Aalto University
School of Science
Master's Programme in Mathematics and Operations Research

Markus Mattila

A cyclic exchange neighborhood search algorithm and transformations for the Generalized Vehicle Routing Problem

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Helsinki, April 6, 2018

Supervisor:	Professor Harri Ehtamo
Instructor:	Professor Enrico Bartolini

Aalto University
School of Science

Master's Programme in
Mathematics and Operations Research

ABSTRACT OF
MASTER'S THESIS

Author:	Markus Mattila		
Title:	A cyclic exchange neighborhood search algorithm and transformations for the Generalized Vehicle Routing Problem		
Date:	April 6, 2018	**Number of pages:**	vii + 66
Department of Mathematics and Systems Analysis			
Professorship:	Systems and Operations Research		
Supervisor:	Prof. Harri Ehtamo		
Advisor:	Prof. Enrico Bartolini		

Vehicle routing problems have numerous applications in fields such as transportation, supply logistics and network design. The optimal design of these routes fall in the category of *NP-hard* optimization problems, meaning that the computational complexity increases extremely fast with increasing problem size.

The Generalized Vehicle Routing Problem (GVRP) is a general problem type that includes a broad variety of other problems as special cases. The main special feature of the GVRP is that the customers are grouped in clusters. For each cluster, only one customer is visited.

In this thesis, we implement a heuristic algorithm to solve GVRP instances in reasonable time. Especially, we include a cyclic exchange method that considers a very large search neighborhood.

In addition, we study the related Capacitated m-Ring-Star Problem (CmRSP). We present the Distance-Constrained Capacitated m-Ring-Star Problem (DCmRSP) and show that it contains the Multivehicle Covering Tour Problem (MCTP) as a special case. We show that DCmRSP instances can be transformed to (distance-constrained) GVRP with minor adaptations and solved with the same heuristic algorithm.

Our algorithm is able to find best known solutions to all GVRP test instances; for two of them, our method shows strict improvement. The transformed CmRSP and MCTP instances are solved successfully by the same algorithm with adequate performance.

Keywords:	Combinatorial optimization, vehicle routing, heuristic algorithm, cyclic exchange
Language:	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Matematiikan ja operaatiotutkimuksen maisteriohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Markus Mattila		
Työn nimi:	Kiertovaihtoalgoritmi ja muunnoksia yleistetylle ajoneuvoreititysongelmalle		
Päiväys:	6. huhtikuuta 2018	**Sivumäärä:**	vii + 66
Matematiikan ja systeemianalyysin laitos			
Pääaine:	Systeemi- ja operaatiotutkimus		
Valvoja:	Prof. Harri Ehtamo		
Ohjaaja:	Prof. Enrico Bartolini		

Ajoneuvoreititysongelmilla on lukuisia sovelluksia muun muassa logistiikan ja verkostosuunnittelun aloilla. Tällaisten reittien optimaalinen ratkaiseminen kuuluu NP-vaikeiden optimointiongelmien kategoriaan, eli ratkaisuun vaadittava laskentateho kasvaa erittäin nopeasti ongelman koon suhteen.

Yleistetty ajoneuvoreititysongelma (Generalized Vehicle Routing Problem, GVRP) on ongelmatyyppi, joka kattaa joukon muita reititysongelmia erikoistapauksina. GVRP:n selkein erityispiirre on asiakkaiden jako ryppäisiin: kussakin ryppäässä on käytävä tasan yhden asiakkaan luona.

Tässä diplomityössä esitellään ja toteutetaan heuristinen algoritmi, joka etsii kohtalaisessa ajassa ratkaisuja GVRP-ongelmiin. Menetelmä sisältää kiertovaihtoalgoritmin, joka kykenee etsimään ratkaisuja hyvin laajasta ympäristöstä.

Tutkimuksen kohteena on lisäksi m-rengastähtiongelma (Capacitated m-Ring-Star Problem, CmRSP). Esittelemme ongelman etäisyysrajoitetun version (DCmRSP), ja näytämme, että kyseiseen ongelmaan sisältyy usean ajoneuvon peittävän reitin ongelma (Multivehicle Covering Tour Problem). Näytämme, että DCmRSP-ongelman pystyy pienin muutoksin muuntamaan GVRP-ongelmaksi ja ratkaisemaan samalla heuristisella algoritmilla.

Algoritmi löytää parhaat tunnetut ratkaisut kaikkiin GVRP-testitehtäviin. Kahdessa tapauksessa ratkaisu on parempi aiemmin löydettyihin nähden. Algoritmi kykenee ratkaisemaan muunnetut CmRSP- ja MCTP-testitehtävät kohtalaisella ratkaisulaadulla.

Avainsanat:	Optimointi, ajoneuvoreititys, heuristiikka, syklinen vaihto
Kieli:	Englanti

Acknowledgements

First of all, I wish to thank my supervisor, professor Harri Ehtamo for his guidance throughout the thesis process. Secondly, I thank my instructor, professor Enrico Bartolini who introduced me to the world of computational optimization already with my Bachelor's Thesis, and whose rigorous advice has provided me with the will and the means to strive for excellence. I thank the entire staff of Systems Analysis Laboratory for an encouraging working community and many years of invaluable experiences.

I express my sincere gratitude to my fellow student friends. I thank the Guild of Physics for the community that made my time in Otaniemi so full of meaning; Fyysikkospeksi for the opportunity to reach for your passions; Fii Fighters for unforgettable moments; and Rissittely for a 24/7 helpdesk, whether regarding thesis advice or garment topology.

My special thanks go to my family for their unconditional support, and to my fiancée and life partner Salla for sharing all the joy and difficulties, study-related or not, during our years together.

Otaniemi, April 6, 2018

Markus Mattila

Contents

Abbreviations

CE	Cyclic exchange
CmRSP	Capacitated m-Ring-Star Problem
CSP	Covering Saleman Problem
CTP	Covering Tour Problem
CVRP	Capacitated Vehicle Routing Problem
DCmRSP	Distance-Constrained Capacitated m-Ring-Star Problem
DGVRP	Distance-Constrained Generalized Vehicle Routing Problem
GTSP	Generalized Traveling Saleman Problem
GVRP	Generalized Vehicle Routing Problem
LNS	Large Neighborhood Search
LS	Local Search
MCTP	Multivehicle Covering Tour Problem
TSP	Traveling Salesman Problem
VLNS	Very Large Neighborhood Search
VRP	Vehicle Routing Problem

Chapter 1

Introduction

It is apparent that there are numerous applications for vehicle routing problems in transportation. Being able to find better routes for cars or airplanes can save significant amounts of fuel, time and money. The constantly improving computer efficiency allows for more complex and large problems to be solved. However, finding optimal solutions remains exceedingly difficult for many real-life problems. Heuristic algorithms that are able to find good solutions quickly are essential in practical applications.

1.1 Problem descriptions

The Generalized Vehicle Routing Problem is a routing problem that contains many simpler problems as special cases, including the Traveling Salesman Problem and the Capacitated Vehicle Routing Problem. Each customer in a GVRP belongs to a cluster and only one customer per cluster can be visited. There is also a constraint on the capacity of each route.

The GVRP can be applied, for instance, to a logistics problem where a company wants to deliver its product to stores in a city. It may be time-consuming and inconvenient to visit each store with a large truck. Instead, a single visit to each neighborhood (cluster of stores) can be more effective as local stores can distribute the product flexibly among themselves.

Another practical application arises in disaster relief. Natural disasters, such as earthquakes or tropical storms can disrupt the road network and leave neighboring cities and villages stranded from other parts of the affected area. Supplies to these village clusters may have to be delivered by aircraft. Optimal routes for airplanes will help in providing the best possible utilization of available resources.

The Distance-Constrained Generalized Vehicle Routing Problem is an even more general version of the GVRP. Adding distance constraints can help to model even more realistic situations concerning working hours, the amount of fuel etc.

The Capacitated m-Ring Star Problem originates from a real-world need to design cable networks. These networks are often circular so that a fault on a single edge will not cut the data flow to the customers. Some customers can also be attached to the ring with a more vulnerable one-way link, resulting in a ring-star shape.

The CmRSP is closely related to vehicle routing; instead of a fleet of vehicles traveling on roads, information travels through cables. In fact, a vehicle routing interpretation of the CmRSP would be very similar to that of the logistic GVRP. Whereas the GVRP includes clusters where only one customer is visited, the trucks in the CmRSP can either visit the customers or one of their possible star-connections.

1.2 Research approach

In this thesis, we have two key focal points. One is to create a competitive heuristic algorithm for solving GVRP instances with good quality and reasonable computing time. As a basis, we use the DGVRP algorithm created for the Bachelor's Thesis. This algorithm is appended with a cyclic exchange search procedure, a simple shortest-path tool and some features to improve the computational efficiency.

The other aim of this thesis is to find other problems that can be solved with the same DGVRP algorithm. Two such similar problems are found: the Capacitated m-Ring-Star Problem and the Multivehicle Covering Tour Problem. Test instances for these problems are transformed into GVRP form and solved. We are especially interested to see if the results, without any ad-hoc adaptations, are comparable to the current research on these problems, concerning both solution quality and computing time.

We also present the Distance-Constrained Capacitated m-Ring-Star Problem (DCmRSP). This problem contains the Multivehicle Covering Tour Problem as a special case. To our knowledge, the DCmRSP has not been previously studied. The addition of distance constraints makes for an interesting problem. Suppose that the distance constraint concerns the length of the ring. Then, a constrained solution would be likely to include short rings with many star assignments, since assignments do not add to the length of the ring. However, a more thorough research of the DCmRSP is left out of the scope of this thesis. The distance-constrained variant is formulated mathematically but not solved exactly or with the heuristic algorithm.

1.3 Thesis structure

In Chapter 2 of this thesis, we review the literature related to vehicle routing problems and especially to the CmRSP, GVRP and MCTP. We also take a look at the development of heuristic search methods for vehicle routing problems, focusing on large neighborhood search methods.

More detailed descriptions and mathematical formulations of the problems are presented in Chapter 3. In this chapter, we present the Distance-Constrained m-Ring-Star Problem and show that both CmRSP and MCTP are special cases of this problem.

Chapter 4 depicts the transformation of a DCmRSP instance to a DGVRP instance. We also discuss some problems of the transformation and add constraints to ensure the feasibility of a GVRP solution transformed back to DCmRSP form.

In Chapter 5 we present the heuristic algorithm. We revise the Bachelor's Thesis algorithm that is built on the *Split* procedure that divides a sequence of all clusters to feasible routes in an optimal fashion. Moreover, the cyclic exchange method is presented in detail. In addition, we describe the implementation of shortest-path and preprocessing algorithms.

The results of the heuristic algorithm can be found in Chapter 6. New previously unknown best solutions for two GVRP instances are presented. Finally, conclusions and ideas for future research can be found in Chapter 7.

Chapter 2

Background

In this chapter, we present the main concepts pertaining to the topic of the thesis. Especially, we familiarize the reader with the history of vehicle routing problems and common metaheuristic methods for solving them. We pay specific attention to the problems examined in this thesis and other closely related problems.

2.1 Traveling Salesman Problem

The *Traveling Saleman Problem* (TSP) is perhaps the most well-known combinatorial optimization problem and it is the basis to all vehicle routing problems. In the standard version of the problem, a salesman must design a least-cost tour to visit a given set of towns and return back to their home town. For a number of n towns, there are $n!$ possible routes for the salesman. Therefore, evaluating all possible routes becomes extremely time consuming with increasing problem size.

Concerning computational complexity, the TSP is $\mathcal{NP}$*-complete*, meaning that it belongs to both complexity classes $\mathcal{NP}$*-hard* and $\mathcal{NP}$. Problems in $\mathcal{NP}$ are solved in 'nondeterministic polynomial' time, as opposed to class $\mathcal{P}$ with known polynomial algorithms. $\mathcal{NP}$*-hard* problems are at least as hard to solve as those in $\mathcal{NP}$. There are no known polynomial-time solution algorithms for problems in $\mathcal{NP}$, but a solution can be verified in polynomial time. Thus, finding a polynomial-time algorithm for the TSP would mean that all problems in $\mathcal{NP}$ can be solved polynomially. So far, no polynomial algorithms for $\mathcal{NP}$-complete problems have been found, nor has it been proved that such an algorithm does not exist. More information about complexity theory is presented e.g. in [6].

This question, whether $\mathcal{P} = \mathcal{NP}$, is one of the most important unanswered questions in the field of mathematics and computer science. Generally, it is considered more likely that $P \neq \mathcal{NP}$, since $\mathcal{NP}$-complete problems contain very complex problems. Currently, the fastest algorithms for solving TSP instances to optimality grow exponentially with respect to problem size.

The TSP has been extensively studied, and it is the subject of numerous variations and generalizations, including e.g. distance constraints, asymmetricity and time windows.

2.2 Vehicle routing problems

The Vehicle Routing Problem (VRP) generalizes the Traveling Saleman Problem. In the VRP, there can be multiple routes to cover all the towns or customers. The VRP was introduced in [9] and it has also attracted much attention, due to its natural applications in vehicle fleet optimization. Current vehicle routing problem types, solution methods and applications can be viewed in [28].

The *Capacitated Vehicle Routing Problem* (CVRP) is a further generalization of the VRP. In the CVRP, each customer has a demand that must be satisfied. In addition, the vehicles have a maximum capacity, which the demand of the customers on a route cannot exceed.

2.3 Generalized Vehicle Routing Problem

The Generalized Vehicle Routing Problem is another generalization of the VRP. To our knowledge, the GVRP is first introduced in [13]. The main component of the GVRP is the presence of *clusters*. Each customer belongs to exactly one cluster and all clusters contain at least one customer. The cluster set is then a partition of the customer set. For each cluster, exactly one customer is visited on a feasible route.

The customers in the standard version of the GVRP have demands. Thus, the GVRP reduces to a CVRP if all the clusters are singletons. A GVRP with only one route reduces to the Generalized Traveling Salesman Problem (GTSP), another well-known optimization problem. More applications of the GVRP are presented in [3], including the Traveling Saleman Problem with profits and the periodic and multi-depot VRP.

In [13], a transformation of the GVRP to a Capacitated Arc Routing Problem is presented with one test instance. The first solution algorithm, an Ant Colony System heuristic is applied to a small set of test instances with up to 20 clusters in [24].

A full set of large-scale (up to more than 100 clusters) test instances for the GVRP is proposed in [5] along with four mathematical formulations. Both exact branch-and-cut and heuristic algorithms are applied. In [20], a tabu search heuristic is applied to the GVRP and the variant with time windows (GVRPTW). An exact algorithm and a metaheuristic based on the *Split* method is presented in both [15] and [1].

The Distance-Constrained Generalized Vehicle Routing Problem (DGVRP) generalizes the GVRP to include distance constraints for the length of the routes. To our knowledge, this variant has only been considered in [19], the Bachelor's Thesis of the author. In the thesis, a set of DGVRP test instances is generated and solved with a heuristic algorithm.

2.4 Capacitated m-Ring-Star Problem

In this thesis, we also consider additional problems closely related to vehicle routing problems. The Capacitated m-Ring Star Problem (CmRSP) is introduced in [4] for an application in the design of a telecommunications cable network. A set of test instances with up to 100 nodes and a branch-and-cut algorithm are introduced.

A heuristic algorithm for these CmRSP instances is presented in [21]. The same authors consider another Integer Linear Programming -based heuristic in [22] and present new test instances with up to 200 nodes. In [29], a memetic heuristic algorithm for solving the CmRSP is presented, along with a set of even larger test instances with up to over 400 nodes.

2.5 Multivehicle Covering Tour Problem

Another problem that we consider is the Multivehicle Covering Tour Problem (MCTP), first presented in [16]. This problem arises from the Covering Saleman Problem (CSP), a variant of the TSP introduced in [8], in which it is not necessary to visit a customer if it is sufficiently close to a customer that is visited. The Covering Tour Problem (CTP), introduced in [11], partitions the set of customers to those that can be visited and those that must be covered. Then, the MCTP is a generalization of the CTP where multiple routes are allowed.

Multiple exact solution methods have been applied for the MCTP [17, 14, 23] along with heuristic approaches [16, 26, 14, 18, 23].

2.6 Cyclic exchange

Due to the fact that vehicle routing problems belong to the class of $\mathcal{NP}$-hard problems, the development of heuristic algorithms is closely associated with the history of these problems. Usually, the local search methods used in the heuristic methods consist of simple moves; insertions, swaps, 2-opt, etc. The *neighborhood* of a solution is the set of solutions that can be reached from the current one with the local search methods. Typically, there is a fundamental trade-off between the size of the neighborhood and the effectiveness of the algorithm; searching a bigger neighborhood is likely to yield better results with the expense of increased complexity and computational time.

The concept of *metaheuristic algorithms* refers to the framework that combines a simple local search procedure with a higher level strategy for creating neighborhoods and escaping from local optima. With metaheuristic methods, the neighborhood of a solution is typically small, but the framework ensures that the entire solutions space is considered in a robust manner. Metaheuristic algorithms often include a procedure that 'shakes' the current solution to enter new neighborhoods, or iteratively create diverse solutions. Multiple metaheuristic methods are presented in detail in [12] and [7], including *tabu search*, *variable neighborhood search* and *simulated annealing*.

It is also possible to consider more complex moves and large neighborhoods, provided that there is an effective algorithm for searching that neighborhood. Such methods are called *large neighborhood search* (LNS) or *very large neighborhood search* (VLNS) methods.

The *cyclic exchange method*, first presented in [27], is a VLNS procedure for solutions that are partitions; for example, most vehicle routing problems are partitions as each customer or cluster belongs to exactly one route. A cyclic exchange move removes an element from one set of the partition and inserts it to another one. Then, an element of the second set is removed and inserted elsewhere. The process continues, until an element is inserted back to the first set, completing the cycle, or if an element is inserted to a set without removal. With the restriction that each set can only be affected once with a single move, there is an effective algorithm for searching improving cycles using an auxiliary improvement graph. A cyclic exchange move can potentially change every route in a VRP solution.

Cyclic exchange is used in [2] in the context of the Capacitated Minimum Spanning Tree Problem. The method is presented in more detail in chapter 13.4 in [7].

Chapter 3

Problem formulations

In this chapter, we present mathematical formulations of the problems studied in the thesis. For the Generalized Vehicle Routing Problem, we use a two-index arc formulation with a two-commodity flow model for the capacity constraints. Possible distance constraints are modelled with a single commodity flow.

We present the mathematical description and a formulation to the new Distance-Constrained Capacitated m-Ring-Star Problem. We show that both the Capacitated m-Ring Star Problem and the Multivehicle Covering Tour Problem are special cases of the DCmRSP.

3.1 Generalized Vehicle Routing Problem formulation

We define the Generalized Vehicle Routing Problem on a directed graph $G = (V, A)$. The vertex set V has n vertices $\{v_1, v_2, ..., v_n\}$ to represent regular customers. In addition, there is a central depot where all routes must start and end. This depot is divided into two vertices v_0 and v_{n+1} so that all routes form paths from v_0 to v_{n+1}. The cost of traversing an arc from $v_i \in V \setminus v_{n+1}$ to $v_j \in V \setminus v_0$ equals the distance d_{ij} between them. In the thesis, we exclusively study the symmetric case, where $d_{ij} = d_{ji}$.

Each regular vertex belongs to exactly one cluster. Each cluster can contain one or more vertices. In total, the cluster set C consists of K clusters: $C = \{C_1, C_2, ..., C_K\}$.

The GVRP is a capacitated problem. This means that each vehicle has a limit Q for the amount of goods it can carry to the customers. In this context, each cluster C_k has a demand $q_k \leq Q$. Since each customer is uniquely associated with one cluster,

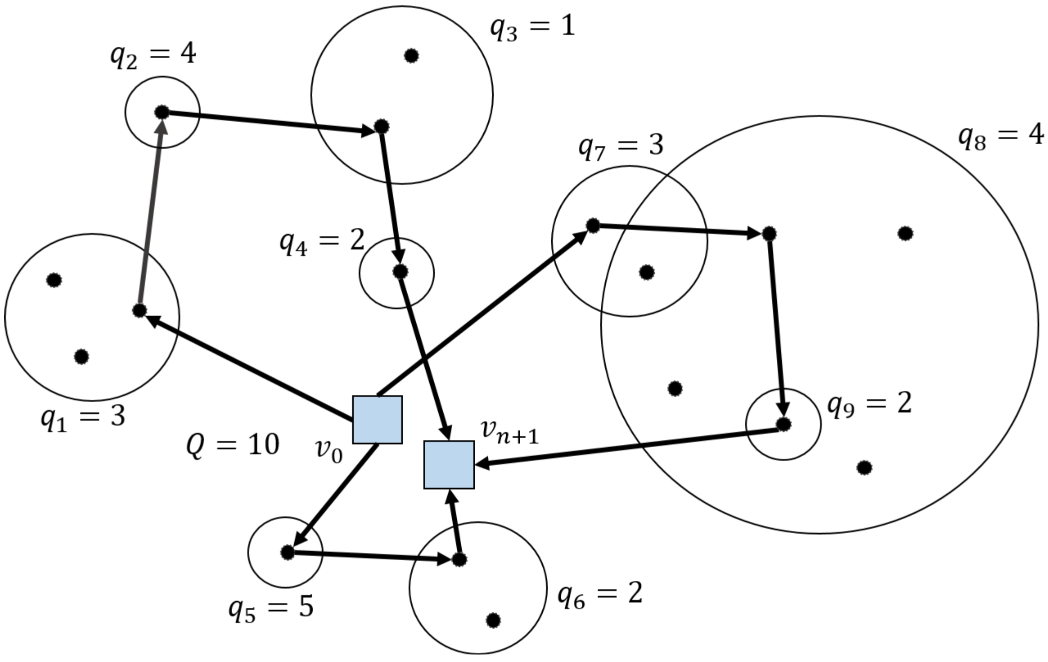

Figure 3.1: An example of a feasible GVRP solution

we can assign a demand $\tilde{q}_i = q_k \, \forall v_i \in C_k$.

In [19], which is the Bachelor's Thesis of the author, we presented the DGVRP that has additional distance or time constraints that the solution must satisfy. The length of each route is bounded by an upper limit T. Each arc (i, j) is assigned with t_{ij}, the time it takes to traverse that arc. If $t_{ij} = d_{ij}$ the problem is strictly distance-constrained, but our formulation allows the distance and time to differ.

A feasible (D)GVRP solution consists of m routes. In most literature, the fleet size is considered as a fixed parameter. We assume that the fleet size is flexible and treat m as a decision variable, as in [14].

To formulate the problem mathematically, we use several sets of decision variables. First, binary variables x_{ij} equal 1 if arc (i, j) is traversed and 0 otherwise. We are forced to use arcs instead of edges in order to model the distance constraints. Binary variables y_i indicate if the customer v_i is visited. Variables f_{ij} model a commodity flow through the network and h_{ij} indicate time flow. More detailed explanations and examples of the flows are presented after the formulation:

$$\min \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} d_{ij} x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i \in C_l} y_i = 1 \quad \forall\, l = 1, 2, \dots, K \tag{3.2}$$

$$\sum_{i=0}^{n} x_{ij} = y_j \quad \forall v_j \in V \setminus \{v_0, v_{n+1}\} \tag{3.3}$$

$$\sum_{k=1}^{n+1} x_{jk} = y_j \quad \forall v_j \in V \setminus \{v_0, v_{n+1}\} \tag{3.4}$$

$$f_{ij} + f_{ji} = Q(x_{ij} + x_{ji}) \quad \forall (i,j) \in A \tag{3.5}$$

$$\sum_{j=1}^{n} f_{0j} = \sum_{k=1}^{K} q_k \tag{3.6}$$

$$\sum_{j=1}^{n} f_{n+1j} = Qm \tag{3.7}$$

$$\sum_{j=0}^{n+1} f_{ji} - \sum_{j=0}^{n+1} f_{ij} = 2\tilde{q}_i y_i \quad \forall v_i \in V \setminus \{v_0, v_{n+1}\}, i \neq j \tag{3.8}$$

$$\sum_{j=0}^{n+1} h_{ij} - \sum_{k=0}^{n+1} h_{ki} = \sum_{j=0}^{n+1} t_{ij} x_{ij} \quad \forall\, v_i \in V \setminus \{v_0, v_{n+1}\}, i \neq j \tag{3.9}$$

$$h_{ij} \leq T x_{ij} \quad \forall\, (i,j) \in A \tag{3.10}$$

$$h_{0\,i} = t_{0\,i} x_{0\,i} \quad \forall v_i \in V \tag{3.11}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{3.12}$$

$$y_i \in \{0,1\} \quad \forall v_i \in V \setminus \{v_0, v_{n+1}\} \tag{3.13}$$

$$m \in \mathbb{Z}_+ \tag{3.14}$$

$$f_{ij} \geq 0 \quad \forall (i,j) \in A \tag{3.15}$$

$$h_{ij} \geq 0 \quad \forall\, (i,j) \in A \tag{3.16}$$

The objective function (3.1) is to minimize the combined cost of each traversed arc. Constraints (3.2) mean that exactly one customer per cluster is visited, and (3.3)-(3.4) enforce that there is exactly one arc entering and leaving each visited customer. No arcs are connected to unvisited customers.

Constraints (3.5)-(3.8) define the two-commodity flow. A simple example of the flow is given in Figure 3.2 where the depot v_0 and its duplicate v_{n+1} are separated for clarity. The blue *forward flow* f_{ij} describes the amount of goods that are onboard a vehicle as it traverses the arc (i, j). The red *backflow* f_{ji} can be interpreted as the empty space on a vehicle traversing the arc (i, j). Thus, the sum of the flows on a given edge must always equal the total vehicle capacity Q if the edge is traversed either way (3.5).

Constraints (3.6) mean that the commodity flow from v_0 is equal to the combined demand of all clusters, and (3.7) that the backflow from v_{n+1} equals the entire capacity of the fleet; each vehicle arrives back at the depot fully unloaded.

The consumption of capacity is modelled by constraints (3.8). The meaning of these constraints is that if a customer v_i is visited, then the sum of flows leaving the customer substracted from the sum of flows entering the customer must equal $2\tilde{q}_i$. In other words, both the forward and the backward flow grow smaller by $\tilde{q}_i$ when visiting v_i. To illustrate, examine customer v_3 in Figure 3.2. Both flows grow smaller by $\tilde{q}_2 = 4$; the formula expresses this as $f_{13} + f_{43} - f_{31} - f_{34} = 9 + 5 - 5 - 1 = 8$.

The two-commodity flow constraints also eliminate subtours. The demands of each cluster can only be positive; therefore, the forward flow value must decrease after every arc traversed. If the same arc would be encountered again on the same route, the flow value would have to be different, creating a contradiction as long as there are no clusters with $q_k = 0$.

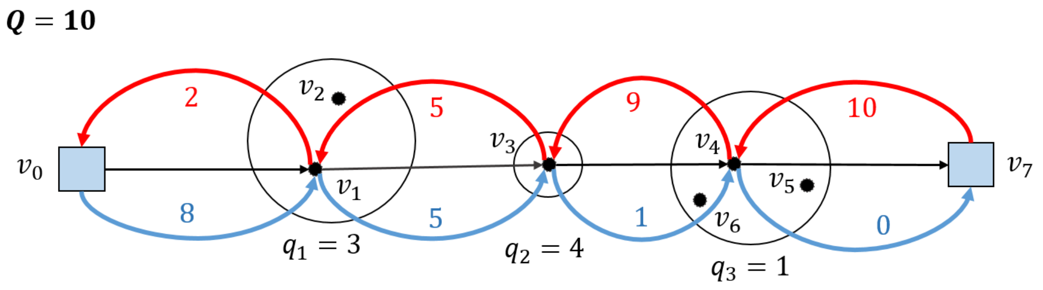

Figure 3.2: Two-commodity flow example for a GVRP, $m = 1$

Distance limits are modelled with time flow constraints (3.9)-(3.11). An example is provided in Figure 3.3 where the distances are marked in black and flow values in green. Constraints (3.11) impose that the time flow h_{0i} on arcs emanating from v_0 must be equal to t_{0i} if the arc is traversed and 0 otherwise. The rest of the flow variables are determined inductively by constraints (3.9); if arc (i,j) is traversed, the time flow leaving v_i must be greater than the flow entering v_i by t_{ij}. For example, in Figure 3.3 we have $h_{13} - h_{01} = t_{13}$: $7 - 4 = 3$. No time flow variable can exceed the limit T (3.10).

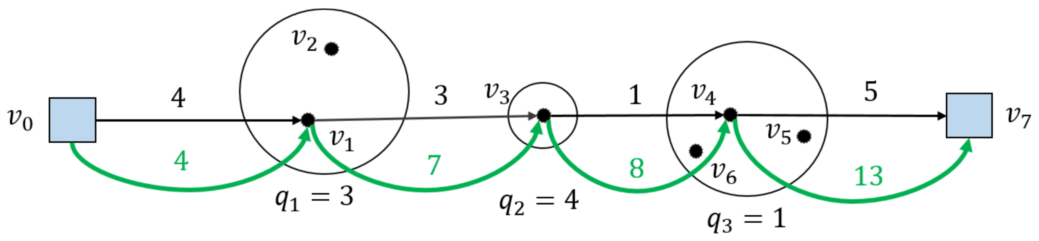

Figure 3.3: Time flow example for a GVRP, $m = 1$

Finally, constraints (3.12)-(3.16) simply define the boundaries of the variables: x_{ij} and y_i are binary, f_{ij} and h_{ij} are non-negative and the number of vehicles m is a positive integer.

3.2 Capacitated m-Ring-Star Problem formulation

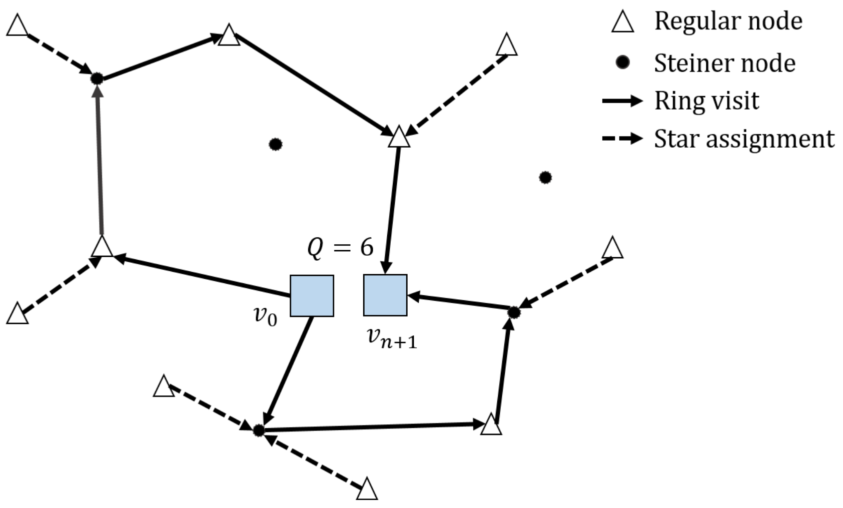

Figure 3.4: An example of a feasible CmRSP solution

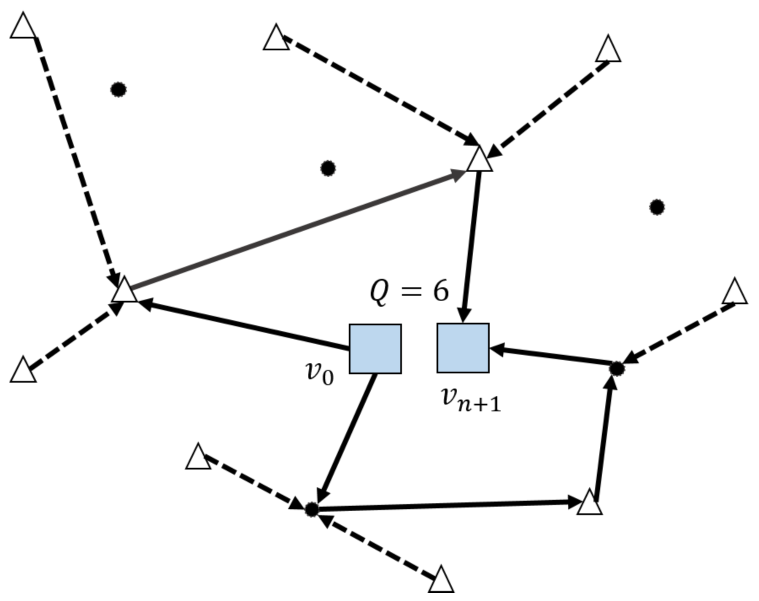

Figure 3.5: An example of a feasible DCmRSP solution

The Distance-Constrained Capacitated m-Ring Star Problem is defined on a directed graph $G = (V, A)$. The definition of the DCmRSP is very similar to that of the DGVRP; the vertex set $V = \{v_0, v_{n+1}, V'\}$ consists of the depot v_0, its duplicate v_{n+1} and the regular vertex set V'. Now V' is further partitioned into two separate sets U and S. Vertices v_i in U represent customers that must be either visited or assigned to another vertex while vertices in S serve as additional assignment points called *Steiner nodes*. The arc set A is divided into two disjoint sets A_1 and A_2 corresponding to ring and star connections, respectively.

Rings on the graph consist of paths traversing from v_0 to v_{n+1}, arcs assigned to that path and all vertices incident to any arc on the path. At most m rings can be used to obtain a solution where every customer in U is either visited by or assigned to exactly one ring. Arcs $A_1 = \{(i, j) |\, v_i, v_j \in V,\, i \neq j\}$ represent possible connections in the graph with non-negative costs $c_{ij} = c_{ji}$. The arc set A_2 represents all possible assignments of the regular nodes to other regular or Steiner nodes. For each $v_i \in U$, the set of feasible assignment points is denoted by $C_i \subset V'$. This arc set is thus $A_2 = \{(i, j) |\, v_i \in U, v_j \in C_i\}$ with non-negative costs a_{ij}. Each v_i belongs to C_i; the customer is assigned to itself with cost $a_{ii} = 0$ if it is directly connected to a ring.

The capacity of each ring-star is bounded by a limit of Q. Each regular node v_i has a demand $q_i = 1$ while Steiner nodes have zero demand. This means that at most Q regular customers can be assigned to one ring-star.

By adding distance constraints to the CmRSP, we can force the rings to become smaller with the expense of more star assignments (Figure 3.5). An upper limit T gives the maximum length of a ring, while t_{ij} describe the distance values of arcs $(i, j) \in A_1$.

To present the DCmRSP mathematically, we use a formulation very similar to the GVRP. It is based on the two-commodity flow formulation presented in [4]. However, in order to model the distance constraints, we must include arc-based decision variables x_{ij} as in the GVRP. We also have the *flow variables* f_{ij}, $(i, j) \in V$. There is one such variable for each arc $(i, j) \in A_1$. Each forward flow variable f_{ij} has a counterpart in a backward flow variable f_{ji}, identically to the previous GVRP formulation.

In the formulation, we have three more sets of decision variables. Binary variables z_{ij} equal 1 for assignment arcs $(i, j) \in A_2$ that are in use. Binary variables s_j for each Steiner node $v_j \in S$ indicate if they are visited. Time flow variables h_{ij} are used exactly as with the GVRP.

The objective function (3.17) is to minimize the combined cost of ring and assignment arcs. Constraints (3.18)-(3.21) model the behavior of the two-commodity flow at the depot. According to constraint (3.18), the flow from v_0 equals the combined demand of each customer in V'. This is a general formula; in the traditional CmRSP, $\sum_{v_i \in V'} q_i = |U|$. The backflow to v_0 must equal the difference between the total vehicle capacity and demand (3.19), the vehicles arrive to v_{n+1} completely empty (3.20) and the backflow from v_{n+1} equals the total fleet capacity.

Constraints (3.22)-(3.23) mean that if a regular or Steiner node is visited, then the sum of flows entering and leaving the node must equal $2Q$. The binary variables x_{ij} are tied to the flow variables f_{ij} with constraints (3.24): the sum of empty space and goods must always equal Q on arcs that are traversed. Every node in U must be assigned to exactly one node (3.25).

$$\min \sum_{(i,j)\in A_1} c_{ij}x_{ij} + \sum_{(i,j)\in A_2} a_{ij}z_{ij} \tag{3.17}$$

$$\sum_{j\in V'} f_{0j} = \sum_{v_i\in V'} q_i \tag{3.18}$$

$$\sum_{j\in V'} f_{j0} = mQ - \sum_{v_i\in V'} q_i \tag{3.19}$$

$$\sum_{j\in V'} f_{j,n+1} = 0 \tag{3.20}$$

$$\sum_{j\in V'} f_{n+1,j} = mQ \tag{3.21}$$

$$\sum_{i\in V}(f_{ij} + f_{ji}) = 2Qz_{jj} \quad \forall v_j \in U \tag{3.22}$$

$$\sum_{i\in V}(f_{ij} + f_{ji}) = 2Qs_j \quad \forall v_j \in S \tag{3.23}$$

$$f_{ij} + f_{ji} = Q(x_{ij} + x_{ji}) \quad \forall (i,j) \in A_1 \tag{3.24}$$

$$\sum_{j\in V'} z_{ij} = 1 \quad \forall v_i \in U \tag{3.25}$$

$$\sum_{i\in V}(f_{ij} - f_{ji}) = 2\sum_{v_i\in U} q_i z_{ij} \quad \forall v_j \in U \tag{3.26}$$

$$\sum_{i\in V}(f_{ij} - f_{ji}) = 2(\sum_{v_i\in U} q_i z_{ij} + q_j s_j) \quad \forall v_j \in S \tag{3.27}$$

$$\sum_{j=0}^{n+1} h_{ij} - \sum_{k=0}^{n+1} h_{ki} = \sum_{j=0}^{n+1} t_{ij}x_{ij} \quad \forall v_i \in V, i \neq j \tag{3.28}$$

$$h_{ij} \leq Tx_{ij} \quad (i,j) \in A_1 \tag{3.29}$$

$$h_{0i} = t_{0i}x_{0i} \quad v_i \in V \tag{3.30}$$

$$x_{ij} \in \{0,1\} \quad \forall v_i, v_j \in V \tag{3.31}$$

$$z_{ij} \in \{0,1\} \quad \forall (i,j) \in A_2 \tag{3.32}$$

$$s_j \in \{0,1\} \quad \forall v_j \in S \tag{3.33}$$

$$f_{ij} \geq 0 \quad \forall v_i, v_j \in V \tag{3.34}$$

$$h_{ij} \geq 0 \quad \forall v_i, v_j \in V \tag{3.35}$$

$$m \in \mathbb{Z}_+ \tag{3.36}$$

Figure 3.6 gives an example of the two commodity flow on a CmRSP. Similarly to the GVRP example, the forward flow is presented with blue arrows and backflow with red arrows. Constraints (3.26)-(3.27) describe the consumption of capacity to regular and Steiner nodes, respectively. For example, there are two regular nodes, v_1 and v_2, assigned to v_2 in the figure. Thus, both the forward and backward flows are diminished by 2 (each node has a demand $q_1 = q_2 = 1$). The flows concerning Steiner nodes behave similarly; we include the possibility that Steiner nodes v_j can also consume capacity with the term $q_j s_j$.

Constraints (3.28)-(3.30) model the distance constraints by adding time flow variables h_{ij} identically to the GVRP. Variable types are reported with constraints (3.31)-(3.36).

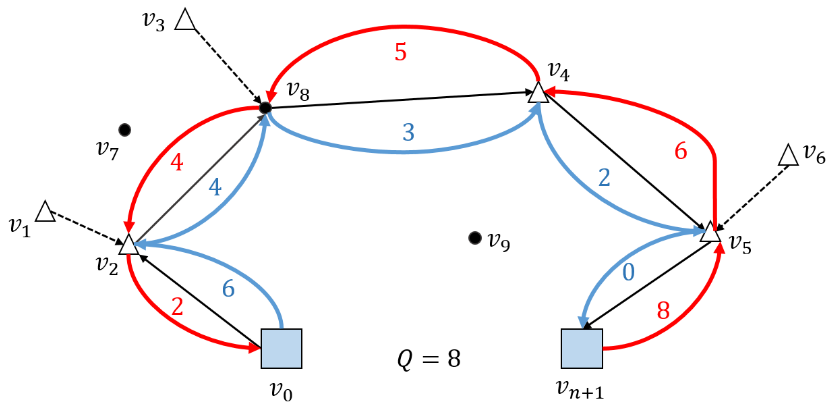

Figure 3.6: Two-commodity flow on a CmRSP example

3.3 Multivehicle Covering Tour Problem formulation

The Multivehicle Covering Tour Problem belongs to the class of vehicle routing problems. It is similar to the Vehicle Routing Problem (VRP) with the objective to design a set of routes having a limit on the number of customers served on a route. In the MCTP, however, it is not necessary to visit every customer; some customers are covered if they lie sufficiently close to the tour. The covered nodes resemble the assigned nodes in the CmRSP. In fact, the MCTP can be viewed as a special case of the Capacitated m-Ring-Star Problem with the exception that every node adds to the capacity limit while Steiner nodes in the CmRSP do not have this property.

More formally, the MCTP is defined on an undirected graph $G = (X, E)$. The customer set X consists of the depot v_0 and its copy v_{n+1} and the set of customers

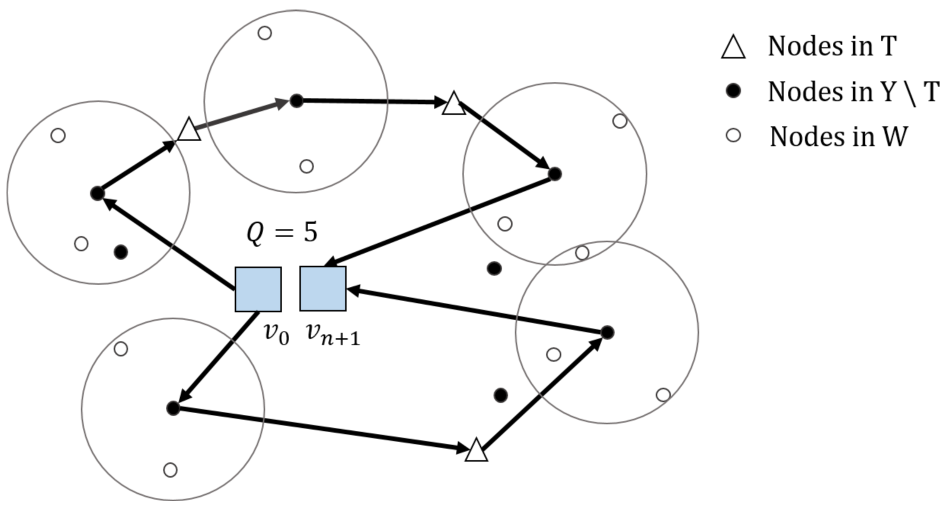

Figure 3.7: An example of a feasible MCTP solution

X'. In X', there is a set W of those customers that must be covered and a set Y of customers that can be visited. In Y, there is a subset T of customers that must be visited. Essentially, the goal is to determine which customers in $Y \setminus T$ to visit in order to cover every node in W. An example of an MCTP instance is given in Figure 3.7.

For each covered node $v_i \in W$ of the MCTP, we define C_i as the set of feasible assignment points consisting of nodes $v_j \in Y$ that lie within the given distance r from v_i. The difference to assigned nodes in the CmRSP is that $v_i \notin C_i$. That is, a covered node can only be assigned to a node in Y and never to itself. For any node $v_j \in T$, the feasible set C_j consists of only v_j itself. Nodes in T must be visited, so no other assignments are possible.

With these distinctions, the assignment arc set is then defined as $A_2 = \{(i,j) | \, i \in W, j \in C_i\} \cup \{(j,j) | \, j \in T\}$. All assignment costs $a_{ij} = 0$. The arc set for nodes that can or must be visited is simply $A_1 = \{(i,j) | \, v_i, v_j \in Y\}$.

The demands q_i for each v_i in X' depend on the type of interpretation of the MCTP. If the capacity limit Q is on the amount of vertices in the ring, then $q_i = 1$ for $v_i \in Y$ and $q_i = 0$ for $v_i \in W$. If the covered nodes are included in the capacity, then also they have demand of 1. Usually only the capacity of the ring is considered, so we choose the former option. Note that whereas the DCmRSP formulation was quite general, this MCTP formulation considers the classical version with fixed demands and no distance constraints.

The formulation is reduced from the two-commodity flow CmRSP formulation with binary arc variables x_{ij} and flow variables f_{ij}. Variables z_{ij} correspond to arc assignments and s_j to optionally visited nodes. In addition to setting the demands to either 1 or 0, we have $a_{ij} = 0$. The set U of regular nodes in the CmRSP is replaced with $W \cup T$ and some redundant constraints are eliminated; for example, the capacity consumption constraints (3.46) only include the set T because $q_i = 0$ for $v_i \in W$. The set of Steiner nodes (S in the CmRSP) is replaced with the set of optional nodes $Y \setminus T$.

With these notations, the mathematical formulation of the MCTP is as follows:

$$\min \sum_{(i,j)\in A_1} c_{ij}x_{ij} \tag{3.37}$$

$$\sum_{j\in X'} f_{0j} = |T| + \sum_{i\in Y\setminus T} s_i \tag{3.38}$$

$$\sum_{j\in X'} f_{j0} = mQ - |T| - \sum_{i\in Y\setminus T} s_i \tag{3.39}$$

$$\sum_{j\in X'} f_{j,n+1} = 0 \tag{3.40}$$

$$\sum_{j\in X'} f_{n+1,j} = mQ \tag{3.41}$$

$$\sum_{i\in X}(f_{ij} + f_{ji}) = 2Q \quad \forall v_j \in T \tag{3.42}$$

$$\sum_{i\in X}(f_{ij} + f_{ji}) = 2Qs_j \quad \forall v_j \in Y \setminus T \tag{3.43}$$

$$\sum_{j\in X'} z_{ij} = 1 \quad \forall v_i \in W \tag{3.44}$$

$$z_{ii} = 1 \quad \forall v_i \in T \tag{3.45}$$

$$\sum_{i\in X}(f_{ij} - f_{ji}) = 2 \quad \forall v_j \in T \tag{3.46}$$

$$\sum_{i\in X}(f_{ij} - f_{ji}) = 2s_j \quad \forall v_j \in Y \setminus T \tag{3.47}$$

$$f_{ij} + f_{ji} = Q(x_{ij} + x_{ji}) \quad \forall (i,j) \in A_1 \tag{3.48}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A_1 \tag{3.49}$$

$$f_{ij} \geq 0 \quad \forall v_i, v_j \in X \tag{3.50}$$

$$z_{ij} \in \{0,1\} \quad \forall (i,j) \in A_2 \tag{3.51}$$

$$s_j \in \{0,1\} \quad \forall v_j \in W \tag{3.52}$$

Chapter 4

Transformations

In this chapter, we present the transformations of the (D)CmRSP instances to (D)GVRP instances and the MCTP instances to DGVRP instances. Inconsistencies in the transformation are discussed and additional constraints are proposed.

4.1 CmRSP to GVRP

A Capacitated m-Ring-Star Problem can be transformed into a Generalized Vehicle Routing Problem with some minor relaxations. This way, we can utilize a competetive heuristic algorithm for the GVRP, constructed in the Bachelor's Thesis of the author. [19]

The main feature of the transformation is expanding the customer set U in a CmRSP to a cluster set C in a GVRP. For each customer, we have mutually exclusive choices to either visit the customer directly or assign it to available customers or Steiner nodes. Each choice for customer $v_i \in U$ is represented as a customer in cluster C_i in the GVRP transformation. The size of the cluster is thus the amount of arcs emanating from v_i plus one. We denote the choice of visiting customer v_i directly by a GVRP vertex u_{ii}, and assigning v_i to another customer v_j by a vertex u_{ij}.

With the customer set in place, we need to define the costs d_{ij} between vertices in the GVRP. First, we note that a CmRSP instance with no feasible assignments ($a_{ij} = \infty \; \forall i \in U, j \in V'$) reduces to a Capacitated Vehicle Routing Problem; the same as a GVRP with only one customer per cluster. Thus, the cost d_{ij} between GVRP customers u_{ii} and u_{jj} is equal to the cost c_{ij} of arc (i, j) in the CmRSP.

For other edges in the GVRP, the assignment costs of the CmRSP must be taken into account. The cost of an edge between GVRP vertices u_{kk} and u_{ij} consists of the cost between v_k and v_j in the CmRSP and half the assignment cost $a_{ij}/2$. Because a visited vertex always has two incident edges, the entire transformation cost is indeed considered. The same principle is applied when both nodes correspond to assignments.

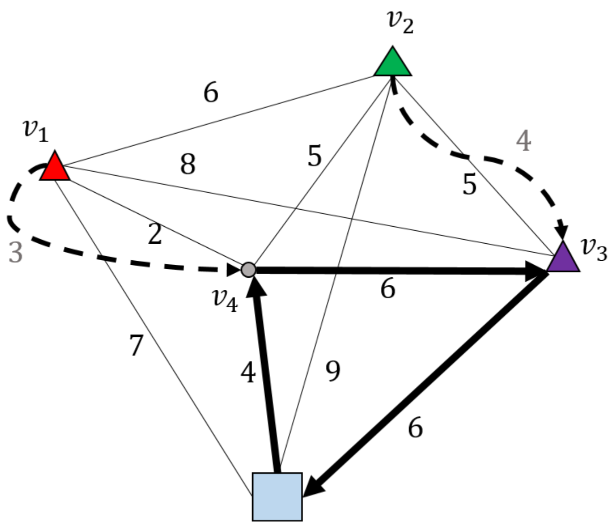

Figure 4.1: A CmRSP instance with three regular nodes

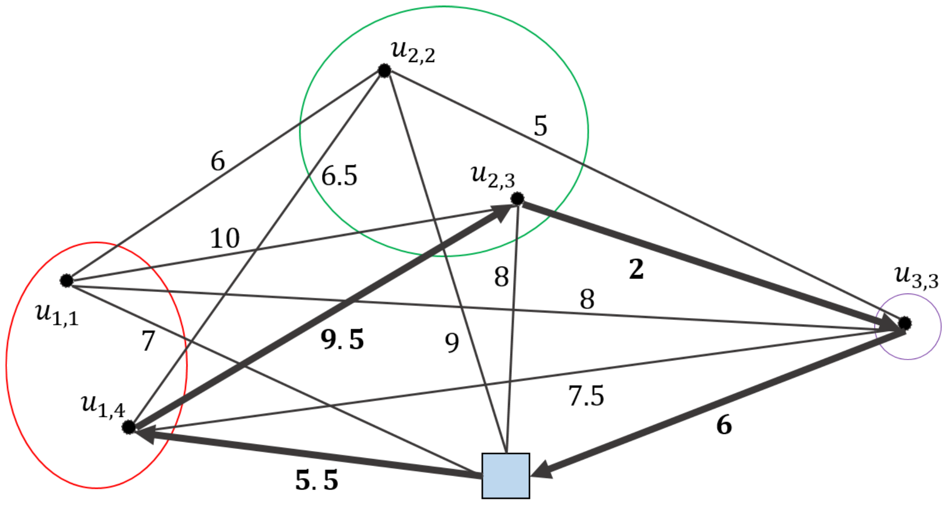

Figure 4.2: The corresponding GVRP instance with three clusters

An example of the new cost structure is given in figures (4.1-4.2). In the CmRSP graph, we have bolded a ring visiting a Steiner node v_4 and a regular node v_3 with v_1 assigned to v_4 and v_2 assigned to v_3. The total cost of this ring is $c_{0,4} + c_{4,3} +$

$c_{3,5} + a_{1,4} + a_{2,3} = 4 + 6 + 6 + 3 + 4 = 23$. In the GVRP graph, node $u_{1,4}$ is visited since v_1 is assigned to v_4. The cost of this edge equals $c_{0,4} + 0.5a_{1,4} = 4 + 1.5 = 5.5$.

Next, v_2 is assigned to v_3; this is equivalent to traversing the edge between $u_{1,4}$ and $u_{2,3}$. The cost of this edge is $0.5a_{1,4} + c_{4,3} + 0.5a_{2,3} = 1.5 + 6 + 2 = 9.5$. Assigning v_3 to the route brings only the additional cost $0.5a_{2,3} = 2$. In the GVRP graph, we move to the third cluster. Finally, the cost of returning to the depot is the same $c_{3,5}$ in both graphs. The cost of the GVRP route is $5.5 + 9.5 + 2 + 6 = 23$; the same as the corresponding CmRSP solution.

Note that in the CmRSP we can arbitrarily choose if v_3 is visited before or after v_2 is assigned to it. In the GVRP, a change in the order will result in a different route, traversing the path $u_{1,4}$, $u_{3,3}$, $u_{2,3}$. The cost of this route still equals $5.5+7.5+2+8 = 23$.

The capacity limit of the CmRSP is for the amount of customers that are either visited or assigned to a ring. Steiner nodes are not included in the capacity. A cluster in the transformed GVRP contains all the possibilities to visit or assign a single customer; therefore, all these options spend one 'capacity unit'. Setting $q_i = 1$ for each cluster C_i in the GVRP results to a correct capacity limit.

If the CmRSP is distance-constrained, then the distance matrix in the DGVRP counterpart is calculated in a similar fashion. The difference is that while assignments can be costly, they do not require any actual distance travelled. For example, the edge between $u_{1,4}$ and $u_{2,3}$ in figure 4.1 would have a distance value of 6 (distance $c_{4,3}$ in Figure 4.2). If the distance values are being used to model time as a resource, then a small constant could be added to each distance matrix value corresponding to assignments, modeling the additional time it might take to handle the assignment.

4.2 MCTP to DGVRP

The transformation of the MCTP to DGVRP form is very similar to the CmRSP, as the problems are nearly identical. There are two kinds of clusters: those that correspond to vertices that must be visited, containing only one customer (denote this set as C^1) and those that model the assignments with one customer for each option (C^2). The DGVRP cost matrix is defined as with the CmRSP. Assignments cost nothing in the MCTP so the transformed costs remain simple.

However, a small difference in capacity interpretations between CmRSP and MCTP leads to a problem in the transformation. In the MCTP, only the visited nodes

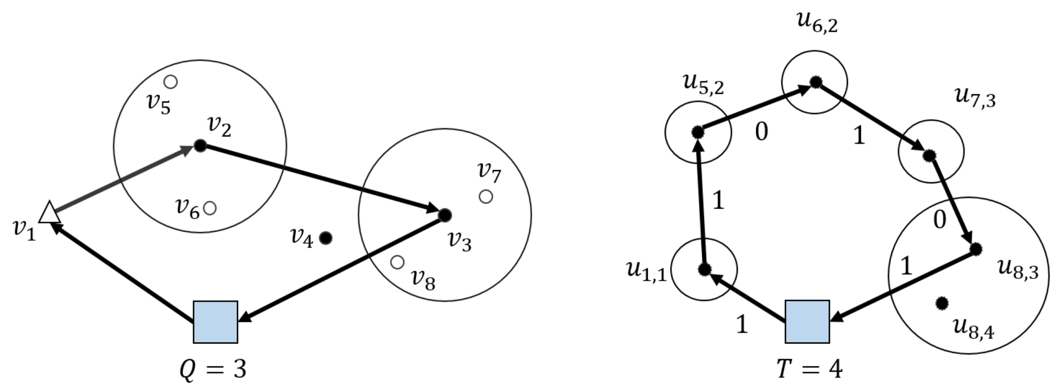

Figure 4.3: MCTP capacity constraint modelled with distance constraints in the DGVRP transformation

(including those that are voluntarily visited) are included in the capacity calculations. This would mean that q_i must be 0 for cluster set C^2 and $q_i = 1$ for C^1. Still, a persisting problem is that the capacity consumed by the MCTP vertices that can be visited (set $Y \setminus T$) is not considered. Because the cluster structure is suitable for mutually exclusive and collectively exhaustive choices, the optional nodes cannot be modelled this way. Instead, we apply a handy work-around: all cluster demands are set to zero or the capacity limit is increased so that the cluster capacities become redundant. The capacity constraints are then implemented by using distance constraints in the DGVRP. The distance between nodes u_{ij} and u_{kj} in the DGVRP is set to 0 and for every other node pair u_{ij}, u_{kl}, $j \neq l$ the distance is 1. Then the distance limit T is set to $Q + 1$. An example of a MCTP instance and the DGVRP transformation with distance values is presented in figure 4.2. Because of this procedure, distance-constrained MCTP instances cannot be transformed to DGVRP form.

4.3 Transformation issues

There are some issues in transforming the (D)CmRSP to a (D)GVRP. The GVRP transformation is, in fact, a relaxation of the original CmRSP. There are two possible ways in which the original requirements of the CmRSP may be violated. We present both these cases separately.

The first problem concerns the possible violation of the CmRSP degree constraints. A feasible solution in the GVRP transformation may correspond to a CmRSP solution with two arcs emanating from and traversing to a node. This is due to the fact that the GVRP formulation 'forgets' that many customers correspond to the same nodes in the CmRSP. The choices are considered independently in each cluster.

An example of such a situation is presented in Figure 4.4 with two regular nodes assigned to a third one. The GVRP cluster sequence (1,2,3,4) corresponds to the following CmRSP route: 'assign v_1 to v_2', 'visit v_2', 'visit v_3', 'assign v_4 to v_2'. Thus, the node v_2 is visited twice which violates constraint (3.22) of the CmRSP; the sum of all flows connected to v_2 now equals $4Q$. However, the GVRP solution is feasible. In practice, such routes will not be present in good-quality solutions. If the triangle inequality holds, it must be suboptimal to visit a node twice.

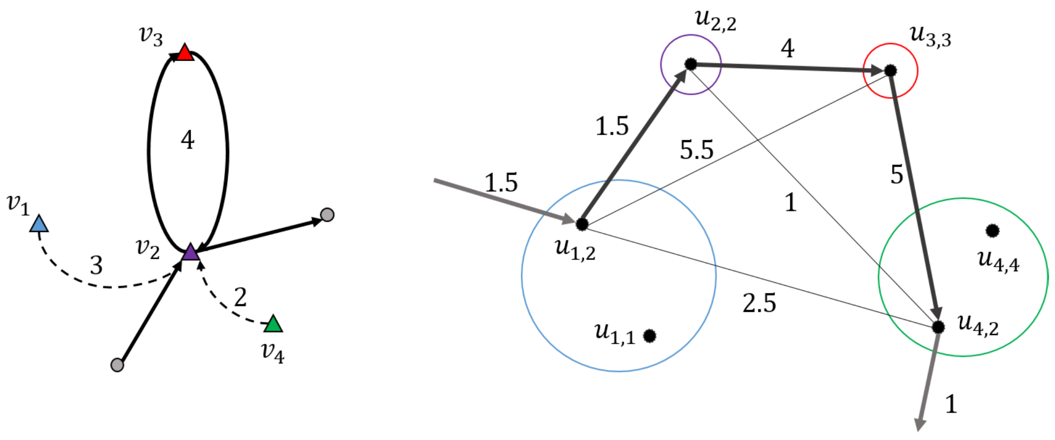

Figure 4.4: An infeasible CmRSP solution (v_2 visited twice) is feasible in the transformed GVRP.

The same problem can also arise if the two paths traversing through the node belong to different routes. This situation is presented in Figure 4.5. Assuming the triangle inequality, it would be cheaper to visit v_2 straight after v_1 on the black route. However, with the additional trip to v_7, the load of customer v_3 is now allocated to the black route instead of the red one. It is not inevitable that v_7 can be skipped on the red route; there may be other customers assigned to it and it is not necessary that the cost of the route is strictly improved. The possibility of redistributing the loads can make this type of violation beneficial.

In the MCTP, this problem can not lead to improved solutions. The loads of covered nodes are set to zero, so redistributing these loads among routes achieves nothing.

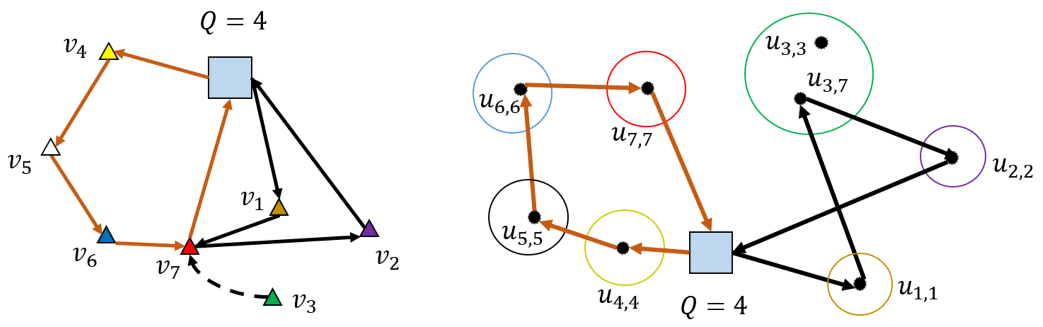

Figure 4.5: An infeasible CmRSP solution (v_7 visited twice) is feasible in the transformed GVRP.

To forbid these violations, we must impose additional constraints in the transformed GVRP. We add binary variables $x_{(i,j),(k,l)}$ that equal 1 if the arc $(u_{i,j}, u_{k,l})$ is traversed and add the constraint:

$$\sum_{v_i, v_k, v_l \in V} x_{(i,j),(k,l)} \leq 1 \quad \forall j \neq l \tag{4.1}$$

These constraints mean that for any node v_j in the CmRSP, there is at most one arc emanating from nodes $u_{i,j}$ such that the CmRSP node is changed to v_l. This means that all assignments to v_j, indicated by visits to nodes $u_{i,j}$ must be made consequently. The situation in Figure 4.4 would be forbidden since there are two arcs, emanating from $u_{2,2}$ and $u_{4,2}$, leading to some node $u_{k,l}$, $l \neq 2$. Similarly, in Figure 4.5, the arcs emanating from $u_{7,7}$ and $u_{3,7}$ violate the constraint.

The second type of violation occurs when a regular node is assigned to another regular node that is already assigned elsewhere. An example of this violation is given in Figure 4.6. The GVRP cluster sequence (1,3,2,4) means the following CmRSP route: 'assign v_1 to v_2', 'visit v_3', 'assign v_2 to v_4', 'visit v_4'. Now v_2 is both assigned to v_4 and visited when assigning v_1 to it. The GVRP solution is feasible. Of course, the assignment inflicts an extra cost that would not be present if v_2 was visited directly after the assignment of v_1. However, it can be the case that v_2 is assigned to another route and thus the capacity of the original route is reduced by one. This can lead to savings greater than the additional assignment cost.

In the MCTP, this type of violation is not present as no node can be both visited and assigned.

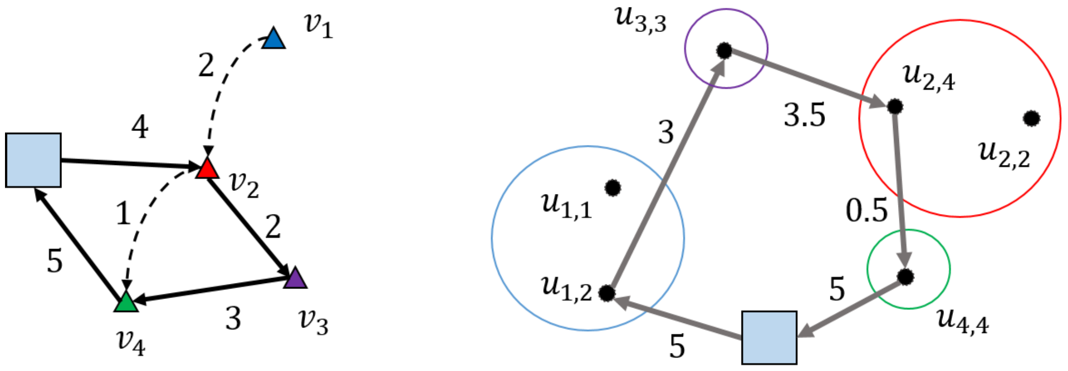

Figure 4.6: An infeasible CmRSP solution (v_2 is both visited and assigned) is feasible in the transformed GVRP.

The violation can be eliminated from the GVRP by adding additional constraints. We use variables $y_{i,j} \in \{0, 1\}$ to indicate if node $u_{i,j}$ is visited.

$$\sum_{i|(i,j)\in A} y_{i,j} \leq M y_{j,j} \quad \forall j \in V \tag{4.2}$$

Constraints 4.2 state that assignments to CmRSP customer v_j can only be made if v_j is visited directly (assigned to itself). Here M is a sufficiently large number to allow any amount of assignments; $M \geq n$.

While these faults in the transformation formulation can be fixed with additional GVRP constraints, we have chosen not to implement them. The focus of this thesis is to see if the GVRP heuristic is capable of solving CmRSP instances with no major ad-hoc adaptations. We also do not try to solve the CmRSP instances exactly based on the GVRP formulation, so the implementation is not necessary. Including the additional constraints in the heuristic algorithm would require significant and computationally expensive changes, since even simple local moves may have large-scale consequences that should be taken into account on every step.

Furthermore, while the violations are not in line with the mathematical CmRSP formulation, the implied alterations can be understood as a part of the application in telecommunications. It should be physically possible to lay two sets of cables through a customer or lay a cable through a specific customer while assigning that customer elsewhere.

With these relaxations, we must monitor if our solutions contain elements that are forbidden in the traditional CmRSP instances to maintain comparability with results found in literature.

Chapter 5

Computational methods

In this chapter, we present the computational methods that were used in order to find solutions for the DGVRP. We briefly describe the heuristic algorithm version used in the Bachelor's Thesis of the author [19] and present a preprocessing algorithm to remove dominated vertices from the GVRP. We present new implementations of a simple shortest-path algorithm and describe the cyclic exchange search method in detail, including the creation of an improvement graph and a label-correcting algorithm to find negative-cost cycles in the graph.

5.1 Heuristic algorithm

In the Bachelor's Thesis of the author [19], we presented a heuristic algorithm that was able to reliably solve small and medium instances to their best known values. The algorithm resembles the one implemented in [14] and mainly consists of a set of local search moves and an exact *Split* method to determine optimal routes for a fixed sequence of clusters.

In this thesis, we use the same basic algorithm structure with some additional features. The entire structure is presented in Algorithm 1. Functions *Preprocessing* and *CyclicExchange* are new features.

InitialRoutes creates an initial set of feasible routes. Two methods are used; the first one creates a route for every cluster, each visiting only one customer. The second initialization method is a randomized greedy procedure that starts from the depot and proceeds to a customer picked randomly from one of the two closest unvisited clusters. The route returns to the depot when capacity or distance constraint limits

Algorithm 1: The metaheuristic algorithm

```
Parameters: $n_i$, $n_m$
Preprocessing()
RoutePool ← ∅
for $i$ ← 1 to $n_i$ do
    Routes ← InitialRoutes($i$)
    Routes ← LocalSearch(Routes)
    Routes ← CyclicExchange(Routes)
    Add Routes to RoutePool
    GiantTour ← Concat(Routes)
    for $j$ ← 1 to $n_m$ do
        MutatedTour ← Mutate(GiantTour)
        Routes ← Split(MutatedTour)
        Add Routes to RoutePool
        Routes ← LocalSearch(Routes)
        Routes ← CyclicExchange(Routes)
        Add Routes to RoutePool
    end
end
Solution ← SetPartitioning(RoutePool)
```

are reached. The combination of these two methods ensures that the initial routes can be diverse and complicated but not always close to feasibility limits.

LocalSearch function is presented in more detail in Algorithm 2. It contains simple moves: *OnePoint* removes a random cluster from the solution and finds the cheapest feasible insertion back to one of the routes. The process is repeated until no new improvements are made in a given amount of iterations. *TwoPoint* chooses a random cluster and determines the best swap with another cluster. This function is also iterated until there are no improvements. The function *ShortestPath* is presented later in more detail. *TwoOpt* is a 2-opt algorithm that is tailored for the GVRP in [10]. A route is called 2-optimal, if there is no improving way of removing a chain of vertices from the route and reinserting them backwards. In practice, a 2-optimal path should not cross itself.

CyclicExchange is, similarly to *ShortestPath*, a new feature and presented later in detail. *Concat* merges the existing locally optimal set of routes to one giant route, preparing the problem for the *Split* algorithm. The same giant route is split multiple times, so there is a function *Mutate* that performs small random perturbations on the giant route.

Algorithm 2: The local search method

```
Input: Routes
Routes ←OnePoint($Routes$)
Routes ←TwoPoint($Routes$)
Routes ←ShortestPath($Routes$)
Routes ←TwoOpt($Routes$)
Routes ←ShortestPath($Routes$)
```

Split is the most important and computationally the most consuming phase of the algorithm. *Split* yields the optimal way to divide a fixed sequence of clusters (the giant tour) into feasible routes. If the distance matrix is not identical or proportionate to the cost matrix, optimality is not guaranteed. The algorithm is presented in more detail in [19], [14] and [1]. In short, the algorithm creates an auxiliary graph where an arc (i, j) represents a route including clusters from the $i + 1$:th to the j:th. The algorithm calculates the costs of these paths dynamically and labels each node with the shortest distance to that node. The label of the final node K is the cost of the optimal solution and the optimal division as well as the visited customers can be backtracked from the final node.

A local search is performed for each solution given by *Split*. Then the original giant route is mutated and split again n_m times. Eventually, new initial routes are created; this happens n_i times. We use values $n_i = 30, n_m = 50$. All the time, locally optimal routes are added to *RoutePool*.

The final part of the algorithm is the *SetPartitioning* procedure. This method utilizes an alternative formulation of the GVRP. Assuming a decision variable z_r for every feasible route for the GVRP with cost c_r, the problem would be to find a least-cost set of routes so that each cluster belongs to exactly one route. The solution space of all feasible solution is enormous; analytically, the problem could be solved with column generation techniques [1]. In this heuristic context, we simply use the set of routes obtained during the entire algorithm, *RoutePool*, and solve the problem with a mixed integer programming solver.

5.2 Preprocessing

5.2.1 GVRP

To reduce the sizes of GVRP instances, we use a preprocessing algorithm (Algorithm 3) proposed in [5]. The algorithm searches for dominated customers. A customer v_i is dominated if any route visiting v_i can be improved by moving to another customer in the cluster. The index of the cluster of v_i is denoted as $\alpha(i)$. More precisely, a customer is dominated if there is no such pair of customers v_k, v_l, $\alpha(i) \neq \alpha(k) \neq \alpha(l) \neq \alpha(i)$, such that a path (v_k, v_i, v_l) with cost $d_{ki} + d_{il}$ is strictly better than any other path (v_k, v_j, v_l), $\alpha(i) = \alpha(j)$.

When a dominated customer is found, it is removed. After the removal, a previously undominated customer may become dominated. This is why the entire preprocessing algorithm is then started again.

Algorithm 3: GVRP preprocessing algorithm

```
Start
for all $C_a \in C$ do
    for all $v_i \in C_a$ do
        IsDominated ← TRUE
        IsNecessary ← TRUE
        for all $v_k \in V \setminus v_{n+1}$, $\alpha(k) \neq \alpha(i)$ do
            for all $v_l \in V \setminus v_0$, $\alpha(l) \neq \alpha(i), \alpha(l) \neq \alpha(k)$ do
                for all $v_j \in C_a$, $i \neq j$ do
                    if $\tilde{q}_i + \tilde{q}_k + \tilde{q}_l \leq Q$ then
                        if $d_{ki} + d_{il} \geq d_{kj} + d_{jl}$ then
                            IsNecessary ← FALSE
                        end
                    end
                end
                if $IsNecessary == TRUE$ then
                    IsDominated ← FALSE
                end
                IsNecessary ← TRUE
            end
        end
        if $IsDominated == TRUE$ then
            Remove $v_i$
            Go to Start
        end
    end
end
```

5.2.2 CmRSP

In the transformation of CmRSP to GVRP, each feasible assignment option adds one customer to the GVRP. Since there may be multiple such options for each regular node in the CmRSP, there is a risk of the GVRP transformation becoming excessively large. To mitigate this effect while maintaining good solution quality, we try eliminating assignments with the greatest cost; these are less likely to be used. We test the effect of the removals with three scenarios: 1 assignment option, 2 assignment options and unlimited assignment options. In case of a tie, all equal-cost assignments are included. In the optimal situation, the heuristic removals succeed in lowering the average computation time while leaving solution quality unchanged.

5.3 Shortest-path algorithm

We append our Local Search algorithm with a simple shortest-path algorithm, presented in Algorithm 4. This procedure makes sure that each route is, in fact, the shortest possible way of traversing the clusters of that route in the given order. The algorithm is based on setting labels L for each customer of the route. The labels are set one cluster at a time, considering all connections between the cluster and its predecessor. The labels are updated to equal the shortest-path cost from the depot to that customer. Finally, the label $L(n+1)$ equals the cost of the shortest path traversing from v_0 to v_{n+1}. The optimal predecessors for each customer are stored with labels P. This way, we can backtrace the optimal path from $P(n+1)$.

Algorithm 4: Shortest-path algorithm

```
Input: sequence of clusters $(C_1, C_2, ..., C_u)$
for $j \leftarrow 1$ to $n$ do
    L(j) $\leftarrow \infty$
    P(j) $\leftarrow \emptyset$
end
L(0) $\leftarrow 0$
for $i \leftarrow 1$ to $u$ do
    for all $v_j \in C_i$ do
        L(j) $\leftarrow \min_{k \in C_{i-1}} \mathrm{L}(k) + d_{kj}$
        P(j) $\leftarrow \operatorname{argmin}_{k \in C_{i-1}} \mathrm{L}(k) + d_{kj}$
    end
end
L(n+1) $\leftarrow \min_{k \in C_u} \mathrm{L}(k) + d_{k,n+1}$
P(n+1) $\leftarrow \operatorname{argmin}_{k \in C_u} \mathrm{L}(k) + d_{k,n+1}$
```

5.4 Cyclic exchange

The methods used in the local search procedure consider changes in the local neighbourhood of the solution. The moves used by the current local search algorithm (relocation, swap and 2-opt) are only capable of altering two routes at a time, making their neighbourhoods relatively small.

Solving the problem to optimality with respect to these small neighbourhoods may not yield the best solutions concerning the whole problem. In order to escape from local optima we use a Very Large Neighbourhood Search method called Cyclic Exchange. This method can potentially alter every route of the solution with each move.

5.4.1 Improvement graph

A feasible GVRP solution consists of m tours covering K clusters so that all clusters are visited exactly once. The solution is thus a partition, in which the cluster set $C = \{C_1, C_2, ..., C_K\}$ is partitioned into m subsets. We denote the set of m routes as $S = \{S_1, S_2, ..., S_m\}$.

It is possible to perform a cyclic exchange neighbourhood search on a feasible solution. In the search, we remove a set of clusters from one route and insert them to another route. A set of clusters is, in turn, removed from this other route and inserted elsewhere. In this implementation, the considered cluster sets are either singletons (one cluster) or pairs of clusters that are connected by an arc.

This process of removals and insertions is completed when a set of clusters is inserted back to the original route, completing the cycle. This is the *cyclic exchange* method.

Each route can only appear in the cycle once. Because of this, we can be sure that the cost of any route after an insertion and a deletion will not be changed later. Thus, the exact cost of each possible insertion/deletion pair can be calculated beforehand.

We can also end the process without completing the cycle, by choosing not to remove any clusters from a route after an insertion. This method is known as *path exchange*. Path exchange is possible if the final route can extend to contain a longer route with more load. If the starting point of the path contains no more clusters than the ones removed, the amount of routes can reduce by one.

Algorithm 5: The improvement graph construction algorithm

```
for $i \leftarrow 1$ to $K$ do
    Add node $p_i$
end
for $j \leftarrow 1$ to $K - m$ do
    Add node $p_j$
end
for $k \leftarrow 1$ to $m$ do
    Add node $s_k$
end
for $i \leftarrow 1$ to $2K - m$ do
    for $j \leftarrow 1$ to $2K - m$, $i \neq j$ do
        Route $\leftarrow S(p_j)$
        NewRoute $\leftarrow$ Route without cluster or pair $j$
        (B($p_i$,$p_j$),mincost) $\leftarrow$ Insert($C_i$,$NewRoute$)
        $\alpha_{ij} \leftarrow$ cost(NewRoute) + mincost – cost(Route)
    end
    for $l \leftarrow 1$ to $m$ do
        Route $\leftarrow S_l$
        (B($p_i$,$s_l$),mincost) $\leftarrow$ Insert($C_i$,$Route$)
        $\alpha_{il} \leftarrow$ mincost
    end
end
for $l \leftarrow 1$ to $m$ do
    for $i \leftarrow 1$ to $2K - m$ do
        NewRoute $\leftarrow$ Route without cluster or pair $i$
        $\alpha_{li} \leftarrow$ cost(NewRoute)-cost(Route)
    end
end
```

To find improving cyclic moves, we create an improvement graph to search negative-cost paths. The graph construction is presented in Algorithms 5-6. The graph contains a set of *regular nodes* p_i for each of the K clusters and another set p_{ij} for the $K - m$ consecutive cluster pairs in the original problem. We denote the route that contains node p_i as $S(p_i)$. Additionally, there are m *pseudonodes* s_k for every route S_k. In total, the improvement graph consists of $2K$ nodes.

An example GVRP solution with 6 clusters and 3 routes and the corresponding improvement graph are presented in Figure 5.1.

Traversing an arc (i, j) between regular nodes p_i and p_j in the improvement graph represents the cluster C_i replacing cluster C_j on route $S(p_j)$. The same principle

Algorithm 6: Cluster insertion algorithm

```
Parameters: ($C_i$, Route)
place ← $\emptyset$
mincost ← $\infty$
if $load(C_i)+RouteLoad(Route) \leq Q$ then
    for all $C_k \in \{Route \cup C_0\}$ do
        $C_{next}$ ← the cluster directly after $C_k$
        insert $C_i$ between $C_k$ and $C_{next}$
        addedcost ← cost(Modified route) – cost(Route)
        if $addedcost < mincost$ then
            mincost=addedcost
            place=$C_k$
        end
    end
end
return place, mincost
```

applies when cluster pairs are involved. For infeasible insertions, the weight α_{ij} of the arc is very large. Otherwise, α_{ij} is equal to the increase in the cost of route $S(p_j)$ after the deletion and the optimal insertion. If the cost of the route is decreased, the arc has a negative weight. The optimal places for replacements are stored in matrix B.

Note that the saving created by removing C_i from its original route is not taken into account in the cost α_{ij}, since it is considered in the arcs leading to p_i. Similarly, the cost of inserting C_j to another route is considered with arcs emanating from p_j.

Arcs (i, k) between regular nodes and pseudonodes describe the insertion of a cluster or pair to route S_k without removing any clusters from it. The weight α_{ik} of a feasible arc equals the increased cost of route S_k after optimal insertion.

With path exhange, we have to separately consider the saving caused by deleting a cluster from the first node in a chain. This is executed by defining additional arcs from each pseudonode s_k to a termination node u with weight 0, and from u to every regular node with negative weight corresponding to the saving obtained by deleting that cluster or pair. This way, also path exhange solutions form cycles in the improvement graph.

To illustrate, we calculate some arc weights for our example improvement graph and examine how the solution is altered. In Figure 5.2, the arc from p_3 to p_5 is for replacing cluster C_5 with cluster C_3 on route S_2. The old cost of route S_2 is 12. With the deletion and insertion, the cost is 9; thus, the weight of the arc is -3.

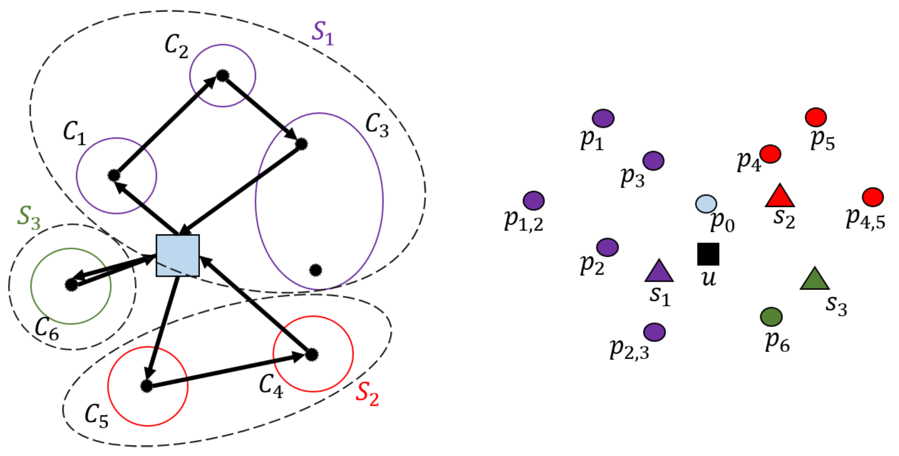

Figure 5.1: A GVRP solution and the corresponding improvement graph

The arc from p_5 to the triangular pseudonode s_3 means that the newly removed cluster C_5 is inserted to S_3 without removing clusters from it. This increases the cost of S_3 from 2 to 4, so the weight of the arc is 4. The pseudonode is connected to the termination node u, which is connected to all regular nodes, including p_3. The weight of this arc is equal to the saving caused by deleting cluster C_3 from route S_1, which has not yet been considered. The saving, according to Figure 5.2, is -2.

The resulting cycle (p_3, p_5, s_3, u, p_3) has a combined weight of -1. This means that executing the cycle improves the solution by 1. After this move, the improvement graph must be updated according to Figure 5.3. Weights need to be recalculated for every arc such that either one of its endpoints belong to routes that were altered. In addition, the route (or 'color') of the regular nodes on the cycle have changed. Also the cluster pairs are now different, but the number of pairs remains the same.

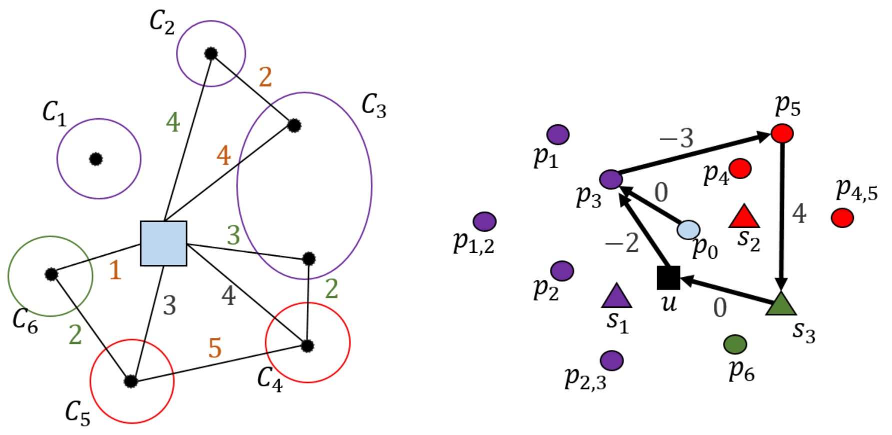

Figure 5.2: The arc weights of the improvement graph are based on the costs and savings in the GVRP.

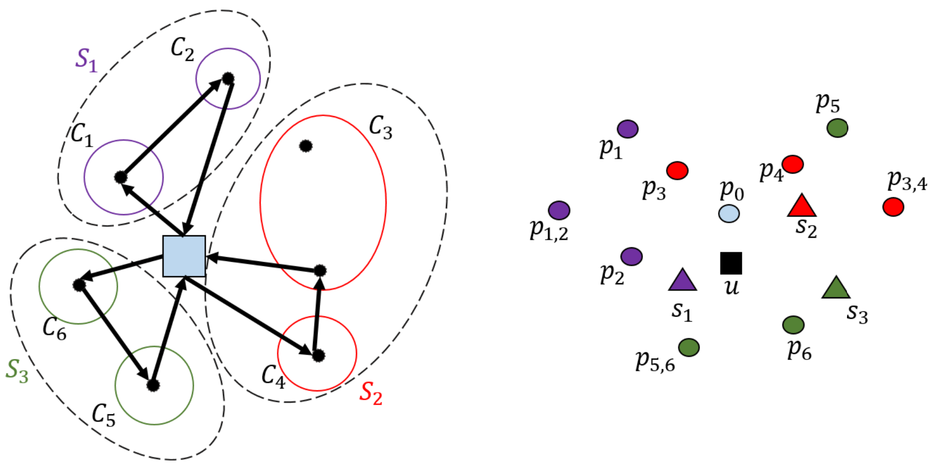

Figure 5.3: The improvement graph is changed after an improving cyclic exchange move.

5.4.2 Label-correcting algorithm

All cyclic and path exhange solutions have a corresponding cycle of arcs in the improvement graph. The total sum of weights on each cycle equals the change in the objective function if the corresponding move is executed. Thus, finding a negative-cost path in the graph is equivalent of finding an improved solution. In order to find such cycles in the improvement graph, we use a label correcting algorithm (Algorithm 7) similar to that used in [2] in the context of the Capacitated Minimum Spanning Tree Problem.

The negative-path search algorithm can be rooted at any node. However, it is not guaranteed that every node can be reached from the root node. In order to search the whole graph at once, we add another artificial root node p_0 that is connected to every regular node p_i with weight $\alpha_{0i} = 0$. This way, every node is included in the initial list and must be examined at least once. There are no arcs leading to p_0 so it cannot be a part of a feasible cycle.

In the algorithm we use distance labels d for all the nodes. These labels are first set to ∞ and corrected as the algorithm proceeds. The predecessors of each node are stored in $Pred$. The path of a node p_i is denoted as $P(i)$. The path consists of all the nodes and arcs that can be backtraced from p_i to p_0.

Algorithm 7: Finding improving cycles

```
for $j \leftarrow 1$ to $(2K)$ do
    d(j) $\leftarrow \infty$
end
d(0)=0
List $\leftarrow \{0\}$
while List not empty do
    i $\leftarrow$ a random node from List
    for all arcs (i,j) do
        if $d(i) + \alpha_{ij} < d(j)$ then
            backtrace path P(i) using labels Pred(i)
            if P(i) already includes a node from the route of j (not j itself)
             then
                do nothing
            end
            else if $j \in P(i)$ then
                an improving cycle $((i,j),\ P(i) \setminus P(j))$ is found; return
            end
            else if j is a regular node then
                $d(j) \leftarrow d(i) + \alpha_{ij}$
                Pred(j) $\leftarrow$ i
                add j to List
            end
            else
                denote $t$ as the second node on path $P(i)$ after $p_0$
                a cycle $(P(i),\ (i,j),\ (j,u),\ (u,t))$ is found
                if the cycle is improving return
            end
        end
    end
    remove i from List
end
```

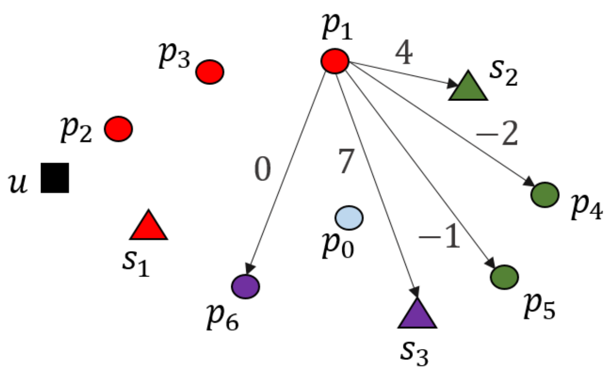

Figure 5.4: The first actual step of the label correcting algorithm

The list of nodes that should be inspected (*List*) contains only p_0 at first. When inspecting a node p_i, all possible values of $d(i) + \alpha_{ij}$ are considered. If $d(i) + \alpha_{ij} < d(j)$ and the path $P(i) \cup (i, j)$ remains subset-disjoint (containing no more than one element from each route), the label $d(j)$ is updated to $d(i) + \alpha_{ij}$ and p_j is added to *List*.

After the first step including p_0, $d(i) = 0$ for all regular nodes p_i and all regular nodes are included in *List*. An example is presented in Figure 5.4. All label values can be seen in Table 5.1.

A random node p_1 is then selected from *List*. Labels $d(j)$ are updated for each node p_j that does not belong to the subset of p_1 and such that $\alpha_{1j} < 0$. When pseudonodes s_2 and s_3 are considered, the potential cycle, e.g., (p_1, s_3, u, p_1), is checked (Figure 5.5). In the example in the figure, $7 - 3 > d(1) = 0$; no improving cycle is found. The algorithm continues; the label for s_3 is updated and p_1 is removed from *List*.

Next, another random node p_4 is selected in Figure 5.6. Because path $P(4)$ already includes a red node, the labels for any red (or green) nodes remain unchanged. The only exception is p_1: if it would be so that $d(4) + \alpha_{41} < d(1)$ then a valid cycle would have been found. This is not the case here. Only the purple nodes p_6 and s_3 can then be feasibly included in the path and their labels are corrected.

Finally, assume p_6 is the next random node from *List* (Figure 5.7). The path $P(6)$ already contains all colors so the options are limited to those nodes that belong to $P(6)$. Of these two, we observe that $d(6)+\alpha_{61} < d(1)$: $-4+2 < 0$. Thus, (p_1, p_4, p_6) is a valid negative cycle and it is returned.

If no improvements were found, the algorithm would continue with another random choice from *List*. Note that although nodes of all colors have now been examined,

Node	List	Label	Pred
p_1	1	0	0
p_2	1	0	0
p_3	1	0	0
p_4	1	0	0
p_5	1	0	0
p_6	1	0	0
s_1	-	∞	-
s_2	-	∞	-
s_3	-	∞	-

Table 5.1: Labels corresponding to the initial situation

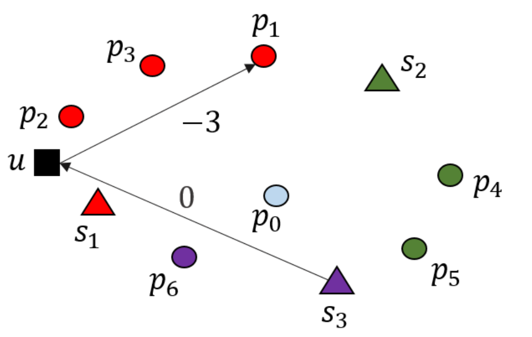

Figure 5.5: Checking a possible path exchange cycle

they are not permanently 'used'; because the subset-disjoint condition only concerns the path of the current node, all totally unexamined nodes (such as p_2 or p_5) are free to form more paths as long as the label-correcting condition $d(i) + \alpha_{ij} < d(j)$ is met. If the label for already examined nodes p_4 or p_6 can be improved, they are again added to *List* and examined at some later time.

The label-correcting terminates whenever an improving cycle is found or when *List* is empty.

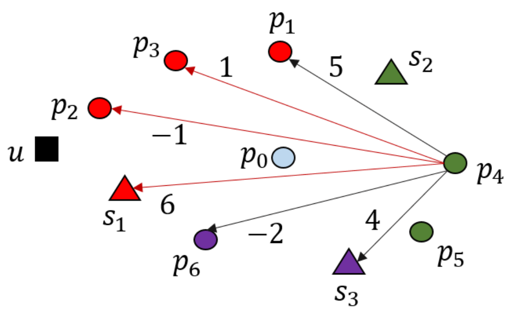

Figure 5.6: Second step: examining p_4

Node	List	Label	Pred
p_1	0	0	0
p_2	1	0	0
p_3	1	0	0
p_4	1	-2	1
p_5	1	-1	1
p_6	1	0	0
s_1	-	∞	-
s_2	-	4	1
s_3	-	7	1

Table 5.2: Labels after examining p_1

The cyclic exchange method is based on the property that the cost of moves is easy to calculate as long as at most one cluster or pair of clusters is removed from one route. Enforcing this subset-disjoint condition in the label-correcting algorithm does narrow the scope of the search. Even with the current restrictions, it is possible that a path will include more than one node of the same color. Assume a node p_i with color label l_i and its path $P(i)$ that contains a node p_j with color label

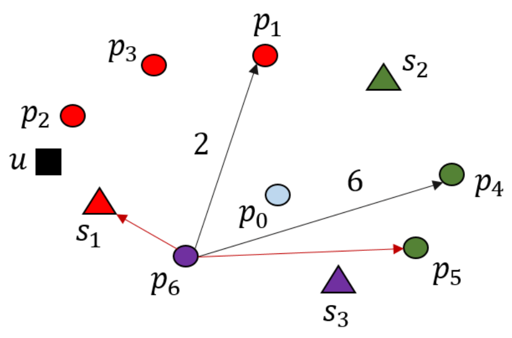

Figure 5.7: Third step: examining p_6

Node	List	Label	Pred
p_1	0	0	0
p_2	1	0	0
p_3	1	0	0
p_4	1	-2	1
p_5	1	-1	1
p_6	1	-4	4
s_1	-	∞	-
s_2	-	4	1
s_3	-	2	4

Table 5.3: Labels after examining p_4

l_j. Now, consider that another node p_k with color l_i is being examined and that $d(k) + \alpha_{kj} < d(j)$. If the path $P(k)$ does not include color l_j, p_j can be appended to path $P(k)$, and thus the path $P(j)$ is updated to $(P(k) \cup p_j)$. Path $P(i)$ now includes two nodes with color l_i.

Based on our previous examples, it may seem that there are very few choices to expand paths and that the random choices strongly determine the outcome. This is true to a certain extent. However, the more there are routes the less nodes are eliminated from consideration. More importantly, typically only a small amount of the arc weights α_{ij} are negative and this condition is required to form initial two-node paths. Thus, the entire graph is not 'exhausted' as quickly as in the example.

It is important to stress that the label-correcting algorithm we use is a heuristic method. The algorithm may go through all of the nodes and find no negative cycles even though some exist. Additionally, our implementation does not try to find the best cycle but returns the first one it finds. This is not that much of a drawback, since the algorithm is executed iteratively until no improvements are found.

Chapter 6

Results

In this chapter, the results of the heuristic algorithm are presented. The algorithm is tested on a set of GVRP instances as well as the transformed CmRSP and MCTP instances. Concerning GVRP, the algorithm outperforms all approaches found in literature and is able to find two previously unknown best solutions. On the CmRSP and MCTP, the algorithm performs adequately but does not reach the level of current literature by means of solution quality or computational efficiency.

6.1 Heuristic performance on GVRP

The revised GVRP algorithm is tested on a set of 158 instances. These instances were introduced in [5] based on a set of CVRP instances. The instances are divided into five sets A, B, P, M and G. Sets A, B and P contain 20-30 small to medium instances each, with the customer amount ranging from 16 to 101. Set M contains four larger instances with 101-200 customers and set G has one instance with 262 customers.

There are two versions of each instance that differ with regard to clusters: the amount of clusters K is $\lceil n/\theta \rceil$ where $\theta \in \{2, 3\}$. All instances where there are two customers per cluster on average belong to set T2 and those with three to T3. Note that the amount of customers per cluster is on average; there are singleton clusters and large clusters in all instances. The number of customers and clusters can be deduced from the instance name; for instance 'P-n45-k5-C23' would include 45 customers and 23 clusters.

The small and medium instances are solved once each and the results are presented in tables 6.1 to 6.6. In these tables, the column 'Instance' reports the instance name and 'n' the number of customers after the preprocessing algorithm may have removed some. The number of removed customers is reported in 'R', the number of clusters in 'K' and the fleet size in 'm'. Column t reports the computing time of each instance. Columns Ub and Ub^* contain the best upper bounds found by our heuristic algorithm and the known optimal solution cost or best known upper bound, respectively. The value of Ub is bolded if the heuristic solution reaches the best known upper bound.

Table 6.1: Heuristic algorithm results for the instances of set T2-A with $K = \lceil n/2 \rceil$

Instance	n	R	K	m	t	Ub	Ub^*
A-n32-k5-C16	30	2	16	3	2.60	**508**	508
A-n33-k5-C17	32	1	17	3	2.54	**451**	451
A-n33-k6-C17	29	4	17	3	2.17	**465**	465
A-n34-k5-C17	30	4	17	3	2.56	**489**	489
A-n36-k5-C18	36	0	18	3	2.79	**502**	502
A-n37-k5-C19	34	3	19	3	2.83	**432**	432
A-n37-k6-C19	36	1	19	3	3.18	**584**	584
A-n38-k5-C19	37	1	19	3	3.08	**476**	476
A-n39-k5-C20	36	3	20	3	3.66	**557**	557
A-n39-k6-C20	36	3	20	3	3.82	**544**	544
A-n44-k6-C22	41	3	22	3	4.85	**608**	608
A-n45-k6-C23	39	6	23	4	4.26	**613**	613
A-n45-k7-C23	41	4	23	4	5.97	**674**	674
A-n46-k7-C23	45	1	23	4	4.65	**593**	593
A-n48-k7-C24	43	5	24	4	5.98	**667**	667
A-n53-k7-C27	49	4	27	4	7.56	**603**	603
A-n54-k7-C27	53	1	27	4	9.19	**690**	690
A-n55-k9-C28	51	4	28	5	7.48	**699**	699
A-n60-k9-C30	55	5	30	5	13.04	**769**	769
A-n61-k9-C31	56	5	31	5	9.57	**638**	638
A-n62-k8-C31	59	3	31	4	15.31	**740**	740
A-n63-k9-C32	62	1	32	5	14.76	**912**	912
A-n63-k10-C32	59	4	32	5	13.16	**801**	801
A-n64-k9-C32	61	3	32	5	13.71	**763**	763
A-n65-k9-C33	59	6	33	5	11.31	**682**	682
A-n69-k9-C35	58	11	35	5	12.81	**680**	680
A-n80-k10-C40	77	3	40	5	27.00	**997**	997

Table 6.2: Heuristic algorithm results for the instances of set T2-B with $K = \lceil n/2 \rceil$

Instance	n	R	K	m	t	Ub	Ub^*
B-n31-k5-C16	27	4	16	3	2.51	**441**	441
B-n34-k5-C17	29	5	17	3	2.47	**472**	472
B-n35-k5-C18	32	3	18	3	2.68	**626**	626
B-n38-k6-C19	35	3	19	3	3.39	**451**	451
B-n39-k5-C20	35	4	20	3	2.72	**357**	357
B-n41-k6-C21	34	7	21	3	3.39	**481**	481
B-n43-k6-C22	40	3	22	3	5.29	**483**	483
B-n44-k7-C22	39	5	22	4	5.43	**540**	540
B-n45-k5-C23	43	2	23	3	4.79	**497**	497
B-n45-k6-C23	39	6	23	4	5.20	**478**	478
B-n50-k7-C25	43	7	25	4	4.81	**449**	449
B-n50-k8-C25	49	1	25	5	8.72	**916**	916
B-n51-k7-C26	44	7	26	4	5.52	**651**	651
B-n52-k7-C26	43	9	26	4	5.41	**450**	450
B-n56-k7-C28	48	8	28	4	7.44	**486**	486
B-n57-k7-C29	46	11	29	4	14.37	**751**	751
B-n57-k9-C29	52	5	29	5	11.95	**942**	942
B-n63-k10-C32	58	5	32	5	11.71	**816**	816
B-n64-k9-C32	56	8	32	5	9.18	**509**	509
B-n66-k9-C33	63	3	33	5	15.94	**808**	808
B-n67-k10-C34	63	4	34	5	14.84	**673**	673
B-n68-k9-C34	58	10	34	5	14.30	**704**	704
B-n78-k10-C39	75	3	39	5	21.89	**803**	803

Table 6.3: Heuristic algorithm results for the instances of set T2-P with $K = \lceil n/2 \rceil$

Instance	n	R	K	m	t	Ub	Ub^*
P-n16-k8-C8	13	3	8	5	0.78	**239**	239
P-n19-k2-C10	16	3	10	2	0.69	**147**	147
P-n20-k2-C10	17	3	10	2	0.84	**154**	154
P-n21-k2-C11	19	2	11	2	1.02	**160**	160
P-n22-k2-C11	20	2	11	2	0.94	**162**	162
P-n22-k8-C11	21	1	11	5	1.02	**314**	314
P-n23-k8-C12	18	5	12	5	1.13	**312**	312
P-n40-k5-C20	37	3	20	3	3.12	**294**	294
P-n45-k5-C23	43	2	23	3	4.08	**337**	337
P-n50-k7-C25	48	2	25	4	5.76	**353**	353
P-n50-k8-C25	48	2	25	5	5.30	**372**	372
P-n50-k10-C25	48	2	25	5	5.38	**410**	410
P-n51-k10-C26	50	1	26	6	6.36	**427**	427
P-n55-k7-C28	53	2	28	4	7.97	**361**	361
P-n55-k8-C28	53	2	28	4	8.81	**361**	361
P-n55-k10-C28	53	2	28	5	7.83	**415**	415
P-n55-k15-C28	53	2	28	9	5.99	**551**	551
P-n60-k10-C30	57	3	30	5	9.24	**443**	443
P-n60-k15-C30	56	4	30	8	7.50	**565**	565
P-n65-k10-C33	65	0	33	5	11.71	**487**	487
P-n70-k10-C35	69	1	35	5	13.61	**485**	485
P-n76-k4-C38	74	2	38	2	19.05	**383**	383
P-n76-k5-C38	74	2	38	3	19.74	**405**	405
P-n101-k4-C51	100	1	51	2	58.96	**455**	455

Table 6.4: Heuristic algorithm results for the instances of set T3-A with $K = \lceil n/3 \rceil$

Instance	n	R	K	m	t	Ub	Ub^*
A-n32-k5-C11	26	6	11	2	1.12	**386**	386
A-n33-k5-C11	26	7	11	2	1.17	**315**	315
A-n33-k6-C11	26	7	11	2	1.20	**370**	370
A-n34-k5-C12	29	5	12	2	1.51	**419**	419
A-n36-k5-C12	31	5	12	2	1.39	**396**	396
A-n37-k5-C13	30	7	13	2	1.44	**347**	347
A-n37-k6-C13	34	3	13	2	1.70	**431**	431
A-n38-k5-C13	32	6	13	2	1.61	**367**	367
A-n39-k5-C13	34	5	13	2	1.71	**364**	364
A-n39-k6-C13	32	7	13	2	1.66	**403**	403
A-n44-k6-C15	41	3	15	3	3.14	**491**	491
A-n45-k6-C15	36	9	15	3	2.16	**474**	474
A-n45-k7-C15	39	6	15	3	2.46	**475**	475
A-n46-k7-C16	40	6	16	3	2.63	**462**	462
A-n48-k7-C16	40	8	16	3	2.63	**451**	451
A-n53-k7-C18	45	8	18	3	3.17	**440**	440
A-n54-k7-C18	49	5	18	3	3.61	**482**	482
A-n55-k9-C19	47	8	19	3	3.77	**473**	473
A-n60-k9-C20	50	10	20	3	5.17	**595**	595
A-n61-k9-C21	51	10	21	4	4.88	**473**	473
A-n62-k8-C21	58	4	21	3	6.25	**596**	596
A-n63-k9-C21	56	7	21	3	5.17	**642**	642
A-n63-k10-C21	50	13	21	4	4.98	**593**	593
A-n64-k9-C22	59	5	22	3	5.96	**536**	536
A-n65-k9-C22	53	12	22	3	5.03	**500**	500
A-n69-k9-C23	58	11	23	3	5.60	**520**	520
A-n80-k10-C27	76	4	27	4	14.40	**710**	710

Table 6.5: Heuristic algorithm results for the instances of set T3-B with $K = \lceil n/3 \rceil$

Instance	n	R	K	m	t	Ub	Ub^*
B-n31-k5-C11	25	6	11	2	1.36	**356**	356
B-n34-k5-C12	27	7	12	2	1.37	**369**	369
B-n35-k5-C12	28	7	12	2	1.32	**501**	501
B-n38-k6-C13	34	4	13	2	2.03	**370**	370
B-n39-k5-C13	24	15	13	2	1.23	**280**	280
B-n41-k6-C14	32	9	14	2	1.96	**407**	407
B-n43-k6-C15	37	6	15	2	2.38	**343**	343
B-n44-k7-C15	34	10	15	3	2.12	**395**	395
B-n45-k5-C15	38	7	15	2	2.61	**410**	410
B-n45-k6-C15	38	7	15	2	2.27	**336**	336
B-n50-k7-C17	41	9	17	3	2.74	**393**	393
B-n50-k8-C17	42	8	17	3	3.38	**598**	598
B-n51-k7-C17	40	11	17	3	3.27	**511**	511
B-n52-k7-C18	33	19	18	3	3.31	**359**	359
B-n56-k7-C19	44	12	19	3	3.63	**356**	356
B-n57-k7-C19	43	14	19	3	4.16	**558**	558
B-n57-k9-C19	48	9	19	3	5.02	**681**	681
B-n63-k10-C21	55	8	21	3	4.87	**599**	599
B-n64-k9-C22	53	11	22	4	5.54	**452**	452
B-n66-k9-C22	47	19	22	3	5.46	**609**	609
B-n67-k10-C23	61	6	23	4	8.45	**558**	558
B-n68-k9-C23	50	18	23	3	6.56	**523**	523
B-n78-k10-C26	69	9	26	4	10.29	**606**	606

Table 6.6: Heuristic algorithm results for the instances of set T3-P with $K = \lceil n/3 \rceil$

Instance	n	R	K	m	t	Ub	Ub^*
P-n16-k8-C6	9	7	6	4	0.54	**170**	170
P-n19-k2-C7	16	3	7	1	0.47	**111**	111
P-n20-k2-C7	15	5	7	1	0.46	**117**	117
P-n21-k2-C7	17	4	7	1	0.48	**117**	117
P-n22-k2-C8	16	6	8	1	0.50	**111**	111
P-n22-k8-C8	16	6	8	4	0.83	**249**	249
P-n23-k8-C8	16	7	8	3	0.69	**174**	174
P-n40-k5-C14	33	7	14	2	1.75	**213**	213
P-n45-k5-C15	39	6	15	2	2.20	**238**	238
P-n50-k7-C17	45	5	17	3	3.14	**261**	261
P-n50-k8-C17	45	5	17	3	3.03	**262**	262
P-n50-k10-C17	45	5	17	4	2.99	**292**	292
P-n51-k10-C17	47	4	17	4	3.04	**309**	309
P-n55-k7-C19	51	4	19	3	4.48	**271**	271
P-n55-k8-C19	51	4	19	3	4.57	**274**	274
P-n55-k10-C19	51	4	19	4	3.79	**301**	301
P-n55-k15-C19	51	4	19	6	3.36	**378**	378
P-n60-k10-C20	53	7	20	4	4.25	**325**	325
P-n60-k15-C20	54	6	20	6	3.82	**374**	374
P-n65-k10-C22	63	2	22	4	5.30	**372**	372
P-n70-k10-C24	67	3	24	4	6.44	**385**	385
P-n76-k4-C26	73	3	26	2	8.89	**309**	309
P-n76-k5-C26	73	3	26	2	9.83	**309**	309
P-n101-k4-C34	100	1	34	2	20.42	**370**	370

The algorithm is able to find the optimal or best known solution on every instance of the sets A, B and P in both T2 and T3. The preprocessing algorithm is able to reduce the problem size on nearly every instance, with up to 19 removals on some instances in T3-B. The smallest instances are solved within a few seconds and the largest ones in about 20 seconds.

A summary of the results and comparison to the Bachelor's Thesis [19] and related literature [5, 20, 14, 1] can be found in Table 6.7. Columns 'Succ' and '$\bar{t}$' report the amount of instances solved to best known values and the average computing time for each instance set with θ values 2 and 3. We also report the CPU clock speed (in GHz) and processor type for each algorithm, and indicate how many times the algorithm was run to obtain the results.

Table 6.7: Summary and comparison of the algorithm performance on small and medium instances

		Master's Thesis		Bachelor's Thesis		Bektaş et al.		Moccia et al.		Hà et al.		Afsar et al.	
Subset	θ	Succ	$\bar{t}$	Succ	$\bar{t}$	Succ	$\bar{t}$	Succ	$\bar{t}$	Succ	$\bar{t}$	Succ	$\bar{t}$
A	2	**27/27**	7.77	**27/27**	16.54	**25/25**	0.48	**25/25**	16.74	**27/27**	26.54	**27/27**	0.74
B	2	**23/23**	8.00	**23/23**	17.18	**23/23**	0.50	21/23	17.07	**23/23**	28.17	20/23	0.88
P	2	**24/24**	8.62	23/24	17.62	22/24	0.53	20/24	20.12	**24/24**	35.18	22/24	0.92
A	3	**27/27**	3.54	**27/27**	12.80	25/26	0.32	25/26	11.34	**27/27**	23.55	**27/27**	0.39
B	3	**23/23**	3.71	**23/23**	12.80	22/23	0.32	**23/23**	11.56	**23/23**	24.16	**23/23**	0.45
P	3	**24/24**	3.97	**24/24**	13.46	21/24	0.34	23/24	10.45	**24/24**	41.59	23/24	0.45
GHz & Runs		2.67	1	2.67	5	2.4	1	1.83	1	2.4	1	3.0	2
Processor type		IC2 Quad Q9400		IC2 Quad Q9400		AMD Opteron 250		Intel Core Duo		Intel Xeon		Intel Core2 Duo	

Compared to the Bachelor's Thesis, we have been able to reduce the average computation time with more than a half on instance set T2 and more than two thirds on set T3. This is because of the preprocessing phase and some in-code optimization. In addition, we were able to reach all best known solutions with a single run whereas there was one instance in T2-P that could not be solved with five trials in the Bachelor's Thesis.

Concerning the results in related literature, only Hà et al. are able to find all best known solutions. Taking the processor types into account, we can compare the computing times as conducted in the Bachelor's Thesis [19]. Our processor is the fastest, with performance similar to that of Afsar et al. The processor of Hà et al. is about 20% slower and the other about 50% slower than the fastest processors. This implies that our algorithm is clearly faster than that of Hà et al. and about as fast as that of Moccia et al. The algorithms of Bektaş et al. and Afsar et al. are clearly the fastest but they are also poorer when it comes to solution quality.

The large instances in sets M and G are solved five times. The summary and comparison of these results are presented and compared in Table 6.8. Columns 'Instance', 'θ' and 'Lb' report the instance name, clustering type and known lower bounds from literature (respectively). Only three of these instances have been solved to optimality. We report the best found upper bounds in columns 'Ub'; concerning this thesis, the average bound of five runs is reported in 'Avg'. Columns 'm' report the number of routes in the best solution found; because we treat m as flexible, the results are not necessarily comparable. Columns '$\bar{t}$' contain the (average) computing times. Best known upper bounds and optimal lower bounds are bolded.

First, we note that our algorithm is able to find all best known solutions, two of which (M-n200-k16-C100 and G-n262-k25-C131) were previously unknown. All of the results are also feasible concerning the problem with fixed fleet size. Each of the algorithms from literature only finds seven of the ten best known values. Compared to the Bachelor's Thesis, there is clear improvement; only half of the best known values were found with five runs. The computing times have also lowered, but not to such an extent as with the smaller instances.

The computation time of our algorithm seems to grow faster with increasing problem size than the other algorithms. Taking processor types into account, the algorithm is faster than that of Hà et. al but slower than the others.

The two new best known solutions for instances M-n200-k16-C100 and G-n262-k25-C131 are presented in figures 6.1 and 6.2.

Table 6.8: Heuristic algorithm result comparison for big instances

			Master's Thesis				Bachelor's Thesis			Bektaş et al.			Moccia et al.			Hà et al.		
Instance	θ	*Lb*	*Ub*	Avg	m	$\bar{t}$	*Ub*	m	$\bar{t}$	*Ub*	m	$\bar{t}$	*Ub*	m	$\bar{t}$	*Ub*	m	$\bar{t}$
M-n101-k10-C51	2	**542.00**	**542**	**542.0**	5	40.11	**542**	5	66.7	**542**	5	1.50	**542**	5	57.8	**542**	5	124.2
M-n121-k7-C61	2	705.84	**719**	**719.0**	4	148.3	721	4	110.9	**719**	4	2.15	720	4	98.3	**719**	4	234.3
M-n151-k12-C76	2	634.65	**659**	**659.0**	6	148.7	**659**	6	219.4	**659**	6	3.24	**659**	6	113.1	**659**	6	306.0
M-n200-k16-C100	2	752.14	**786**	787.4	8	299.0	791	9	499.3	791	8	5.34	805	8	158.7	789	9	454.0
G-n262-k25-C131	2	2945.02	**3248**	3253	12	811.9	3285	13	1204.4	3249	12	6.22	3319	12	193.6	3303	13	822.8
M-n101-k10-C34	3	**458.00**	**458**	**458.0**	4	19.35	**458**	4	46.4	**458**	4	0.86	**458**	4	36.8	**458**	4	152.3
M-n121-k7-C41	3	**527.00**	**527**	**527.0**	3	53.13	531	3	79.5	**527**	3	1.19	**527**	3	63.3	**527**	3	238.2
M-n151-k12-C51	3	474.32	**483**	**483.0**	4	87.45	**483**	4	155.8	**483**	4	1.89	**483**	4	85.5	**483**	4	434.3
M-n200-k16-C67	3	572.81	**605**	**605.0**	6	157.9	**605**	6	354.8	**605**	6	3.03	**605**	6	108.4	**605**	6	602.0
G-n262-k25-C88	3	2239.50	**2463**	2470.2	9	383.8	2486	9	830.7	2476	9	4.92	**2463**	9	134.3	2477	9	861.7
CPU speed (GHz) & Runs			2.67	5			2.67	5		2.4	1		1.83	1		2.4	1	
Processor			Intel Core2 Quad Q9400				IC2 Quad Q9400			AMD Opteron 250			Intel Core Duo			Intel Xeon		

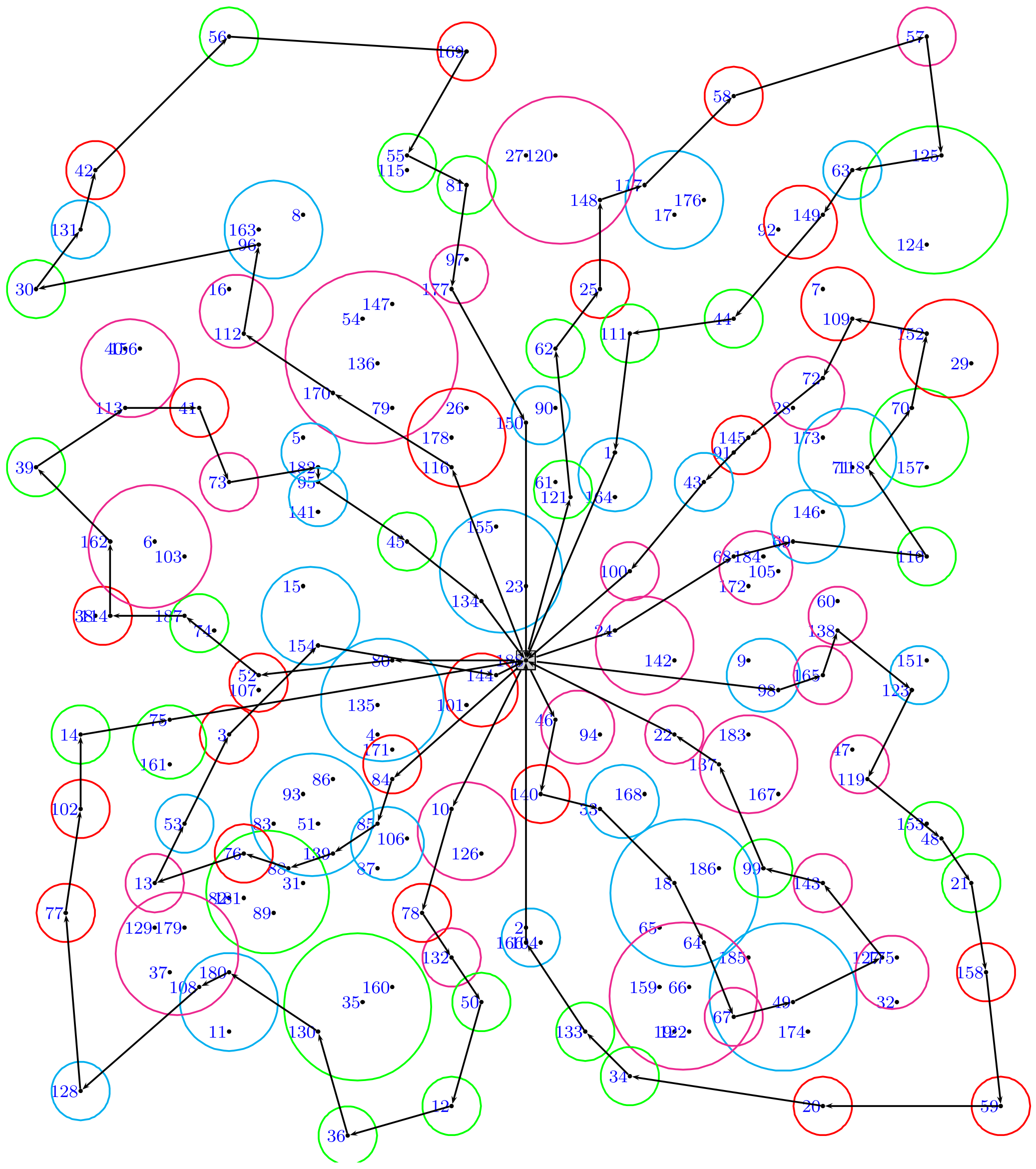

Figure 6.1: Best known solution for instance M-n200-k16-C100

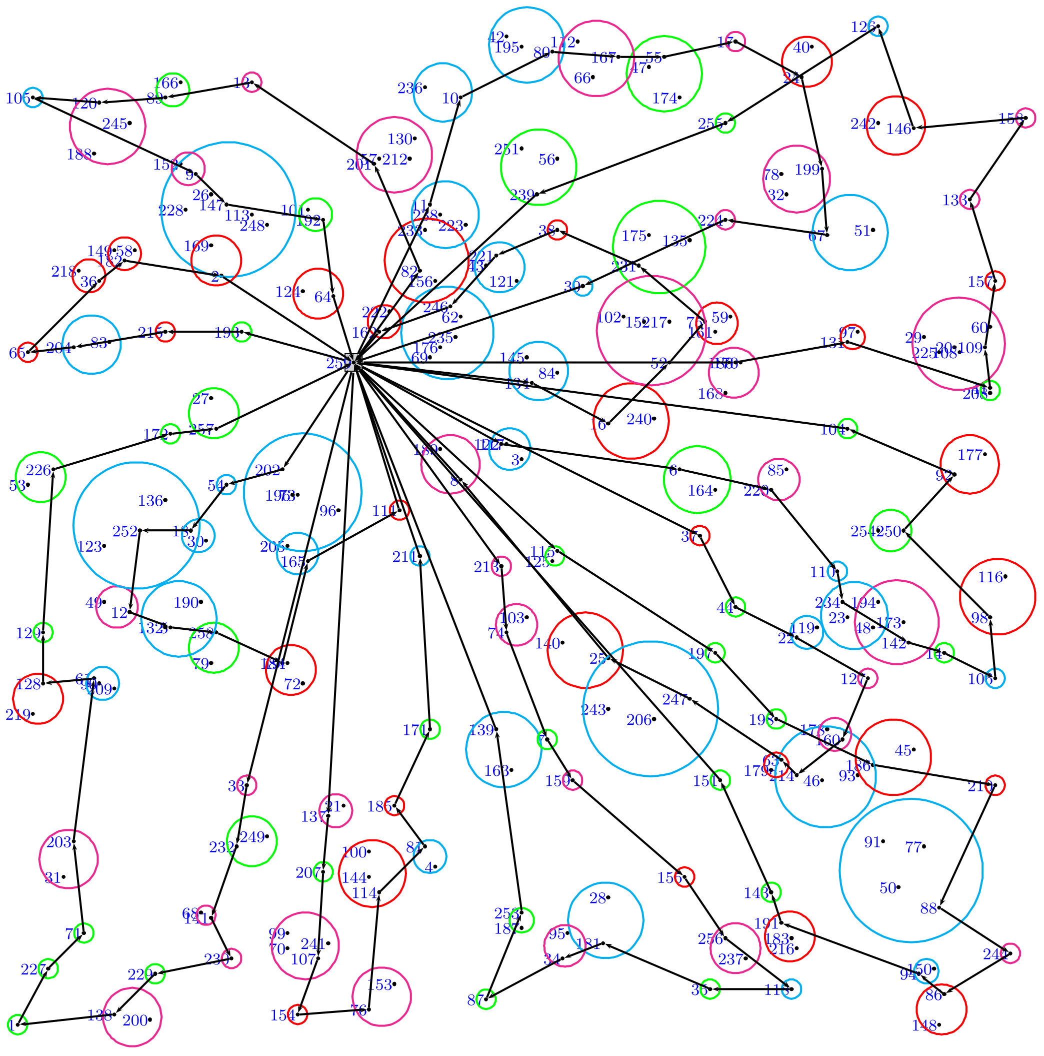

Figure 6.2: Best known solution for instance G-n262-k25-C131

6.2 Heuristic performance on CmRSP

The tests instances used for the CmRSP are generated in [4] using three existing Traveling Salesman Problem instances *eil51*, *eil76* and *eil101* [25]. An additional smaller instance set is constructed by choosing the first 26 nodes of *eil51*. The instances can be divided into two sets: A and B. Given the Euclidian costs e_{ij} in the original TSP, set A has equal ring and assignment costs; $c_{ij} = a_{ij} = e_{ij}$. In set B, assignments are cheaper than ring connections; $c_{ij} = 7e_{ij}$, $a_{ij} = 3e_{ij}$.

For each instance in A and B, there are two additional variables. First, the capacity limit Q is set tight in order to generate exactly $m \in \{3, 4, 5\}$ routes. Second, a variable fraction γ of the nodes are made regular nodes (the rest are Steiner nodes); $\gamma \in \{0.25, 0.5, 0.75, 1\}$. Note that the number of regular nodes is equal to the number of clusters in the GVRP transformation. In total, there are 9 versions of instance *eil26* and 12 versions each of the other instances.

The GVRP algorithm results on transformed CmRSP instances are reported in Tables 6.9 to 6.14. Each instance is solved three times. Column 'Instance' reports the instance name. Columns 'n', 'R' and 'K' report the number of customers, customers removed in preprocessing, and clusters, respectively. In the transformed GVRP instances, the number of routes is given in column 'm'. The average computing time is under '$\bar{t}$'. Columns 'Ub' and 'Avg' report the best and average upper bounds found by the algorithm, respectively. Finally, column 'Ub*' reports the best known solution values, as reported in [29], for each instance.

Tables 6.9-6.10 contain the results for the case where only the cheapest assignment is allowed. The case with two cheapest assignments is presented in Tables 6.11-6.12 and the unrestricted case in Tables 6.13-6.14. If our algorithm finds the best known solution, that value in column 'Ub' is bolded, as well as the average in 'Avg' if the best known value is reached on all the solutions.

Table 6.9: Heuristic results on CmRSP-A with only 1 assignment option

Instance	n	R	K	m	$\bar{t}$	Ub	Avg	Ub*
eil26.tsp.3.12.5.A	14	0	12	3	1.25	**242**	**242.0**	242
eil26.tsp.4.12.4.A	14	0	12	3	1.11	(251)	251.0	261
eil26.tsp.5.12.3.A	14	0	12	4	1.03	(279)	279.0	292
eil26.tsp.3.18.7.A	21	0	18	3	2.34	**301**	**301.0**	301
eil26.tsp.4.18.5.A	21	0	18	4	2.24	**339**	**339.0**	339
eil26.tsp.5.18.4.A	21	0	18	5	2.15	**375**	**375.0**	375
eil26.tsp.3.25.10.A	30	0	25	3	4.43	**325**	**325.0**	325
eil26.tsp.4.25.7.A	30	0	25	4	4.71	**362**	**362.0**	362
eil26.tsp.5.25.6.A	30	0	25	5	4.76	**382**	**382.0**	382
eil51.tsp.3.12.5.A	17	1	12	3	1.26	**242**	**242.0**	242
eil51.tsp.4.12.4.A	17	1	12	3	1.18	(251)	251.0	261
eil51.tsp.5.12.3.A	17	1	12	4	1.09	(279)	279.0	286
eil51.tsp.3.25.10.A	38	0	25	3	5.25	**322**	**322.0**	322
eil51.tsp.4.25.7.A	38	0	25	4	5.48	**360**	**360.0**	360
eil51.tsp.5.25.6.A	38	0	25	5	5.40	**379**	**379.0**	379
eil51.tsp.3.37.14.A	56	1	37	3	16.40	(373)	**373.0**	373
eil51.tsp.4.37.11.A	56	1	37	4	15.82	(405)	**405.0**	405
eil51.tsp.5.37.9.A	56	1	37	5	15.37	(432)	**432.0**	432
eil51.tsp.3.50.19.A	78	1	50	3	47.66	(458)	**458.0**	458
eil51.tsp.4.50.14.A	78	1	50	4	46.97	(490)	**490.0**	490
eil51.tsp.5.50.12.A	78	1	50	5	44.51	(520)	**520.0**	520
eil76.tsp.3.18.7.A	32	0	18	3	2.99	**330**	**330.0**	330
eil76.tsp.4.18.5.A	32	0	18	4	3.40	**385**	**385.0**	385
eil76.tsp.5.18.4.A	32	0	18	5	3.19	**448**	**448.0**	448
eil76.tsp.3.37.14.A	66	4	37	3	23.65	**402**	**402.0**	402
eil76.tsp.4.37.11.A	66	4	37	4	22.44	**457**	457.3	457
eil76.tsp.5.37.9.A	66	4	37	5	24.05	**479**	**479.0**	479
eil76.tsp.3.56.21.A	108	3	56	3	103.89	**471**	471.3	471
eil76.tsp.4.56.16.A	108	3	56	4	93.21	**519**	**519.0**	519
eil76.tsp.5.56.13.A	108	3	56	5	91.75	**545**	545.7	545
eil76.tsp.3.75.28.A	150	0	75	3	353.95	567	568.0	564
eil76.tsp.4.75.21.A	150	0	75	4	299.63	604	605.7	602
eil76.tsp.5.75.17.A	150	0	75	5	272.45	**640**	641.7	640
eil101.tsp.3.25.10.A	49	2	25	3	6.18	**363**	**363.0**	363
eil101.tsp.4.25.7.A	49	2	25	4	6.28	418	418.0	415
eil101.tsp.5.25.6.A	49	2	25	5	5.85	**448**	**448.0**	448
eil101.tsp.3.50.19.A	95	1	50	3	50.23	**500**	**500.0**	500
eil101.tsp.4.50.14.A	95	1	50	4	45.75	529	529.7	528
eil101.tsp.5.50.12.A	95	1	50	5	44.90	**567**	**567.0**	567
eil101.tsp.3.75.28.A	152	1	75	3	278.52	596	597.7	595
eil101.tsp.4.75.21.A	152	1	75	4	254.44	625	625.3	623
eil101.tsp.5.75.17.A	152	1	75	5	219.66	**657**	**657.0**	657
eil101.tsp.3.100.38.A	230	0	100	3	1110.12	656	658.3	646
eil101.tsp.4.100.28.A	230	0	100	4	976.67	681	682.0	679
eil101.tsp.5.100.23.A	230	0	100	5	837.87	705	706.7	700

Table 6.10: Heuristic results on CmRSP-B with only 1 assignment option

Instance	n	R	K	m	$\bar{t}$	Ub	Avg	Ub*
eil26.tsp.3.12.5.B	14	0	12	3	1.07	**1684**	**1684.0**	1684
eil26.tsp.4.12.4.B	14	0	12	3	1.12	(1757)	1757.0	1827
eil26.tsp.5.12.3.B	14	0	12	4	1.01	(1950)	1950.0	2041
eil26.tsp.3.18.7.B	21	0	18	3	2.25	**2104**	**2104.0**	2104
eil26.tsp.4.18.5.B	21	0	18	4	2.31	**2370**	**2370.0**	2370
eil26.tsp.5.18.4.B	21	0	18	5	2.12	**2615**	**2615.0**	2615
eil26.tsp.3.25.10.B	30	0	25	3	4.42	**2251**	**2251.0**	2251
eil26.tsp.4.25.7.B	30	0	25	4	4.83	**2510**	**2510.0**	2510
eil26.tsp.5.25.6.B	30	0	25	5	4.60	**2674**	**2674.0**	2674
eil51.tsp.3.12.5.B	18	0	12	3	1.25	**1681**	**1681.0**	1681
eil51.tsp.4.12.4.B	18	0	12	3	1.22	(1751)	1751.0	1821
eil51.tsp.5.12.3.B	18	0	12	4	1.08	(1923)	1923.0	1972
eil51.tsp.3.25.10.B	38	0	25	3	5.65	**2176**	**2176.0**	2176
eil51.tsp.4.25.7.B	38	0	25	4	5.75	**2470**	**2470.0**	2470
eil51.tsp.5.25.6.B	38	0	25	5	5.34	**2579**	**2579.0**	2579
eil51.tsp.3.37.14.B	57	0	37	3	18.04	**2490**	**2490.0**	2490
eil51.tsp.4.37.11.B	57	0	37	4	17.99	**2721**	**2721.0**	2721
eil51.tsp.5.37.9.B	57	0	37	5	15.65	**2908**	**2908.0**	2908
eil51.tsp.3.50.19.B	79	0	50	3	58.74	**3015**	**3015.0**	3015
eil51.tsp.4.50.14.B	79	0	50	4	53.21	**3260**	**3260.0**	3260
eil51.tsp.5.50.12.B	79	0	50	5	50.89	(3401)	3402.0	3404
eil76.tsp.3.18.7.B	32	0	18	3	2.99	**2253**	**2253.0**	2253
eil76.tsp.4.18.5.B	32	0	18	4	3.29	2625	2625.0	2620
eil76.tsp.5.18.4.B	32	0	18	5	3.05	**3059**	**3059.0**	3059
eil76.tsp.3.37.14.B	67	3	37	3	26.37	**2720**	**2720.0**	2720
eil76.tsp.4.37.11.B	67	3	37	4	24.39	**3100**	**3100.0**	3100
eil76.tsp.5.37.9.B	67	3	37	5	25.90	3291	3291.0	3284
eil76.tsp.3.56.21.B	109	2	56	3	136.84	3064	3064.0	3044
eil76.tsp.4.56.16.B	109	2	56	4	112.73	3440	3444.3	3415
eil76.tsp.5.56.13.B	109	2	56	5	110.29	3649	3649.0	3631
eil76.tsp.3.75.28.B	150	0	75	3	447.75	3669	3670.0	3652
eil76.tsp.4.75.21.B	150	0	75	4	398.25	4003	4003.0	3964
eil76.tsp.5.75.17.B	150	0	75	5	333.05	**4217**	**4217.0**	4217
eil101.tsp.3.25.10.B	50	1	25	3	6.54	2445	2445.0	2434
eil101.tsp.4.25.7.B	50	1	25	4	6.59	2837	2837.0	2782
eil101.tsp.5.25.6.B	50	1	25	5	6.30	3019	3019.0	3009
eil101.tsp.3.50.19.B	95	1	50	3	56.70	3328	3328.0	3322
eil101.tsp.4.50.14.B	95	1	50	4	53.20	3549	3549.0	3533
eil101.tsp.5.50.12.B	95	1	50	5	48.52	3868	3868.3	3834
eil101.tsp.3.75.28.B	152	1	75	3	340.77	3894	3898.0	3887
eil101.tsp.4.75.21.B	152	1	75	4	299.97	4087	4087.7	4082
eil101.tsp.5.75.17.B	152	1	75	5	238.70	4365	4371.7	4358
eil101.tsp.3.100.38.B	230	0	100	3	1515.00	(4156)	4164.0	4109
eil101.tsp.4.100.28.B	230	0	100	4	1270.27	4389	4397.0	4355
eil101.tsp.5.100.23.B	230	0	100	5	1044.47	4608	4608.7	4565

Table 6.11: Heuristic results on CmRSP-A with 2 assignment options

Instance	n	R	K	m	$\bar{t}$	Ub	Avg	Ub*
eil26.tsp.3.12.5.A	14	0	12	3	1.38	**242**	**242.0**	242
eil26.tsp.4.12.4.A	14	0	12	3	1.16	(251)	251.0	261
eil26.tsp.5.12.3.A	14	0	12	4	1.10	(279)	279.0	292
eil26.tsp.3.18.7.A	21	0	18	3	2.23	**301**	**301.0**	301
eil26.tsp.4.18.5.A	21	0	18	4	2.28	**339**	**339.0**	339
eil26.tsp.5.18.4.A	21	0	18	5	2.18	**375**	**375.0**	375
eil26.tsp.3.25.10.A	30	0	25	3	4.43	**325**	**325.0**	325
eil26.tsp.4.25.7.A	30	0	25	4	4.80	**362**	**362.0**	362
eil26.tsp.5.25.6.A	30	0	25	5	4.71	**382**	**382.0**	382
eil51.tsp.3.12.5.A	17	1	12	3	1.24	**242**	**242.0**	242
eil51.tsp.4.12.4.A	17	1	12	3	1.18	(251)	251.0	261
eil51.tsp.5.12.3.A	17	1	12	4	1.09	(279)	279.0	286
eil51.tsp.3.25.10.A	38	0	25	3	5.28	**322**	**322.0**	322
eil51.tsp.4.25.7.A	38	0	25	4	5.39	**360**	**360.0**	360
eil51.tsp.5.25.6.A	38	0	25	5	5.25	**379**	**379.0**	379
eil51.tsp.3.37.14.A	56	1	37	3	16.82	(373)	**373.0**	373
eil51.tsp.4.37.11.A	56	1	37	4	16.06	(405)	**405.0**	405
eil51.tsp.5.37.9.A	56	1	37	5	15.64	(432)	**432.0**	432
eil51.tsp.3.50.19.A	78	1	50	3	44.98	(458)	458.3	458
eil51.tsp.4.50.14.A	78	1	50	4	44.64	(490)	**490.0**	490
eil51.tsp.5.50.12.A	78	1	50	5	43.34	(520)	**520.0**	520
eil76.tsp.3.18.7.A	33	0	18	3	3.02	**330**	**330.0**	330
eil76.tsp.4.18.5.A	33	0	18	4	3.33	**385**	**385.0**	385
eil76.tsp.5.18.4.A	33	0	18	5	3.24	**448**	**448.0**	448
eil76.tsp.3.37.14.A	70	4	37	3	23.67	**402**	**402.0**	402
eil76.tsp.4.37.11.A	70	4	37	4	23.98	**457**	**457.0**	457
eil76.tsp.5.37.9.A	70	4	37	5	24.90	**479**	**479.0**	479
eil76.tsp.3.56.21.A	117	3	56	3	115.87	**471**	**471.0**	471
eil76.tsp.4.56.16.A	117	3	56	4	103.82	**519**	**519.0**	519
eil76.tsp.5.56.13.A	117	3	56	5	100.88	**545**	**545.0**	545
eil76.tsp.3.75.28.A	162	0	75	3	375.06	567	568.7	564
eil76.tsp.4.75.21.A	162	0	75	4	339.94	604	605.0	602
eil76.tsp.5.75.17.A	162	0	75	5	269.61	**640**	**640.0**	640
eil101.tsp.3.25.10.A	54	1	25	3	6.68	**363**	**363.0**	363
eil101.tsp.4.25.7.A	54	1	25	4	6.64	**415**	**415.0**	415
eil101.tsp.5.25.6.A	54	1	25	5	6.16	**448**	**448.0**	448
eil101.tsp.3.50.19.A	102	1	50	3	59.57	**500**	**500.0**	500
eil101.tsp.4.50.14.A	102	1	50	4	51.99	**528**	**528.0**	528
eil101.tsp.5.50.12.A	102	1	50	5	46.57	**567**	**567.0**	567
eil101.tsp.3.75.28.A	163	1	75	3	337.41	596	598.7	595
eil101.tsp.4.75.21.A	163	1	75	4	302.51	624	625.3	623
eil101.tsp.5.75.17.A	163	1	75	5	250.51	**657**	657.7	657
eil101.tsp.3.100.38.A	254	0	100	3	1327.32	656	657.3	646
eil101.tsp.4.100.28.A	254	0	100	4	1170.23	681	682.7	679
eil101.tsp.5.100.23.A	254	0	100	5	1013.73	705	706.0	700

Table 6.12: Heuristic results on CmRSP-B with 2 assignment options

Instance	n	R	K	m	$\bar{t}$	Ub	Avg	Ub*
eil26.tsp.3.12.5.B	14	0	12	3	1.08	**1684**	**1684.0**	1684
eil26.tsp.4.12.4.B	14	0	12	3	1.10	(1757)	1757.0	1827
eil26.tsp.5.12.3.B	14	0	12	4	1.01	(1950)	1950.0	2041
eil26.tsp.3.18.7.B	21	0	18	3	2.29	**2104**	**2104.0**	2104
eil26.tsp.4.18.5.B	21	0	18	4	2.21	**2370**	**2370.0**	2370
eil26.tsp.5.18.4.B	21	0	18	5	2.11	**2615**	**2615.0**	2615
eil26.tsp.3.25.10.B	30	0	25	3	4.53	**2251**	**2251.0**	2251
eil26.tsp.4.25.7.B	30	0	25	4	4.86	**2510**	**2510.0**	2510
eil26.tsp.5.25.6.B	30	0	25	5	4.72	**2674**	**2674.0**	2674
eil51.tsp.3.12.5.B	18	0	12	3	1.22	**1681**	**1681.0**	1681
eil51.tsp.4.12.4.B	18	0	12	3	1.24	(1751)	1751.0	1821
eil51.tsp.5.12.3.B	18	0	12	4	1.08	(1923)	1923.0	1972
eil51.tsp.3.25.10.B	38	0	25	3	5.53	**2176**	**2176.0**	2176
eil51.tsp.4.25.7.B	38	0	25	4	6.03	**2470**	**2470.0**	2470
eil51.tsp.5.25.6.B	38	0	25	5	5.76	**2579**	**2579.0**	2579
eil51.tsp.3.37.14.B	57	0	37	3	18.35	**2490**	**2490.0**	2490
eil51.tsp.4.37.11.B	57	0	37	4	17.59	**2721**	**2721.0**	2721
eil51.tsp.5.37.9.B	57	0	37	5	15.98	**2908**	**2908.0**	2908
eil51.tsp.3.50.19.B	79	0	50	3	56.68	**3015**	**3015.0**	3015
eil51.tsp.4.50.14.B	79	0	50	4	52.31	**3260**	**3260.0**	3260
eil51.tsp.5.50.12.B	79	0	50	5	50.22	(3401)	3411.3	3404
eil76.tsp.3.18.7.B	33	0	18	3	3.16	**2253**	**2253.0**	2253
eil76.tsp.4.18.5.B	33	0	18	4	3.47	**2620**	**2620.0**	2620
eil76.tsp.5.18.4.B	33	0	18	5	3.09	**3059**	**3059.0**	3059
eil76.tsp.3.37.14.B	71	3	37	3	27.86	2721	2721.0	2720
eil76.tsp.4.37.11.B	71	3	37	4	27.11	**3100**	**3100.0**	3100
eil76.tsp.5.37.9.B	71	3	37	5	27.54	**3284**	**3284.0**	3284
eil76.tsp.3.56.21.B	118	2	56	3	151.96	**3044**	3046.7	3044
eil76.tsp.4.56.16.B	118	2	56	4	124.18	**3415**	3422.3	3415
eil76.tsp.5.56.13.B	118	2	56	5	122.40	**3631**	**3631.0**	3631
eil76.tsp.3.75.28.B	162	0	75	3	532.86	3654	3654.3	3652
eil76.tsp.4.75.21.B	162	0	75	4	448.80	4000	4001.0	3964
eil76.tsp.5.75.17.B	162	0	75	5	378.51	**4217**	4221.7	4217
eil101.tsp.3.25.10.B	54	1	25	3	7.30	**2434**	**2434.0**	2434
eil101.tsp.4.25.7.B	54	1	25	4	6.95	2785	2785.0	2782
eil101.tsp.5.25.6.B	54	1	25	5	6.65	3012	3012.0	3009
eil101.tsp.3.50.19.B	102	1	50	3	63.97	3323	3324.7	3322
eil101.tsp.4.50.14.B	102	1	50	4	57.67	3536	3536.0	3533
eil101.tsp.5.50.12.B	102	1	50	5	54.39	3846	3846.0	3834
eil101.tsp.3.75.28.B	163	1	75	3	387.41	**3887**	3903.3	3887
eil101.tsp.4.75.21.B	163	1	75	4	343.51	**4082**	4091.0	4082
eil101.tsp.5.75.17.B	163	1	75	5	294.99	4360	4371.0	4358
eil101.tsp.3.100.38.B	254	0	100	3	1856.14	4141	4160.0	4109
eil101.tsp.4.100.28.B	254	0	100	4	1506.09	4379	4383.3	4355
eil101.tsp.5.100.23.B	254	0	100	5	1327.77	4587	4600.3	4565

Table 6.13: Heuristic results on CmRSP-A with unlimited assignment options

Instance	n	R	K	m	$\bar{t}$	Ub	Avg	Ub*
eil26.tsp.3.12.5.A	14	0	12	3	2.03	**242**	**242.0**	242
eil26.tsp.4.12.4.A	14	0	12	3	1.11	(251)	251.0	261
eil26.tsp.5.12.3.A	14	0	12	4	1.02	(279)	279.0	292
eil26.tsp.3.18.7.A	21	0	18	3	2.28	**301**	**301.0**	301
eil26.tsp.4.18.5.A	21	0	18	4	2.27	**339**	**339.0**	339
eil26.tsp.5.18.4.A	21	0	18	5	2.12	**375**	**375.0**	375
eil26.tsp.3.25.10.A	30	0	25	3	4.38	**325**	**325.0**	325
eil26.tsp.4.25.7.A	30	0	25	4	4.71	**362**	**362.0**	362
eil26.tsp.5.25.6.A	30	0	25	5	4.69	**382**	**382.0**	382
eil51.tsp.3.12.5.A	17	1	12	3	1.23	**242**	**242.0**	242
eil51.tsp.4.12.4.A	17	1	12	3	1.18	(251)	251.0	261
eil51.tsp.5.12.3.A	17	1	12	4	1.09	(279)	279.0	286
eil51.tsp.3.25.10.A	38	0	25	3	5.23	**322**	**322.0**	322
eil51.tsp.4.25.7.A	38	0	25	4	5.28	**360**	**360.0**	360
eil51.tsp.5.25.6.A	38	0	25	5	5.21	**379**	**379.0**	379
eil51.tsp.3.37.14.A	56	1	37	3	16.82	(373)	**373.0**	373
eil51.tsp.4.37.11.A	56	1	37	4	15.76	(405)	**405.0**	405
eil51.tsp.5.37.9.A	56	1	37	5	16.04	(432)	**432.0**	432
eil51.tsp.3.50.19.A	78	1	50	3	46.37	(459)	**459.0**	459
eil51.tsp.4.50.14.A	78	1	50	4	44.25	(490)	**490.0**	490
eil51.tsp.5.50.12.A	78	1	50	5	40.78	(520)	**520.0**	520
eil76.tsp.3.18.7.A	33	0	18	3	3.02	**330**	**330.0**	330
eil76.tsp.4.18.5.A	33	0	18	4	3.38	**385**	**385.0**	385
eil76.tsp.5.18.4.A	33	0	18	5	3.19	**448**	**448.0**	448
eil76.tsp.3.37.14.A	71	4	37	3	25.14	**402**	**402.0**	402
eil76.tsp.4.37.11.A	71	4	37	4	24.10	**457**	**457.0**	457
eil76.tsp.5.37.9.A	71	4	37	5	24.89	**479**	**479.0**	479
eil76.tsp.3.56.21.A	121	3	56	3	124.75	**471**	**471.0**	471
eil76.tsp.4.56.16.A	121	3	56	4	104.42	**519**	**519.0**	519
eil76.tsp.5.56.13.A	121	3	56	5	103.60	**545**	**545.0**	545
eil76.tsp.3.75.28.A	172	0	75	3	433.58	568	568.7	564
eil76.tsp.4.75.21.A	172	0	75	4	360.81	604	605.3	602
eil76.tsp.5.75.17.A	172	0	75	5	331.90	**640**	**640.0**	640
eil101.tsp.3.25.10.A	62	2	25	3	7.83	**363**	**363.0**	363
eil101.tsp.4.25.7.A	62	2	25	4	7.28	**415**	**415.0**	415
eil101.tsp.5.25.6.A	62	2	25	5	7.25	**448**	**448.0**	448
eil101.tsp.3.50.19.A	117	1	50	3	71.84	**500**	**500.0**	500
eil101.tsp.4.50.14.A	117	1	50	4	63.30	**528**	**528.0**	528
eil101.tsp.5.50.12.A	117	1	50	5	58.85	**567**	**567.0**	567
eil101.tsp.3.75.28.A	191	1	75	3	486.64	597	599.0	595
eil101.tsp.4.75.21.A	191	1	75	4	390.98	**623**	**623.0**	623
eil101.tsp.5.75.17.A	191	1	75	5	335.98	658	658.0	657
eil101.tsp.3.100.38.A	349	0	100	3	2680.85	(651)	652.3	646
eil101.tsp.4.100.28.A	349	0	100	4	2312.63	(681)	682.0	679
eil101.tsp.5.100.23.A	349	0	100	5	1943.48	(705)	707.0	700

Table 6.14: Heuristic results on CmRSP-B with unlimited assignment options

Instance	n	R	K	m	$\bar{t}$	Ub	Avg	Ub*
eil26.tsp.3.12.5.B	14	0	12	3	1.08	**1684**	**1684.0**	1684
eil26.tsp.4.12.4.B	14	0	12	3	1.11	(1757)	1757.0	1827
eil26.tsp.5.12.3.B	14	0	12	4	1.03	(1950)	1950.0	2041
eil26.tsp.3.18.7.B	21	0	18	3	2.27	**2104**	**2104.0**	2104
eil26.tsp.4.18.5.B	21	0	18	4	2.31	**2370**	**2370.0**	2370
eil26.tsp.5.18.4.B	21	0	18	5	2.10	**2615**	**2615.0**	2615
eil26.tsp.3.25.10.B	30	0	25	3	4.59	**2251**	**2251.0**	2251
eil26.tsp.4.25.7.B	30	0	25	4	4.86	**2510**	**2510.0**	2510
eil26.tsp.5.25.6.B	30	0	25	5	4.71	**2674**	**2674.0**	2674
eil51.tsp.3.12.5.B	18	0	12	3	1.21	**1681**	**1681.0**	1681
eil51.tsp.4.12.4.B	18	0	12	3	1.25	(1751)	1751.0	1821
eil51.tsp.5.12.3.B	18	0	12	4	1.09	(1923)	1923.0	1972
eil51.tsp.3.25.10.B	38	0	25	3	5.46	**2176**	**2176.0**	2176
eil51.tsp.4.25.7.B	38	0	25	4	5.96	**2470**	**2470.0**	2470
eil51.tsp.5.25.6.B	38	0	25	5	5.53	**2579**	**2579.0**	2579
eil51.tsp.3.37.14.B	57	0	37	3	18.59	**2490**	**2490.0**	2490
eil51.tsp.4.37.11.B	57	0	37	4	17.99	**2721**	**2721.0**	2721
eil51.tsp.5.37.9.B	57	0	37	5	15.86	**2908**	**2908.0**	2908
eil51.tsp.3.50.19.B	79	0	50	3	55.56	**3015**	**3015.0**	3015
eil51.tsp.4.50.14.B	79	0	50	4	54.94	**3260**	**3260.0**	3260
eil51.tsp.5.50.12.B	79	0	50	5	51.20	(3401)	3401.0	3404
eil76.tsp.3.18.7.B	33	0	18	3	3.02	**2253**	**2253.0**	2253
eil76.tsp.4.18.5.B	33	0	18	4	3.43	**2620**	**2620.0**	2620
eil76.tsp.5.18.4.B	33	0	18	5	3.18	**3059**	**3059.0**	3059
eil76.tsp.3.37.14.B	72	3	37	3	27.04	**2720**	**2720.0**	2720
eil76.tsp.4.37.11.B	72	3	37	4	26.44	**3100**	**3100.0**	3100
eil76.tsp.5.37.9.B	72	3	37	5	27.35	**3284**	**3284.0**	3284
eil76.tsp.3.56.21.B	122	2	56	3	159.81	**3044**	**3044.0**	3044
eil76.tsp.4.56.16.B	122	2	56	4	132.91	**3415**	3422.7	3415
eil76.tsp.5.56.13.B	122	2	56	5	130.53	**3631**	**3631.0**	3631
eil76.tsp.3.75.28.B	172	0	75	3	612.36	3655	3657.7	3652
eil76.tsp.4.75.21.B	172	0	75	4	462.80	3970	3980.7	3964
eil76.tsp.5.75.17.B	172	0	75	5	417.26	**4217**	4218.0	4217
eil101.tsp.3.25.10.B	63	1	25	3	8.93	**2434**	**2434.0**	2434
eil101.tsp.4.25.7.B	63	1	25	4	7.92	**2782**	**2782.0**	2782
eil101.tsp.5.25.6.B	63	1	25	5	7.56	**3009**	**3009.0**	3009
eil101.tsp.3.50.19.B	117	1	50	3	78.28	**3322**	**3322.0**	3322
eil101.tsp.4.50.14.B	117	1	50	4	76.05	**3533**	3535.3	3533
eil101.tsp.5.50.12.B	117	1	50	5	63.44	**3834**	**3834.0**	3834
eil101.tsp.3.75.28.B	191	1	75	3	566.04	**3887**	3891.0	3887
eil101.tsp.4.75.21.B	191	1	75	4	481.30	**4082**	**4082.0**	4082
eil101.tsp.5.75.17.B	191	1	75	5	403.62	**4358**	4367.3	4358
eil101.tsp.3.100.38.B	349	0	100	3	4102.98	(4134)	4140.0	4109
eil101.tsp.4.100.28.B	349	0	100	4	3380.47	(4376)	4381.0	4355
eil101.tsp.5.100.23.B	349	0	100	5	2773.91	(4575)	4581.3	4565

The results on the six cases are mutually quite similar; the small instances are solved quickly to best known values, while solving the large instances can take up to an hour and yield suboptimal results. With the largest instances in the unrestricted case, there are 349 customers; a significant increase to the 100 nodes in the CmRSP. Allowing only two cheapest assignments, the customer amount is less by a hundred.

In all of the six cases, there are four instances based on *eil26* and *eil51* that yield better results than the current best known value. This is because our results can have a flexible amount of routes; the capacity limit in these instances is not tight, allowing us to use one route (or ring) less than the solutions in the literature. Concerning these instances, our best known values are in parenthesis. Similarly, parentheses are used every time our best solution is not a feasible CmRSP solution as explained in Chapter 4. This occurs systematically for six *eil51*-based instances in set A. However, the instances are small and the results are always equal to the upper bound that is known to be optimal; this implies that we happen to find solutions that are nearly identical to the correct one with small zero-cost alterations. There is only one instance (*eil51*.tsp.5.50.12.B) for which we are able to find strictly improving infeasible solutions. Overall, only a handful of our solutions are infeasible in the original CmRSP; we are able to compare the results to current literature.

To examine the effect of the restriction of assignment options, observe Table 6.15. In columns 'Best' and '$\bar{t}$', we report the success rate and average computing time for sets A and B and for each of the options. The success rate is the amount of instances that we were able to solve to best known values. From the total 45 instances, we include only those with comparable results in the comparison.

Table 6.15: Algorithm performance summary on the CmRSP with different constrictions on the amount of assignments

	1 option		2 options		Unrestricted	
Set	Best	$\bar{t}$	Best	$\bar{t}$	Best	$\bar{t}$
A	26/35	119.1	28/35	137.6	28/32	225.2
B	21/39	150.5	28/40	178.2	35/37	315.9

It seems clear that the solution quality increases as more assignment options are included. Apparently, not only the cheapest assignments are used in the solutions; especially in the large ones. In set B, the assignments are relatively cheaper than in set A. This means that good solutions are likely to use more assignments in set B. It is then not a surprise that restricting assignment options leads to clearly poorer solutions on set B. With the unrestricted case, we are able to find the best known solutions with adequate reliability, especially with set B. It must be noted that the three biggest instances are not included in the comparison in the unrestricted case due to infeasibility.

Comparison to current literature is performed in Table 6.16. with familiar columns 'Best' and '$\bar{t}$'. We use our results from the unrestricted case.

Table 6.16: CmRSP Results compared to literature

	Thesis		Baldacci et al.		Naji-Azimi et. al		Zhang et. al	
Set	Best	$\bar{t}$	Best	$\bar{t}$	Best	$\bar{t}$	Best	$\bar{t}$
A	28/32	225.2	36/45	2098	44/45	2.02	**45/45**	59.6
B	35/37	315.9	29/45	2953	**44/45**	2.29	**44/45**	85.5

The table includes the results from the exact method of Baldacci et al. and the heuristics of Naji-Azimi et. al and Zhang et. al. In a sense, it is not obvious that comparing heuristics with exact methods gives any useful insight; especially concerning computing times, as exact algorithms are typically terminated after a fixed time limit if no optimal solutions are found (7200 seconds in this case). Here, we can merely state that our algorithm is able to find better solutions to set B than the exact algorithm.

Compared to the heuristic algorithms, our algorithm is clearly inferior. They are able to find nearly all best known solutions with little time. This time, we do not have sufficient information on the processor types, so computation time comparisons are made in a rough manner. As our algorithm appears to be at least four times slower than the slower of the two heuristics and over a hundred times slower than the other, it should be safe to say that our algorithm is nowhere near the current level of research.

6.3 Heuristic performance on MCTP

The heuristic GVRP algorithm was tested on available MCTP instances. These instances are based on TSP instances $kroA100$, $kroB100$, $kroC100$ and $kroD100$. They were generated in [17] and later used in [14] and [18].

In the instances, the parameter $|Y| \in \{25, 50\}$ denotes the amount of customers belonging to set Y (customers that can be visited). The depot v_0 is included in this set and it is the only node that must be visited: $T = \{v_0\}$. The second parameter implies the amount of remaining nodes that must be assigned. Finally, there is a customer limit $p \in \{4, 5, 6, 8\}$ for all routes. The names of the CTP instances are then constructed as follows: 'TSP instance name'-'$|Y|$'-'$|W|$'-'p'. For example, the first instance 'kroA100-25-75-4' is an instance derived from the TSP instance 'kroA100' with 25 potential customers to visit, 75 customers to cover and a limit of 4 customers per route.

Each customer can be covered by a node within a radius r. Here r is set in such a way that there are at least two options for each node. Otherwise, there would be no other choice than to visit the only possible assignment node; then that node would belong to T and the covered node could be eliminated altogether.

We do not consider cutting out any assignments in the transformation as with the CmRSP. This is because the assignments have no cost; we cannot assume that long-distance assignments would be made less frequently. Thus, the clusters in the transformed GVRP instances contain as many nodes as there are visitable nodes within radius r of the corresponding covered node. In many cases, there are clusters with a size of 10 customers or more.

Each instance is solved three times. The computational results can be found in table 6.17. In addition to our own results, we report the upper bounds and computing times by the heuristic methods of [14] and [18]. As before, 'Instance', 'n', 'K', 'Q' and 'm' stand for instance name, number of customers, number of clusters, capacity and number of routes. For each method, '$\bar{t}$' and 'Ub' report the (average) computing time and best upper bound.

The algorithm is able to find 23 of the 32 best known solutions. These solutions are also optimal as they are all solved to optimality in [14]. The GVRP problems are very large compared to the original instances. For example, the solution to instance kroB100-25-75-6 contains one route with at most 6 nodes visited; in the transformation, there are 75 clusters and 459 customers.

Table 6.17: Heuristic algorithm performance and comparison on 32 MCTP instances

					Thesis			Hà et al.		Kammoun et. al	
Instance	n	K	Q	m	$\bar{t}$	Ub	Avg	$\bar{t}$	Ub	$\bar{t}$	Ub
kroA100-25-75-4	416	75	4	2	420.5	**8479**	**8479.0**	0.16	**8479**	0.016	**8479**
kroA100-25-75-5	416	75	5	2	469.5	**8479**	**8479.0**	0.17	**8479**	0.016	**8479**
kroA100-25-75-6	416	75	6	2	538.4	**8479**	8479.7	0.16	**8479**	0.013	**8479**
kroA100-25-75-8	416	75	8	1	595.9	**7985**	7985.7	0.16	**7985**	0.014	**7985**
kroA100-50-50-4	301	50	4	3	174.3	**10271**	**10271.0**	0.80	**10271**	0.022	**10271**
kroA100-50-50-5	301	50	5	2	196.5	**9220**	**9220.0**	0.78	**9220**	0.017	**9220**
kroA100-50-50-6	301	50	6	2	208.7	**9130**	**9130**	0.81	**9130**	0.023	**9130**
kroA100-50-50-8	301	50	8	2	234.6	**9130**	**9130**	0.81	**9130**	0.018	**9130**
kroB100-25-75-4	459	75	4	2	531.3	**7146**	7179.3	0.22	**7146**	0.004	**7146**
kroB100-25-75-5	459	75	5	2	598.5	**6901**	6917.7	0.18	**6901**	0.005	**6901**
kroB100-25-75-6	459	75	6	1	610.4	**6450**	6483.3	0.23	**6450**	0.004	**6450**
kroB100-25-75-8	459	75	8	1	582.5	**6450**	**6450.0**	0.20	**6450**	0.004	**6450**
kroB100-50-50-4	385	50	4	3	312.5	10524	11207.0	0.62	**10107**	0.012	**10107**
kroB100-50-50-5	385	50	5	2	336.0	9732	9852.3	0.64	**9723**	0.009	**9723**
kroB100-50-50-6	385	50	6	2	361.0	**9382**	**9382.0**	0.58	**9382**	0.016	**9382**
kroB100-50-50-8	385	50	8	2	417.4	8552	8977.0	0.58	**8348**	0.016	**8348**
kroC100-25-75-4	617	75	4	1	964.4	**6161**	6499.3	0.16	**6161**	0.004	**6161**
kroC100-25-75-5	617	75	5	1	1067.8	**6161**	**6161.0**	0.16	**6161**	0.004	**6161**
kroC100-25-75-6	617	75	6	1	1065.6	**6161**	**6161.0**	0.15	**6161**	0.004	**6161**
kroC100-25-75-8	617	75	8	1	903.1	**6161**	**6161.0**	0.17	**6161**	0.004	**6161**
kroC100-50-50-4	287	50	4	3	208.9	**11372**	**11372.0**	0.64	**11372**	0.028	**11372**
kroC100-50-50-5	287	50	5	2	224.9	**9900**	**9900.0**	0.67	**9900**	0.013	**9900**
kroC100-50-50-6	287	50	6	2	246.9	**9895**	**9895.0**	0.67	**9895**	0.017	**9895**
kroC100-50-50-8	287	50	8	2	275.1	8712	9008.3	0.65	**8699**	0.007	**8699**
kroD100-25-75-4	469	75	4	2	532.0	**7671**	**7671.0**	0.16	**7671**	0.020	**7671**
kroD100-25-75-5	469	75	5	2	599.0	7666	7666.0	0.16	**7465**	0.022	**7465**
kroD100-25-75-6	469	75	6	1	674.1	**6651**	6670.3	0.15	**6651**	0.015	**6651**
kroD100-25-75-8	469	75	8	1	634.0	**6651**	**6651.0**	0.16	**6651**	0.014	**6651**
kroD100-50-50-4	276	50	4	3	168.8	12170	12634.0	0.93	**11606**	0.021	**11606**
kroD100-50-50-5	276	50	5	3	191.3	11143	11206.3	0.85	**10770**	0.263	**10770**
kroD100-50-50-6	276	50	6	2	207.0	10820	10896.0	0.82	10680	0.026	**10525**
kroD100-50-50-8	276	50	8	2	241.9	9790	9910.7	0.93	**9361**	0.028	**9361**

The heuristic of Hà et al. can find all but one optimal solutions while that of Kammoun et al. finds an optimal solution to all instances. Generally, the computing time per instance is less than one second for Hà et al. and less than a tenth of a second for Kammoun et al. In comparison, our heuristic algorithm could not find any solutions in under two minutes (120s), and most instances took hundreds of seconds to solve. In fact, even the exact branch-and-cut method of Hà et al. was faster than our heuristic one on every instance.

The solutions contain few routes. In all cases, $m \in \{1, 2, 3\}$. This is one factor explaining the poor performance of the heuristic; with $m = 1$, both the cyclic exchange and split algorithms are practically useless. To improve solution quality and reduce computing time, some ad-hoc local search manoeuvres would be necessary.

Our heuristic solves the instances with 50 visited and 50 covered nodes faster than those with 25 visited and 75 covered customers. This is probably due to 50-50 instances containing less clusters. It is an opposite trend to the other heuristics which perform faster on the 25-75 instances. However, we are able to find fewer optimal solutions for the 50-50 instances.

Chapter 7

Conclusions

This thesis concentrates on two main points: building a competitive algorithm for the GVRP and utilizing this algorithm to solve CmRSP and MCTP instances. Both goals can be claimed to have been met.

The revised GVRP heuristic algorithm with cyclic exchange, shortest-path and preprocessing methods is very competitive. In terms of solution quality, it outperforms all methods in current literature. It can reliably find optimal solutions to small and medium instances with computing times that are also competitive. The algorithm is able to find best known solutions to all large instances with two new unique best solutions discovered. The computational load of the algorithm does increase faster with problem size than its rivals. However, even with the largest instances of 262 customers, the algorithm cannot be considered actually slow.

We have shown that the Capacitated m-Ring-Star Problem and the Multivehicle Covering Tour Problem can be transformed into a GVRP and solved with the same algorithm. The algorithm is able to find best known solutions to most small CmRSP instances, but the quality for bigger ones is poor concerning both the upper bounds and computing time. This is, in part, due to the GVRP problem size increasing rapidly with respect to the original CmRSP. In addition, the cost structure of the GVRP becomes peculiar in the transformation process; the triangle equality generally does not hold, which is a setback to most heuristic moves and the Split algorithm. Keeping in mind the competitiveness of the algorithm in the GVRP, the algorithm does not seem like a sensible way to solve CmRSP instances practically. The CmRSP instances considered in this thesis only contained up to one hundred nodes, and the computing times could reach one hour. In the related literature, instances with over 400 nodes are considered. This algorithm would not be capable of solving these instances; at least, not without a major renovation of the GVRP metaheuristic framework and search moves to better utilize the characteristics of the CmRSP.

Utilizing the CmRSP transformation, the MCTP instances could be turned into GVRP instances in a very straight-forward fashion. The algorithm could successfully solve these instances as well, but the problems encountered with the CmRSP were even stronger in this context. Even simple instances resulted in excessively large transformations, and the zero-cost assignments were also a drawback. Typical solutions to these MCTP instances can only include one route, which makes the cyclic exchange search mostly redundant. The exact algorithms used in current literature can solve the MCTP instances faster than the heuristic. This implies that our heuristic algorithm is not a sensible method for solving the MCTP, but the transformation itself can be useful if solved with different algorithms.

In addition to these two main goals, we presented the Distance-Constrained Capacitated m-Ring Star Problem. The problem was formulated in general form, allowing for general demands to both regular and Steiner nodes. The MCTP was shown to be a special case of this problem. The DCmRSP would be an interesting subject for further study; a constraint on the length of the rings would encourage the utilization of star assignments. This problem might be relevant in the original context of the CmRSP in telecommunications, or in the vehicle routing interpretation.

Bibliography

[1] Afsar, H. M., Prins, C., and Santos, A. C. Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. *International Transactions in Operational Research 21*, 1 (2014), 153–175.

[2] Ahuja, R. K., Orlin, J. B., and Sharma, D. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming 91*, 1 (2001), 71 – 97.

[3] Baldacci, R., Bartolini, E., and Laporte, G. Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society 61* (2010), 1072–1077.

[4] Baldacci, R., Dell'Amico, M., and Salazar-González, J. The capacitated m-ring-star problem. *Operations Research 55*, 6 (2007), 1147 – 1162.

[5] Bektaş, T., Erdoğan, G., and Røpke, S. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science 45*, 3 (2011), 299–316.

[6] Bertsimas, D., and Tsitsiklis, J. *Introduction to linear optimization.* Athena Scientific, 1997.

[7] Burke, E. K., and Kendall, G., Eds. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, 2 ed. Springer US, 2014.

[8] Current, J. R., and Schilling, D. A. The covering salesman problem. *Transportation Science 23*, 3 (1989), 208–213.

[9] Dantzig, G., and Ramser, J. The truck dispatching problem. *Management Science 6* (1959), 80–91.

[10] Fischetti, M., González, J. J. S., and Toth, P. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research 45*, 3 (1997), 378–394.

[11] Gendreau, M., Laporte, G., and Semet, F. The covering tour problem. *Operations Research 45*, 4 (1997), 568–576.

[12] Gendreau, M., and Potvin, J.-Y., Eds. *Handbook of Metaheuristics*, 2 ed. Springer US, 2010.

[13] Ghiani, G., and Improta, G. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research 122*, 1 (2000), 11 – 17.

[14] Hà, M. H., Bostel, N., Langevin, A., and Rousseau, L.-M. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *European Journal of Operational Research 226*, 2 (2013), 211 – 220.

[15] Hà, M. H., Bostel, N., Langevin, A., and Rousseau, L.-M. An exact algorithm and a metaheuristic for the generalized vehicle routing problem with flexible fleet size. *Computers & Operations Research 43* (2014), 9 – 19.

[16] Hachicha, M., Hodgson, M. J., Laporte, G., and Semet, F. Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research 27*, 1 (2000), 29 – 42.

[17] Jozefowiez, N. A column generation approach for the multi-vehicle covering tour problem. In *Proceedings of the 12th ROADEF Conference* (2011).

[18] Kammoun, M., Derbel, H., Ratli, M., and Jarboui, B. A variable neighborhood search for solving the multi-vehicle covering tour problem. *Electronic Notes in Discrete Mathematics 47* (2015), 285 – 292. The 3rd International Conference on Variable Neighborhood Search (VNS'14).

[19] Mattila, M. The distance-constrained generalized vehicle routing problem. `http://sal.aalto.fi/publications/pdf-files/tmat16_public.pdf`, 2016. [Online; accessed February 19th, 2018].

[20] Moccia, L., Cordeau, J.-F., and Laporte, G. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society 63*, 2 (2012), 232–244.

[21] Naji-Azimi, Z., Salari, M., and Toth, P. A heuristic procedure for the capacitated m-ring-star problem. *European Journal of Operational Research 207* (2010), 1227 – 1234.

[22] Naji-Azimi, Z., Salari, M., and Toth, P. An integer linear programming based heuristic for the capacitated m-ring-star problem. *European Journal of Operational Research 217* (2012), 17 – 25.

[23] Pham, T. A., Hà, M. H., and Nguyen, X. H. Solving the multi-vehicle multi-covering tour problem. *Computers & Operations Research 88* (2017), 258 – 278.

[24] Pop, P., Pintea, C., Zelina, I., and Dumitrescu, D. Solving the generalized vehicle routing problem with an ACS-based algorithm. In *AIP Conference Proceedings* (2009), vol. 1117, pp. 157–162.

[25] Reinelt, G. Tsplib—a traveling salesman problem library. *ORSA Journal on Computing 3*, 4 (1991), 376–384.

[26] Salari, M., and Naji-Azimi, Z. An integer programming-based local search for the covering salesman problem. *Computers & Operations Research 39*, 11 (2012), 2594 – 2602.

[27] Thompson, P. M., and Orlin, J. B. The theory of cyclic transfers. Massachusetts Institute of Technology, Operations Research Center.

[28] Toth, P., and Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*, 2 ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2014.

[29] Zhang, Z., Qin, H., and Lim, A. A memetic algorithm for the capacitated m-ring-star problem. *Applied Intelligence 40*, 305 (2014).