which is denoted by SSB1, is defined by (3.1)-(3.4), (3.30) and (3.36)-(3.39), and it has $\mathcal{O}(n^2)$ variables and $\mathcal{O}(n^3)$ constraints.

Formulation based in Sherali et al. (2006)

The last formulation adapted to fit the d-relaxed priority rule was proposed by Sherali et al. (2006). The authors deduce a reformulation for a model defined by (3.1)-(3.4), (3.30), (3.36)-(3.37) and (3.41)-(3.43), where the last constraints are defined as follows:

$$y_{ij} \geqslant x_{0i}, \quad \forall i, j \in V : i, j > 0, i \neq j, \tag{3.41}$$

$$y_{ji} \geqslant x_{i0}, \quad \forall i, j \in V : i, j > 0, i \neq j, \tag{3.42}$$

$$-(1 - x_{ik}) \leqslant y_{ij} - y_{kj} \leqslant 1 - x_{ik}, \quad \forall i, j, k \in V : i, j, k > 0, i \neq j \neq k, \tag{3.43}$$

where inequalities (3.41) and (3.42) are a disaggregated version of (3.39) (notice that summing up the two expressions imply the same constraints as in (3.39) as a consequence of (3.37)), and inequalities (3.43) ensure that when $x_{ik} = 1$ for some $i, k \in V$, $i, k > 0$, $i \neq k$, than the relation $y_{ij} = y_{kj}$ will be valid, for all $j \in V$, $j > 0$, $i \neq j \neq k$. In other words, the successors of i different from k will be successors of k too, and the same applies to the predecessors of k different from i.

The new formulation is obtained by applying a *Reformulation-linearization technique* (RLT) to the following non-linear relations:

$$(0 \leqslant y_{kj} \leqslant 1) \cdot x_{ik}, \quad \forall i, j, k \in V : i, j, k > 0, i \neq j \neq k,$$

$$\left[\sum_{k=0, k \neq i}^{n-1} x_{ik} = 1 \right] \cdot y_{ij}, \quad \forall i, j \in V : i, j > 0, i \neq j,$$

$$\left[\sum_{i=0, i \neq k}^{n-1} x_{ik} = 1 \right] \cdot y_{kj}, \quad \forall j, k \in V : j, k > 0, j \neq k.$$

To deduce the new constraints sets for the problem based on the non-linear constraints described above, a set of auxiliary variables must be defined representing the product of the variables x_{ik} and y_{kj} in the following way:

$$t_{ij}^k = x_{ik} y_{kj}, \quad \forall i, j, k \in V : i, j, k > 0, i \neq j \neq k. \tag{3.44}$$

After applying the RLT, the new constraints obtained are:

$$0 \leqslant t_{ij}^{k} \leqslant x_{ik}, \quad \forall i,j,k \in V : i,j,k \neq 0, i \neq j \neq k \tag{3.45}$$

$$\sum_{\substack{k \in V \setminus \{0\}: \\ k \neq i,j}} t_{ij}^{k} + x_{ij} = y_{ij}, \quad \forall i,j \in V : i,j > 0, i \neq j \tag{3.46}$$

$$x_{0k} + \sum_{\substack{i \in V \setminus \{0\}: \\ i \neq j,k}} t_{ij}^{k} = y_{kj}, \quad \forall j,k \in V : j,k > 0, j \neq k \tag{3.47}$$

The details involved in obtaining these new linear constraints can be found in the paper by Sherali et al. (2006), and all the steps are not reproduced here since this would not contribute to the goal of relating the d-relaxed priority rule and the already known formulations for the TSP.

- The formulation is defined by (3.1)-(3.4), (3.30), (3.37)-(3.38), (3.41)-(3.42), (3.45)-(3.47) and is denoted by SST1. Note that this formulation is bigger than all the formulations previously described, since it has $\mathcal{O}(n^3)$ variables and $\mathcal{O}(n^3)$ constraints.

3.2.1 Proposed valid inequalities for the precedence variables-based formulations

In both ways of formulating the TSP, i.e., using the position variables u_j or the precedence variables y_{ij}, our main goal is to have variables representing global relationships among the vertices in a tour, which allows the avoiding of subtours and the formulation of more complex variants of the problem, for example the CTSP-d discussed in this thesis. How variables u_j and y_{ij} are related is presented next, showing how the valid inequalities proposed for the formulation MTZ1 in Section 3.1.1 can be adapted and used to strengthen the GP1, SSB1 and SST1 formulations.

The first point to emphasize is that any feasible solution is a tour where the origin vertex always figures before and after all the other vertices. The MTZ formulation follows the convention $u_0 = 0$, which is ensured by the subtour elimination constraints. In the precedence-based formulations, the constraints do not define values for the y_{ij} variables, and so two new sets of constraints are imposed as follows:

$$y_{i0} = 0, \quad \forall i \in V \setminus \{0\} \tag{3.48}$$

$$y_{0j} = 1, \quad \forall j \in V \setminus \{0\} \tag{3.49}$$

These two sets of fixed values for y_{ij} allow an analogy between the way variables u_j

relate to the feasible solutions in the MTZ-based formulations and the way variables y_{ij} relate to them in the precedence-based formulations.

Constraints set (3.49) represents the fact that the vertex 0 is always the first one in any tour and it plays the same role as constraint (3.24) for the u_j variables. Since the variables u_j cannot represent the fact that 0 is the final vertex in the tour too (as long as u_0 cannot assume two different values in the same solution), (3.48) is added to the precedence variable models and so the following relationship is valid:

$$u_j = \sum_{\substack{k \in V: \\ (k,j) \in A}} y_{kj}, \quad \forall j \in V, j \neq 0. \tag{3.50}$$

In other words, the values variables u_j assume, for each $j \in V$, represent how many vertices there are before j in a solution (except for the final vertex 0). Having this simple relationship well defined, any valid inequality proposed for the MTZ1 formulation can be directly adapted to any of the GP1, SSB1 or SST1 formulations through a direct substitution.

Analyzing expressions (3.9)-(3.12)

Another fact that can be used to strengthen the precedence variable models is that constraints sets (3.9)-(3.12) only rely on variables x_{ij}, which are used in all the formulations presented, and so they can be directly added to GP1, SSB1 and SST1. Proposition 3 shows that equation (3.9) does not need to be added into SSB1 and SST1.

Proposition 3. *Constraints* (3.9) *are redundant in SSB1 and SST1.*

Proof. Let i and j be two vertices such that $i \in V_p$ and $j \in V_q$, where $q > p + d$. Constraints (3.30) imply $y_{ij} = 1$, and so $y_{ji} = 0$ by (3.37). But this conclusion leads to $x_{ji} = 0$, as a consequence of (3.36). And since i and j are arbitrary, $x_{ji} = 0$, for any $i \in V_p$, $j \in V_q$ such that $q > p + d$. Summing all these variables gives

$$\sum_{i \in V_p} \sum_{\substack{j \in V_q: \\ (i,j) \in A}} x_{ji} = 0, \quad \forall p, q \in P, q > p + d,$$

which is exactly the result in (3.9), and so these constraints would not make any difference to models SSB1 and SST1 if added. ■

For the GP1 formulation, the condition in (3.37) is not ensured by its constraints sets, and so the conclusion reached for the two other precedence variables formulations is not valid. But since this condition must be valid for any feasible solution, add (3.37)

to the new CTSP-d formulation based on GP1 and the result follows, since constraint sets (3.36) and (3.31) are the same. Moreover, in a formulation including (3.31) and (3.37), (3.32) is not needed, since it will follow from a direct substitution of (3.37) in (3.31).

Analyzing variables bounds (3.14)-(3.15) **and** (3.21)-(3.23)

Constraint sets (3.14) and (3.15) can be adapted to produce lower and upper bounds to the sum of the y_{ij} variables in the following way:

$$\sum_{\substack{k\in V:\\(k,j)\in A}} y_{kj} \geqslant \sum_{r=0}^{p-d-1} |V_r|+1, \quad \forall j \in V_p, d+1 \leqslant p \leqslant P_{max} \tag{3.51}$$

$$\sum_{\substack{k\in V:\\(k,j)\in A}} y_{kj} \leqslant \sum_{r=0}^{p+d} |V_r|, \quad \forall j \in V_p, 0 \leqslant p \leqslant P_{max}-d-1 \tag{3.52}$$

However, this would not lead to a better formulation, which is proved in Proposition 4.

Proposition 4. *Constraints* (3.51) *and* (3.52) *are redundant in the precedence-variable models.*

Proof. In fact, for each fixed $j \in V\backslash\{0\}$, $j \in V_p$, for some p value, and so it follows from (3.30), (3.37) and (3.49) that

$$\sum_{\substack{k\in V:\\(k,j)\in A}} y_{kj} = y_{0j} + \sum_{\substack{k\in V_r:\\r=0,\ldots,p-d-1}} y_{kj} + \sum_{\substack{k\in V_r:\\r=p-d,\ldots,p+d}} y_{kj} + \sum_{\substack{k\in V_r:\\r=p+d+1,\ldots,P_{max}}} y_{kj} =$$

$$= 1 + \sum_{r=0}^{p-d-1} |V_r| + \sum_{\substack{k\in V_r:\\r=p-d,\ldots,p+d}} y_{kj} \Rightarrow 1 + \sum_{r=0}^{p-d-1} |V_r| \leqslant \sum_{\substack{k\in V:\\(k,j)\in A}} y_{kj} \leqslant \sum_{r=0}^{p+d} |V_r|.$$

The lower and upper bounds found are exactly the same as in (3.51) and (3.52), and so the constraints are redundant for the models, as stated. ■

Identity (3.50) can be applied to (3.21), (3.22) and (3.23), which leads to new constraints sets:

$$\sum_{\substack{k\in V:\\(k,i)\in A}} y_{ki} \leqslant n-(n-2)x_{0i} - \sum_{(i,j)\in A: j\neq 0} x_{ij}, \quad \forall i \in V : (0,i) \in A \tag{3.53}$$

$$\sum_{\substack{k\in V:\\(k,i)\in A}} y_{ki} \geqslant (n-2)x_{i0} + \sum_{(j,i)\in A: j\neq 0} x_{ji}, \quad \forall i \in V : (i,0) \in A \tag{3.54}$$

$$\sum_{\substack{k\in V:\\(k,i)\in A}} y_{ki} \geqslant 1 + \sum_{(j,i)\in A: j\neq 0} x_{ji}, \quad \forall i \in V_p, 0 \leqslant p \leqslant d. \tag{3.55}$$

Adapting expressions (3.27) **and** (3.28)

Note that (3.28) is a generalization of the result in (3.39) for all the vertices $i \in V$ and can be added to the models, since it only depends on the x_{ij} variables. But this would not produce any improvement in SSB1 and SST1, as shown in Proposition 5.

Proposition 5. *Constraints* (3.28) *are redundant for SSB1 and SST1.*

Proof. Both the SSB1 and SST1 formulations consider the constraints set (3.36). This implies that, for any $i, j \in V$, $i \neq j$, $i, j > 0$, the following condition must hold:

$$\begin{cases} x_{ij} \leqslant y_{ij} \\ x_{ji} \leqslant y_{ji} \end{cases} \Rightarrow x_{ij} + x_{ji} \leqslant y_{ij} + y_{ji} = 1, \quad \forall i, j \in V : i, j > 0, i \neq j \tag{3.56}$$

In formulation SSB1, the case where $i = 0$ or $j = 0$ is directly considered as the constraints set (3.39). Finally, for the formulation SST1, summing the constraints in (3.41) and (3.42) gives a similar conclusion to that in (3.56) and so the result is valid. ■

Constraint set (3.37) is included in all precedence-variable reformulations and so the result in (3.56) will be valid for the GP1-based reformulation too. In this case, the only constraints that can be added to this new formulation are (3.39) or (3.41) and (3.42), which include the initial vertex.

The last constraint set proposed for MTZ2 to be analyzed is (3.27). In this case, applying the (3.50) relationship derives valid inequalities to the precedence-variable formulations. This gives the result presented in (3.57).

$$\sum_{(i,j)\in A} y_{ij} = \frac{n(n-1)}{2}. \tag{3.57}$$

Since subtour elimination is already part of the original models, and the d-relax priority rule is defined in the models through (3.30), a summary of the new formulations for the precedence-variable formulations can be presented.

- (GP2) is given by (3.1)-(3.4), (3.10)-(3.12), (3.30), and (3.31), (3.33)-(3.34), (3.37), (3.39), (3.53)-(3.55), (3.57).

- (SSB2) is defined by (3.1)-(3.4), (3.10)-(3.12), (3.30) and (3.36)-(3.39), (3.53)-(3.55), (3.57).
- (SST2) is given by (3.1)-(3.4), (3.10)-(3.12), (3.30), and (3.37)-(3.38), (3.41)-(3.42), (3.45)-(3.47), (3.53)-(3.55), (3.57).

The size order of the formulations, the section in which each formulation is described, as well as its reference numbers are shown in Table 3.1. Besides the fact that adding valid inequalities increases the formulation size, the size order of the formulations is still the same for all of them.

	Variables	Constraints	Formulation
H2020	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	(3.1)-(3.4),(3.7)-(3.12)
MTZ1	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	(3.1)-(3.4),(3.7)-(3.8)
MTZ2	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	(3.1)-(3.4),(3.9)-(3.12),(3.14)-(3.17),(3.21)-(3.25),(3.27)-(3.28)
GP1	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	(3.1)-(3.4),(3.30)-(3.34)
GP2	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	(3.1)-(3.4),(3.10)-(3.12),(3.30)-(3.31),(3.33)-(3.34),(3.37),(3.39),(3.53)-(3.55),(3.57)
SSB1	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	(3.1)-(3.4),(3.30),(3.36)-(3.39)
SSB2	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	(3.1)-(3.4),(3.10)-(3.12),(3.30),(3.36)-(3.39),(3.53)-(3.55),(3.57)
SST1	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	(3.1)-(3.4),(3.30),(3.37)-(3.38),(3.41)-(3.42),(3.45)-(3.47)
SST2	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	(3.1)-(3.4),(3.10)-(3.12),(3.30),(3.37)-(3.38),(3.41)-(3.42),(3.45)-(3.47),(3.53)-(3.55),(3.57)

Table 3.1: Size order and reference numbers of each formulation

4 Formulations and a heuristic approach for the CluVRP-d

The aim of the VRP-RPR, considered in this thesis, is to find up to G_{max} vehicle routes starting and ending at the depot, such that: each vertex is visited exactly once and its demand (d_j) is satisfied; the total demand quantity of any route does not exceed the vehicle capacity cap_k; the d-relaxed priority rule is satisfied (locally or globally); a weighted summation of the total travel cost (c_{ijk}) and fixed cost (C_k) is minimized. In this chapter we will call the VRP-PRP version considered here as the Clustered Vehicle Routing Problem with d-relaxed priority rule (CluVRP-d).

The extension of the d-relaxed priority rule in the context of VRP can be divided in two approaches, namely the Local Timing model and the Global Timing model. The first case refers to the application of the rule on the tours described by each working vehicle in isolation, and so the rule is obeyed by each vehicle only over the areas that vehicle visits. However, this rule does not consider any global hierarchy constraints; that is, once a vehicle finishes its work on the higher priority areas in which it was assigned to work, the vehicle may go to the lower priority areas in its tour, which may occur regardless of the existence of high priority areas still not visited in the other vehicles routes. On the other hand, the Global Timing model requires that the d-relaxed priority rule must be obeyed not only by each vehicle individually, but it includes a global hierarchy constraint across all the tours; that is, a vehicle can only visit a lower priority area after all the higher priority areas have been visited by some vehicle. That is, the priority rule must be observed across the fleet of vehicles.

Both the Local and the Global Timing Models were initially described in Panchamgam (2011), and the Local Timing case was treated in Doan et al. (2021) too. In both cases the main aspects described are: each vertex may be visited, at most, once, each vehicle has a maximum capacity and there are maximum lengths for the routes of each vehicle. Considering these features, the priorities must satisfy two priority rules, namely the "d-relaxed priority rule" and the "order of demand fulfillment (ODF)", which states that "at the end of the planning horizon, all the higher priority vertices must be visited before the vehicles can move to any lower priority vertex". The

objective considered is to minimize the total traveling costs in the tours, as well as minimize the total demand not met in the vertices not visited.

An important property proposed by Doan et al. (2021) is to present a problem where not all areas must be visited and, in this case, the d-relaxed priority rule allows tours that strongly weaken the original priorities hierarchy. It can be seen that this rule may allow the construction of tours in which a vehicle will only visit vertices with medium or lower priorities, mixing the different priorities requirements in a very aggressive approach. Moreover, supposing the different priority levels have relevant practical implications, to accept solutions focusing on lower priority vertices instead of the higher priority ones may lead to solutions that cannot be used in real world contexts. It motivates the definition of the ODF rule.

According to Panchamgam (2011) and Doan et al. (2021), the ODF imposes that at the end of the planning horizon, if a vehicle has sufficient capacity, then all vertices with higher priorities must be visited before the vehicle be allowed to visit a vertex with lower priority level. In other words, combining the d-relaxed priority rule and the ODF in a Vehicle Routing Problem in which not all vertices must be visited ensures that, in the solution routes, at least the areas with higher priority levels were visited. We emphasize that it does not extinguish the possibility of a vehicle visiting only areas with medium or lower priorities. However, we can impose that the other vehicles visit the areas with higher priorities, not allowing the construction of routes that "ignore" the existence of priorities to reach smaller/cheaper routes or visits only to vertices with lower demands.

In this study we propose the Clustered Vehicle Routing Problem with d-relaxed priority rule (CluVRP-d) as a direct generalization of the Clustered Traveling Salesman Problem with d-relaxed priority rule (CTSP-d), described in Panchamgam (2011), Hà et al. (2020), Doan et al. (2023) and Teixeira and de Araujo (2024), which have stricter constraints than the VRP-RPR studied in Doan et al. (2021). In particular, we impose that all vertices in the problem must be visited, there is a capacity for the vehicles but not a maximum route length, and the d-relaxed priority rule must be valid in any feasible solution. Requiring that all vertices must be visited removes any consequences of not visiting a vertex, a possibility which was considered by Doan et al. (2021). It is also sufficient to ensure that the solutions do not skip the priority levels, and so removes the need to consider the ODF rule.

4.1 Formulations for the CluVRP-d

Both the proposed models (Local Timing and Global Timing) are based on a common vehicle routing problem structure, which we present below, leaving the details related to each approach for its own section.

• Sets:

$G = \{1, 2, ..., G_{max}\}$: Set of available vehicles.

• Parameters:

d_j: Total demand of vertex $j \in C$.

C_k: Fixed cost of vehicle $k \in G$;

cap_k: Total demand vehicle $k \in G$ can meet over the planning horizon (total working capacity of vehicle $k \in G$);

c_{ijk}: Cost to move vehicle $k \in G$ from vertex $i \in V$ to vertex $j \in V$;

• Variables:

$x_{ijk} = 1$ when arc $(i, j) \in A$ is included in the tour assigned to vehicle $k \in G$, or 0 otherwise.

$y_k = 1$ when there is work assigned for vehicle $k \in G$, or 0 otherwise.

• Formulation:

$$\min \sum_{(i,j)\in A} \sum_{k\in G} c_{ijk} x_{ijk} + \sum_{k\in G} C_k y_k \tag{4.1}$$

$$\sum_{\substack{i\in V:\\(i,j)\in A}} \sum_{k\in G} x_{ijk} = 1, \quad \forall j \in C \tag{4.2}$$

$$\sum_{\substack{i\in V:\\(i,h)\in A}} x_{ihk} = \sum_{\substack{j\in V:\\(h,j)\in A}} x_{hjk}, \quad \forall h \in C, k \in G \tag{4.3}$$

$$\sum_{(i,j)\in A} d_j x_{ijk} \leqslant cap_k, \quad \forall k \in G \tag{4.4}$$

$$\sum_{j\in V:(0,j)\in A} x_{0jk} = y_k, \quad \forall k \in G \tag{4.5}$$

The function defined in (4.1) represents the objective of minimizing the logistics costs involved in the problem, given by the sum of the traveling costs between the vertices and the fixed costs associated with the use of each vehicle in the fleet. The feasible solutions for the problem are given by a set of disjoint tours which start at vertex 0 and such that every vertex different from 0 is visited only once. The constraints in (4.2) ensure the visiting condition of the vertices, while the constraints in (4.3) define the flow condition on the problem - a vertex can be visited by a vehicle if,

and only if, it is then left by this same vehicle. We consider in the formulations the application of the d-relaxed priority rule in the capacitated vehicle routing problem with heterogeneous fleet, that is, each vehicle $k \in G$ has its own maximum working capacity cap_k associated, which must not be exceeded. The constraints in (4.4) ensure this condition. The variables y_k represent the decision about using or not the available vehicles $k \in G$. The constraints in (4.5) impose that vehicle $k \in G$ leaves the origin to visit some vertex if, and only if, vehicle k was chosen for the assignment of some tour, that is, the associated variable y_k has a value of 1.

4.1.1 Local Timing Model

The Local Timing Model refers to the case where the d-relaxed priority rule is applied to the route of each working vehicle in isolation, which means that the rule must only be satisfied by each vehicle over the vertices it was assigned to visit. In practical contexts, this can imply that once a vehicle finishes its work on the higher priority vertices in which it was assigned, it may move to the lower priority vertices it needs to visit, even if there are other high priority vertices not visited in the other vehicle tours. In other words, the vehicles must only satisfy the d-relaxed priority rule locally on their own routes, which motivates the name of the model.

To propose a formulation for the Local Timing Model, we will add a set of auxiliary position variables to the common formulation (4.1)-(4.5) to model the d-relaxed priority rule. Here, we describe the new variables as well as the constraints relating them to the original x_{ijk} variables.

• New variable set:

u_{jk}: Variable representing the position of vertex j in the tour of vehicle k when vertex j is assigned to the vehicle k tour, or 0 otherwise.

Since we are supposing not all the vehicles need to be used in a solution but they all are available in the depot, we may fix $u_{0k} = 0$, for all $k \in G$. In this case, the first constraints we consider are

$$0 \leqslant u_{jk} \leqslant n - 1, \quad \forall j \in V, k \in G \tag{4.6}$$

$$u_{0k} = 0, \quad \forall k \in G \tag{4.7}$$

• Visiting order constraints/Sub-tour elimination: as described above, variables u_{jk} represent the order of visiting each vertex in the tours of the working vehicles. Constraints (4.8) are the lifted version of the classical Miller et al. (1960) constraints, which was proposed for the Traveling Salesman Problem by Desrochers and Laporte (1991), and they suffice to eliminate sub-tours in the problem. By adding the constraints

set (4.9), we impose that when j immediately follows i in the tour of a vehicle, then $u_{jk} = u_{ik} + 1$, that is, the variables represent the succession relationship.

$$u_{ik} - u_{jk} + nx_{ijk} + (n-2)x_{jik} \leqslant n-1, \quad \forall (i,j) \in A, k \in G : j \neq 0 \tag{4.8}$$

$$u_{ik} - u_{jk} - nx_{ijk} - (n+2)x_{jik} \geqslant -n-1, \quad \forall (i,j) \in A, k \in G : i \neq 0 \tag{4.9}$$

• Relationship between u_{jk} and x_{ijk}: since we are considering that the variables u_{jk} will represent the position of the vertex j in the tour of vehicle k, it can only be different from 0 when, in fact, the vertex j belongs to the tour of vehicle k, and so we need to consider the following constraints set:

$$u_{jk} \leqslant (n-1) \sum_{\substack{i \in V: \\ (i,j) \in A}} x_{ijk}, \quad \forall j \in C, k \in G. \tag{4.10}$$

• d-relaxed priority rule: our main goal is to apply the d-relaxed priority rule to the vehicle routing problem. In the case of the local timing model, it can be done as in equations (4.11), where we impose that when $i, j \in C$ are included in the tour of a vehicle k, then they must satisfy the d-relaxed priority rule condition.

$$u_{ik} + 1 \leqslant u_{jk} + n \left(2 - \sum_{\substack{h \in V: \\ (h,j) \in A}} x_{hjk} - \sum_{\substack{h \in V: \\ (h,i) \in A}} x_{hik} \right),$$

$$\forall i \in V_p, j \in V_q, k \in G : p, q \in P, q > p + d. \tag{4.11}$$

4.1.2 Global Timing Model

Different from the Local Timing case, in the Global Timing Model the d-relaxed priority rule must be observed not only in the tours of each vehicle individually, but also across the tours; that is, a vehicle can only visit a lower priority vertex after all the higher priority vertices have been visited by some vehicle. That means all the vehicles must satisfy the d-relaxed priority rule globally in the problem, which leads to the name of the model.

In the Global Timing Model, the auxiliary variables needed to formulate the problem are cumulative time variables, which allows to measure when all the higher priority vertices were visited and allow the lower priority vertices to be visited. We describe below the extra variables and parameters used in the formulation, and the constraints proposed.

• New parameters:

t_{ijk}: Time vehicle k spends to travel between vertices $i \in V$ and $j \in V$;

M_{ij}: A set of big numbers used as auxiliary parameters to define the cumulative time variables;

M: A sufficiently big number, representing the maximum value any cumulative time variable may assume. Since we are not imposing any total planning horizon timing for any vehicle, this value must be big enough to not prohibit any feasible tour. In the situation that the solutions should have an upper bound on the total time of the tours, this value can be used to impose it.

• New variable:

s_{jk}: Variable representing the total cumulative traveling time of vehicle k from the depot node 0 up to vertex j, if j belongs to the tour of vehicle k, or 0 otherwise.

• Cumulative time constraints/Sub-tour elimination: analogous to constraints (4.8) and (4.9) in the local timing model, here in the global timing model we have constraints based on the lifted version of the Miller et al. (1960) model to eliminate sub-tours and impose a relationship between a pair of variables s_{ik} and s_{jk} when arc $(i, j) \in A$ is included in the tour assigned to vehicle k. In this case, constraints (4.12) and (4.13) guarantee that when j succeeds i in a tour, the equality $s_{jk} \geqslant s_{ik} + t_{ij}$ must hold, and $s_{0k} = 0$, for all $k \in G$.

$$s_{ik} - s_{jk} + M_{ij}x_{ijk} \leqslant M_{ij} - t_{ij}, \quad \forall (i,j) \in A, k \in G : j \neq 0 \tag{4.12}$$

$$s_{0k} = 0, \quad k \in G \tag{4.13}$$

• s_{ik} variables setup/tour time limit: the constraints presented in (4.14) ensure two important conditions to the model: when the vehicle k does not visit the vertex j, we will have $s_{jk} = 0$, otherwise if the variables s_{jk} have an upper bound, it can be imposed using the M parameter value.

$$s_{jk} \leqslant M \sum_{\substack{i \in V: \\ (i,j) \in A}} x_{ijk}, \quad \forall j \in C, k \in G. \tag{4.14}$$

• d-relaxed priority rule: as discussed above, the difference between the Local Timing Model and the Global Timing Model is that, in the first case, the d-relaxed priority rule must be satisfied by each vehicle $k \in G$ in an isolated manner, while in the global case the rule is obeyed by all the vehicles across the tours. In this sense, constraints (4.15) ensure the validity of the d-relaxed priority rule by imposing that given two vertices $i, j \in V$, if $i \in V_p$, $j \in V_q$ and $q > p + d$, then a vehicle can only visit

j after the vertex i has already been visited.

$$\sum_{k \in G} s_{ik} \leqslant \sum_{k \in G} s_{jk}, \quad \forall i \in V_p, j \in V_q : p, q \in P, q > p + d. \tag{4.15}$$

4.1.3 Valid Inequalities

We included some common sets of valid inequalities in both the Local and Global Timing Models to build stronger formulations, which are presented below.

$$\sum_{i \in V_p} \sum_{\substack{j \in V_q: \\ (i,j) \in A}} x_{jik} = 0, \quad \forall k \in G, p, q \in P : q > p + d \tag{4.16}$$

$$\sum_{i \in V_p} \sum_{\substack{j \in V_q: \\ (i,j) \in A}} x_{ijk} \leqslant 1, \quad \forall k \in G, p, q \in P : q > p + d \tag{4.17}$$

$$\sum_{k \in G} (x_{ijk} + x_{jik}) \leqslant 1, \quad \forall (i,j) \in A. \tag{4.18}$$

Constraints in (4.16) represent the fact that the vertices in a set V_q can only be visited after all the vertices in the V_p sets have all been visited first, and it must hold for all $q > p + d$. Moreover, for any vehicle $k \in G$, when a vertex belonging to a V_p set is left, it is possible that the next vertex in the tour belongs to a V_q set such that $q > p+d$, but it can only occur, at most, once, which is represented in (4.17). Both constraints (4.16) and (4.17) were proposed by Doan et al. (2021), as generalizations for the multi-vehicle case of the valid inequalities proposed by Hà et al. (2020) for the CTSP-d. Finally, the constraints set (4.18) represents the fact that each arc (i, j) can be included in the solution assigned to a single vehicle, and it can be traversed in only one direction, that is, if a vehicle $k \in G$ goes from i to j, then it cannot go back from j to i.

4.2 A heuristic approach for the Local Timing case

Considering the Local Timing case, we proposed a three-phase heuristic procedure based on clustering the vertices and after that, generating an initial feasible solution and then improving it through a variable neighborhood search method. This is a common approach for the Vehicle Routing Problem, as pointed out by Laporte et al. (2014). To cluster the vertices and produce the initial solution we solve a mixed integer linear programming problem based on the p-Medians problem, after that the variable neighborhood search procedure intra-route and inter-route moves are applied. The subject of metaheuristics for variants of the VRP is very comprehensive, including many different approaches based on local search methods, population based methods,

matheuristics, as well as other kinds of solution procedures. A broader discussion can be found in Laporte et al. (2014) and Desaulniers et al. (2014).

4.2.1 Clustering Phase

The goal of the clustering phase is to pack the vertices of the problem into clusters in a way that, for each cluster, it is possible to order its vertices and produce a valid initial route. It is not our intent to discuss the topic "clustering algorithms" but just show how to apply the method as part of the initialization phase of our heuristic. The reader can find deeper analysis about the algorithm in Ikotun et al. (2023). The clustering procedure is split in two steps that are described as follows.

Uncapacitated clustering step

The first step in the proposed heuristic is clustering the vertices according to their coordinates. To do so, we first use the K-means method (Lloyd, 1982), a classical and computationally efficient clustering algorithm, which can be applied in Python using the SciKit Learn (sklearn) module (`https://scikit-learn.org/stable/`) treating the data in the following way - the coordinates of the vertices are put in a table, where each vertex is in a line, the x coordinate is in the first column, while its y coordinate is in the second column.

The K-means method is an iterative procedure based on clustering vertices using centroids that are updated at each iteration until the algorithm stops. The centroids of the clusters are initially created in determined positions and each vertex is assigned to the cluster with the nearest centroid position in a way that each vertex belongs to only one cluster. In the next step, for each cluster, the mean of its vertices features are calculated, the position of the centroid is changed to this value and after that the vertices are reassigned to the clusters with the nearest centroids again. In Algorithm 1 we describe the complete K-means method, summarizing its two main steps - assign the vertices to the clusters and move the centroids to their new position. The stopping criterion of the procedure is when it reaches a maximum number of iterations, but the method can be stopped when the clusters are no longer being improved after an iteration, or the change in the centroids positions are smaller than a given value.

We highlight that there are different ways to initialize the clusters positions in the K-means method, and so the final result obtained by the method may vary accordingly. In our case we are initializing the clusters positions randomly, using the sklearn embedded functionalities, and so the results tend to be slightly different each time a problem is solved. The final result will be a set of K clusters that do not have any

Algorithm 1 K-Means procedure

Input: A set of vertices $\{x_1, ..., x_n\} \in \mathbb{R}^m$, a number of centroids K and a maximum number of iterations $IterQty$.

Output: A set of disjoint clusters $S_1, ..., S_K$ with centroids coordinates $\mu_1, .., \mu_K \in \mathbb{R}^m$.

$t \leftarrow 0$
Randomly initialize $\mu_1^0, ..., \mu_K^0$
repeat
    $t \leftarrow t + 1$
    for $k = 1$ **to** K **do**
        $S_k^t \leftarrow \emptyset$
    end for
    for $j = 1$ **to** n **do**
        $k^* \leftarrow \underset{k=1,...,K}{\arg\min} \{\|x_j - \mu_k^t\|^2\}$ // Assign vertex x_j to the nearest centroid
        $S_{k^*}^t \leftarrow S_{k^*}^t \cup \{x_j\}$
    end for
    for $k = 1$ **to** K **do**
        $\mu_k^t \leftarrow \frac{1}{|S_k^t|} \sum_{x_j \in S_k^t} x_j$ // Update centroids position
    end for
until $t \geqslant IterQty$
for $k = 1$ **to** K **do**
    $S_k \leftarrow S_k^t$
    $\mu_k \leftarrow \mu_k^t$
end for

vertices in common and all vertices are assigned to exactly one cluster. Moreover, since the positions of the clusters are defined iteratively using the mean of the coordinates of the vertices assigned to each of them, in the general case, the coordinates of the clusters do not need to correspond to the coordinates of the areas in the Routing Problem.

It gives a clustering procedure for the problem vertices and, after this step, for each cluster, it is only necessary to assign a vehicle to its areas and the Routing Problem is reduced to a set of Traveling Salesman Problems. The impracticality of using this approach solely is that, by doing so, we are not taking into account the vehicles working capacity. By just clustering the vertices based on (x, y) coordinates some clusters might represent infeasible problems, including a number of vertices that would require a working capacity level that none of the vehicles available could meet. To overcome this, we propose a second step in the clustering procedure, which is described as follows.

Capacitated clustering step

The second step to cluster the vertices is to build a large number of clusters using only the coordinates of the vertices through the K-means method, as described previously, and then to use the centroids coordinates as parameters in an optimization

model based on the p-Medians Problem to merge these clusters, obtaining a smaller number of clusters needed to solve the Routing Problem, but imposing the working capacity as constraints in this model.

The p-Medians problem is a classical integer programming problem in the field of discrete location that can be briefly described as follows: given a set of n users that must be supplied by exactly p facilities (usually called medians in the context of the problem), how to choose these facilities among a bigger set of m potential facilities in a way that minimizes the allocation costs of the users to the p medians. We can model this problem representing the m potential facilities and the n users as the vertices of a graph, and when a facility is chosen to be part of the solution, the vertices assigned to it are related to the facility by edges in the graph. A common case in the literature is to consider the set of users as the set of potential facilities, which is not the case we are considering. For more details about the problem the reader can refer to Marín and Pelegrín (2019).

Our goal in this step of the heuristic is to choose a set of exactly r clusters representing the r vehicles that will be used to solve the problem. Considering the vertices that must be visited in the Routing problem are the users in the p-Medians problem, by pre-clustering the vertices using the K-means method, we obtained a set of potential facilities as the K centroids selected, and the present step is to select exactly r of these centroids aiming to minimize the sum of the distances among the vertices to the selected centroids and the fixed costs of the vehicles associated to them. It can be seen as an application of the p-Medians problem, where for the sake of notation we are representing the p parameter as r. To model the problem we will call the set of centroids selected by the K-means as *centroids candidates* and the centroids chosen to be part of the final solution as *definitive centroids*.

To present our model based in the p-Medians problem, we need to define the following auxiliary parameters and variables:

• New set:

$\mathbb{K}$: Set of K centroids candidates selected by the K-means method.

• New parameter:

D_{jk}: Distance between vertex $j \in V$ and the centroid candidate $k \in \mathbb{K}$.

• New variables:

$x_{kl} = 1$ if $k \in \mathbb{K}$ is chosen as one of the definitive centroids used to built an initial solution for the problem, and in this case the vehicle $l \in G$ is assigned to it, and 0 otherwise.

$y_{jk} = 1$ if vertex $j \in V$ is assigned to the cluster associated with the definitive centroid $k \in \mathbb{K}$, and 0 otherwise.

$$\min \quad \sum_{j \in V} \sum_{k \in \mathbb{K}} D_{jk} y_{jk} + \sum_{k \in \mathbb{K}} \sum_{l \in G} C_l x_{kl} \tag{4.19}$$

$$\text{s. t.} \quad \sum_{k \in \mathbb{K}} y_{jk} = 1, \quad \forall j \in V \tag{4.20}$$

$$y_{jk} \leqslant \sum_{l \in G} x_{kl}, \quad \forall j \in V, k \in \mathbb{K} \tag{4.21}$$

$$\sum_{l \in G} x_{kl} \leqslant 1, \quad \forall k \in \mathbb{K} \tag{4.22}$$

$$\sum_{j \in V} d_j y_{jk} \leqslant \sum_{l \in G} cap_l x_{kl}, \quad \forall k \in \mathbb{K} \tag{4.23}$$

$$\sum_{k \in \mathbb{K}} \sum_{l \in G} x_{kl} = r \tag{4.24}$$

$$x_{lk} \in \{0, 1\}, \quad \forall l \in G, \quad \forall k \in \mathbb{K} \tag{4.25}$$

$$y_{jk} \in \{0, 1\}, \quad \forall j \in V, \quad \forall k \in \mathbb{K}. \tag{4.26}$$

Objective function (4.19) represents the minimization of distances between the candidate centroids and the vertices assigned to them added to the total fixed cost of the vehicles. Constraints (4.20) ensure that each vertex $j \in V$ is assigned to exactly one cluster, defined by the centroid $k \in \mathbb{K}$. Constraints (4.21) are set up constraints imposing that a vertex $j \in V$ can only be assigned to a cluster if the associated centroid $k \in \mathbb{K}$ was chosen to group the vertices a vehicle $l \in G$ will visit. Since the clusters are used to group the vertices a vehicle will visit, we must have that each vehicle $l \in G$ can be assigned to at most one cluster. This condition is guaranteed by constraints (4.22). This is a capacitated problem, where each vehicle has a defined maximum working capacity that must not be exceeded in a solution. This is ensured by constraints (4.23). Constraints (4.24) are the p-facilities constraints of the p medians problem, imposing that the number of clusters in a solution must be exactly r. Finally, constraints (4.25) and (4.26) define the variables domains.

4.2.2 Initial Feasible Solution Phase

After clustering the vertices, we need to order them in a way they define a feasible solution, which will be improved in the later steps of the procedure. There are different ways this can be done. The simplest way to define a solution that will be feasible for any value of d is to first line up all the vertices with priority level 0, after that all vertices with priority level 1, then priority level 2 and so on. The order of the vertices inside the priority level is not relevant. This will lead to an arbitrary feasible solution

to the problem, built in a very specific way, which is interesting when $d = 0$, where all the feasible solutions will satisfy this kind of ordering.

Since the heuristic has an improvement phase, the initial solution chosen is not the most important point to be considered. On the other hand, starting the procedure with a solution very distant from optimality may lead to more local search steps to reach a high-quality solution, increasing the computational time. In a local search procedure, it is always important to initialize the solutions with the most general solution available, since if all the starting solutions were near to each other, the procedure may end up being trapped in the same set of local optima.

To initialize the heuristic with a randomly generated solution that really explores the d-value of the problem, we built the initial solutions by adding the vertices to the route randomly, respecting the problem constraints. The random initialization procedure is presented in Algorithm 2.

Algorithm 2 Random Initialization procedure

Input: Vertices assigned to each vehicle $k \in G$.
Output: A feasible solution s for the CluVRP-d, where each s_k is the route designed for vehicle k.

1: **for** $k = 1$ **to** $k = G_{max}$ **do**
2: Initialize Candidate List (CL_k) with all vertices allowed for vehicle k
3: $r \leftarrow 0$
4: $s_k \leftarrow s_k \cup \{0\}$
5: $CL_k \leftarrow CL_k \backslash \{0\}$
6: **while** $CL_k \neq \emptyset$ **do**
7: Updated Restricted Candidate List for vehicle k (RCL_k) with all vertices in CL_k that can follow r in the route without violating the d-relaxed priority rule
8: Choose a vertex $v \in RCL_k$ randomly
9: $r \leftarrow v$
10: $s \leftarrow s \cup \{r\}$
11: $CL_k \leftarrow CL_k \backslash \{r\}$
12: **end while**
13: $s_k \leftarrow s_k \cup \{0\}$
14: **end for**

The main challenge in Algorithm 2 is to ensure the d-relaxed priority rule will be respected while the vertices are added to each vehicle solution, since the rest of the procedure is basically a random initialization for the classic TSP. The central aspect in the d-relaxed priority rule is to impose that, given a feasible d value and a priority level p, if a vertex has priority level q greater than $p + d$, it can only be visited after all vertices with priority levels $0, 1, ..., q - d - 1$ have all been visited. This condition is guaranteed by constraints (4.11), but it can be seen in the following way: the greatest visiting order position a vertex in V_p is in the route must be inferior to the lowest

visiting order position a vertex in V_q is, for all $p, q \in P$ such that $q > p + d$. It can be represented mathematically by:

$$\max_{i \in V_p}\{u_{ik}\} + 1 \leqslant \min_{\substack{j \in V_q: \\ q > p+d}} \{u_{jk}\}, \quad \forall k \in G, p \in P. \tag{4.27}$$

Besides the fact this is not trivial to solve by exact methods, to see clearly how the variables in the priority sets V_p are related to each other is the central point to develop heuristic procedures for both the CTSP-d and for the CluVRP-d, mainly in the Local Timing Model, where each vehicle describes a route equivalent to a CTSP-d. The constraint in inequality (4.27) is a simple condition to be verified computationally.

Finally, we highlight that the steps described in Algorithm 2 can be easily adapted into a naive greedy search initialization method or a more general Greedy Randomized Adaptive Search Procedure (GRASP) (Feo and Resende, 1995). In the first case, we can obtain the greedy start by changing step 8, where instead of selecting vertex v randomly, the algorithm can be defined to select the vertex v nearest to r. The GRASP method is defined by changing step 8 to select vertex v randomly among a restricted set of the nearest vertices of r, where the size of the set is a user defined parameter. Any of these strategies can be adopted and can lead to better or worse results in each case. In the work by Hà et al. (2020) the authors choose to use the GRASP procedure to generate an initial solution for the CTSP-d and then use it successfully in an Iterated Local Search procedure. In our case, we used the random initialization, which is a simpler procedure that can lead to a broader range of initial solutions.

4.2.3 Improvement Phase

After clustering the vertices and build an initial feasible solution for the problem, the next step is to improve the solution. There are different ways to do it, based on different kinds of methods. In the Vehicle Routing Problem literature we find improvement heuristics based on local search moves (Máximo et al., 2022), adaptive large neighborhood approaches (Shi et al., 2023), population based methods (Wu and Gao, 2023), among others.

Our proposal is to work with an improvement strategy based on a two steps local search algorithm. In one step, the neighborhoods are built based on changing the positions of the vertices inside the route designed for each vehicle. This step is named the intra-route search step. The second step is to search for improvements in the problem solution by reassigning some vertices that are in the route of a given vehicle to the route of a different vehicle. After that, both steps are repeated individually until the method do not find better solutions after a given number of tries.

There are some interesting advantages of working with local search movements: they are, in general, easy to understand, easy to implement computationally, and do not consume lots of time and resource to run. In this sense, understanding the details of the problem and what is needed to build the neighborhoods, using this kind of heuristics may be a very promising way to obtain good solutions for problems, even for instances with higher number of vertices. In which follows we describe in details each of the heuristic steps.

Intra-route search step

Intra-route neighborhoods contain solutions in which a single route is changed with respect to the problem solution (Desaulniers et al., 2014). In this sense, for each route, the search steps to be performed can be the same commonly used to solve the TSP, and so we can apply methods already developed for the CTSP-d to improve the solutions locally. Our approach is based on the method presented in Hà et al. (2020), which uses the GILS-RVND framework, proposed by Silva et al. (2012) in the context of the minimum latency problem. Since the method is described as a general metaheuristic method, it can be adapted to a large range of problems by adapting the main initialization and neighborhood steps.

The GILS-RVND framework is a combination of heuristic components based on a GRASP initialization, where the solutions are built using a randomized greedy search procedure, with an Iterated Local Search (ILS) procedure based on a Random Variable Neighborhood Descent (RVND) approach, where the method interchanges improvement search steps and random perturbation steps. In our case, both search and perturbation steps are based in six neighborhood types, which are described below:

- **Relocate (1):** A vertex is relocated to a different position in the route. Given two vertices in positions i and j, we can relocate the vertex in position i to the right of the vertex in position j by removing arcs $(i-1, i)$, $(i, i+1)$ and $(j, j+1)$ and adding arcs $(i-1, i+1)$, (j, i) and $(i, j+1)$.

- **Relocate (2):** An arc is relocated to a different position in the route. Given an arc $(i, i+1)$ and a vertex in position j, we can relocate the arc to the right of the vertex by removing arcs $(i-1, i)$, $(i+1, i+2)$, $(j, j+1)$ and adding arcs $(i-1, i+2)$, (j, i), $(i+1, j+1)$.

- **Swap (1, 1):** Two different vertices of the route are interchanged. Given two vertices in positions i and j, we can swap them by removing arcs $(i-1, i)$, $(i, i+1)$, $(j-1, j)$, $(j, j+1)$ and adding arcs $(i-1, j)$, $(j, i+1)$, $(j-1, i)$, and $(i, j+1)$.

- **Swap (2, 1):** An arc and a vertex of the route are interchanged. Given an arc $(i, i+1)$ and a vertex in position j, we can swap them by removing arcs $(i-1, i)$, $(i+1, i+2)$, $(j-1, j)$ and $(j, j+1)$ and adding arcs $(i-1, j)$, $(j, i+2)$, $(j-1, i)$ and $(i+1, j+1)$.

- **Swap (2, 2):** Two different arcs of the route are interchanged. Given two arcs $(i, i+1)$ and $(j, j+1)$, we can swap them by removing arcs $(i-1, i)$, $(i+1, i+2)$, $(j-1, j)$ and $(j+1, j+2)$ and adding arcs $(i-1, j)$, $(j+1, i+2)$, $(j-1, i)$ and $(i+1, j+2)$.

- **2-Opt:** A slice of the route is reverted. Given two vertices in positions i and j, where $j > i$, the route of the vehicle is kept unchanged up to vertex i, the order of the vertices between $i+1$ and j are reverted, and the order of the final vertices is kept unchanged. It can be done by removing arcs $(i, i+1)$ and $(j, j+1)$ and adding arcs (i, j) and $(i+1, j+1)$. To obtain a feasible route, it is necessary to revert the order of the remaining arcs between i and j.

On the main search step, all the six neighborhoods are considered, and since we are working in an RVND format, the order in which the neighborhoods will be chosen varies at each iteration. In the intra-route step, we will perform at most a given I_{ILS} number of iterations before stopping the search and for each neighborhood we will look for the greatest improvement. If a better solution is found for some neighborhood, the best solution s^* is updated and the iterations counter is restarted. On the perturbation step, only Swap (1, 1) and Relocate (1) movements are considered. The complete intra route procedure is described in Algorithm 3.

Algorithm 3 Intra Route Search procedure

Input: Initial solution s and maximum number of iterations I_{ILS} without improvement.
Output: A solution s^* for the CTSP-d.

```
1: s* ← s
2: j ← 0
3: while j < I_ILS do
4:     s ← intra_route_local_search(s)
5:     if objective_value(s) < objective_value(s*) then
6:         s* ← s
7:         j ← 0
8:     end if
9:     s ← perturbation(s*)
10:    j ← j + 1
11: end while
```

The main difference between the original GILS-RVND method and the method we are using is that, in our case, we are supposing that we already have an initial solution that must be improved when the procedure starts, and so the GRASP initialization step is unnecessary. Moreover, when the intra-route step does not produce a better solution after I_{ILS} iterations, instead of producing a new starting solution and repeating all the search and perturbation steps, as originally proposed by Silva et al. (2012), in our method we do proceed to an inter-route search step, and what is fully repeated is the complete search loop, including the generation of a new solution and both intra and inter-route steps. This is adopted as a way of spending more time effectively improving the initial solution, instead of keep producing a lot of intermediate different solutions, which could lead to a very time consuming algorithm, and would lead to a method where we do not best explore each solution produced.

Inter-route search step

In the inter-route search step, the goal is to look for better solutions by changing the route to which a vertex belongs (or to which a set of vertices belong). As in the case of the intra-route step, we do so in a local search approach based on a set of neighborhoods, which are used to improve the solutions by a sequential procedure that looks for better solutions locally and then apply small perturbations to the best solution to enlarge the search space.

To move the vertices across the routes we propose the following local search steps:

- **Cross (1, 1):** This is a local movement analogous to Swap (1, 1) in the intra-route case. In Cross (1, 1), a vertex in position i in the route of a vehicle k_1 and a vertex in position j in the route of a different vehicle k_2 are swapped.

- **Cross (2, 2):** Analogous to the Swap (2, 2) neighborhood in the intra-route case, in Cross (2, 2), an arc $(i, i+1)$ in the route of a vehicle k_1 and an arc $(j, j+1)$ in the route of a different vehicle k_2 are swapped.

- **Cross (3, 3):** Similarly to the previous Swap (1, 1) and Swap (2, 2) cases, in the Cross (3, 3) neighborhood, a sequence of three consecutive vertices in positions i, $i+1$ and $i+2$ in the route of a vehicle k_1 is swapped with a sequence of three consecutive vertices in positions j, $j+1$ and $j+2$ in the route of a different vehicle k_2.

- **Path-Relocation (1):** This is a local movement analogous to Relocate (1) in the intra-route case. In Path-Relocation (1), a vertex i in the route of a vehicle

k_1 is relocated to the right position of a vertex j in the route of a different vehicle k_2.

- **Path-Relocation (2):** Analogous to the Relocate (2) neighborhood in the intra-route case, the Path-Relocation (2) is a neighborhood built by relocating an arc $(i,\ i+1)$ in the route of a vehicle k_1 to the right position of a vertex j in the route of a different vehicle k_2.

- **Path-Relocation (3):** Similarly to the Path-Relocation (1) and Path-Relocation (2) cases, we define the Path-Relocation (3) solutions by relocating a sequence of three consecutive vertices in position i, $i+1$ and $i+2$ in the route of a vehicle k_1 to the right position of a vertex j in the route of a different vehicle k_2.

- **2-Opt*:** Let i be the position of a given vertex in the route of a vehicle k_1 and j the position of a given vertex in the route of a different vehicle k_2. In the 2-Opt* movement, the route of vehicle k_1 will be kept for all vertices up to position i, and in the right positions we will put the vertices of k_2, starting with j and all the other vertices after it up to the last one. On the route of k_2, we will keep the vertices up to position $j-1$, and then in the right positions we will have the vertices from k_1 route starting with position $i+1$ and all the next ones up to the last. In other words, in the 2-Opt* movement we swap all the final vertices in k_1 route with all the initial vertices in k_2 route.

Since the inter-route search is a time-consuming procedure, we do not repeat the full search and perturbation sequence for a large number of times before stopping. However, it will be repeated until no better solution is found. In the main search step, we consider the Cross (1, 1), Cross (2, 2), Cross (3, 3) and 2-Opt* neighborhoods and in the perturbation phase the neighborhoods used are Path-Relocation (1), Path-Relocation (2) and Path-Relocation (3). The complete inter-route step is described in Algorithm 4.

4.2.4 The complete heuristic procedure

By assembling the steps described in the previous sub-sections, we obtain the complete heuristic solution method proposed, presented in Algorithm 5. In summary, it shows the three main components of the method: before applying any specific search step, cluster the vertices to define feasible solutions; for a specified quantity of complete search steps Q, produce an initial feasible solution; proceed to the two different kinds of search steps - intra-route and inter-route - and repeat them both a fixed number of times, defined by the parameter L.

Algorithm 4 Inter Route Search procedure

Input: Initial solution s.
Output: A solution s^* for the CluVRP-d.

```
1: s* ← s
2: continue_search ← true
3: while continue_search do
4:     improved_solution ← false
5:     s ← inter_route_local_search(s*, neighborhood)
6:     if objective_value(s) < objective_value(s*) then
7:         s* ← s
8:         improvedSolution ← true
9:     end if
10:    s ← perturbation(s*)
11:    if improvedSolution = false then
12:        continueSearch ← false
13:    end if
14: end while
```

Algorithm 5 Complete heuristic procedure

Input: Parameters Q, L and I_{ILS} for the number of iterations in each step of the method.
Output: A solution s^* for the CluVRP-d.

```
1: cluster_candidates ← K_Means(points, n_clusters)
2: clusters ← clusters_merge(CVRP_d, clusters_candidates)
3: for q = 1 to Q do
4:     s ← generate_initial_solution(CVRP_d, clusters)
5:     for l = 1 to L do
6:         for k = 1 to G_max do
7:             s_{k_v} ← intra_route_search(s, I_ILS)
8:         end for
9:         s ← inter_route_search(s)
10:        if objective_value(s) < objective_value(s*) then
11:            s* ← s
12:        end if
13:    end for
14: end for
```

5 Computational Results for the CTSP-d

All the formulations described in Chapter 3 are based on already known Asymmetric Traveling Salesman Problems formulations, and, according to Öncan et al. (2009), by not considering the d-relaxed priority rule, it is possible to prove that formulation SST1 is stronger than SSB1, while SSB1 is not comparable to GP1 and all the precedence-variable formulations are stronger than MTZ1. On the other hand, the sizes of the models are different, as shown in Table 3.1, and so it is not possible to know a priori which model would perform better when solving benchmark instances. In this chapter, computational results from the different formulations of Chapter 3 are presented and discussed to offer a clearer view about their use and the improvements obtained by adding valid inequalities to each formulation.

5.1 Benchmark instances and experimental settings

The dataset used in this chapter is the one proposed by Hà et al. (2020) and made available by the authors at ORLab webpage: `http://orlab.com.vn`. The data was generated based on selected instances of the TSPLIB (Reinelt, 1991) adding different numbers of priority sets, where the vertices were grouped in two different kinds of instances: random and clustered. In the first type of instances, vertices are distributed into the priority groups in a random way, which is useful to model situations of distribution of commercial products, where the priority may be defined by storage levels, for example. The other kind of instances were built by clustering nearby vertices into priority groups, simulating natural disasters situations like earthquakes or tsunamis, where the level of the impact of the fact is related to its propagation direction.

More specifically, the instances are identified by a five-field notation *-*-*-*-*, where:

- the first field is the name of the original TSPLIB instances. Here the number of vertices is given, which is 42, 52, 100 or 200;
- the second one represents the clustering procedure: C (clustered) or R (random);

- the third one is the number of priority groups which can be 3 or 5, representing instances for the CTSP-d;
- the fourth one represents the d parameter value adopted in the d-relaxed priority rule that, in these computational tests, can be 0, 1, 2 or 3 depending on the instance;
- finally, the last field is the replication of the instances (a, b and c), representing different instances built based on the same TSPLIB instance and the same priority sets. These instances were randomly generated respecting the clustering procedure of the instance (clustered or random), and differ from each other in the way the vertices were assigned to the priority groups.

All the formulations were implemented in the Python programming language and solved using Gurobi Optimizer 9.5. The experiments were performed on a virtual Linux Ubuntu 20.04 machine operating with a Quad-Core 5th generation Intel Processor at 2.6 GHz with 16 GB of RAM. The total Runtime imposed for the smaller instances (described in Section 5.2 and 5.3) was 3600 seconds and, in the case of the bigger instances (described in Section 5.4), it was 18000 seconds.

For the sake of readability, in the next sections we present the computational results through figures based in performance profiles, as proposed by Dolan and Morè (2002), for the runtime and the gap of the solver in the instances, and tables that summarize the results found. In Tables 5.1, 5.2 and 5.3, we present five important aspects of the solutions found by the solver in a summarized display. The "AvObj" row shows the average objective value found by the solver for each formulation in the benchmark instances. This information can be compared among the formulations to illustrate which one reached the smaller values. Similarly, the rows "AvRuntime" and "AvGAP" bring the mean values for these two parameters, indicating which formulation performed better in each criteria for the tested instances. The BKS row brings the number of best known solutions found by the solver in each formulation, including the cases in which it could prove the optimality and the cases in which it was not possible to prove optimality in the given runtime limit. Finally, the exact number of instances in which the solver proved optimality is shown in the "Proved Optimality" row. In the Appendix A.1 we are including tables with detailed results.

The performance profile proposed by Dolan and Morè (2002) is a tool for visually evaluating and comparing the performance of optimization software. In this thesis we use this tool to compare the performance of the solver when solving the proposed formulations. The idea is to analyze the ratio of the computational solution time (and the GAP) of a specific formulation versus the best computational time (GAP) of all

of the formulations as the performance metric. More specifically, denoting by t_{fb} the computational time required to solve (or the GAP obtained after solving) instance $b \in B$ by formulation $f \in F$, we can compare a set of formulations F over a set of benchmark instances B using the performance ratio, defined by

$$r_{fb} = \frac{t_{fb}}{min\{t_{fb} : f \in F\}}. \tag{5.1}$$

Using the performance ratio, and representing as $P_f(\tau)$ the fraction of instances that formulation f can be solved with at most τ many times the computational time (or the GAP) of the best formulation, the performance function of a formulation f is computed by

$$P_f(\tau) = \frac{|\{b \in B : r_{fb} \leqslant \tau\}|}{|B|}. \tag{5.2}$$

This result can be interpreted as follows: considering specifically the computational time, $P_f(1)$ is the number of instances that the computational time to solve formulation f is better than, or as good as the other formulations in F. Therefore in the figures representing the performance profiles, the y-axis is presented by $P_f(\tau)$, whereas, the x-axis is represented by τ. In this way, an (x, y) point of (3, 0.8) means that the examined solver successfully solved 80% of the instances 3 times as long as the fastest formulation. Then, the formulation with the best computational time would be found close to the top part of the plot.

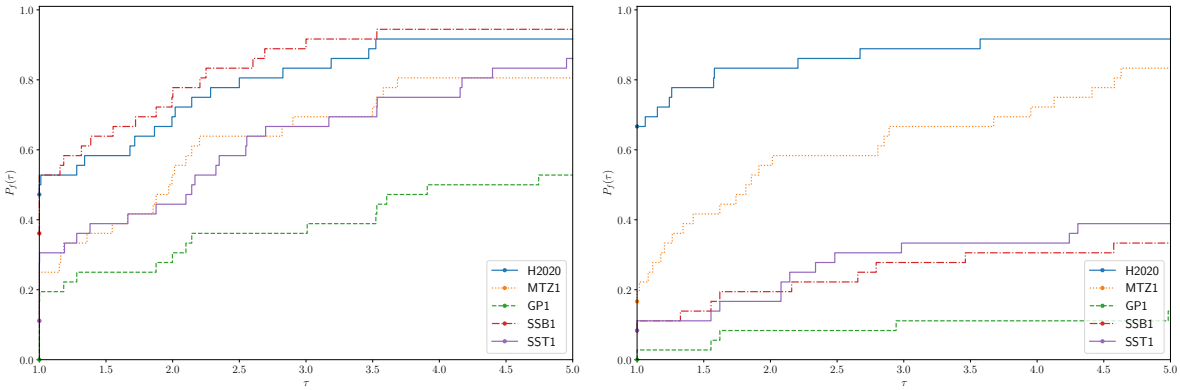
5.2 Comparison of the results from the formulations H2020, MTZ1, GP1, SSB1 and SST1 for smaller instances

The analysis starts by comparing the results obtained for the H2020 formulation (proposed by Hà et al. (2020)) and the four formulations proposed without any valid inequalities, whose results of the computational experiments, representing the random and the clustered instances, are presented in Figures 5.1 and 5.2, as well as in Tables 5.1, A.1 and A.2.

The performance profiles shown in Figures 5.1 and 5.2 offer an overview of the runtime and GAP performance across the formulations, respectively. In the first one we see that the solver presented the best runtime performance for the randomly generated instances with the SSB1 formulation, where the probability that SSB1 is the winner on a given instance is higher than the other formulation for almost all the factors τ. For the clustered instances the best runtime performance was reached for the H2020

formulation with the probability to be the winner on a given instance ($\tau = 1$) of about 65%. It worth to notice that in the clustered case the result for formulation SSB1 were not competitive with H2020, and the second best profile was in the MTZ1 case. The results for the GAP criteria, according to Figure 5.2, show that the SSB1 formulation obtained the best results for most part of the τ values, with H2020 performing with very similar results. For the clustered instances, the best performance was reached by the solver running the H2020 formulation, followed by SSB1 and MTZ1.

When comparing the results in Table 5.1 we can get some more insights about the solver performance. The average objective values reached by Gurobi were very similar for all but SST1 formulation, in both random and clustered instances cases, and we see that this worst performance running SST1 was specially large for $d = 3$ in the random instances case, and for $d = 2$ and $d = 3$ for the clustered case. On the other hand, we see that the best mean runtime was found in the SSB1 case for the random instances and in the H2020 case for the clustered instances. This results are similar to the conclusions drawn from Figure 5.2, showing that, in fact, the solver performance on each formulation depends on the dispersion of the points in the instance.



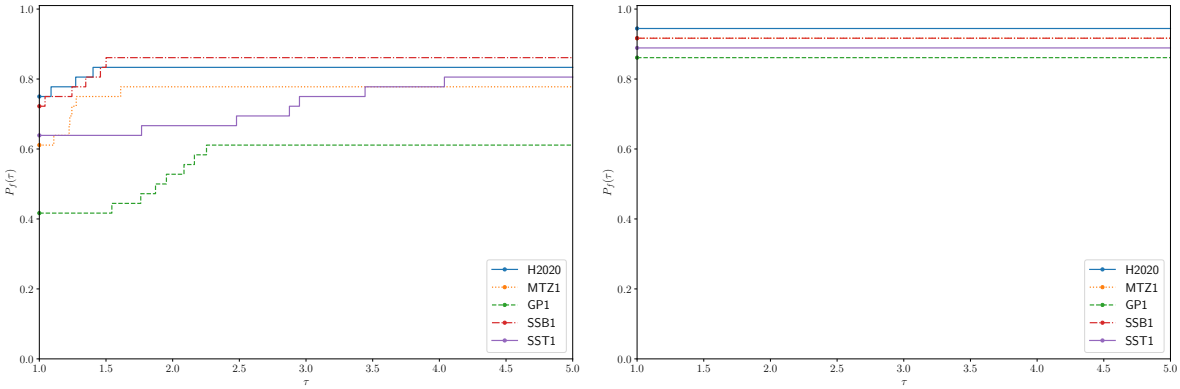
(a) Random Instances (b) Clustered Instances

Figure 5.1: H2020, MTZ1, GP1, SSB1 and SST1 runtime performance profile for smaller instances

Three important measures to evaluate the quality of the solutions found by the solver are the GAP, the number of instances it reached the best known solution (BKS) and the quantity of instances it proved optimality for each formulation. These values can help to analyze not only the performance of the solver in reaching optimality, but also can give an overall information about how it performed when it did not proved optimality. In Table 5.1 we see that for the random case SSB1 had the best results for all average GAP, BKS and number of instances with optimality proved criteria, tied with H2020 in the BKS case. On the clustered instances case, formulations H2020, MTZ1 and SSB1 performed very well, but with H2020 proving optimality for the greatest

		Random				Clustered			
		All	$d=0$	$d=1$	$d=2$	$d=3$	All	$d=0$	$d=1$	$d=2$	$d=3$
AvObj	H2020	6437.03	8027.92	5959.5	5643.17	5004.17	4819.33	5162.25	4753.25	4585.17	4499.83
	MTZ1	6436.50	8027.83	5955.92	5658.83	4992.67	4819.33	5162.25	4753.25	4585.17	4499.83
	GP1	6466.19	8027.92	5987.75	5742.83	5023.00	4822.17	5162.25	4753.25	4602.17	4499.83
	SSB1	6437.36	8027.92	5959.83	5643.17	5005.50	4819.33	5162.25	4753.25	4585.17	4499.83
	SST1	7653.44	8027.92	6188.17	6282.00	11206.50	5986.47	5162.25	5070.83	8085.50	7367.17
AvRuntime	H2020	1694.28	4.16	2336.76	2628.11	2855.71	343.29	328.56	7.71	785.79	601.44
	MTZ1	2006.38	5.72	2815.65	3082.72	3312.85	415.17	356.10	86.74	1001.87	603.52
	GP1	2235.99	16.11	3091.45	3600.29	3600.52	663.59	756.30	259.88	1282.04	667.18
	SSB1	1568.35	4.66	1838.07	2284.13	3440.51	534.61	29.38	422.92	1346.40	956.66
	SST1	1745.11	7.15	1803.25	3249.82	3600.02	888.55	94.17	974.97	1823.68	1369.35
AvGAP	H2020	1.81	0.00	2.46	3.42	2.50	0.18	0.48	0.00	0.00	0.10
	MTZ1	2.15	0.00	3.05	3.87	2.91	0.19	0.47	0.00	0.14	0.07
	GP1	3.52	0.00	4.48	6.32	5.83	0.44	0.65	0.00	0.96	0.35
	SSB1	1.40	0.00	1.23	2.53	3.43	0.18	0.00	0.00	0.65	0.40
	SST1	8.19	0.00	2.53	7.14	36.93	6.62	0.00	2.72	17.64	16.67
BKS	H2020	31	12	9	5	5	36	12	12	6	6
	MTZ1	30	12	10	5	3	36	12	12	6	6
	GP1	25	12	7	3	3	34	12	12	4	6
	SSB1	31	12	9	5	5	36	12	12	6	6
	SST1	23	12	9	2	0	32	12	11	4	5
Proved Optimality	H2020	23	12	6	3	2	34	11	12	6	5
	MTZ1	21	12	5	2	2	33	11	12	5	5
	GP1	16	12	4	0	0	31	10	12	4	5
	SSB1	25	12	8	3	2	33	12	12	4	5
	SST1	23	12	9	2	0	32	12	11	4	5

Table 5.1: Average values of Objective, Runtime and GAP, and number of Best Known Solutions (BKS) found by the solver for the smaller instances running formulations H2020, MTZ1, GP1, SSB1 and SST1



(a) Random Instances (b) Clustered Instances

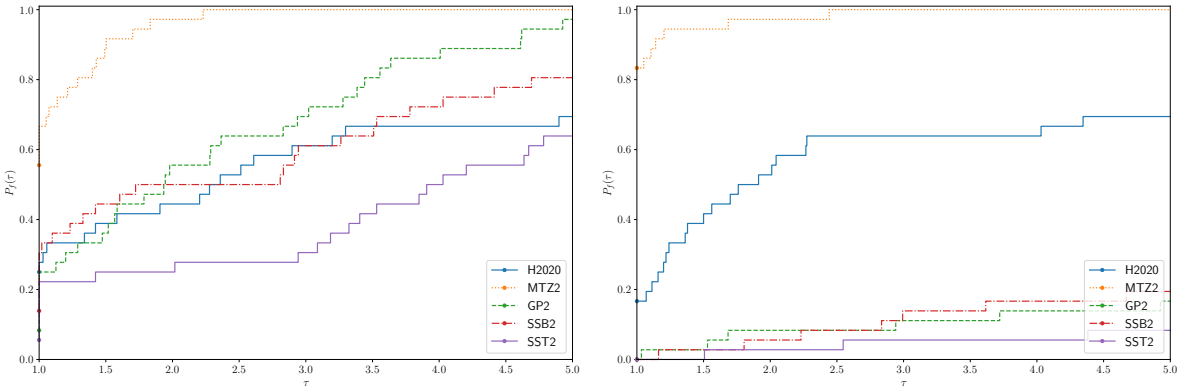
Figure 5.2: H2020, MTZ1, GP1, SSB1 and SST1 GAP performance profile for smaller instances

number of instances. Another aspect we see in Table 5.1 is that instances with $d = 0$ were easier for the solver to prove optimality with all the formulations, and in both random and clustered cases. Instances with $d = 2$ and $d = 3$ were harder for the solver to reach and prove optimality, mainly in the random instances cases.

In summary, in the random instances the solver had the best performances running SSB1 and H2020, with SSB1 being a slightly better option while in the clustered cases, besides the fact Gurobi performed very well with H2020, MTZ1 and SSB1, the best results were found running H2020, with which it had the lowest average runtime, reached the best average GAP value and proved optimality for most instances. The worst values of average runtime and number of solutions where optimality was proved were reached by the solver with GP1 formulation for the random instances. In the next Section we analyze how adding valid inequalities boosted the performance of the solver, with the results found for GP2 being better than the results found with GP1. The average GAP running SST1 were the worst in both random and clustered instances cases because, in practice, SST1 was an unstable option, where the solver sometimes performed much better than all the others formulations while, at the same time, it performed very poorly in other instances, mainly in the ones where $d = 2$ and $d = 3$. The results reached for each instance can be seen in the Appendix A.1.

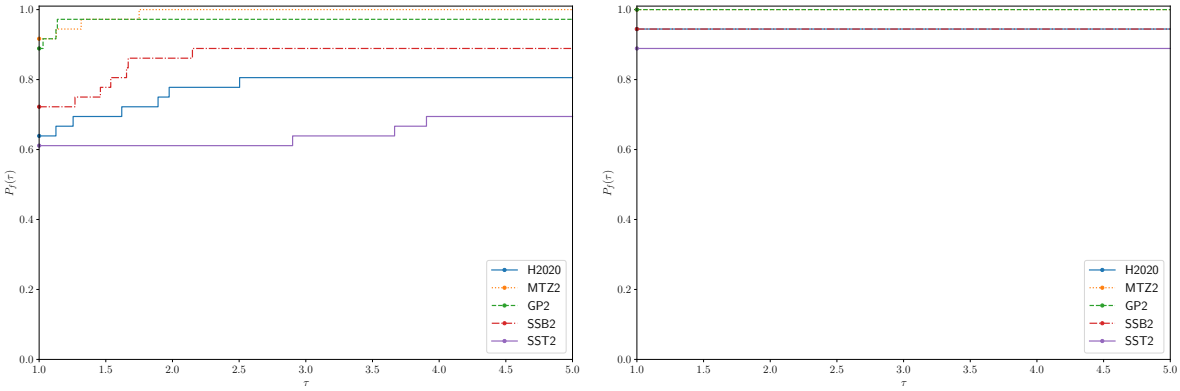
5.3 Comparison of the results from the formulations H2020, MTZ2, GP2, SSB2 and SST2 for smaller instances

As discussed in Chapter 3, we proposed formulations MTZ2, GP2, SSB2 and SSB2 after adding valid inequalities to formulations MTZ1, GP1, SSB1 and SSB1, respectively. In this Section we analyze how the changes in the models impacted the computational results and compare the four proposals with valid inequalities and the literature formulation H2020. Detailed resuls can be found in the Appendix A.1 (Tables A.3 and A.4).



(a) Random Instances (b) Clustered Instances

Figure 5.3: H2020, MTZ2, GP2, SSB2 and SST2 runtime performance profile for smaller instances



(a) Random Instances (b) Clustered Instances

Figure 5.4: H2020, MTZ2, GP2, SSB2 and SST2 GAP performance profile for smaller instances

In Figures 5.3 and 5.4 we see the performance profiles for the runtime and GAP, respectively, for formulations H2020, MTZ2, GP2, SSB2 and SST2. Different of the

		Random					Clustered				
		All	$d=0$	$d=1$	$d=2$	$d=3$	All	$d=0$	$d=1$	$d=2$	$d=3$
AvObj	H2020	6437.03	8027.92	5959.50	5643.17	5004.17	4819.33	5162.25	4753.25	4585.17	4499.83
	MTZ2	6438.50	8027.92	5952.50	5671.83	4998.33	4819.33	5162.25	4753.25	4585.17	4499.83
	GP2	6429.19	8027.92	5952.33	5639.67	4975.00	4819.33	5162.25	4753.25	4585.17	4499.83
	SSB2	6439.75	8027.92	5952.33	5656.50	5021.50	4823.61	5162.25	4753.25	4585.17	4525.50
	SST2	7708.86	8027.92	6162.08	6501.33	11371.83	6215.08	5162.25	4879.33	9807.00	7400.33
AvRuntime	H2020	1694.28	4.16	2336.76	2628.11	2855.71	343.29	328.56	7.71	785.79	601.44
	MTZ2	919.05	2.41	895.84	1432.08	2285.75	19.94	1.08	3.85	20.48	89.28
	GP2	1169.83	4.98	1106.08	1822.63	2974.25	184.60	8.16	71.15	459.30	489.65
	SSB2	1545.57	4.27	1774.81	2261.73	3453.52	341.11	10.60	158.70	1012.87	695.21
	SST2	1804.68	6.98	2142.14	2929.77	3600.03	947.98	129.48	988.43	1847.35	1604.71
AvGAP	H2020	1.81	0.00	2.46	3.42	2.50	0.18	0.48	0.00	0.00	0.10
	MTZ2	0.77	0.00	0.14	2.30	2.03	0.0	0.00	0.00	0.00	0.00
	GP2	0.71	0.00	0.12	1.95	2.08	0.0	0.00	0.00	0.00	0.00
	SSB2	1.27	0.00	0.69	2.46	3.78	0.17	0.00	0.00	0.22	0.78
	SST2	8.42	0.00	2.20	10.25	35.90	6.91	0.00	1.36	22.04	16.67
BKS	H2020	31	12	9	5	5	36	12	12	6	6
	MTZ2	32	12	11	4	5	36	12	12	6	6
	GP2	35	12	12	6	5	36	12	12	6	6
	SSB2	33	12	12	4	5	35	12	12	6	5
	SST2	22	12	8	2	0	32	12	11	4	5
Proved Optimality	H2020	23	12	6	3	2	34	11	12	6	5
	MTZ2	30	12	11	4	3	36	12	12	6	6
	GP2	30	12	11	4	3	36	12	12	6	6
	SSB2	26	12	8	4	2	34	12	12	5	5
	SST2	22	12	8	2	0	32	12	11	4	5

Table 5.2: Average values of Objective, Runtime and GAP, and number of Best Known Solutions (BKS) found by the solver for the smaller instances running formulations H2020, MTZ2, GP2, SSB2 and SST2

cases presented in Figure 5.1, where the best formulation for the random instances was different from the best formulation for the clustered case, in Figure 5.3 the best runtime results was reached by the solver running MTZ2 in both cases, and for all τ values. For the GAP criteria, we see in Figure 5.4 that MTZ2 and GP2 performed in a very similar manner in the random instances, and both reached zero GAP in all clustered instances.

Table 5.2 allows us to develop a deeper analysis about the solver results. At first we see the addition of valid inequalities did not impact the average objective value in a relevant manner, since the results in Tables 5.1 and 5.2 are very similar on this criterion. The formulation SST2 did not performed well in the random instances where $d = 3$, as well as in the clustered instances with $d = 2$ and $d = 3$, the same pattern we saw when analyzing SST1. On the other hand, an important improvement of the use of valid inequalities was on the average runtime. Comparing the results on Tables 5.1 and 5.2 we see the solver reached much better results when running MTZ2 and GP2 in comparison with MTZ1 and GP1, respectively, in both random and clustered instances cases. The most expressive difference was on the clustered instances for the MTZ2 formulation, where the solver took more than 95% less time, on average, to reach the optimal solutions, and in the MTZ2 the solver could prove the optimality for all instances too, fact that did not happened for MTZ1.

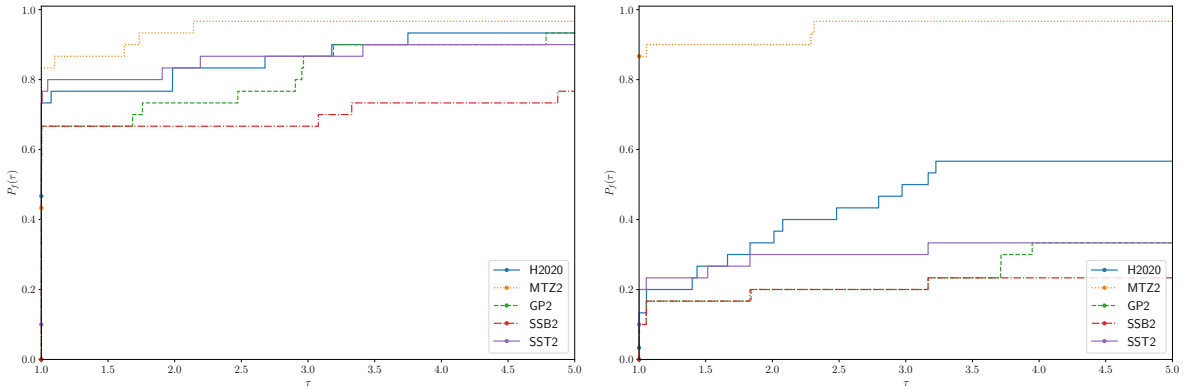
The GAP values for all formulations in Table 5.2 were very low, mainly for MTZ2, GP2 and SSB2 when we consider only the instances with $d = 0$ and $d = 1$. For the cases with $d = 2$ and $d = 3$ the results for all but SST2 were all good, being less than 1% in the clustered instances cases. Finally, the valid inequalities had an important boost in the number of instances the solver could prove optimality or, at least, reached the best known solution, when comparing the results found for MTZ1 and GP1 with the results the solver showed running MTZ2 and GP2, respectively. In fact, the best results for these criteria was reached by Gurobi for MTZ2 and GP2, proving optimality for all clustered instances, and for 30 out of 36 instances for the random cases. It is interesting to notice that the valid inequalities did not lead to better results in the SST2 formulation case, and the number of instances the solver proved optimality in this case decreased from 23 in the SST1 to 22 in the SST2 case.

Since MTZ1 and constraints (3.9)-(3.12) are based on the models proposed in Hà et al. (2020), our final comparison is between the results obtained using the best formulation proposed in that paper and the ones presented in Tables 5.1 and 5.2. MTZ1 is equivalent to the (F2) formulation described in Hà et al. (2020) and comparing the computational results here with the results those authors described and considering only the instance used in both papers, the average values of Objective, Runtime and

GAP are very similar. On the other hand, comparing the computational results here for MTZ2 and H2020, there are visible differences: the average runtime find for MTZ2 for the random instances is nearly 46% smaller than that from H2020 case, and the GAP difference is nearly 57%. For the clustered instances, the solver performed 94% faster on average running MTZ2, while with H2020 it could not solve all the instances and the average GAP obtained was 0.18.

5.4 Comparison of the results from the formulations H2020, MTZ2, GP2, SSB2 and SST2 for large instances

All the four stronger formulations - MTZ2, GP2, SSB2 and SST2 - as well as the literature formulation H2020, were applied to the bigger instances including 100 and 200 cities. The total Runtime limit for each instance was set to five hours and the results obtained for these cases are presented in Figures 5.5 and 5.6 and Tables 5.3, A.5 and A.6 for the cases with 100 cities and Tables 5.4 and 5.5 for the instances with 200 cities. In the cases presented in this Section, a new notation was added to represent the lack of feasible solutions: table entries in which there is a numeric value to the Runtime column, but a "-" in the Obj and GAP columns, represent cases where the solver could start the solution procedure but it did not find any solution. In these cases, the average values are calculated using only the existing values, that is, the average Runtime considers all the table entries, but the Obj and GAP averages are obtained considering only the instances in which the solver could find a feasible solution.

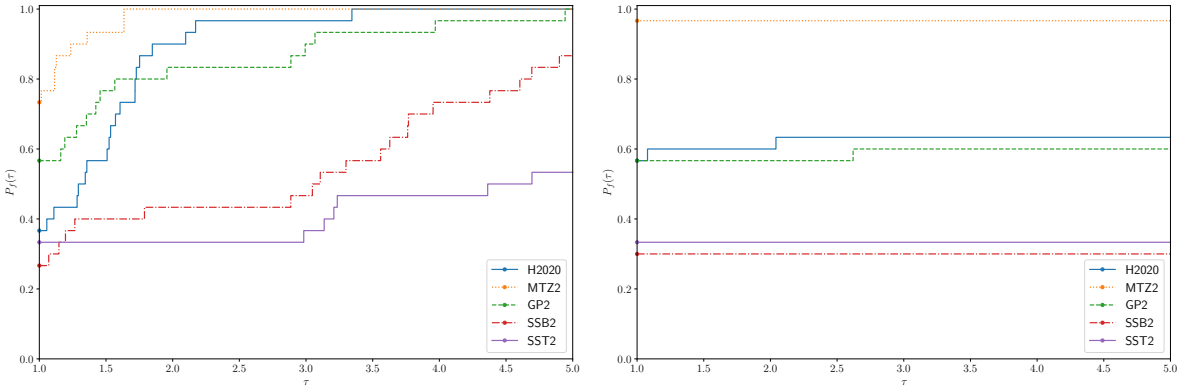


(a) Random Instances (b) Clustered Instances

Figure 5.5: H2020, MTZ2, GP2, SSB2 and SST2 runtime performance profile for 100 vertices instances

		Random				Clustered			
		All	$d=0$	$d=1$	$d=2$	$d=3$	All	$d=0$	$d=1$	$d=2$	$d=3$
AvObj	H2020	36329.23	44487.3	33632.9	32855.4	28879.6	23175.3	24690.2	22733.8	22391.6	21812.2
	MTZ2	35439.03	44487.3	33324.7	29800.8	27209.4	23166.1	24671.5	22724.9	22391.6	21812.2
	GP2	38606.47	44487.3	33598.2	30769.6	44698.2	25326.63	24671.5	23804.7	22419.0	38588.4
	SSB2	44251.93	44510.5	41451.3	46088.0	47500.0	27276.07	24671.5	26312.8	26142.4	35545.4
	SST2	98776.47	44487.3	88137.8	162517.0	164891.6	57701.97	24914.3	44177.1	66096.6	141932.4
AvRuntime	H2020	12582.05	1744.71	18000.18	18001.45	18001.05	9481.52	7414.88	9256.88	8121.07	15424.52
	MTZ2	12482.78	1446.22	18000.76	18000.78	18001.97	4023.99	508.25	4406.23	5047.95	9267.00
	GP2	12847.74	2524.47	18002.59	18015.94	18016.37	10911.93	2194.36	13589.45	15836.25	18067.72
	SSB2	14570.79	7700.33	18010.42	18000.15	18003.10	13569.52	5996.67	16708.22	18006.08	18001.26
	SST2	12501.66	1487.96	18015.66	18000.67	18002.03	14027.41	6573.48	17502.13	18001.13	18012.09
AvGAP	H2020	12.69	0.00	15.94	24.13	20.14	1.33	1.62	1.36	0.57	1.45
	MTZ2	8.85	0.00	13.56	12.01	13.99	0.29	0.00	0.52	0.36	0.34
	GP2	14.85	0.00	13.12	14.86	48.00	8.63	0.00	4.76	1.67	40.60
	SSB2	26.32	0.21	29.92	45.13	52.51	14.93	0.22	15.67	16.66	41.13
	SST2	48.47	0.00	53.93	82.96	100.00	38.59	1.03	33.15	63.19	100.00
BKS	H2020	10	10	0	0	0	27	8	9	5	5
	MTZ2	10	10	0	0	0	30	10	10	5	5
	GP2	10	10	0	0	0	20	10	6	4	0
	SSB2	8	8	0	0	0	12	10	2	0	0
	SST2	10	10	0	0	0	10	8	2	0	0
Proved Optimality	H2020	10	10	0	0	0	17	7	6	3	1
	MTZ2	10	10	0	0	0	26	10	8	4	4
	GP2	10	10	0	0	0	17	10	5	2	0
	SSB2	8	8	0	0	0	9	8	1	0	0
	SST2	10	10	0	0	0	10	8	2	0	0

Table 5.3: Average values of Objective, Runtime and GAP, and number of Best Known Solutions (BKS) found by the solver for the 100 areas instances running formulations H2020, MTZ2, GP2, SSB2 and SST2



(a) Random Instances (b) Clustered Instances

Figure 5.6: H2020, MTZ2, GP2, SSB2 and SST2 GAP performance profile for 100 vertices instances

Figure 5.5 shows that, for the clustered instances, the solver clearly presented the best runtime performance with MTZ2 formulation with a probability to be the winner on a given instance higher than 85% for all the factors τ. Considering the random instances, the solver also presented the best runtime performance with MTZ2 but, given the difficulties on solving these instances, the performance of the solver applied to the other formulations are not so distant from MTZ2, which can be confirmed on the average runtime presented on Table 5.3. Similarly, in Figure 5.6, the results reached by the solver were clearly better on the clustered instances when running MTZ2, but just slightly superior on the random instances, being similar to the results found for GP2 or H2020.

Analyzing more detailed results for the random instances, Table 5.3 shows that the cases where $d = 0$ were the easiest ones for all the formulations and the solver reached and proved optimality in all instances for all but the SSB2 formulations. Gurobi did not perform well running the SST2 formulation in the instances where $d > 0$ but it lead to the second best performance on average runtime where $d = 0$. Considering the cases where $d = 1$, running H2020, MTZ2 and GP2, the solver produced good feasible solutions for all these instances and the average GAP values were 15.94%, 13.56% and 13.12%, respectively, which may still be acceptable for some practical purposes. For the SSB2 and SST2 formulations, on the other hand, instances with more priority sets were easier to solve: Gurobi produced good solutions with SSB2 when the number of priority sets was equal to 5 and $d = 1$, with GAP values similar to those found using H2020, MTZ2 and GP2. However, in the instances with only 3 priority sets, both SSB2 and SST2 led to poor results. In those cases where $d = 2$, SSB2 and SST2 led to very weak solutions with all the GAPs being higher than 40% for SSB2 and higher than 80% for SST2. Looking at the results for the other three formulations, there is no best

option for all the instances. For those instances where $d = 3$, only H2020 and MTZ2 led to good solutions with Gurobi performing better for one specific instance with H2020 formulation, and in all the other instances the solver had the best performance running MTZ2.

As for the smaller instances, when comparing the 100 cities cases with random and clustered instances, the latter were easier for the solver, with smaller average GAP values for all the formulations and a much higher number of instances with optimality proved. Like the random case, instances with $d = 0$ were easier for the solver and optimality was reached and proved for all these instances using MTZ2 and GP2. With all the other formulations, good solutions were found for all the instances. In particular, in the SSB2 case, the solver found the optimal solutions in all the instances but it did not prove optimality in two of them, and in both cases H2020 and SST2, it did not find the optimal solution for only two instances. The clustered instances with $d = 1$ also showed results similar to the random instances, with the cases with three priority sets being harder for the solver when running SSB2 and SST2 formulations while, in the cases with five priority sets, it performed well with all formulations. In the instances where $d = 2$ and $d = 3$, the solver performed well for all instances only when running formulations H2020 and MTZ2, with MTZ2 showing the best results in GAP and total runtime in all the tests.

For the 200-city instances, only H2020 and MTZ2 could be tested due to lack of memory to run the tests on higher dimension models and the results for these instances are shown in Tables 5.4 and 5.5. For the random instances, the solver could not prove optimality with none of the formulations and so the average times for both are approximately the full five-hour runtime limit. On the other hand, the average objective values and GAPs are different, with MTZ2 leading to better results in both. Table 5.4 shows that the solver did not perform well running H2020 on the instances with $d > 0$, producing no feasible solution in 4 out of the 12 instances and a feasible solution with a considerably higher value than in the MTZ2 case in the instances where $d = 3$. Where $d = 0$, Gurobi produced good results with H2020, leading to better solutions than MTZ2 for the kroA200-R-5-0-a and kroB200-R-5-0-a instances. Running the MTZ2 formulation it found good feasible solutions for all the random instances, with a maximum GAP of 21.46%.

Table 5.5 shows that the clustered instances were easier to solve when using H2020 and MTZ2. The solver reached optimality in both cases for the kroA200-C-5-0-a and kroB200-C-5-0-a instances. In particular, with MTZ2 it proved the optimality of the solution for the kroA200-C-5-0-a instance in about 45 minutes, which was very fast when compared with the other instances. The average values of the objective value

and GAP were similar for the two formulations, with MTZ2 leading to a few better results. With each formulation Gurobi found at least one best-known solution from the literature in those cases where it could not prove optimality with none of the models. In this sense, both formulations performed well for clustered instances but, in general, MTZ2 was a better option.

Table 5.4: Solver results from the H2020 and MTZ2 formulations for the 200-city random instances

	(H2020)			(MTZ2)		
Instance	Obj	Runtime	GAP	Obj	Runtime	GAP
kroA200-R-3-0-a	52082	18000.10	2.11	51708	18000.06	1.19
kroA200-R-3-1-a	40439	18000.05	18.63	40136	18000.08	15.80
kroA200-R-5-0-a	67079	18000.06	1.06	67831	18000.05	2.50
kroA200-R-5-1-a	-	18000.01	-	50254	18000.65	17.95
kroA200-R-5-2-a	-	18000.85	-	41182	18001.90	15.50
kroA200-R-5-3-a	68176	18000.16	57.17	36452	18003.10	16.80
kroB200-R-3-0-a	53802	18002.16	1.98	53588	18000.06	1.00
kroB200-R-3-1-a	45416	18000.04	25.79	40249	18000.07	14.80
kroB200-R-5-0-a	73616	18000.07	3.30	76765	18000.12	7.65
kroB200-R-5-1-a	-	18000.01	-	54287	18000.09	21.46
kroB200-R-5-2-a	-	18000.17	-	41148	18000.37	15.28
kroB200-R-5-3-a	45947	18000.38	34.78	39066	18000.05	20.14
Mean Values	55819.62	18000.34	18.10	49388.83	18000.55	12.51

Table 5.5: Solver results from the H2020 and MTZ2 formulations for the 200-city clustered instances

	(H2020)			(MTZ2)		
Instance	Obj	Runtime	GAP	Obj	Runtime	GAP
kroA200-C-3-0-a	29913**	18000.63	0.52	30219	18000.11	2.18
kroA200-C-3-1-a	29368**	18000.06	1.51	29970	18001.19	3.29
kroA200-C-5-0-a	32224*	18000.05	0.30	**32224**	2722.67	0.00
kroA200-C-5-1-a	31968	18000.09	4.55	31383	18000.06	2.04
kroA200-C-5-2-a	31408	18000.07	6.60	30863	18000.07	3.27
kroA200-C-5-3-a	30769	18003.63	4.55	30023**	18000.08	0.62
kroB200-C-3-0-a	30989*	18003.99	0.58	**30989**	17923.22	0.00
kroB200-C-3-1-a	30895	18000.04	2.34	30442**	18000.07	0.32
kroB200-C-5-0-a	39011	18000.08	11.96	38216	18001.53	2.75
kroB200-C-5-1-a	34358	18000.07	11.50	33704	18001.52	8.70
kroB200-C-5-2-a	36473	18000.14	19.96	32610	18000.44	8.96
kroB200-C-5-3-a	30317	18000.11	1.74	30103	18000.40	0.73
Mean Values	32307.75	18000.75	5.51	31728.83	16720.95	2.74

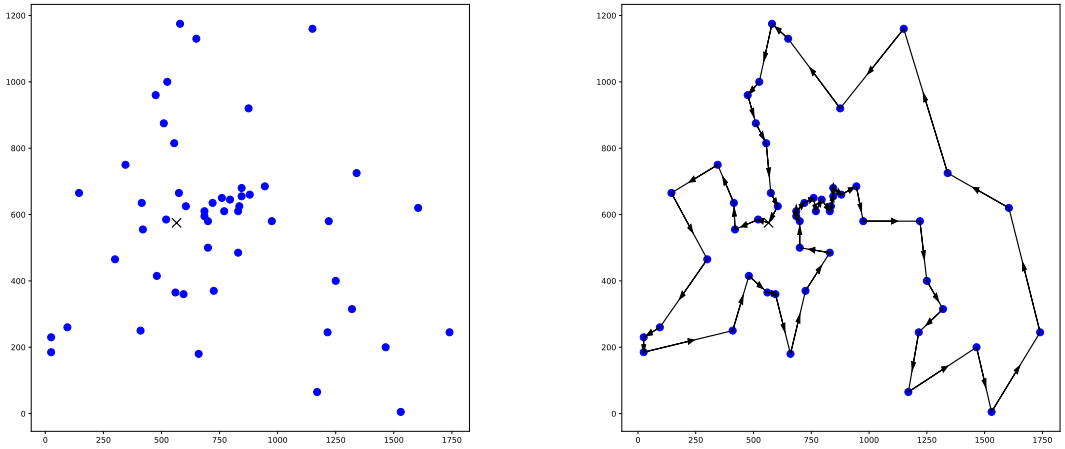
5.5 How the TSP, CTSP-PO and the CTSP-d are related

As discussed previously in this thesis, the CTSP-d can be seen as a generalization of both TSP and CTSP-PO, since the case where $d = 0$ corresponds to the CTSP-PO, and cases where d is big enough it becomes the classic TSP. In this section, the results obtained in the computational tests for some selected instances are compared to see how the different values of d affect the same instance and how the problem changes from the CTSP-PO into the TSP from a practical point of view. We are not aiming to compare the proposed formulations against formulations that are developed specifically to the TSP nor to the CTSP-PO problems.

In the following analysis, we will adopt a generic reference for the value of the d parameter in the instances and so, instead of using berlin52-R-5-0-b, berlin52-R-5-1-b, and so on, we will simply write berlin52-R-5-d-b. The same applies to all other instances described in this section. Since the V_p will be the same when we only change the value of d, this general notation allows a clearer description of the data.

All the instances described in this section are those solved and discussed in the earlier sections, based on instances from the TSPLIB. Instances which are described via points coordinates in the TSPLIB can be obtained from the original dataset and images representing the tours produced after solving the CTSP-d formulations, which

include the TSP and the CTSP-PO as particular cases. Figures 5.7 and 5.8 show solutions based on the berlin52 instance and all the solutions have the same starting vertex, denoted by a black cross. The coordinates of the points of the berlin52 instance and the TSP solution are shown in Figure 5.7.



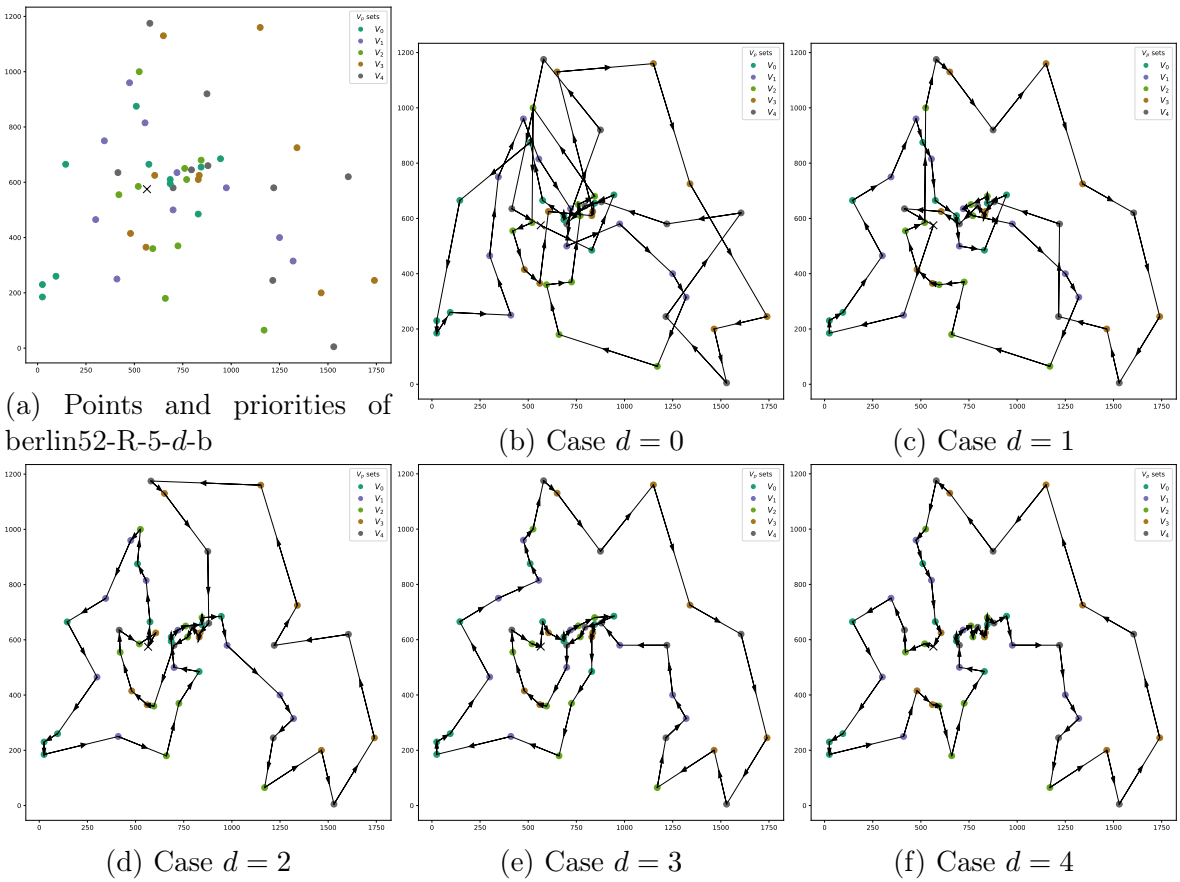
(a) Instance: berlin52 (b) Optimal route

Figure 5.7: The berlin52 TSP instance and its solution

As expected from the CTSP-d description and seen in the computational results, when the number of priority sets V_p in an instance of the CTSP-d is fixed, as the value of the d parameter increases, the value of the optimal solution tends to decrease or stay the same. In other words, as the relaxation of the priorities increases, the problem becomes increasingly similar to the TSP and the best possible tour becomes smaller. This is shown in Figure 5.8, where the solutions for fixed V_p sets are analyzed as the value of d increases. Figure 5.9 shows the effect of varying d on the total optimal tour length for some selected instances.

Figure 5.8 shows the case where $d = 0$, when the problem is equivalent to the CTSP-PO, the optimal solution involves a much longer tour when compared to the TSP solution, with a greater number of crossing arcs, since the visits to the vertices must obey the priority levels in a strict manner. The case where $d = 1$ has an optimal solution that is clearly different from the TSP but introducing some level of priority relaxation led to a shorter optimal tour without ignoring the existence of visit priorities among the vertices by just adjusting the value of d instead of redefining all the V_p sets of the problem and trying to fit the relaxed priorities into a new instance of the CTSP-PO. It is interesting that building new formulations of CTSP-PO with different priority sets would lead to a different approach, harder to apply and stricter in the results. Where $d = 2$ the optimal tour is even shorter than for the cases where $d = 0$ and $d = 1$ and the number of crossing arcs decreases. The tour becomes visually different from the previous cases and becomes even more similar to the TSP solution. The same

remarks apply to the case where $d = 3$ and, finally, where $d = 4$, the problem becomes equivalent to the TSP, since the priorities are not taken into account anymore.

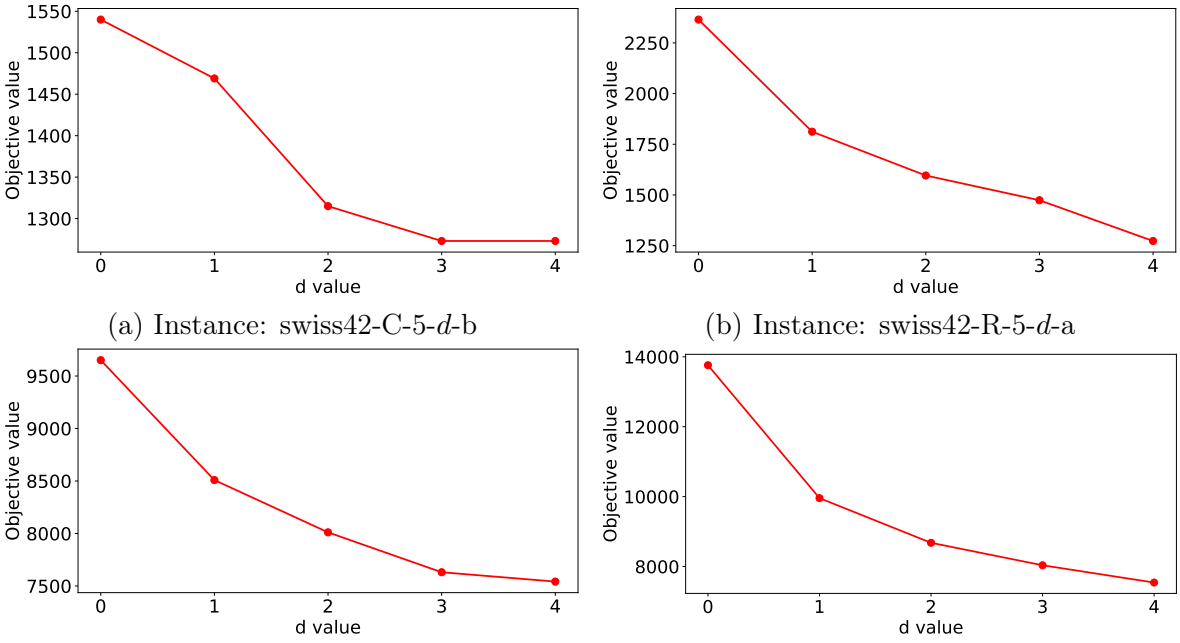


(a) Points and priorities of berlin52-R-5-d-b (b) Case $d = 0$ (c) Case $d = 1$

(d) Case $d = 2$ (e) Case $d = 3$ (f) Case $d = 4$

Figure 5.8: Optimal tours obtained for different values of d for the berlin52-R-5-d-b instances

Figure 5.9 shows the comparison of the total tour length for the swiss42-C-5-d-b, swiss42-R-5-d-a, berlin52-C-5-d-c and berlin52-R-5-d-b instances for d ranging from 0 to 4. It shows that there is no absolute pattern that occurs in all CTSP-d instances and this is a consequence of the main difference between this problem and the TSP, that is the existence of priority sets but which nodes will be included in each set is a situation-dependent question. The greatest slope in sub-figure (a) occurs between values $d = 1$ and $d = 2$, while in the other sub-figures it happens between values $d = 0$ and $d = 1$. Another specific aspect of sub-figure (a) is that the optimal value found for the cases $d = 3$ and $d = 4$ are the same and so the line plot in this range is a horizontal line. In the case of the sub-figures (c) and (d), the slope of the line between two consecutive values of d is always decreasing as the values of d increase but this does not happen in sub-figures (a) and (b). On the other hand, this decreasing in the total tour length is related to the relaxation of the priorities and so, in a practical decision-making situation, this analysis of how the d value impacts the tour length must be related to how it impacts the treatment of priorities.



(a) Instance: swiss42-C-5-d-b

(b) Instance: swiss42-R-5-d-a

(c) Instance: berlin52-C-5-d-c

(d) Instance: berlin52-R-5-d-b

Figure 5.9: Line plots showing how the optimal tour length decreases as the value of d increases for some selected instances

The second parameter to be analyzed when comparing the impact of varying the d parameter on the instances is how the demands are fulfilled during the traveling salesman tour. This is important to measure the trade-off between minimizing the total tour length and the delays produced over the initial priorities set to each vertex in the problem. Table 5.6 summarizes the results for each one of the instances compared in this section. The columns of the Table represent the priority levels $p \in P$ of the sets in the instances, and for each fixed column, each row presents four pieces of information about the tour built in the format [* (*), * (*)], where:

- the first one represents the first position after the origin in which a vertex with the priority level p appears;
- the second one is the total accumulated distance traveled in the tour until the first vertex with priority p appears;
- the third one represents the last position after the origin containing a vertex with priority level p;
- the fourth one is the total accumulated distance of the tour since the origin up to the last vertex with priority level p.

Since, in all instances, we are considering the optimal solution, the instances with $d = 0$ represent the best scenarios for the priorities fulfillment and so the first vertex

Table 5.6: Fulfilling priorities in the selected instances

Instance	Priority 0	Priority 1	Priority 2	Priority 3	Priority 4
swiss42-C-5-0-b	[1 (90), 6 (236)]	[7 (273), 11 (387)]	[12 (408), 27 (756)]	[28 (792), 35 (1079)]	[36 (1164), 41 (1399)]
swiss42-C-5-1-b	[1 (92), 6 (238)]	[23 (635), 27 (756)]	[7 (263), 22 (614)]	[34 (1110), 41 (1397)]	[28 (806), 33 (1025)]
swiss42-C-5-2-b	[12 (277), 18 (434)]	[3 (104), 22 (548)]	[1 (37), 41 (1300)]	[29 (898), 36 (1185)]	[23 (612), 28 (813)]
swiss42-C-5-3-b	[15 (517), 22 (681)]	[23 (710), 34 (1148)]	[1 (37), 41 (1258)]	[2 (86), 10 (416)]	[27 (841), 32 (1060)]
swiss42-R-5-0-a	[1 (90), 9 (525)]	[10 (549), 17 (971)]	[18 (1022), 25 (1360)]	[26 (1477), 33 (1892)]	[34 (1973), 41 (2328)]
swiss42-R-5-1-a	[1 (43), 15 (670)]	[2 (94), 17 (753)]	[20 (972), 29 (1328)]	[18 (816), 39 (1717)]	[30 (1393), 41 (1775)]
swiss42-R-5-2-a	[4 (93), 20 (813)]	[1 (15), 24 (1019)]	[2 (34), 39 (1544)]	[21 (894), 40 (1562)]	[25 (1059), 41 (1573)]
swiss42-R-5-3-a	[4 (93), 16 (578)]	[1 (15), 40 (1397)]	[2 (34), 31 (1072)]	[5 (165), 38 (1316)]	[17 (617), 41 (1437)]
berlin52-C-5-0-c	[1 (46), 13 (2030)]	[14 (2522), 17 (3037)]	[18 (3185), 40 (6908)]	[41 (7767), 45 (8328)]	[46 (8660), 51 (9411)]
berlin52-C-5-1-c	[1 (46), 13 (1690)]	[26 (4487), 29 (5002)]	[14 (1876), 45 (7271)]	[30 (5453), 34 (6041)]	[46 (7772), 51 (8269)]
berlin52-C-5-2-c	[1 (46), 25 (4138)]	[2 (143), 33 (5658)]	[8 (521), 51 (7845)]	[26 (4528), 30 (5089)]	[34 (5878), 39 (6375)]
berlin52-C-5-3-c	[1 (64), 37 (4211)]	[4 (342), 51 (7540)]	[17 (2604), 44 (6584)]	[3 (262), 10 (1358)]	[45 (6892), 50 (7389)]
berlin52-R-5-0-b	[1 (280), 11 (2164)]	[12 (2479), 21 (4528)]	[22 (4820), 31 (7011)]	[32 (7163), 41 (10180)]	[42 (10386), 51 (13597)]
berlin52-R-5-1-b	[2 (750), 17 (2896)]	[1 (360), 24 (3756)]	[18 (2921), 33 (5702)]	[29 (4937), 50 (9605)]	[34 (5885), 51 (9795)]
berlin52-R-5-2-b	[1 (91), 23 (3180)]	[2 (242), 26 (3728)]	[4 (443), 50 (8521)]	[29 (4364), 51 (8615)]	[27 (3854), 49 (8405)]
berlin52-R-5-3-b	[1 (91), 23 (2977)]	[5 (284), 45 (7338)]	[6 (327), 51 (7990)]	[2 (141), 48 (7642)]	[26 (3316), 50 (7874)]

visited will always belong to set V_0 and the first vertex in V_1 will be visited exactly after the last one in V_0, and so on. The impacts of bigger values of d may be different, depending on the positions of the vertices.

Taking the instances with $d = 1$, a clear priorities clustered pattern is seen in the tour for the swiss42-C-5-1-b instance: in the optimal solution, the traveling salesman starts by visiting all the vertices with priority 0, as in the case $d = 0$ but with different accumulated distances, showing that the vertices are visited in a different order. After that, it goes directly to the priority 2 set and visits all of its vertices before starting to visit the vertices with priority 1. At this point of the tour, by the d-relaxed priority rule, the salesman can only visit priority 1 vertices before starting to visit sets with higher priority indexes (V_3 and V_4 in this example). After finishing the visits to priority 1 vertices, the next set is the priority 4 vertices, which are all visited before the priority 3 set, which are left until the end of the tour. For the swiss42-R-5-1-a instance, the pattern observed is not so rigid. The positions in Table 5.6 show that the first vertex visited belongs to the priority 0 set and the second one belongs to the priority 1 vertices. The tour continues by visiting vertices with these two different priority levels alternatively. After finishing the visits to all vertices of both sets, the traveling salesman goes to a vertex with priority 3. The tour is not built priority set by priority set, as in the case of the swiss42-C-5-1-b instance but it is developed visiting different priority orders in an interchangeable manner.

Despite the fact the berlin52-C-5-1-c instance is defined by priority sets clustered by the distance between their vertices, the pattern of its optimal solution it is not strictly clustered as in the case of the swiss42-C-5-1-b instance. The data in Table 5.6 allow us to see that the optimal solution found for the berlin52-C-5-1-c instance follows this order: the traveling salesman visits all vertices with priority 0, part of the vertices with priority 2, all the vertices with priority 1, all the vertices with priority 3, the remainder of the vertices with priority 2 and, finally, all the vertices with priority 4. It shows that the priority clustering based on the distance among the points may lead to an optimal solution following the same clustering, as in the swiss42-C-5-1-b instance, but this is not the general case, as shown by the berlin52-C-5-1-c instance.

Analyzing the randomly grouped berlin52-R-1-b instance, the exact sequence of the priority sets being visited through the tour cannot be seen and the accumulated distance up to the start or the end of each priority level varies differently when compared to this instance with $d = 0$ case given by the berlin52-R-0-b instance. The final position of the vertices in all the priority sets when $d = 1$ are similar to the values found when $d = 0$, but with higher values in most cases. Depending on how the priorities impact the practical result, this situation may be more interesting or less interesting, since the

reduction in the total tour length was significant.

The bigger values of the d parameter allow even bigger variances in the priorities met on the optimal tours found and the difference between the solutions when $d = 0$ and when $d = 2$ and $d = 3$ clearly shows how relaxing the priorities produces solutions with weaker respect for the priority level of the vertices. In the swiss42-C-5-2-b and swiss42-C-5-3-b instances, for example, the optimal route found in both cases starts by visiting a priority 2 vertex and ends returning to the origin coming from a vertex with priority 2. Moreover, in all these instances, there are big changes in all the parameters described in Table 5.6 as d increases from 0 up to 3, which shows in practice how the objectives of minimizing the total tour length and satisfying the priority levels can be very conflicting objectives in practice.

The possibility of visiting all the priority levels from the start of the route, except priority 4, impacts the instances in different ways and may produce undesirable results in some real scenarios. For the swiss42-C-5-3-b and berlin52-C-5-3-c instances, the first priority level in which all the vertices are visited is level 3, that is, the route is focusing on vertices that have less urgency to be visited and, in the swiss42-R-5-2-a and berlin52-R-5-3-b instances, vertices with priority 2 are among the first and the last ones to be visited, meaning that the priorities are only weakly impacting the tour. These examples are important to highlight how the value of d impacts the final result. This must be carefully defined in real-world decision-making scenarios, where to relax the priorities too much may lead to impractical results.

6 Computational Results for the CluVRP-d

In this chapter, computational results from the two formulations and the heuristic approach described in Chapter 4 are presented.

6.1 Benchmark instances and experimental settings

The instances used in this chapter are based on the ones used in Doan et al. (2021), that can be found at `http://orlab.com.vn/home/download`, identified as "The Vehicle Routing Problem with relaxed priority rules". The dataset was generated based on selected instances of the TSPLIB (Reinelt, 1991) including different numbers of priority sets. The way the points were grouped into the priority sets follows two types of clustering, namely random and clustered instances. In the first case, vertices are distributed into the priority groups randomly, which is useful to model cases of commercial product distribution, where the priorities are defined by storage levels, for example. The other type of instance is obtained by clustering nearby vertices, simulating natural disaster situations like earthquakes or tsunamis, where the level of the impact of the disaster is related to its propagation direction. The name of the instances follows a four-field notation *-*-*-*, where:

- the first field is the adapted name of the original TSPLIB instances, including the letters HVRP before the name of the instances. Here the number of vertices is given, which is 20, 30, 100 or 200;
- the second field represents the clustering type: C (clustered) or R (random);
- the third one is the quantity of priority groups in the instance, which may be 3 or 5;
- the fourth one represents the d parameter value used to define the d-relaxed priority rule. It can be 0, 1, 2 or 3 depending on the instance.

As described in the modeling section, we are considering a heterogeneous fleet on the CluVRP-d, and so the vehicles available to visit the areas are different in values

of costs and working capacity. To make the fleets standardized while having different vehicles, we proposed to work with three different types of vehicles, according to the following specifications: Type 1 vehicle with $C_1 = 2000$ and $cap_1 = 500$, Type 2 vehicle with $C_2 = 1300$ and $cap_2 = 300$ and Type 3 vehicle with $C_3 = 900$ and $cap_3 = 200$. The way these vehicles are gathered define the fleets used in the computational experiments.

In the smaller instances case, for each instance proposed in Doan et al. (2021), we built four different instances by defining four different types of vehicle fleets. The idea is to test if varying the vehicles used impacts the difficult in solving the instances using either the solver or the heuristic. Since the smaller instances could be solved using both Gurobi and our heuristic, this aspect of the problem could be developed and tested. The fleets proposed are defined as:

- Fleet 1: 2 Type 1 vehicles, 2 Type 2 vehicles and 2 Type 3 vehicles;
- Fleet 2: 3 Type 1 vehicles and 2 Type 2 vehicles;
- Fleet 3: 5 Type 1 vehicles;
- Fleet 4: 1 Type 1 vehicle, 3 Type 2 vehicles and 5 Type 3 vehicles;

Since the different fleets did not have much impact on the solution of the problem, for the larger instances, we worked with only one type of fleet for each case, namely, a fleet with 6 Type 1 vehicles, 6 Type 2 vehicles and 7 Type 3 vehicles on the instances with 100 vertices, and a fleet with 11 Type 1 vehicles, 12 Type 2 vehicles and 12 Type 3 vehicles for the instances with 200 vertices.

For the computational tests, the instances were divided in three groups, namely the small instances, including the cases with 20 and 30 vertices, the medium instances, with all the 100 vertices cases, and the large instances, including the 200 vertices cases. All instances in each group were solved using the same fixed input parameters for the heuristic, which are presented below, except for the r parameter of the capacitated clustering model, which was set to a range of values, and the complete heuristic was run for each value. Moreover, since the heuristic includes random moves, for each combination of input parameters, r values and vehicle fleet, the method was run a total of five times, and the results presented include the best solution found across all tests, and the average objective value and runtime. Cases where the r parameter did not allow a feasible solution to be found in the clustering step are not shown in the results tables.

- Small instances: Number of clusters in the K-means method was set to 15. Local search parameters: $I_{ILS} = 200$, $Q = 5$ and $L = 100$. Values tested for parameter r: $\{2, 3, 4, 5, 6, 7, 8, 9\}$.

- Medium instances: Number of clusters in the K-means method was set to 35. Local search parameters: $I_{ILS} = 500$, $Q = 5$ and $L = 500$. Values tested for parameter r: $\{17, 18, 19, 20, 21, 22, 23\}$.

- Large instances: Number of clusters in the K-means method was set to 70. Local search parameters: $I_{ILS} = 1000$, $Q = 5$ and $L = 600$. Values tested for parameter r: $\{33, 34, 35, 36, 37, 38, 39\}$.

All the experiments were implemented in Python 3.10 programming language, with the proposed formulations solved using Gurobi Optimizer 9.5. In the heuristic tests, the K-means method was run using the SciKit Learn module version 1.2.2, the clustering optimization model was solved using Gurobi, and the search moves were compiled using Numba module version 0.56.4. The total runtimes presented in the results tables include all the heuristic steps. The first time the heuristic runs, the Numba module applies a JIT compilation process to the steps, which took a few minutes - this time is not included in the tables, since it is not part of the solution process. The experiments were performed on a virtual Linux Ubuntu 20.04 machine operating with a Quad-Core Intel Processor at 2.6 GHz with 16 GB of RAM. The total Runtime imposed was 3600 seconds for the small instances. The larger instances could not be solved using Gurobi due to lack of memory and so we only present results for them using the proposed heuristic.

6.2 Computational results obtained by the proposed formulations

The first results analyzed here were obtained by running both the Local Timing (LTM) and the Global Timing (GTM) Models on the smaller instances using Gurobi. The detailed results are presented in Tables A.7, A.8, A.9 and A.10 (see Appendix A.2), and a summary of the results is given in Table 6.1. The two main features in the results are the long runtimes required by the solver to stop the solution procedure, meaning it was not able to prove optimality for the most part of the instances, and the high values of GAP, which implies the solver could not find good bounds during the branch-and-cut search. Hence, it was not possible to verify if the solutions produced were near optimality or not. It does not mean the results obtained are far from optimality or could not be useful for practical applications but it makes any conclusions drawn about the quality of the results uncertain.

		Random				Cluster			
		F1	F2	F3	F4	F1	F2	F3	F4
Objective Value	LTM	27763.29	26755.71	26848.06	30222.51	26222.40	25033.76	24866.72	28914.35
	GTM	28042.27	27196.84	26805.71	30191.05	26279.21	25283.81	24933.87	29043.57
Runtime	LTM	3321.96	3115.10	3152.99	3536.74	3600.03	3600.03	3600.02	3600.03
	GTM	3462.39	3257.92	3401.33	3541.56	3600.02	3600.01	3600.01	3600.02
GAP	LTM	33.13%	30.26%	30.16%	39.50%	46.28%	42.40%	41.35%	51.40%
	GTM	36.58%	34.00%	32.29%	41.58%	48.83%	45.89%	45.78%	53.72%

Table 6.1: Summary of results for the MILP models proposed, showing the means of objective value, runtime and GAP for each vehicle fleet.

6.3 Computational results obtained by the proposed heuristic

The two main advantages of using a heuristic approach to solve the CluVRP-d are the possibility of tackling larger problems, with numbers of vertices that were not possible to load and solve using Gurobi, and the option of a faster solution procedure, that can lead to local optimal solutions requiring shorter runtimes. We start our analysis comparing the results obtained by the solver and the heuristic on the instances with 20 and 30 vertices, and then we use the findings to interpret and draw conclusions about the performance of the heuristic on the larger cases. The results obtained running the heuristic on the proposed instances are presented in Tables A.11, A.12, A.13 and A.14 (see Appendix A.2), and the comparison between the results reached and the findings obtained with Gurobi (MILP columns) on the smaller instances for the Local Timing Model case is given in Table 6.2.

		Objective Value		Runtime		Best Solution	
		MILP	Heuristic	MILP	Heuristic	MILP	Heuristic
Random	All	27897.39	27207.99	3281.69	43.71	40.00%	87.50%
	d=0	28604.38	28283.44	2963.37	32.85	52.50%	80.00%
	d=1	27190.41	26132.53	3600.02	54.56	27.50%	95.00%
Cluster	All	26259.30	25661.13	3600.03	61.87	41.25%	90.00%
	d=0	26258.48	25839.12	3600.03	49.79	42.50%	90.00%
	d=1	26260.13	25483.14	3600.02	73.95	40.00%	90.00%

Table 6.2: Average objective value, average runtime and percentage of instances the best solution was found for each solution approach on the smaller instances considering the Local Timing Model.

The values in Table 6.2 shows the heuristic had a promising performance. For all cases, the mean of the objective function values of the instances produced by the heuristic are lower than the results obtained using the solver, showing a superior average performance in producing feasible solutions. Moreover, looking at the total runtime column, we see the heuristic found local optima needing, on average, less than 2 minutes in all cases, while the solver took about 50 to 60 minutes to stop the solving procedure with inferior results in most cases. Finally, the Best Solution column shows the following information: among the solutions found using both the heuristic and the solver, in how many cases the solver reached the best solution, and in how many cases the heuristic found the best solution, in percentage. When both solution approaches found the same solution, it is included for both counts. As in the average objective

value column, the heuristic showed the best results in the best solutions column too, being the best option to produce solutions in most part of the cases. The MILP solver did not reach the best solutions for most instances in almost all groups, except for the Random instances with $d = 0$. For the Random instances with $d = 1$, on the other hand, the solver did not produce the best solutions in most cases, reaching the best results in only 27.50% of the instances.

The solver could not solve the larger instances (100 and 200 vertices) so there is no data for comparison. The results obtained running the heuristic in these cases are presented in Tables A.13 and A.14, and are summarized in Table 6.3. There are two main findings we can highlight about the heuristic results. First, greater values of the d parameter are associated with greater average total runtimes. This makes sense, since in instances with greater d values each search step in the heuristic must evaluate greater neighborhood sets. Second, for the same d values and number of vertices, the Cluster instances required greater runtimes than the Random ones.

		100 Vertices		200 Vertices	
		Objective Value	Runtime	Objective Value	Runtime
Random	All	86771.84	733.76	161613.62	3579.89
	d=0	90618.06	508.10	167766.44	2504.29
	d=1	86104.08	749.79	159316.46	3655.28
	d=2	84245.50	839.18	158671.30	4048.18
	d=3	82941.28	1047.62	156844.62	5112.01
Cluster	All	83196.30	1074.01	156242.28	5306.55
	d=0	83811.90	903.19	156893.16	4865.22
	d=1	83019.31	1109.74	155906.87	5439.85
	d=2	82639.57	1145.43	155955.92	5469.27
	d=3	82875.77	1272.73	155897.71	5759.90

Table 6.3: Average best objective value and total runtime when using the heuristic approach on the instances with 100 and 200 vertices.

7 Conclusion and future research

In the first part of this thesis, we propose new formulations for the Clustered Traveling Salesman Problem with d-relaxed Priority Rule (CTSP-d) and analyze a computational study of them. The d-relaxed priority rule is a tool that allows the Clustered Traveling Salesman Problem with Prespecified Order (CTSP-PO) to be relaxed very easily and it can be the basis for many decision support systems in different areas.

The new formulations are based on valid inequalities and precedence variables. These formulations not only improve a MIP model proposed in Hà et al. (2020) but also introduce a new point of view to the problem described as a particular case of the TSP with precedence variables. The new approaches performed very well in the benchmark instances, allowing to prove optimality for most of the tested instances and to find high quality solutions for all of them. Comparing the results from just adding the d-relaxed priority rule to TSP formulations and those after adding valid inequalities to them show great improvements in two of the four formulations proposed, which highlights how the work with reformulation techniques is a good direction for research to improve the MILP solver performance for some problems as found in the CTSP-d case. According to our results, the formulation that led to the best computational results was MTZ2 in both random and clustered instances cases. It is interesting to notice that the clustered instances seemed to be easier for the solver, being solved, in general, in smaller runtimes than the random ones. Other point we highlight is that clustered instances could be solved faster through formulations based on the Miller et al. (1960) model, while in the random cases there is not a clear pattern of best formulation approach.

In the second part of this thesis, we presented the CluVRP-d as a generalization of the CTSP-d proposed by Panchamgam (2011) and developed in Hà et al. (2020), Doan et al. (2023) and Teixeira and de Araujo (2024). A formulation was proposed for each way the priorities can be treated across the vehicles, namely the Local Timing Model and Global Timing Model. A heuristic procedure was developed for the Local Timing case. Results of computational experiments are presented to compare the performance of both a commercial solver and the heuristic in obtaining solutions for instances of the problem. One important finding of second part of the study was that the models

proposed in Chapter 5 were not strong enough to lead the solver to prove optimality for most part of the instances, and after one hour of computational work, in most cases, the GAP value was not near to zero. We emphasize this does not mean Gurobi produced bad quality solutions and the fact the heuristic reached similar results shows these results are local optima, and possibly global optima.

Future research proposals for the first part of the thesis include the development of other possible ways to formulate/interpret the problem, which may be useful for modeling purposes or direct solution procedures using commercial solvers, as well as it may allow the fitting of known efficient techniques used to solve clustered variants of the TSP to the CTSP-d. In this research we developed CTSP-d formulations based on classical TSP models by Miller et al. (1960), Gouveia and Pires (1999), Sarin et al. (2005) and Sherali et al. (2006), but the field of formulations for the TSP is very large, and there are many other models that can be adapted. The formulations proposed in Godinho et al. (2014), Gouveia et al. (2018) and Balma et al. (2018), for example, can be adopted to propose some different formulations for the problem, which may lead to different results. Time dependent models (or layered graph models in a more generic way) (see, for example, Gouveia et al. (2019)) can be investigated, since they are especially suited to model issues related to the d-relaxed priority rule. Solution approaches like exact methods (Schulz and Pfeiffer, 2023) and dynamic programming (Salii, 2017) may be interesting ways to future study. It may be an initial step to the development of different kinds of solution approaches, such as decomposition techniques based on the Benders decomposition or column generation, as well as model-based heuristics such as Relax-and-fix and Fix-and-optimize methods.

The field of heuristics for the TSP has been widely explored but it is still limited for the CTSP-d. The main kind of approach used in the literature is the Iterated Local Search (ILS), proposed in Hà et al. (2020) and in Doan et al. (2023), where the first authors use the GILS-RVND scheme of Silva et al. (2012). Other kinds of heuristics that have already been proposed for the classic TSP and showed good performance could also be applied, for example Tabu Search approaches, Evolutionary Algorithms, Simulated Annealing and Ant Colony methods, among others.

Considering the second part of the thesis, another relevant matters that can be used to motivate future research are the development of new formulations for the CluVRP-d and the application of heuristics to the CluVRP-d, for both the Local Timing and Global Timing Models. The lack of optimality proof shows the necessity to improve/reformulate the problem. The availability of solutions with zero GAP, or near zero GAP, would be useful to make a deeper comparison between the results produced by exact methods and results obtained via heuristic methods. On the heuristics subject,

in this study we proposed an Iterated Local Search method based on the GILS-RVND framework by Silva et al. (2012), obtaining promising results in considerably less time than the commercial solver, or even being the only resource available to reach high quality solutions for the problem, when the quantity of vertices of the instance did not allow the use of the exact method. An interesting topic to be explored is the development of different solution strategies based on other methods for VRP variants. One possible research direction would be to test different neighborhood systems or search strategies on the same framework, looking for a new procedure that leads to better solutions requiring shorter times. Another interesting subject is to explore other types of heuristic based on different solution approaches, like large neighborhood search, population based methods (like genetic algorithms and ant colony optimization), local search methods applying tabu search or guided search strategies, methods based on mixing heuristics and exact approaches (matheuristics), among others. Finally, it is worth to highlight that the VRP-RPR it is a problem still not yet fully explored, and it could be considered in different real applications by making small adaptations. In Chapter 4 we proposed the CluVRP-d as a direct generalization of the CTSP-d considering a heterogeneous fleet and capacity constraints. The same problem could be studied considering a homogeneous fleet and/or other practical limitations on the problem, like duration of the routes. In particular, on the Global Timing case, we are allowing the vehicles to wait in a vertex before moving to the next one, if it is needed to respect the priority rule. Another practical situation that could be considered is the minimization of the total idle time of the vehicles, which may be a costly issue depending on the application.

References

Nezih Altay and Walter G. Green. OR/MS research in disaster operations management. *European Journal of Operational Research*, 175(1):475–493, 2006. ISSN 0377-2217.

Claudia Archetti and Ivana Ljubić. Comparison of formulations for the inventory routing problem. *European Journal of Operational Research*, 2022. ISSN 0377-2217.

Claudia Archetti, M. Grazia Speranza, and Daniele Vigo. *Chapter 10: Vehicle Routing Problems with Profits*, chapter 10, pages 273–297. In: Toth, P., Vigo, D. (eds) Vehicle Routing - Problems, Methods and Applications. MOS-SIAM Series on Optimization. 2nd Edition., 2014.

Ali Balma, Safa Ben Salem, Mehdi Mrad, and Talel Ladhari. Strong multi-commodity flow formulations for the asymmetric traveling salesman problem. *European Journal of Operational Research*, 271(1):72–79, 2018. ISSN 0377-2217.

Pouya Baniasadi, Mehdi Foumani, Kate Smith-Miles, and Vladimir Ejov. A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. *European Journal of Operational Research*, 285(2):444–457, 2020. ISSN 0377-2217.

Gulay Barbarosoglu, Linet Ozdamar, and Ahmet Cevik. An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. *European Journal of Operational Research*, 140:118–133, 2002.

Maria Battarra, Güneş Erdoğan, and Daniele Vigo. Exact algorithms for the clustered vehicle routing problem. *Operations Research*, 62(1):58–71, 2014.

Benita Beamon and Burcu Balcik. Performance measure in humanitarian relief chains. *International Journal of Public Sector Management*, 21:4–25, 2008.

Ali Kourank Beheshti, Seyed Reza Hejazi, and Mehdi Alinaghian. The vehicle routing problem with multiple prioritized time windows: A case study. *Computers & Industrial Engineering*, 90:402–413, 2015. ISSN 0360-8352.

Tolga Bektaş and Luis Gouveia. Requiem for the Miller-Tucker-Zemlin subtour elimination constraints? *European Journal of Operational Research*, 236(3):820–832, 2014. ISSN 0377-2217.

Raquel Bernardino and Ana Paias. Solving the family traveling salesman problem. *European Journal of Operational Research*, 267(2):453–466, 2018. ISSN 0377-2217.

Douglas R. Bish. Planning for a bus-based evacuation. *OR Spectrum*, 33:629–654, 2011.

Sylvain Boussier, Dominique Feillet, and Michel Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5:211–230, 2007.

S. Bretschneider and A. Kimms. A basic mathematical model for evacuation problems in urban areas. *Transportation Research Part A: Policy and Practice*, 45(6):523–539, 2011. ISSN 0965-8564.

D. Briskorn, A. Kimms, and D Olschok. Simultaneous planning for disaster road clearance and distribution of relief goods: a basic model and an exact solution method. *OR Spectrum*, 42:591–619, 2020.

Teobaldo Bulhões, Minh Hoàng Hà, Rafael Martinelli, and Thibaut Vidal. The vehicle routing problem with service level constraints. *European Journal of Operational Research*, 265(2):544–558, 2018. ISSN 0377-2217.

Ann Melissa Campbell, Dieter Vandenbussche, and William Hermann. Routing for relief efforts. *Transportation science*, 42(2):127–145, 2008.

I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996. ISSN 0377-2217.

Antônio Augusto Chaves, José Fernando Gonçalves, and Luiz Antonio Nogueira Lorena. Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. *Computers & Industrial Engineering*, 124: 331–346, 2018. ISSN 0360-8352.

James A. Chisman. The clustered traveling salesman problem. *Computers & Operations Research*, 2(2):115–119, 1975. ISSN 0305-0548.

Yi-Chang Chiu and Hong Zheng. Real-time mobilization decisions for multi-priority emergency response resources and evacuation groups: Model formulation and solution. *Transportation Research Part E: Logistics and Transportation Review*, 43(6): 710–736, 2007. ISSN 1366-5545. Challenges of Emergency Logistics Management.

Andre A. Cire and Willem-Jan van Hoeve. Multivalued decision diagrams for sequencing problems. *Operations Research*, 61(6):1411–1428, 2013.

Ovidiu Cosma, Petrică C. Pop, and Laura Cosma. An effective hybrid genetic algorithm for solving the generalized traveling salesman problem. In Hugo Sanjurjo González, Iker Pastor López, Pablo García Bringas, and Emilio Quintián, Héctor Corchado, editors, *Hybrid Artificial Intelligent Systems*, pages 113–123, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86271-8.

Tiago Tiburcio da Silva, Antônio Augusto Chaves, Horácio Hideki Yanasse, and Henrique Pacca Loureiro Luna. The multicommodity traveling salesman problem with priority prizes: a mathematical model and metaheuristics. *Computational and Applied Mathematics*, 38(188), 2019.

G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954. ISSN 00963984.

Guy Desaulniers, Oli B.G. Madsen, and Stefan Ropke. *Chapter 5: The Vehicle Routing Problem with Time Windows*, chapter 5, pages 119–159. In: Toth, P., Vigo, D. (eds) Vehicle Routing - Problems, Methods and Applications. MOS-SIAM Series on Optimization. 2nd Edition., 2014.

Martin Desrochers and Gilbert Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1): 27–36, 1991. ISSN 0167-6377.

Thanh Tan Doan, Nathalie Bostel, and Minh Hoàng Hà. The vehicle routing problem with relaxed priority rules. *EURO Journal on Transportation and Logistics*, 10: 100039, 2021. ISSN 2192-4376.

T.T. Doan, N. Bostel, M.H. Hà, and V.H.V. Nguyen. New mixed integer linear programming models and an iterated local search for the clustered traveling salesman problem with relaxed priority rule. *Journal of Combinatorial Optimization*, 46(1), 2023. ISSN 1573-2886.

Elizabeth D. Dolan and Jorge J. Morè. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002. ISSN 1436-4646.

Yvan Dumas, Jacques Desrosiers, Eric Gelinas, and Marius M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 42(2):367–371, 1995.

L.F. Escudero. An inexact algorithm for the sequential ordering problem. *European Journal of Operational Research*, 37(2):236–249, 1988. ISSN 0377-2217.

Christopher Expósito-Izquierdo, André Rossi, and Marc Sevaux. A two-level solution approach to solve the clustered capacitated vehicle routing problem. *Computers & Industrial Engineering*, 91:274–289, 2016. ISSN 0360-8352.

Thomas A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995. ISSN 1573-2916.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. W. H. Freeman, 1979. ISBN 9780716710448.

Gianpaolo Ghiani and Gennaro Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122(1):11–17, 2000. ISSN 0377-2217.

Mario Gnägi and Philipp Baumann. A matheuristic for large-scale capacitated clustering. *Computers & Operations Research*, 132:105304, 2021. ISSN 0305-0548.

Maria Teresa Godinho, Luis Gouveia, and Pierre Pesneau. Natural and extended formulations for the time-dependent traveling salesman problem. *Discrete Applied Mathematics*, 164:138–153, 2014. ISSN 0166-218X. Combinatorial Optimization.

Luis Gouveia and Jose Manuel Pires. The asymmetric travelling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints. *European Journal of Operational Research*, 112(1):134–146, 1999. ISSN 0377-2217.

Luis Gouveia and Mario Ruthmair. Load-dependent and precedence-based models for pickup and delivery problems. *Computers & Operations Research*, 63:56–71, 2015. ISSN 0305-0548.

Luis Gouveia, Pierre Pesneau, Mario Ruthmair, and Daniel Santos. Combining and projecting flow models for the (precedence constrained) asymmetric traveling salesman problem. *Networks*, 71(4):451–465, 2018.

Luis Gouveia, Markus Leitner, and Mario Ruthmair. Layered graph approaches for combinatorial optimization problems. *Computers & Operations Research*, 102:22–38, 2019.

Erica Gralla and Jarrod Goentzel. Humanitarian transportation planning: Evaluation of practice-based heuristics and recommendations for improvement. *European Journal of Operational Research*, 269(2):436–450, 2018. ISSN 0377-2217.

Minh Hoàng Hà, Nathalie Bostel, André Langevin, and Louis-Martin Rousseau. An exact algorithm and a metaheuristic for the generalized vehicle routing problem with flexible fleet size. *Computers & Operations Research*, 43:9–19, 2014. ISSN 0305-0548.

Minh Hoàng Hà, Hoa Nguyen Phuong, Huyen Tran Ngoc Nhat, André Langevin, and Martin Trépanier. Solving the clustered traveling salesman problem with d-relaxed priority rule. *International Transactions in Operational Research*, 2020.

Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023. ISSN 0020-0255.

Evin Uzun Jacobson, Nilay Tanik Argon, and Serhan Ziya. Priority assignment in emergency response. *Operations Research*, 60(4):813–832, 2012.

Daniil Khachai, Ruslan Sadykov, Olga Battaia, and Michael Khachay. Precedence constrained generalized traveling salesman problem: Polyhedral study, formulations, and branch-and-cut algorithm. *European Journal of Operational Research*, 309(2): 488–505, 2023. ISSN 0377-2217.

A. Kimms and M. Maiwald. Bi-objective safe and resilient urban evacuation planning. *European Journal of Operational Research*, 269(3):1122–1136, 2018. ISSN 0377-2217.

Gilbert Laporte and Yves Nobert. Generalized travelling salesman problem through n sets of nodes: An integer programming approach. *INFOR: Information Systems and Operational Research*, 21(1):61–75, 1983.

Gilbert Laporte, Hélène Mercure, and Yves Nobert. Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. *Discrete Applied Mathematics*, 18(2):185–197, 1987. ISSN 0166-218X.

Gilbert Laporte, Jean-Yves Potvin, and Florence Quilleret. A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. *Journal of Heuristics*, 2:187–200, 1997.

Gilbert Laporte, Stefan Ropke, and Thibaut Vidal. *Chapter 4: Heuristics for the Vehicle Routing Problem*, chapter 4, pages 87–116. In: Toth, P., Vigo, D. (eds) Vehicle Routing - Problems, Methods and Applications. MOS-SIAM Series on Optimization. 2nd Edition., 2014.

Adam Letchford and Juan José Salazar González. Stronger multi-commodity flow formulations of the (capacitated) sequential ordering problem. *European Journal of Operational Research*, 251:74–84, 2016.

Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28: 129–136, 1982.

Alfredo Marín and Mercedes Pelegrín. *Chapter 2: p-Median Problems*, chapter 2, pages 25–50. In: Laporte, G., Nickel, S., Saldanha da Gama, F. (eds) Location Science. Springer International Publishing, 2nd Edition, Cham, 2019. ISBN 978-3-030-32177-2.

Anna Martínez-Gavara, Dario Landa-Silva, Vicente Campos, and Rafael Martí. Randomized heuristics for the capacitated clustering problem. *Information Sciences*, 417:154–168, 2017. ISSN 0020-0255.

Vinícius R. Máximo, Jean-François Cordeau, and Mariá C.V. Nascimento. An adaptive iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 148:105954, 2022. ISSN 0305-0548.

Pablo A. Maya Duque, Irina S. Dolinskaya, and Kenneth Sörensen. Network repair crew scheduling and routing for emergency relief distribution problem. *European Journal of Operational Research*, 248(1):272–285, 2016. ISSN 0377-2217.

Mário Mestria. New hybrid heuristic algorithm for the clustered traveling salesman problem. *Computers & Industrial Engineering*, 116:1–12, 2018. ISSN 0360-8352.

C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960. ISSN 0004-5411.

Alfredo Moreno, Pedro Munari, and Douglas Alem. A branch-and-benders-cut algorithm for the crew scheduling and routing problem in road restoration. *European Journal of Operational Research*, 275(1):16–34, 2019. ISSN 0377-2217.

Alfredo Moreno, Pedro Munari, and Douglas Alem. Decomposition-based algorithms for the crew scheduling and routing problem in road restoration. *Computers & Operations Research*, 119:104935, 2020. ISSN 0305-0548.

Temel Öncan, İ. Kuban Altinel, and Gilbert Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654, 2009. ISSN 0305-0548.

Linet Özdamar and Onur Demir. A hierarchical clustering and routing procedure for large scale disaster relief logistics planning. *Transportation Research Part E: Logistics and Transportation Review*, 48(3):591–602, 2012. ISSN 1366-5545.

Kiran Venkata Panchamgam. *Essays in Retail Operations and Humanitarian Logistics*. PhD thesis, University of Maryland, College Park, 2011.

Camelia-M. Pintea, Petrică C. Pop, and Camelia Chira. The generalized traveling salesman problem solved with ant algorithms. *Complex Adaptive Systems Modeling*, 5(8), 2017.

Petrica Pop. New integer programming formulations of the generalized travelling salesman problem. *American Journal of Applied Sciences*, 4, 2007.

Petrica C. Pop, Levente Fuksz, Andrei Horvat Marc, and Cosmin Sabo. A novel two-level optimization approach for clustered vehicle routing problem. *Computers & Industrial Engineering*, 115:304–318, 2018. ISSN 0360-8352.

Jean-Yves Potvin and François Guertin. *A Genetic Algorithm for the Clustered Traveling Salesman Problem with a Prespecified Order on the Clusters*, chapter 11, pages 287–299. In: Woodruff, David L. Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research. Springer US, 1998. ISBN 978-1-4757-2807-1.

Vitoria Pureza, Reinaldo Morabito, and Henrique P. Luna. Modeling and solving the traveling salesman problem with priority prizes. *Pesquisa Operacional*, 38(3):499–522, 2018.

Gerhard Reinelt. TSPLIB–a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

Cosmin Sabo, Petrică C. Pop, and Andrei Horvat-Marc. On the selective vehicle routing problem. *Mathematics*, 8(5), 2020. ISSN 2227-7390.

Yaroslav Salii. Revisiting dynamic programming for precedence-constrained traveling salesman problem and its time-dependent generalization. *European Journal of Operational Research*, 2017.

Raad Salman, Fredrik Ekstedt, and Peter Damaschke. Branch-and-bound for the precedence constrained generalized traveling salesman problem. *Operations Research Letters*, 48(2):163–166, 2020. ISSN 0167-6377.

Subhash C. Sarin, Hanif D. Sherali, and Ajay Bhootra. New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Operations Research Letters*, 33(1):62–70, 2005. ISSN 0167-6377.

H. Schmitz and S. Niemann. *A Bicriteria Traveling Salesman Problem with Sequence Priorities*, chapter 1, pages 1–14. In: Sörensen K., Sevaux M., Habenicht W., Geiger M. (eds) Metaheuristics in the Service Industry. Lecture Notes in Economics and Mathematical Systems, vol 624. Springer, Berlin, Heidelberg., 2009. ISBN 978-3-642-00939-6.

Arne Schulz and Christian Pfeiffer. Using fixed paths to improve branch-and-cut algorithms for precedence-constrained routing problems. *European Journal of Operational Research*, 2023. ISSN 0377-2217.

Hanif D. Sherali, Subhash C. Sarin, and Pei-Fang Tsai. A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Discrete Optimization*, 3(1):20–32, 2006. ISSN 1572-5286.

Vijay K. Shetty, Moises Sudit, and Rakesh Nagi. Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles. *Computers & Operations Research*, 35(6):1813–1828, 2008. ISSN 0305-0548. Part Special Issue: OR Applications in the Military and in Counter-Terrorism.

Jiuh-Biing Sheu. An emergency logistics distribution approach for quick response to urgent relief demand in disasters. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):687–709, 2007. ISSN 1366-5545. Challenges of Emergency Logistics Management.

Jiuh-Biing Sheu. Dynamic relief-demand management for emergency logistics operations under large-scale disasters. *Transportation Research Part E: Logistics and Transportation Review*, 46(1):1–17, 2010. ISSN 1366-5545.

Yong Shi, Wenheng Liu, and Yanjie Zhou. An adaptive large neighborhood search based approach for the vehicle routing problem with zone-based pricing. *Engineering Applications of Artificial Intelligence*, 124:106506, 2023. ISSN 0952-1976.

Marcos Melo Silva, Anand Subramanian, Thibaut Vidal, and Luiz Satoru Ochi. A simple and effective metaheuristic for the minimum latency problem. *European Journal of Operational Research*, 221(3):513–520, 2012. ISSN 0377-2217.

Natalie Simpson and Philip Hancock. Fifty years of operational research and emergency response. *JORS*, 60, 2009.

SS Srivastava, Santosh Kumar, RC Garg, and Prasenjit Sen. Generalized traveling salesman problem through n sets of nodes. *CORS journal*, 7(2):97, 1969.

Fernando Stefanello, Olinto C. B. de Araújo, and Felipe M. Müller. Matheuristics for the capacitated p-median problem. *International Transactions in Operational Research*, 22(1):149–167, 2015.

Eduardo Teixeira and Silvio de Araujo. Formulations for the clustered traveling salesman problem with d-relaxed priority rule. *Computers & Operations Research*, 161: 106402, 2024. ISSN 0305-0548.

Eduardo Teixeira et al. Problema do Caixeiro Viajante Hierárquico: Uma Aplicação no Controle de Formigas em uma Indústria de Papel e Celulose. In: ANAIS DO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 2021, João Pessoa. Anais eletrônicos... Campinas, Galoá, 2021. Disponível em: <https://proceedings.science/sbpo/sbpo-2021/trabalhos/problema-do-caixeiro-viajante-hierarquico-uma-aplicacao-no-controle-de-formigas> Acesso em: 19 mar. 2024.

L N Van Wassenhove. Humanitarian aid logistics: supply chain management in high gear. *Journal of the Operational Research Society*, 57(5):475–489, 2006.

Hongguang Wu and Yuelin Gao. An ant colony optimization based on local search for the vehicle routing problem with simultaneous pickup-delivery and time window. *Applied Soft Computing*, 139:110203, 2023. ISSN 1568-4946.

Ala-Eddine Yahiaoui, Aziz Moukrim, and Mehdi Serairi. The clustered team orienteering problem. *Computers & Operations Research*, 111:386–399, 2019. ISSN 0305-0548.

Zhiwei Yang, Michael Emmerich, and Thomas Bäck. Ant based solver for dynamic vehicle routing problem with time windows and multiple priorities. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2813–2819, 2015.

Qing Zhou, Una Benlic, Qinghua Wu, and Jin-Kao Hao. Heuristic search to the capacitated clustering problem. *European Journal of Operational Research*, 273(2): 464–487, 2019. ISSN 0377-2217.

Appendix A - Numerical results

A.1 CTSP-d Results

In Tables A.1 to A.6 there are three features presented for each solved instance and divided by formulation, for which the meaning and details are as follows. The "Obj" column refers to the best value the solver could find for the objective function in the given time limit. The "Runtime" column shows the solving time reported by the solver, not including the time to instantiate the formulation. Finally, an important information directly related to the solving time is the "GAP" for each instance, that is shown as percentages in the Tables and can be used as a measure of the quality of the solutions obtained when the solver was not able to prove optimality in the given time limit. The best solutions are highlighted in the following way: Solutions where the solver proved optimality within the given time limit are denoted by bold type. When the solver obtained the optimal solution but was not able to prove it, the instance is marked with one star and when the solver reached the best solution for an instance but it was not proved if this solution is optimal or not, this is indicated with two stars. The best solutions from the literature considered are described in Doan et al. (2021). Since we did not find results in the literature for the cases with $d = 2$, we have no data to compare them, so we marked the best results found in this study with the same two stars notation.

Table A.1: Solver results from the H2020, MTZ1, GP1, SSB1 and SST1 formulations for the smaller random instances

	(H2020)			(MTZ1)			(GP1)			(SSB1)			(SST1)		
Instance	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP
berlin52-R-3-0-a	**12765**	2.45	0.00	**12765**	9.03	0.00	**12765**	24.51	0.00	**12765**	12.67	0.00	**12765**	16.00	0.00
berlin52-R-3-1-a	**9473**	3043.43	0.00	9473*	3600.00	3.53	9473*	3600.16	5.52	9498	3600.02	2.73	10105	3617.79	7.04
berlin52-R-3-0-b	**12668**	5.61	0.00	**12668**	3.34	0.00	**12668**	22.77	0.00	**12668**	5.75	0.00	**12668**	17.93	0.00
berlin52-R-3-1-b	9442	3600.00	2.64	9419*	3600.01	3.23	9507	3600.53	5.95	9482	3600.01	3.85	10497	3600.02	10.66
berlin52-R-3-0-c	**12483**	10.66	0.00	**12483**	10.54	0.00	**12483**	38.00	0.00	**12483**	16.37	0.00	**12483**	17.54	0.00
berlin52-R-3-1-c	9577**	3600.00	3.68	9586	3600.02	4.08	9779	3607.58	7.96	9579	3600.01	4.58	10697	3600.02	12.67
berlin52-R-5-0-a	**16414**	2.49	0.00	**16414**	1.48	0.00	**16414**	16.84	0.00	**16414**	1.09	0.00	**16414**	4.53	0.00
berlin52-R-5-1-a	11662	3600.02	8.09	11651*	3600.02	9.06	11651*	3600.09	9.78	**11651**	426.03	0.00	**11651**	1088.91	0.00
berlin52-R-5-2-a	10386**	3600.01	10.90	10501	3600.01	12.51	10730	3600.72	16.20	10386**	3600.01	7.77	12230	3600.02	19.26
berlin52-R-5-3-a	9240	3600.01	8.04	9150	3600.02	7.39	9325	3600.07	11.41	9248	3600.06	11.09	27106	3600.04	100.00
berlin52-R-5-0-b	**13759**	14.02	0.00	**13759**	24.97	0.00	**13759**	31.03	0.00	**13759**	5.73	0.00	**13759**	4.96	0.00
berlin52-R-5-1-b	10009	3600.01	5.91	9957*	3600.01	6.18	10037	3606.60	8.06	9957*	3600.06	3.58	**9957**	1804.89	0.00
berlin52-R-5-2-b	**8679**	1919.35	0.00	8679*	3600.01	0.96	8679*	3600.03	1.83	8679*	3601.75	1.13	9005	3600.02	3.72
berlin52-R-5-3-b	8036*	3600.01	0.36	**8036**	3232.80	0.00	8036*	3600.78	2.22	**8036**	2811.20	0.00	26429	3600.03	69.80
berlin52-R-5-0-c	**14131**	6.76	0.00	**14131**	4.27	0.00	**14131**	20.22	0.00	**14131**	2.12	0.00	**14131**	4.98	0.00
berlin52-R-5-1-c	10940*	3600.01	7.34	10974	3603.70	7.76	10994	3600.11	9.52	**10940**	1415.65	0.00	**10940**	1021.66	0.00
berlin52-R-5-2-c	9958	3600.00	7.98	9937**	3600.03	7.80	10210	3600.22	11.04	9958	3600.02	6.27	11616	3606.52	18.03
berlin52-R-5-3-c	8224**	3600.00	2.10	8229	3600.02	2.58	8224**	3601.55	3.93	8224**	3600.03	2.19	8401	3600.02	3.71
swiss42-R-3-0-a	**2256**	0.47	0.00	**2256**	2.96	0.00	**2256**	2.23	0.00	**2256**	1.41	0.00	**2256**	2.82	0.00
swiss42-R-3-1-a	**1652**	166.97	0.00	**1652**	1228.26	0.00	**1652**	1847.04	0.00	**1652**	434.47	0.00	**1652**	734.43	0.00
swiss42-R-3-0-b	**2153**	1.76	0.00	**2152**	6.30	0.00	**2153**	12.41	0.00	**2153**	3.96	0.00	**2153**	6.22	0.00
swiss42-R-3-1-b	**1607**	341.41	0.00	**1607**	990.64	0.00	**1607**	2701.42	0.00	**1607**	2261.41	0.00	**1607**	2292.38	0.00
swiss42-R-3-0-c	**2080**	2.25	0.00	**2080**	2.61	0.00	**2080**	13.86	0.00	**2080**	4.96	0.00	**2080**	5.23	0.00
swiss42-R-3-1-c	**1525**	945.58	0.00	**1525**	2666.62	0.00	**1525**	2845.06	0.00	**1525**	2544.18	0.00	**1525**	1305.11	0.00
swiss42-R-5-0-a	**2365**	1.55	0.00	**2365**	1.03	0.00	**2365**	4.47	0.00	**2365**	0.62	0.00	**2365**	1.58	0.00
swiss42-R-5-1-a	**1812**	245.57	0.00	**1812**	425.47	0.00	**1812**	888.63	0.00	**1812**	160.03	0.00	**1812**	121.63	0.00
swiss42-R-5-2-a	**1596**	1881.10	0.00	**1596**	2834.31	0.00	1596*	3600.15	3.13	**1596**	542.02	0.00	**1596**	1719.00	0.00
swiss42-R-5-3-a	**1474**	1714.75	0.00	1474*	3600.00	1.76	1474*	3600.07	4.82	**1474**	3431.72	0.00	1785	3600.02	18.04
swiss42-R-5-0-b	**2567**	1.23	0.00	**2567**	1.30	0.00	**2567**	2.58	0.00	**2567**	0.66	0.00	**2567**	1.78	0.00
swiss42-R-5-1-b	1905*	3600.00	1.84	1905*	3600.00	2.78	1906	3600.06	4.98	**1905**	111.74	0.00	**1905**	242.16	0.00
swiss42-R-5-2-b	**1639**	1168.19	0.00	**1639**	1261.96	0.00	1639*	3600.11	1.46	**1639**	680.97	0.00	**1639**	3373.33	0.00
swiss42-R-5-3-b	1541**	3600.02	4.48	1557	3600.00	5.72	1566	3600.63	8.75	1541**	3600.02	6.04	1725	3600.01	13.22
swiss42-R-5-0-c	**2694**	0.71	0.00	**2694**	0.82	0.00	**2694**	4.43	0.00	**2694**	0.53	0.00	**2694**	2.21	0.00
swiss42-R-5-1-c	**1910**	1698.08	0.00	**1910**	3273.04	0.00	1910*	3600.07	1.99	**1910**	303.23	0.00	**1910**	2209.98	0.00
swiss42-R-5-2-c	1601*	3600.01	1.62	1601*	3600.01	1.94	1603	3600.52	4.24	**1601**	1680.01	0.00	1606	3600.06	1.81
swiss42-R-5-3-c	**1510**	1019.49	0.00	**1510**	2244.24	0.00	1513	3600.02	3.83	1510*	3600.01	1.26	1793	3600.01	16.79
Mean Values	6437.03	1694.28	1.81	6436.50	2006.38	2.15	6466.19	2235.99	3.52	6437.36	1568.35	1.40	7653.44	1745.11	8.19

Table A.2: Solver results from the H2020, MTZ1, GP1, SSB1 and SST1 formulations for the smaller clustered instances

	(H2020)			(MTZ1)			(GP1)			(SSB1)			(SST1)		
Instance	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP
berlin52-C-3-0-a	**8144**	1.88	0.00	**8144**	2.10	0.00	**8144**	97.22	0.00	**8144**	80.70	0.00	**8144**	768.61	0.00
berlin52-C-3-1-a	**7952**	3.66	0.00	**7952**	35.83	0.00	**7952**	211.94	0.00	**7952**	1892.89	0.00	**7952**	2210.32	0.00
berlin52-C-3-0-b	**8016**	1.73	0.00	**8016**	8.01	0.00	**8016**	56.64	0.00	**8016**	10.01	0.00	**8016**	15.74	0.00
berlin52-C-3-1-b	**7596**	2.72	0.00	**7596**	5.48	0.00	**7596**	54.15	0.00	**7596**	214.80	0.00	**7596**	759.91	0.00
berlin52-C-3-0-c	**9085**	149.99	0.00	**9085**	334.56	0.00	9085*	3600.13	1.59	**9085**	56.13	0.00	**9085**	241.67	0.00
berlin52-C-3-1-c	**7984**	1.99	0.00	**7984**	1.58	0.00	**7984**	69.18	0.00	**7984**	845.78	0.00	11795	3600.03	32.65
berlin52-C-5-0-a	**9430**	29.73	0.00	**9430**	74.39	0.00	**9430**	497.99	0.00	**9430**	18.82	0.00	**9430**	40.37	0.00
berlin52-C-5-1-a	**8811**	5.14	0.00	**8811**	9.33	0.00	**8811**	169.13	0.00	**8811**	220.05	0.00	**8811**	375.78	0.00
berlin52-C-5-2-a	**7907**	2458.20	0.00	**7907**	2316.23	0.00	7986	3600.17	2.97	7907*	3600.04	1.20	24031	3600.03	67.30
berlin52-C-5-3-a	7907*	3600.00	0.61	7907*	3600.01	0.40	7907*	3600.14	2.12	7907*	3600.04	2.38	25111	3600.05	100.00
berlin52-C-5-0-b	**8669**	6.92	0.00	**8669**	12.05	0.00	**8669**	54.70	0.00	**8669**	76.91	0.00	**8669**	14.39	0.00
berlin52-C-5-1-b	**7948**	2.21	0.00	**7948**	9.75	0.00	**7948**	78.71	0.00	**7948**	186.60	0.00	**7948**	421.68	0.00
berlin52-C-5-2-b	**7614**	0.54	0.00	**7614**	1.54	0.00	**7614**	122.37	0.00	**7614**	295.44	0.00	**7614**	883.31	0.00
berlin52-C-5-3-b	**7614**	1.49	0.00	**7614**	6.15	0.00	**7614**	94.25	0.00	**7614**	1091.35	0.00	**7614**	1226.67	0.00
berlin52-C-5-0-c	9651*	3600.01	5.71	9651*	3600.01	5.64	9651*	3600.17	6.26	**9651**	80.59	0.00	**9651**	10.55	0.00
berlin52-C-5-1-c	**8509**	13.99	0.00	**8509**	8.89	0.00	**8509**	119.69	0.00	**8509**	78.85	0.00	**8509**	394.91	0.00
berlin52-C-5-2-c	**8011**	2221.57	0.00	8011*	3600.01	0.85	8034	3601.04	2.76	8011*	3600.06	2.73	12889	3600.03	38.53
berlin52-C-5-3-c	**7631**	1.33	0.00	**7631**	1.89	0.00	**7631**	137.27	0.00	**7631**	538.55	0.00	**7631**	1219.12	0.00
swiss42-C-3-0-a	**1347**	2.36	0.00	**1347**	6.62	0.00	**1347**	11.76	0.00	**1347**	8.17	0.00	**1347**	10.01	0.00
swiss42-C-3-1-a	**1301**	7.91	0.00	**1301**	8.57	0.00	**1301**	135.09	0.00	**1301**	68.84	0.00	**1301**	872.10	0.00
swiss42-C-3-0-b	**1505**	8.98	0.00	**1505**	7.56	0.00	**1505**	38.33	0.00	**1505**	4.07	0.00	**1505**	12.14	0.00
swiss42-C-3-1-b	**1344**	8.61	0.00	**1344**	2.41	0.00	**1344**	182.86	0.00	**1344**	495.25	0.00	**1344**	2558.55	0.00
swiss42-C-3-0-c	**1467**	1.95	0.00	**1467**	1.57	0.00	**1467**	4.62	0.00	**1467**	2.08	0.00	**1467**	8.98	0.00
swiss42-C-3-1-c	**1357**	4.64	0.00	**1357**	13.41	0.00	**1357**	146.47	0.00	**1357**	133.86	0.00	**1357**	376.79	0.00
swiss42-C-5-0-a	**1561**	114.85	0.00	**1561**	190.58	0.00	**1561**	1019.37	0.00	**1561**	8.10	0.00	**1561**	2.90	0.00
swiss42-C-5-1-a	**1434**	14.17	0.00	**1434**	16.69	0.00	**1434**	102.78	0.00	**1434**	64.82	0.00	**1434**	29.46	0.00
swiss42-C-5-2-a	**1343**	6.53	0.00	**1343**	7.87	0.00	**1343**	105.10	0.00	**1343**	80.46	0.00	**1343**	334.54	0.00
swiss42-C-5-3-a	**1273**	0.72	0.00	**1273**	0.91	0.00	**1273**	6.55	0.00	**1273**	22.19	0.00	**1273**	344.84	0.00
swiss42-C-5-0-b	**1540**	1.16	0.00	**1540**	1.56	0.00	**1540**	18.20	0.00	**1540**	3.08	0.00	**1540**	2.88	0.00
swiss42-C-5-1-b	**1469**	25.25	0.00	**1469**	926.93	0.00	**1469**	1825.13	0.00	**1469**	860.67	0.00	**1469**	59.04	0.00
swiss42-C-5-2-b	**1315**	11.85	0.00	**1315**	12.06	0.00	**1315**	81.43	0.00	**1315**	195.07	0.00	**1315**	1543.53	0.00
swiss42-C-5-3-b	**1273**	1.38	0.00	**1273**	5.07	0.00	**1273**	48.52	0.00	**1273**	40.53	0.00	**1273**	494.44	0.00
swiss42-C-5-0-c	**1532**	23.11	0.00	**1532**	34.14	0.00	**1532**	76.42	0.00	**1532**	3.93	0.00	**1532**	1.82	0.00
swiss42-C-5-1-c	**1334**	2.27	0.00	**1334**	1.97	0.00	**1334**	23.38	0.00	**1334**	12.57	0.00	**1334**	41.06	0.00
swiss42-C-5-2-c	**1321**	16.05	0.00	**1321**	73.50	0.00	**1321**	182.12	0.00	**1321**	307.35	0.00	**1321**	980.64	0.00
swiss42-C-5-3-c	**1301**	3.69	0.00	**1301**	7.06	0.00	**1301**	116.33	0.00	**1301**	447.30	0.00	**1301**	1330.99	0.00
Mean Values	4819.33	343.29	0.18	4819.33	415.17	0.19	4822.17	663.59	0.44	4819.33	534.61	0.18	5986.47	888.55	6.62

Table A.3: Solver results from the H2020, MTZ2, GP2, SSB2 and SST2 formulations for the smaller random instances

	(H2020)			(MTZ2)			(GP2)			(SSB2)			(SST2)		
Instance	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP
berlin52-R-3-0-a	**12765**	2.45	0.00	**12765**	3.50	0.00	**12765**	8.91	0.00	**12765**	6.88	0.00	**12765**	15.45	0.00
berlin52-R-3-1-a	**9473**	3043.43	0.00	**9473**	480.26	0.00	**9473**	1410.57	0.00	9473*	3600.02	1.40	9849	3600.06	4.15
berlin52-R-3-0-b	**12668**	5.61	0.00	**12668**	2.15	0.00	**12668**	10.83	0.00	**12668**	6.33	0.00	**12668**	15.83	0.00
berlin52-R-3-1-b	9442	3600.00	2.64	**9419**	419.58	0.00	**9419**	1419.96	0.00	9419*	3600.01	2.51	9702	3600.05	3.53
berlin52-R-3-0-c	**12483**	10.66	0.00	**12483**	3.68	0.00	**12483**	14.75	0.00	**12483**	16.24	0.00	**12483**	17.19	0.00
berlin52-R-3-1-c	9577**	3600.00	3.68	9579	3600.01	1.66	9577**	3600.04	1.47	9577**	3600.02	3.16	11416	3600.03	18.21
berlin52-R-5-0-a	**16414**	2.49	0.00	**16414**	1.19	0.00	**16414**	1.77	0.00	**16414**	1.13	0.00	**16414**	4.75	0.00
berlin52-R-5-1-a	11662	3600.02	8.09	**11651**	260.14	0.00	**11651**	925.01	0.00	**11651**	1048.16	0.00	**11651**	1206.09	0.00
berlin52-R-5-2-a	10386**	3600.01	10.90	10434	3600.02	6.73	10386**	3600.17	7.65	10432	3600.06	8.54	13057	3600.03	24.67
berlin52-R-5-3-a	9240	3600.01	8.04	9205	3600.01	7.14	9065	3600.08	7.35	9344	3600.02	11.91	26598	3600.04	100.00
berlin52-R-5-0-b	**13759**	14.02	0.00	**13759**	10.23	0.00	**13759**	6.69	0.00	**13759**	6.13	0.00	**13759**	5.58	0.00
berlin52-R-5-1-b	10009	3600.01	5.91	**9957**	3541.29	0.00	**9957**	2530.46	0.00	9957*	3600.11	1.25	9976	3600.12	0.46
berlin52-R-5-2-b	**8679**	1919.35	0.00	**8679**	369.19	0.00	**8679**	1705.20	0.00	**8679**	3370.99	0.00	9009	3600.03	3.83
berlin52-R-5-3-b	8036*	3600.01	0.36	**8036**	309.85	0.00	**8036**	1428.52	0.00	**8036**	3253.64	0.00	28112	3600.03	71.61
berlin52-R-5-0-c	**14131**	6.76	0.00	**14131**	1.38	0.00	**14131**	2.67	0.00	**14131**	1.70	0.00	**14131**	5.56	0.00
berlin52-R-5-1-c	10940*	3600.01	7.34	**10940**	1209.36	0.00	**10940**	1125.77	0.00	**10940**	1148.38	0.00	**10940**	3312.62	0.00
berlin52-R-5-2-c	9958	3600.00	7.98	10082	3600.03	7.07	9937**	3600.33	4.04	9992	3600.09	6.21	11905	3600.02	20.26
berlin52-R-5-3-c	8224**	3600.00	2.10	8224**	3600.01	1.46	8224**	3601.02	1.11	8224**	3600.03	1.62	8364	3600.05	3.22
swiss42-R-3-0-a	**2256**	0.47	0.00	**2256**	0.80	0.00	**2256**	1.42	0.00	**2256**	1.37	0.00	**2256**	2.87	0.00
swiss42-R-3-1-a	**1652**	166.97	0.00	**1652**	70.80	0.00	**1652**	161.45	0.00	**1652**	332.24	0.00	**1652**	713.77	0.00
swiss42-R-3-0-b	**2153**	1.76	0.00	**2153**	1.71	0.00	**2153**	4.84	0.00	**2153**	6.00	0.00	**2153**	6.68	0.00
swiss42-R-3-1-b	**1607**	341.41	0.00	**1607**	322.84	0.00	**1607**	576.71	0.00	**1607**	2174.90	0.00	**1607**	2504.32	0.00
swiss42-R-3-0-c	**2080**	2.25	0.00	**2080**	2.73	0.00	**2080**	5.32	0.00	**2080**	3.61	0.00	**2080**	4.54	0.00
swiss42-R-3-1-c	**1525**	945.58	0.00	**1525**	161.59	0.00	**1525**	796.03	0.00	**1525**	1618.15	0.00	**1525**	772.92	0.00
swiss42-R-5-0-a	**2365**	1.55	0.00	**2365**	0.47	0.00	**2365**	0.93	0.00	**2365**	0.81	0.00	**2365**	1.60	0.00
swiss42-R-5-1-a	**1812**	245.57	0.00	**1812**	146.32	0.00	**1812**	145.06	0.00	**1812**	171.15	0.00	**1812**	128.83	0.00
swiss42-R-5-2-a	**1596**	1881.10	0.00	**1596**	421.63	0.00	**1596**	189.05	0.00	**1596**	714.69	0.00	**1596**	1527.72	0.00
swiss42-R-5-3-a	**1474**	1714.75	0.00	**1474**	1083.20	0.00	**1474**	2107.94	0.00	**1474**	3067.37	0.00	1613	3600.02	9.42
swiss42-R-5-0-b	**2567**	1.23	0.00	**2567**	0.54	0.00	**2567**	0.82	0.00	**2567**	0.54	0.00	**2567**	1.72	0.00
swiss42-R-5-1-b	1905*	3600.00	1.84	**1905**	117.31	0.00	**1905**	100.64	0.00	**1905**	78.05	0.00	**1905**	240.95	0.00
swiss42-R-5-2-b	**1639**	1168.19	0.00	**1639**	132.67	0.00	**1639**	303.36	0.00	**1639**	754.39	0.00	**1639**	1650.75	0.00
swiss42-R-5-3-b	1541**	3600.02	4.48	1541**	3600.00	3.57	1541**	3600.11	4.02	1541**	3600.02	5.91	1743	3600.01	13.94
swiss42-R-5-0-c	**2694**	0.71	0.00	**2694**	0.53	0.00	**2694**	0.84	0.00	**2694**	0.53	0.00	**2694**	2.04	0.00
swiss42-R-5-1-c	**1910**	1698.08	0.00	**1910**	420.54	0.00	**1910**	481.26	0.00	**1910**	326.58	0.00	**1910**	2425.98	0.00
swiss42-R-5-2-c	1601*	3600.01	1.62	**1601**	468.96	0.00	**1601**	1537.66	0.00	**1601**	1530.17	0.00	1802	3600.07	12.76
swiss42-R-5-3-c	**1510**	1019.49	0.00	**1510**	1521.45	0.00	**1510**	3507.83	0.00	1510*	3600.01	3.25	1801	3600.01	17.21
Mean Values	6437.03	1694.28	1.81	6438.50	919.05	0.77	6429.19	1169.83	0.71	6439.75	1545.57	1.27	7708.86	1804.68	8.42

Table A.4: Solver results from the H2020, MTZ2, GP2, SSB2 and SST2 formulations for the smaller clustered instances

	(H2020)			(MTZ2)			(GP2)			(SSB2)			(SST2)		
Instance	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP
berlin52-C-3-0-a	**8144**	1.88	0.00	**8144**	1.38	0.0	**8144**	37.82	0.0	**8144**	78.50	0.00	**8144**	1217.17	0.00
berlin52-C-3-1-a	**7952**	3.66	0.00	**7952**	3.05	0.0	**7952**	167.70	0.0	**7952**	580.99	0.00	**7952**	2043.90	0.00
berlin52-C-3-0-b	**8016**	1.73	0.00	**8016**	1.42	0.0	**8016**	10.54	0.0	**8016**	4.25	0.00	**8016**	14.01	0.00
berlin52-C-3-1-b	**7596**	2.72	0.00	**7596**	2.86	0.0	**7596**	51.55	0.0	**7596**	100.81	0.00	**7596**	725.41	0.00
berlin52-C-3-0-c	**9085**	149.99	0.00	**9085**	1.05	0.0	**9085**	14.08	0.0	**9085**	13.79	0.00	**9085**	208.83	0.00
berlin52-C-3-1-c	**7984**	1.99	0.00	**7984**	1.86	0.0	**7984**	159.27	0.0	**7984**	408.71	0.00	9497	3600.05	16.35
berlin52-C-5-0-a	**9430**	29.73	0.00	**9430**	0.46	0.0	**9430**	7.09	0.0	**9430**	5.52	0.00	**9430**	45.32	0.00
berlin52-C-5-1-a	**8811**	5.14	0.00	**8811**	2.92	0.0	**8811**	60.52	0.0	**8811**	44.74	0.00	**8811**	451.27	0.00
berlin52-C-5-2-a	**7907**	2458.20	0.00	**7907**	54.76	0.0	**7907**	902.53	0.0	**7907**	1768.60	0.00	26387	3600.03	70.22
berlin52-C-5-3-a	7907*	3600.00	0.61	**7907**	522.37	0.0	**7907**	2573.29	0.0	8061	3600.02	4.66	25310	3600.07	100.00
berlin52-C-5-0-b	**8669**	6.92	0.00	**8669**	0.33	0.0	**8669**	6.13	0.0	**8669**	2.66	0.00	**8669**	16.38	0.00
berlin52-C-5-1-b	**7948**	2.21	0.00	**7948**	2.52	0.0	**7948**	15.92	0.0	**7948**	109.11	0.00	**7948**	422.80	0.00
berlin52-C-5-2-b	**7614**	0.54	0.00	**7614**	0.91	0.0	**7614**	42.21	0.0	**7614**	92.13	0.00	**7614**	1005.93	0.00
berlin52-C-5-3-b	**7614**	1.49	0.00	**7614**	1.08	0.0	**7614**	46.26	0.0	**7614**	105.10	0.00	**7614**	1475.57	0.00
berlin52-C-5-0-c	9651*	3600.01	5.71	**9651**	0.51	0.0	**9651**	5.73	0.0	**9651**	2.98	0.00	**9651**	9.73	0.00
berlin52-C-5-1-c	**8509**	13.99	0.00	**8509**	8.96	0.0	**8509**	68.53	0.0	**8509**	56.22	0.00	**8509**	365.97	0.00
berlin52-C-5-2-c	**8011**	2221.57	0.00	**8011**	50.67	0.0	**8011**	1490.44	0.0	8011*	3600.04	1.32	20862	3600.04	62.02
berlin52-C-5-3-c	**7631**	1.33	0.00	**7631**	1.47	0.0	**7631**	93.61	0.0	**7631**	193.32	0.00	**7631**	1849.56	0.00
swiss42-C-3-0-a	**1347**	2.36	0.00	**1347**	2.84	0.0	**1347**	3.61	0.0	**1347**	6.69	0.00	**1347**	12.64	0.00
swiss42-C-3-1-a	**1301**	7.91	0.00	**1301**	1.82	0.0	**1301**	45.81	0.0	**1301**	51.68	0.00	**1301**	618.35	0.00
swiss42-C-3-0-b	**1505**	8.98	0.00	**1505**	0.94	0.0	**1505**	4.74	0.0	**1505**	4.39	0.00	**1505**	11.68	0.00
swiss42-C-3-1-b	**1344**	8.61	0.00	**1344**	7.43	0.0	**1344**	110.55	0.0	**1344**	364.67	0.00	**1344**	3060.01	0.00
swiss42-C-3-0-c	**1467**	1.95	0.00	**1467**	0.86	0.0	**1467**	3.20	0.0	**1467**	3.11	0.00	**1467**	9.97	0.00
swiss42-C-3-1-c	**1357**	4.64	0.00	**1357**	2.27	0.0	**1357**	42.07	0.0	**1357**	53.23	0.00	**1357**	413.68	0.00
swiss42-C-5-0-a	**1561**	114.85	0.00	**1561**	1.17	0.0	**1561**	1.97	0.0	**1561**	2.61	0.00	**1561**	2.98	0.00
swiss42-C-5-1-a	**1434**	14.17	0.00	**1434**	7.41	0.0	**1434**	76.76	0.0	**1434**	68.15	0.00	**1434**	32.47	0.00
swiss42-C-5-2-a	**1343**	6.53	0.00	**1343**	1.62	0.0	**1343**	46.08	0.0	**1343**	56.29	0.00	**1343**	317.80	0.00
swiss42-C-5-3-a	**1273**	0.72	0.00	**1273**	0.48	0.0	**1273**	9.73	0.0	**1273**	26.95	0.00	**1273**	490.63	0.00
swiss42-C-5-0-b	**1540**	1.16	0.00	**1540**	0.51	0.0	**1540**	1.50	0.0	**1540**	0.92	0.00	**1540**	2.81	0.00
swiss42-C-5-1-b	**1469**	25.25	0.00	**1469**	3.25	0.0	**1469**	42.63	0.0	**1469**	51.49	0.00	**1469**	81.91	0.00
swiss42-C-5-2-b	**1315**	11.85	0.00	**1315**	6.97	0.0	**1315**	138.71	0.0	**1315**	455.87	0.00	**1315**	1561.82	0.00
swiss42-C-5-3-b	**1273**	1.38	0.00	**1273**	1.24	0.0	**1273**	46.27	0.0	**1273**	44.60	0.00	**1273**	500.15	0.00
swiss42-C-5-0-c	**1532**	23.11	0.00	**1532**	1.48	0.0	**1532**	1.53	0.0	**1532**	1.72	0.00	**1532**	2.23	0.00
swiss42-C-5-1-c	**1334**	2.27	0.00	**1334**	1.83	0.0	**1334**	12.54	0.0	**1334**	14.54	0.00	**1334**	45.31	0.00
swiss42-C-5-2-c	**1321**	16.05	0.00	**1321**	7.98	0.0	**1321**	135.80	0.0	**1321**	104.28	0.00	**1321**	998.50	0.00
swiss42-C-5-3-c	**1301**	3.69	0.00	**1301**	9.01	0.0	**1301**	168.76	0.0	**1301**	201.27	0.00	**1301**	1712.30	0.00
Mean Values	4819.33	343.29	0.18	4819.33	19.94	0.0	4819.33	184.60	0.0	4823.61	341.11	0.17	6215.08	947.98	6.91

Table A.5: Solver results from the H2020, MTZ2, GP2, SSB2 and SST2 formulations for the 100-city random instances

	(H2020)			(MTZ2)			(GP2)			(SSB2)			(SST2)		
Instance	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP
kroA100-R-3-0-a	**38814**	1529.42	0.00	**38814**	481.27	0.00	**38814**	2615.01	0.00	**38814**	12730.00	0.00	**38814**	3589.75	0.00
kroA100-R-3-1-a	30321	18000.02	15.33	30098	18000.32	13.82	31115	18008.64	16.04	50722	18000.53	50.14	166471	18000.35	83.43
kroA100-R-5-0-a	**50192**	5198.86	0.00	**50192**	6521.38	0.00	**50192**	7342.72	0.00	**50192**	17679.22	0.00	**50192**	342.73	0.00
kroA100-R-5-1-a	38071	18000.63	19.02	37331	18000.13	16.52	37913	18000.31	14.82	37676	18000.18	17.00	61305	18000.23	44.21
kroA100-R-5-2-a	31930	18006.92	23.03	29520	18000.02	12.24	29466	18048.36	10.98	51095	18000.16	51.53	168822	18000.24	83.69
kroA100-R-5-3-a	27718	18004.76	19.25	25408	18000.03	10.42	46492	18001.91	51.51	44217	18000.30	51.06	165477	18000.35	100.00
kroB100-R-3-0-a	**37706**	2391.26	0.00	**37706**	3878.75	0.00	**37706**	7630.52	0.00	37706	18000.60	0.65	**37706**	2505.31	0.00
kroB100-R-3-1-a	29957	18000.01	13.82	29155	18004.42	10.19	30255	18005.61	13.79	47464	18000.12	46.91	40744	18133.48	32.96
kroB100-R-5-0-a	**50781**	2828.06	0.00	**50781**	525.89	0.00	**50781**	899.53	0.00	**50781**	3670.97	0.00	**50781**	303.26	0.00
kroB100-R-5-1-a	37334	18000.04	16.46	38014	18000.47	15.66	36377	18000.78	9.58	36516	18000.25	12.13	49666	18000.37	30.74
kroB100-R-5-2-a	33853	18000.29	25.65	28958	18000.02	7.67	31214	18021.41	15.01	43214	18000.10	40.19	161996	18002.18	82.81
kroB100-R-5-3-a	30391	18000.36	24.46	27805	18000.02	15.94	44170	18000.41	47.72	48052	18001.30	52.60	155681	18002.37	100.00
kroC100-R-3-0-a	**37953**	548.34	0.00	**37953**	510.87	0.00	**37953**	860.56	0.00	**37953**	1700.04	0.00	**37953**	1742.81	0.00
kroC100-R-3-1-a	30123	18000.01	17.55	28741	18000.02	11.52	30504	18001.13	16.40	37484	18030.41	35.11	166954	18005.23	84.01
kroC100-R-5-0-a	**50085**	551.16	0.00	**50085**	147.01	0.00	**50085**	434.16	0.00	**50085**	716.32	0.00	**50085**	322.33	0.00
kroC100-R-5-1-a	35676	18000.01	18.16	35007	18001.52	14.88	35207	18000.44	12.04	34557	18003.48	12.89	52300	18005.64	37.76
kroC100-R-5-2-a	30256	18000.03	20.34	28909	18000.04	11.61	30864	18000.33	18.18	43551	18000.11	43.68	160043	18000.37	83.10
kroC100-R-5-3-a	26471	18000.10	13.07	26960	18000.02	13.26	38751	18037.09	40.09	44348	18013.64	49.25	172566	18005.18	100.00
kroD100-R-3-0-a	**38110**	2881.08	0.00	**38110**	1451.89	0.00	**38110**	3591.43	0.00	38342	18051.44	1.42	**38110**	2768.27	0.00
kroD100-R-3-1-a	28219	18000.02	13.93	28383	18000.01	13.19	29374	18006.88	15.71	44179	18000.77	46.95	158163	18000.27	83.24
kroD100-R-5-0-a	**49100**	1160.58	0.00	**49100**	476.71	0.00	**49100**	762.19	0.00	**49100**	1333.90	0.00	**49100**	433.45	0.00
kroD100-R-5-1-a	36884	18000.06	19.10	37494	18000.60	18.18	35320	18000.91	11.12	35023	18000.20	13.30	63852	18000.20	48.51
kroD100-R-5-2-a	32812	18000.01	24.84	30612	18000.03	16.21	29931	18008.57	14.38	41471	18000.11	41.50	152942	18000.21	82.12
kroD100-R-5-3-a	29928	18000.01	24.42	28226	18000.03	18.16	47685	18042.09	52.42	50760	18000.12	56.42	159223	18001.89	100.00
kroE100-R-3-0-a	**37935**	175.79	0.00	**37935**	376.36	0.00	**37935**	841.16	0.00	**37935**	2201.51	0.00	**37935**	2344.99	0.00
kroE100-R-3-1-a	29734	18000.95	10.38	29513	18000.06	8.03	30686	18000.84	11.69	49059	18039.32	46.94	53772	18010.63	47.61
kroE100-R-5-0-a	**54197**	182.57	0.00	**54197**	92.04	0.00	**54197**	267.38	0.00	**54197**	919.31	0.00	**54197**	526.67	0.00
kroE100-R-5-1-a	40010	18000.01	15.68	39511	18000.02	13.56	39231	18000.33	9.98	41833	18028.94	17.85	68151	18000.23	46.85
kroE100-R-5-2-a	35426	18000.02	26.78	31005	18003.80	12.33	32373	18001.05	15.77	51109	18000.27	48.74	168782	18000.36	83.10
kroE100-R-5-3-a	29890	18000.04	19.52	27648	18009.76	12.16	46393	18000.37	48.27	50123	18000.12	53.24	171511	18000.35	100.00
Mean Values	36329.23	12582.05	12.69	35439.03	12482.78	8.85	38606.47	12847.74	14.85	44251.93	14570.79	26.32	98776.47	12501.66	48.47

Table A.6: Solver results from the H2020, MTZ2, GP2, SSB2 and SST2 formulations for the 100-city clustered instances

	(H2020)			(MTZ2)			(GP2)			(SSB2)			(SST2)		
Instance	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP	Obj	Runtime	GAP
kroA100-C-3-0-a	**24049**	1306.29	0.00	**24049**	910.67	0.00	**24049**	3595.28	0.00	24049*	18000.15	0.91	25831	18000.20	7.26
kroA100-C-3-1-a	22845*	18000.16	3.67	**22845**	1842.05	0.00	33352	18018.03	34.35	33090	18019.00	34.74	110741	18000.45	100.00
kroA100-C-5-0-a	24863	18002.62	7.14	**24745**	792.59	0.00	**24745**	1272.17	0.00	**24745**	6052.80	0.00	**24745**	342.80	0.00
kroA100-C-5-1-a	22589**	18000.86	3.77	22589**	18000.03	0.53	22617	18055.75	3.94	22696	18000.14	5.54	23170	18000.22	2.91
kroA100-C-5-2-a	21955**	18000.27	1.93	21955**	18000.03	1.79	22092	18000.80	4.69	26097	18000.22	21.76	85169	18000.41	100.00
kroA100-C-5-3-a	**21443**	5122.54	0.00	**21443**	995.29	0.00	27099	18203.21	23.12	40360	18000.17	49.76	153741	18056.47	100.00
kroB100-C-3-0-a	**24887**	284.07	0.00	**24887**	101.58	0.00	**24887**	594.38	0.00	**24887**	1021.45	0.00	**24887**	1628.56	0.00
kroB100-C-3-1-a	**22141**	1026.63	0.00	**22141**	318.26	0.00	**22141**	12030.87	0.00	27636	18000.13	21.49	34277	18000.22	35.80
kroB100-C-5-0-a	**24794**	255.98	0.00	**24794**	33.42	0.00	**24794**	296.18	0.00	**24794**	695.56	0.00	**24794**	1459.44	0.00
kroB100-C-5-1-a	**23159**	12289.15	0.00	**23159**	1348.61	0.00	**23159**	15003.68	0.00	23159*	18026.79	1.98	28413	18000.53	18.57
kroB100-C-5-2-a	**22206**	2340.37	0.00	**22206**	943.26	0.00	22206*	18058.27	0.82	28790	18008.42	24.94	89834	18004.43	100.00
kroB100-C-5-3-a	22141*	18000.02	1.14	**22141**	9827.63	0.00	37108	18083.18	41.95	34190	18000.42	38.37	105484	18002.95	100.00
kroC100-C-3-0-a	**21340**	486.50	0.00	**21340**	241.90	0.00	**21340**	1554.06	0.00	**21340**	3140.97	0.00	**21340**	9702.62	0.00
kroC100-C-3-1-a	**20910**	268.41	0.00	**20910**	191.92	0.00	**20910**	11988.51	0.00	28986	18000.12	30.14	32162	18000.25	35.79
kroC100-C-5-0-a	**24040**	127.96	0.00	**24040**	11.71	0.00	**24040**	390.05	0.00	**24040**	1290.52	0.00	**24040**	13897.48	0.00
kroC100-C-5-1-a	22827*	18000.02	0.74	**22827**	394.25	0.00	**22827**	4775.55	0.00	25380	18023.30	12.72	23903	18010.78	5.38
kroC100-C-5-2-a	**22383**	1563.18	0.00	**22383**	277.43	0.00	**22383**	16634.30	0.00	29161	18001.03	25.44	106626	18000.37	100.00
kroC100-C-5-3-a	21278*	18000.02	1.83	**21278**	429.48	0.00	62197	18029.65	66.60	34331	18000.11	40.18	152104	18000.36	100.00
kroD100-C-3-0-a	23809*	18000.03	2.08	**23809**	1309.37	0.00	**23809**	11357.88	0.00	23809*	18002.74	1.29	**23809**	1983.27	0.00
kroD100-C-3-1-a	**21944**	5655.96	0.00	**21944**	3400.68	0.00	21944*	18000.55	1.30	24938	18000.16	16.05	32663	18000.21	33.03
kroD100-C-5-0-a	28297	18000.01	6.98	**28228**	1355.93	0.00	**28228**	1331.70	0.00	**28228**	3300.78	0.00	**28228**	214.36	0.00
kroD100-C-5-1-a	25191	18000.03	5.41	25102*	18000.02	4.68	25191	18000.44	6.04	25265	18005.74	10.07	**25102**	17057.27	0.00
kroD100-C-5-2-a	22284*	18003.16	0.94	**22284**	5682.79	0.00	22284*	18000.49	2.85	23414	18000.10	8.73	24067	18000.23	8.23
kroD100-C-5-3-a	21744**	18000.03	3.45	21744**	18000.02	1.69	30850	18022.19	32.42	33825	18000.12	38.56	155087	18000.39	100.00
kroE100-C-3-0-a	**24383**	1848.87	0.00	**24383**	293.55	0.00	**24383**	1090.24	0.00	**24383**	8142.83	0.00	25029	18000.19	3.03
kroE100-C-3-1-a	**22121**	1266.15	0.00	**22121**	426.00	0.00	22295	18025.00	1.93	28367	18000.14	23.95	107729	18007.47	100.00
kroE100-C-5-0-a	**26440**	15836.42	0.00	**26440**	31.80	0.00	**26440**	461.66	0.00	**26440**	318.89	0.00	**26440**	505.87	0.00
kroE100-C-5-1-a	**23611**	61.40	0.00	**23611**	140.49	0.00	**23611**	1996.11	0.00	**23611**	5006.73	0.00	**23611**	13943.87	0.00
kroE100-C-5-2-a	**23130**	698.38	0.00	**23130**	336.22	0.00	**23130**	8487.37	0.00	23250	18020.65	2.44	24787	18000.22	7.73
kroE100-C-5-3-a	22455*	18000.01	0.84	**22455**	17082.60	0.00	35688	18000.38	38.92	35021	18005.50	38.80	143246	18000.27	100.00
Mean Values	23175.3	9481.52	1.33	23166.1	4023.99	0.29	26326.63	10911.93	8.63	27276.07	13569.52	14.93	57701.97	14027.41	38.59

A.2 CluVRP-d Results

Tables A.7 to A.14 show the results obtained in the computational tests. Tables A.7 to A.10 show the results obtained running the proposed formulations using Gurobi where, for each vehicle fleet, there are the objective value (column ObjVal), the total runtime of the solving process (column Runtime) and the GAP presented in the solver output (column GAP, presented as percentage). The other tables give the results obtained by the proposed heuristic, including the best result found over the five times the heuristic was run, the average objective value and the average runtime.

Table A.7: Small random instances - Local Timing Model Case

	F1			F2			F3			F4		
Name	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP
HVRP_kroA20-R-3-0	22250.39	488.52	0.0%	22250.39	115.05	0.0%	23262.12	941.92	0.0%	22250.39	2334.20	0.0%
HVRP_kroA20-R-3-1	21204.94	3600.01	34.0%	22583.09	3600.01	37.3%	21236.01	3600.01	35.4%	22107.24	3600.01	41.8%
HVRP_kroA30-R-3-0	30898.50	3600.08	16.1%	29637.50	3600.03	11.0%	29987.29	3600.03	14.4%	32839.54	3600.04	22.9%
HVRP_kroA30-R-3-1	30316.61	3600.02	50.8%	28277.10	3600.01	48.7%	29350.34	3600.03	47.4%	30160.16	3600.08	50.7%
HVRP_kroB20-R-3-0	23072.27	3600.01	6.0%	23072.27	907.16	00.0%	23072.27	726.16	0.0%	25495.08	3600.01	18.0%
HVRP_kroB20-R-3-1	20844.47	3600.01	48.2%	20880.37	3600.01	45.4%	21185.76	3600.01	46.0%	22447.39	3600.04	50.5%
HVRP_kroB30-R-3-0	33340.79	3600.02	28.4%	31907.03	3600.03	25.4%	31896.65	3600.03	19.2%	36563.44	3600.04	36.1%
HVRP_kroB30-R-3-1	32453.84	3600.02	59.6%	27483.31	3600.02	52.9%	29499.10	3600.02	57.0%	35658.41	3600.03	64.3%
HVRP_kroC20-R-3-0	21468.27	3600.03	10.0%	21468.26	3600.01	5.9%	21468.27	3600.03	5.8%	22594.82	3600.02	20.3%
HVRP_kroC20-R-3-1	19534.82	3600.01	30.5%	19534.82	3600.00	30.2%	19534.82	3600.01	28.4%	21343.67	3600.04	38.7%
HVRP_kroC30-R-3-0	31077.50	3600.02	27.1%	28125.16	3600.06	18.6%	28125.16	3600.05	18.4%	31413.32	3600.03	28.3%
HVRP_kroC30-R-3-1	28892.54	3600.01	51.5%	29106.81	3600.02	52.0%	26068.46	3600.02	45.7%	33268.26	3600.04	58.1%
HVRP_kroD20-R-3-0	24906.42	3600.01	18.2%	24906.42	3600.01	18.0%	25606.42	3600.01	21.0%	28000.50	3600.02	28.5%
HVRP_kroD20-R-3-1	25161.83	3600.01	52.8%	24115.63	3600.01	50.7%	24815.62	3600.01	52.6%	27209.54	3600.02	56.3%
HVRP_kroD30-R-3-0	38689.55	3600.04	36.1%	37074.88	3600.03	31.4%	36929.31	3600.03	29.9%	42901.76	3600.05	42.7%
HVRP_kroD30-R-3-1	36046.30	3600.08	60.7%	33822.53	3600.02	58.3%	33592.66	3600.01	57.6%	40002.09	3600.06	65.1%
HVRP_kroE20-R-3-0	21459.54	1150.13	0.0%	21459.54	79.37	0.0%	21459.54	191.29	0.0%	24324.74	3600.02	12.6%
HVRP_kroE20-R-3-1	20892.85	3600.01	37.3%	20704.12	3600.01	33.1%	20777.76	3600.01	36.4%	24266.13	3600.01	46.9%
HVRP_kroE30-R-3-0	37332.98	3600.03	36.0%	35589.90	3600.03	29.2%	35377.06	3600.02	31.6%	40619.90	3600.03	41.6%
HVRP_kroE30-R-3-1	35421.37	3600.04	59.3%	33115.15	3600.06	57.0%	33716.56	3600.04	56.3%	40983.91	3600.03	66.5%

Table A.8: Small clustered instances - Local Timing Model Case

	F1			F2			F3			F4		
Name	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP
HVRP_kroA20-C-3-0	19581.30	3600.02	28.6%	19727.09	3600.02	23.7%	19751.07	3600.01	24.6%	20972.39	3600.02	34.9%
HVRP_kroA20-C-3-1	20940.94	3600.01	47.0%	19935.31	3600.03	40.6%	19609.81	3600.03	39.1%	19727.10	3600.03	36.2%
HVRP_kroA30-C-3-0	26227.31	3600.03	33.3%	27801.76	3600.03	36.1%	24987.96	3600.02	28.9%	29390.15	3600.04	43.8%
HVRP_kroA30-C-3-1	27889.37	3600.01	49.4%	29091.19	3600.05	51.3%	25012.72	3600.01	43.6%	29378.31	3600.03	52.4%
HVRP_kroB20-C-3-0	20595.70	3600.01	43.5%	20595.70	3600.01	39.9%	20595.70	3600.01	38.6%	22986.99	3600.02	49.2%
HVRP_kroB20-C-3-1	22241.94	3600.01	55.6%	20025.12	3600.01	48.6%	20991.35	3600.01	51.0%	22241.89	3600.02	54.4%
HVRP_kroB30-C-3-0	28224.70	3600.04	47.7%	28636.36	3600.03	48.7%	28092.47	3600.02	47.5%	32777.79	3600.04	55.1%
HVRP_kroB30-C-3-1	29044.53	3600.02	57.4%	27091.65	3600.02	52.8%	28059.11	3600.02	54.5%	33911.27	3600.05	63.4%
HVRP_kroC20-C-3-0	19416.84	3600.01	09.0%	19416.84	3600.01	15.6%	19416.84	3600.01	01.7%	21487.59	3600.02	23.0%
HVRP_kroC20-C-3-1	18734.63	3600.01	30.6%	18734.63	3600.02	29.9%	18734.63	3600.02	30.5%	22082.11	3600.01	43.1%
HVRP_kroC30-C-3-0	27880.82	3600.09	48.9%	24912.81	3600.07	42.2%	24413.46	3600.02	40.8%	30517.85	3600.06	53.5%
HVRP_kroC30-C-3-1	30608.70	3600.03	56.6%	24260.11	3600.02	43.5%	24272.25	3600.03	43.4%	30522.45	3600.03	56.0%
HVRP_kroD20-C-3-0	25018.15	3600.02	42.8%	23880.74	3600.01	36.7%	24580.74	3600.01	38.7%	26946.11	3600.01	51.3%
HVRP_kroD20-C-3-1	24235.90	3600.01	49.8%	23922.48	3600.01	49.4%	24501.76	3600.01	50.7%	27329.62	3600.02	55.6%
HVRP_kroD30-C-3-0	35894.26	3600.07	54.3%	32439.98	3600.07	49.0%	33345.93	3600.02	52.5%	40707.37	3600.05	60.5%
HVRP_kroD30-C-3-1	36244.04	3600.03	62.9%	32335.31	3600.01	55.0%	33082.67	3600.01	57.8%	39658.32	3600.03	67.8%
HVRP_kroE20-C-3-0	21641.27	3600.01	40.4%	21646.35	3600.01	36.0%	21365.07	3600.01	32.5%	25396.97	3600.02	51.3%
HVRP_kroE20-C-3-1	21106.63	3600.01	48.3%	20674.91	3600.01	35.4%	20807.17	3600.01	37.4%	24209.67	3600.01	48.2%
HVRP_kroE30-C-3-0	32722.01	3600.05	54.8%	33921.91	3600.04	56.6%	33428.54	3600.01	55.1%	38996.15	3600.08	62.4%
HVRP_kroE30-C-3-1	36198.87	3600.04	64.6%	31624.88	3600.03	56.9%	32285.06	3600.03	58.1%	39046.92	3600.03	65.8%

Table A.9: Small random instances - Global Timing Model Case

	F1			F2			F3			F4		
Name	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP
HVRP_kroA20-R-3-0	22250.39	1689.59	0.0%	22250.39	186.22	0.0%	23262.12	2793.87	0.0%	22250.39	2430.94	00.0%
HVRP_kroA20-R-3-1	20136.04	3600.00	36.5%	22664.88	3600.00	44.6%	22245.58	3600.00	38.9%	22274.64	3600.01	41.8%
HVRP_kroA30-R-3-0	31488.50	3600.02	29.1%	29637.50	3600.05	21.9%	29128.65	3600.01	10.1%	31380.52	3600.01	29.4%
HVRP_kroA30-R-3-1	29528.87	3600.02	51.8%	28795.31	3600.04	50.8%	28734.85	3600.01	49.8%	30165.96	3600.01	53.4%
HVRP_kroB20-R-3-0	23072.26	3600.02	9.5%	23072.27	2824.79	0.0%	23072.27	3600.00	05.2%	25239.82	3600.01	19.9%
HVRP_kroB20-R-3-1	21076.29	3600.00	49.3%	21192.56	3600.00	47.1%	20880.36	3600.00	49.0%	22447.37	3600.01	55.1%
HVRP_kroB30-R-3-0	32457.04	3600.01	29.7%	32191.36	3600.03	26.5%	31760.58	3600.01	25.5%	36007.60	3600.02	38.0%
HVRP_kroB30-R-3-1	29878.41	3600.02	58.8%	27483.29	3600.01	55.3%	28501.44	3600.01	57.2%	33849.66	3600.02	63.5%
HVRP_kroC20-R-3-0	21468.27	3600.04	15.9%	21468.27	3600.01	15.0%	21468.26	3600.00	13.5%	22594.82	3600.01	21.7%
HVRP_kroC20-R-3-1	21360.62	3600.01	39.6%	21721.53	3600.00	38.2%	19534.82	3600.01	29.4%	21676.79	3600.01	40.6%
HVRP_kroC30-R-3-0	30518.19	3600.01	26.1%	28125.14	3600.01	19.8%	28125.14	3600.02	19.0%	32606.32	3600.01	32.0%
HVRP_kroC30-R-3-1	33366.28	3600.01	59.1%	27533.68	3600.01	49.8%	26111.06	3600.01	47.0%	31657.86	3600.02	56.1%
HVRP_kroD20-R-3-0	24997.55	3600.03	21.0%	24997.55	3600.01	19.6%	25606.42	3600.01	22.7%	28613.78	3600.02	31.1%
HVRP_kroD20-R-3-1	24969.16	3600.01	53.2%	24384.32	3600.00	51.7%	25023.26	3600.00	53.2%	27001.05	3600.01	57.1%
HVRP_kroD30-R-3-0	40728.33	3600.02	40.8%	39184.93	3600.02	35.5%	37110.57	3600.01	31.0%	43960.51	3600.01	45.0%
HVRP_kroD30-R-3-1	36902.79	3600.01	66.0%	34302.98	3600.03	62.8%	33499.52	3600.01	59.3%	42085.18	3600.02	70.2%
HVRP_kroE20-R-3-0	21459.54	2758.03	0.0%	21459.54	947.21	00.0%	21459.54	432.58	00.0%	24508.02	3600.01	13.1%
HVRP_kroE20-R-3-1	20720.64	3600.01	44.5%	20541.20	3600.01	41.8%	20541.20	3600.01	42.3%	24260.33	3600.01	53.7%
HVRP_kroE30-R-3-0	38117.90	3600.01	38.2%	38800.15	3600.01	39.5%	35403.94	3600.01	33.4%	41002.64	3600.02	43.7%
HVRP_kroE30-R-3-1	36348.33	3600.01	62.5%	34130.00	3600.01	60.1%	34644.61	3600.01	59.2%	40237.73	3600.02	66.1%

Table A.10: Small clustered instances - Global Timing Model Case

	F1			F2			F3			F4		
Name	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP	ObjVal	Runtime	GAP
HVRP_kroA20-C-3-0	19535.30	3600.01	30.0%	19727.10	3600.01	30.5%	19751.07	3600.07	32.2%	19750.85	3600.01	31.7%
HVRP_kroA20-C-3-1	19581.31	3600.00	41.1%	19609.81	3600.01	37.7%	20635.31	3600.01	45.6%	20686.74	3600.01	46.2%
HVRP_kroA30-C-3-0	30180.42	3600.01	45.6%	26855.93	3600.01	35.8%	25161.14	3600.01	33.7%	30634.59	3600.01	47.6%
HVRP_kroA30-C-3-1	30212.12	3600.03	54.4%	24904.24	3600.01	42.5%	25054.82	3600.01	43.0%	30124.86	3600.01	54.0%
HVRP_kroB20-C-3-0	20595.70	3600.00	45.5%	20595.69	3600.01	45.9%	20590.15	3600.01	46.1%	22809.91	3600.01	52.0%
HVRP_kroB20-C-3-1	20025.10	3600.01	49.8%	20025.12	3600.00	52.7%	20025.11	3600.00	52.3%	22241.94	3600.01	57.8%
HVRP_kroB30-C-3-0	28144.16	3600.03	51.4%	29036.26	3600.01	52.8%	28092.46	3600.01	51.0%	32477.85	3600.02	57.6%
HVRP_kroB30-C-3-1	29381.74	3600.02	58.6%	27175.74	3600.01	54.7%	28555.68	3600.01	57.2%	33679.98	3600.02	63.9%
HVRP_kroC20-C-3-0	19536.83	3600.04	23.9%	19536.83	3600.00	16.0%	19416.83	3600.01	13.8%	21713.83	3600.07	28.5%
HVRP_kroC20-C-3-1	18734.62	3600.03	34.2%	18734.61	3600.01	31.2%	18734.63	3600.00	31.0%	21708.28	3600.02	43.2%
HVRP_kroC30-C-3-0	28057.00	3600.01	50.1%	25984.23	3600.01	45.9%	24413.45	3600.01	41.2%	31419.46	3600.02	55.5%
HVRP_kroC30-C-3-1	27583.34	3600.01	51.6%	24272.25	3600.01	44.5%	24551.81	3600.01	44.3%	29295.06	3600.01	54.4%
HVRP_kroD20-C-3-0	25459.82	3600.08	47.1%	23880.74	3600.00	41.1%	24580.74	3600.01	44.9%	26946.11	3600.04	49.8%
HVRP_kroD20-C-3-1	24686.97	3600.01	52.1%	23863.27	3600.00	49.9%	24563.26	3600.02	51.5%	26474.16	3600.01	55.0%
HVRP_kroD30-C-3-0	36826.54	3600.01	60.8%	33620.12	3600.01	57.0%	33370.48	3600.01	56.5%	44238.57	3600.01	67.3%
HVRP_kroD30-C-3-1	39678.87	3600.00	68.4%	33998.61	3600.01	62.4%	33293.47	3600.01	62.0%	40322.30	3600.02	68.9%
HVRP_kroE20-C-3-0	21641.26	3600.00	50.8%	21365.07	3600.00	47.3%	21365.07	3600.00	48.2%	25036.00	3600.01	57.6%
HVRP_kroE20-C-3-1	20706.39	3600.01	44.7%	20715.04	3600.15	45.4%	20715.04	3600.00	44.9%	24209.67	3600.01	53.8%
HVRP_kroE30-C-3-0	32805.64	3600.02	56.4%	35176.42	3600.01	59.5%	33521.95	3600.01	56.8%	38482.31	3600.02	63.0%
HVRP_kroE30-C-3-1	32211.02	3600.00	60.0%	36599.19	3600.02	64.9%	32285.01	3600.01	59.3%	38618.87	3600.01	66.6%

Table A.11: Small random instances - Local Timing Model Heuristic

	F1			F2			F3			F4		
Name	Best Obj	Mean Obj	Runtime	Best Obj	Mean Obj	Runtime	Best Obj	Mean Obj	Runtime	Best Obj	Mean Obj	Runtime
HVRP_kroA20-R-3-0	22250.39	22379.12	22.21	22841.18	22884.23	27.70	23541.18	23541.18	20.48	22250.39	22498.83	25.04
HVRP_kroA20-R-3-1	20311.07	20455.32	42.83	20536.04	20536.04	46.87	21236.01	21236.01	38.30	20446.72	20949.56	31.59
HVRP_kroA30-R-3-0	29539.38	29958.93	28.26	29539.38	29920.01	34.52	29908.39	30489.72	47.70	31122.93	31813.65	34.91
HVRP_kroA30-R-3-1	27334.86	27352.35	51.36	26740.46	27059.48	62.15	26650.23	26803.67	79.91	29861.98	30087.64	58.75
HVRP_kroB20-R-3-0	23398.82	23751.36	27.27	23072.27	23710.77	29.17	23072.27	23633.99	33.11	25310.33	25413.14	22.74
HVRP_kroB20-R-3-1	21185.76	21186.93	52.02	21185.76	21190.43	48.81	21071.28	21164.03	52.12	22447.39	22447.39	36.80
HVRP_kroB30-R-3-0	31617.73	31898.53	39.84	31348.55	31749.93	42.97	31547.69	31819.49	52.39	34392.10	34744.92	35.09
HVRP_kroB30-R-3-1	28129.97	28129.97	57.78	27483.31	27489.46	98.16	27466.80	27480.01	75.72	30821.69	31079.08	64.41
HVRP_kroC20-R-3-0	21468.26	21468.26	31.13	21468.26	21468.26	26.81	21468.26	21468.26	27.74	22594.82	22594.82	29.51
HVRP_kroC20-R-3-1	19534.82	19534.82	64.77	19534.82	19534.82	46.33	19534.82	19534.82	47.50	21343.67	21343.67	33.26
HVRP_kroC30-R-3-0	29504.88	29728.64	35.13	28516.63	29000.07	45.20	27915.52	28555.89	32.10	31401.99	31682.21	37.67
HVRP_kroC30-R-3-1	27415.92	27424.43	61.74	25990.11	26087.48	62.12	25959.25	25983.94	72.49	29923.47	29936.22	43.46
HVRP_kroD20-R-3-0	24906.42	24976.56	25.21	24906.42	24906.42	25.58	25606.42	25745.77	23.56	27327.94	27340.15	22.95
HVRP_kroD20-R-3-1	24115.63	24143.21	39.03	24115.63	24115.63	31.06	24815.62	24815.62	32.12	26909.30	26909.30	35.17
HVRP_kroD30-R-3-0	39455.85	39853.71	27.64	37275.37	37584.48	36.45	36599.45	36604.95	41.08	42198.35	42198.35	30.27
HVRP_kroD30-R-3-1	35908.97	36270.59	47.81	32628.99	32628.99	44.38	33042.60	33208.05	67.37	40000.63	40023.11	32.65
HVRP_kroE20-R-3-0	21459.54	21924.85	35.83	21977.03	22028.35	35.63	21459.54	21795.70	31.36	24724.74	25003.78	24.86
HVRP_kroE20-R-3-1	20541.20	20541.20	59.59	20541.20	20577.09	53.80	20541.20	20541.20	59.29	24260.34	24311.86	44.02
HVRP_kroE30-R-3-0	35413.52	35593.87	35.19	34258.54	34686.01	48.10	34958.54	35319.51	48.68	39718.34	39858.45	32.91
HVRP_kroE30-R-3-1	32998.61	33208.32	68.96	32184.35	32190.08	82.67	32908.53	32919.62	92.25	37642.29	37787.23	63.14

Table A.12: Small cluster instances - Local Timing Model Heuristic

	F1			F2			F3			F4		
Name	Best Obj	Mean Obj	Runtime	Best Obj	Mean Obj	Runtime	Best Obj	Mean Obj	Runtime	Best Obj	Mean Obj	Runtime
HVRP_kroA20-C-3-0	19325.51	19575.32	41.13	19725.51	19725.51	39.19	19751.07	19751.07	54.30	19727.10	20360.02	41.08
HVRP_kroA20-C-3-1	19325.51	19335.01	73.78	19609.81	19609.81	82.32	19609.81	19609.81	83.30	19727.10	20307.64	52.40
HVRP_kroA30-C-3-0	26227.31	26227.31	62.40	24987.96	25545.83	73.76	24914.85	25035.07	72.14	29214.96	29248.64	54.95
HVRP_kroA30-C-3-1	26227.31	26227.31	75.97	24987.96	25381.80	99.98	24781.22	24915.17	92.51	29214.96	29229.94	65.06
HVRP_kroB20-C-3-0	20595.70	20595.70	31.31	20595.70	20595.70	31.57	20595.70	20595.70	31.62	22794.56	22855.56	32.77
HVRP_kroB20-C-3-1	20025.12	20070.93	55.93	20025.12	20095.29	58.41	20025.12	20093.83	52.34	22241.89	22241.89	45.50
HVRP_kroB30-C-3-0	27795.88	27972.61	56.42	27819.56	27918.28	78.68	27819.56	27918.28	81.90	31861.66	32320.99	58.00
HVRP_kroB30-C-3-1	27589.75	27611.77	81.06	27067.58	27072.04	126.26	27067.58	27072.04	124.10	30487.49	30766.17	116.54
HVRP_kroC20-C-3-0	19416.84	19464.84	53.10	19416.84	19464.84	53.46	19416.84	19416.84	51.31	21669.12	21676.70	46.67
HVRP_kroC20-C-3-1	18734.63	18734.63	78.28	18734.63	18734.63	78.05	18734.63	18734.63	84.23	21191.46	21191.46	47.29
HVRP_kroC30-C-3-0	26592.69	27017.39	54.15	24413.46	24413.46	91.27	24413.46	24413.46	89.73	29841.89	30038.25	51.88
HVRP_kroC30-C-3-1	26618.53	26750.71	83.58	24301.12	24331.85	121.67	24272.25	24289.57	118.14	29323.81	29455.07	68.03
HVRP_kroD20-C-3-0	23821.68	23821.68	31.84	23880.74	23880.74	26.60	24580.74	24605.35	27.49	26474.16	26514.79	27.14
HVRP_kroD20-C-3-1	23801.77	23809.73	42.26	23801.77	23801.77	43.79	24501.76	24501.76	43.18	26474.16	26474.16	35.86
HVRP_kroD30-C-3-0	36057.26	36738.86	48.89	32439.98	32439.98	51.82	33364.66	33487.08	47.49	39790.27	39790.27	45.19
HVRP_kroD30-C-3-1	36458.61	36618.43	56.67	32439.99	32439.99	74.24	33130.24	33228.02	71.29	39658.32	39763.88	56.25
HVRP_kroE20-C-3-0	21365.07	21365.07	33.33	21365.07	21365.07	32.82	21365.07	21365.07	32.60	25113.94	25235.80	34.21
HVRP_kroE20-C-3-1	20674.91	20674.91	64.53	20674.91	20674.91	63.96	20674.91	20674.91	64.42	24209.67	24445.85	45.92
HVRP_kroE30-C-3-0	32925.18	32960.76	66.60	32135.11	32234.15	55.09	32648.99	32769.20	50.32	37303.14	37731.70	47.28
HVRP_kroE30-C-3-1	32211.04	32299.64	92.74	31585.06	31688.10	85.90	32221.94	32262.37	81.41	36882.19	37033.16	70.77

Table A.13: 100 vertices instances - Local Timing Model Heuristic

Name	Best Obj	Mean Obj	Runtime	Name	Best Obj	Mean Obj	Runtime
HVRP_kroA100-R-3-0	82965.25	84716.88	566.30	**HVRP_kroA100-C-3-0**	77047.10	78911.32	1000.21
HVRP_kroA100-R-3-1	80037.29	80905.67	863.43	**HVRP_kroA100-C-3-1**	77669.52	78251.13	1084.03
HVRP_kroA100-R-5-0	86839.55	87733.86	456.91	**HVRP_kroA100-C-5-0**	77896.63	79731.61	756.36
HVRP_kroA100-R-5-1	81737.12	83112.22	584.48	**HVRP_kroA100-C-5-1**	77190.11	79123.59	1003.29
HVRP_kroA100-R-5-2	78876.82	80345.72	757.31	**HVRP_kroA100-C-5-2**	76595.73	78777.38	1076.31
HVRP_kroA100-R-5-3	76455.25	78619.65	951.93	**HVRP_kroA100-C-5-3**	78098.82	78987.89	1115.16
HVRP_kroB100-R-3-0	78314.17	80442.87	580.97	**HVRP_kroB100-C-3-0**	73807.22	74519.94	1094.02
HVRP_kroB100-R-3-1	75945.38	77257.68	1037.06	**HVRP_kroB100-C-3-1**	72097.44	73218.14	1566.02
HVRP_kroB100-R-5-0	83210.32	85345.77	464.17	**HVRP_kroB100-C-5-0**	74086.24	75666.71	826.35
HVRP_kroB100-R-5-1	76292.50	78897.98	699.55	**HVRP_kroB100-C-5-1**	73964.33	75582.60	1101.06
HVRP_kroB100-R-5-2	72678.30	75415.76	979.93	**HVRP_kroB100-C-5-2**	71701.42	72803.59	1334.38
HVRP_kroB100-R-5-3	72644.04	75230.78	1293.95	**HVRP_kroB100-C-5-3**	73155.04	73850.43	1604.88
HVRP_kroC100-R-3-0	92534.78	93163.36	463.14	**HVRP_kroC100-C-3-0**	88674.62	88874.93	907.59
HVRP_kroC100-R-3-1	89622.30	89945.03	757.15	**HVRP_kroC100-C-3-1**	88579.12	88978.98	996.62
HVRP_kroC100-R-5-0	96959.97	98513.93	402.53	**HVRP_kroC100-C-5-0**	89337.35	89995.65	753.74
HVRP_kroC100-R-5-1	91397.01	92000.52	564.88	**HVRP_kroC100-C-5-1**	89496.15	89955.00	847.02
HVRP_kroC100-R-5-2	89331.79	90736.23	732.19	**HVRP_kroC100-C-5-2**	88967.78	89192.98	883.58
HVRP_kroC100-R-5-3	88734.27	88923.13	876.28	**HVRP_kroC100-C-5-3**	88590.94	88751.12	958.52
HVRP_kroD100-R-3-0	94272.59	95128.63	576.79	**HVRP_kroD100-C-3-0**	89826.05	90545.86	830.09
HVRP_kroD100-R-3-1	91300.33	91962.04	808.23	**HVRP_kroD100-C-3-1**	88297.48	89852.17	1047.84
HVRP_kroD100-R-5-0	98811.02	100025.64	457.56	**HVRP_kroD100-C-5-0**	91811.03	92908.02	870.06
HVRP_kroD100-R-5-1	91808.28	93351.24	614.00	**HVRP_kroD100-C-5-1**	90473.04	91649.40	913.08
HVRP_kroD100-R-5-2	90817.32	91511.26	857.89	**HVRP_kroD100-C-5-2**	89306.11	90250.49	1129.07
HVRP_kroD100-R-5-3	90338.62	90759.46	1010.41	**HVRP_kroD100-C-5-3**	88277.72	89366.89	1214.77
HVRP_kroE100-R-3-0	93005.61	95525.58	627.36	**HVRP_kroE100-C-3-0**	87518.53	88937.63	1049.38
HVRP_kroE100-R-3-1	90057.22	92363.09	939.88	**HVRP_kroE100-C-3-1**	85698.65	87670.98	1412.73
HVRP_kroE100-R-5-0	99267.37	101076.87	485.24	**HVRP_kroE100-C-5-0**	88114.27	90391.56	944.13
HVRP_kroE100-R-5-1	92843.38	96403.49	629.23	**HVRP_kroE100-C-5-1**	86727.27	88267.22	1125.75
HVRP_kroE100-R-5-2	89523.27	93100.53	868.58	**HVRP_kroE100-C-5-2**	86626.82	87239.89	1303.81
HVRP_kroE100-R-5-3	86534.22	91069.99	1105.55	**HVRP_kroE100-C-5-3**	86256.35	87246.57	1470.31

Table A.14: 200 vertices instances - Local Timing Model Heuristic

Name	Best Obj	Mean Obj	Runtime	Name	Best Obj	Mean Obj	Runtime
HVRP_kroA200-R-3-0	168170.02	169907.66	2706.17	**HVRP_kroA200-C-3-0**	159781.73	160656.01	4532.83
HVRP_kroA200-R-3-1	161010.96	162184.80	3921.07	**HVRP_kroA200-C-3-1**	159457.28	160717.18	5402.17
HVRP_kroA200-R-5-0	173161.66	174524.28	2198.93	**HVRP_kroA200-C-5-0**	160342.24	161215.83	4099.79
HVRP_kroA200-R-5-1	165253.96	166307.80	2971.57	**HVRP_kroA200-C-5-1**	160052.97	160966.68	4589.41
HVRP_kroA200-R-5-2	161840.09	163080.73	3669.24	**HVRP_kroA200-C-5-2**	160433.87	160861.68	4987.47
HVRP_kroA200-R-5-3	160427.97	160923.96	4804.22	**HVRP_kroA200-C-5-3**	159767.08	160454.24	5079.56
HVRP_kroB200-R-3-0	163017.81	164459.92	2918.93	**HVRP_kroB200-C-3-0**	152620.80	153522.26	5233.56
HVRP_kroB200-R-3-1	153007.75	155015.08	4424.37	**HVRP_kroB200-C-3-1**	150042.73	152515.97	6046.16
HVRP_kroB200-R-5-0	166716.26	167813.98	2193.11	**HVRP_kroB200-C-5-0**	154827.87	155991.45	5594.70
HVRP_kroB200-R-5-1	157993.17	161186.71	3304.12	**HVRP_kroB200-C-5-1**	154074.51	154706.43	5721.67
HVRP_kroB200-R-5-2	155502.52	156530.46	4427.11	**HVRP_kroB200-C-5-2**	151477.96	154528.55	5951.06
HVRP_kroB200-R-5-3	153261.28	155321.38	5419.80	**HVRP_kroB200-C-5-3**	152028.34	153544.15	6440.25