A Hybrid Metaheuristic for a Multi-Objective Mixed Capaciated General Routing Problem

Ingvild Lyckander

Master of Science in Physics and Mathematics

Submission date:	July 2014
Supervisor:	Markus Grasmair, MATH
Co-supervisor:	Elin Halvorsen-Weare, SINTEF
	Geir Hasle, SINTEF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

In this thesis, we have studied a bi-objective variant of the Mixed Capacitated General Routing Problem (MCGRP). The MCGRP is a generalization of other well known routing problems. It is defined on a mixed, weighted graph, where a homogeneous fleet of vehicles with capacity constraints services a set of required entities. These entities can be nodes, directed arcs and undirected edges. The aim of the problem is to find a set of vehicle routes so that every required entity is serviced exactly once and the total route cost is minimized. In the current work, a bi-objective variant of the MCGRP is proposed, where also route balance is optimized.

To solve the problem, a hybrid metaheuristic solution method is proposed. The aim of the method is to find a diversified set of potential Pareto optimal solutions with high quality objective values. The solution method is a variant of a genetic algorithm to obtain a diversified set of solutions, combined with local search based heuristics to improve the quality of the objective values.

To our knowledge, this is the first study of multi-objective variants of the MCGRP, hence the results cannot be compared directly with results from other studies. Instead, the performance of the method is evaluated by visual inspection of the plotted potential Pareto front and by comparing the quality of the objective values with the best known solutions to the single-objective MCGRP.

The solution method is conducted on 23 instances. Solutions that are as good as the best known solutions of the single-objective MCGRP was found for two of them, of which one is known to be optimal. The solutions were well spread out along the potential Pareto front for most of the instances, but a large population size or multiple runs of the same instance is necessary to obtain a good approximation of the Pareto front. For most of the instances, the objectives are conflicting, meaning they cannot be simultaneously optimized.

There is still a lot of research potential for the multi-objective MCGRP, and we hope this thesis will motivate further research.

Sammendrag

I denne masteroppgaven har vi studert en bi-objektiv variant av ruteproblemet kjent som “Mixed Capacitated General Routing Problem” (MCGRP). Dette problemet er en generalisering av andre kjente rute-optimaliserings-problemer. Problemet er definert på en vektet, blandet graf. Noen av enhetene i grafen må betjenes. For å betjene disse enhetene disponeres en homogen flåte med kjøretøy, der hvert kjøretøy har en begrenset betjeningskapasitet. Målet er å finne et optimalt sett med kjøreruter, slik at hver pålagt enhet er betjent nøyaktig én gang og den totale rutekostnaden er minimert. Vi har studert en bi-objektiv variant av MCGRP, hvor også rutebalanse er optimalisert.

For å løse problemet har vi foreslått en hybrid metaheuristisk løsningsmetode. Formålet med metoden er å finne et sett av potensielle Paretooptimale løsninger, med god spredning langs en approksimert Paretofront, minimal rutekostnad og god rutebalanse. For å oppnå god spredning av løsninger har vi valgt å basere løsningsmetoden på en genetisk algoritme, kombinert med lokalsøk-baserte heuristikker for å forbedre kvaliteten på objektivverdiene.

Vi har ikke funnet noen andre studier på varianter av MCGRP med mer en ett objektiv, hvilket betyr at vi ikke kan sammenligne resultatene våre direkte med resultater fra andre studier. Derimot har vi evaluert resultatene ved å grafisk fremstille den approksimerte Paretofronten og ved å sammenligne objektivverdiene med de beste kjente løsninger til den enkelt-objektive MCGRP der total rutekostnad er minimert.

Løsningsmetoden ble gjennomført for 23 datasett. Løsninger med like god rutekostobjektiv som de beste kjente løsningene for enkelt-objektiv MCGRP ble funnet for to datasett, hvorav én er bevist å være optimal. For de fleste datasettene viste løsningene seg å ha god spredning langs den approksimerte Paretofronten. Dog vil det trolig være nødvendig å øke populasjonsstørrelsen eller utføre et søk flere ganger for samme datasett for å oppnå en god approksimasjon av den optimale Paretofronten. De to objektivene er tilsynelatende i konflikt for de fleste datasettene, hvilket betyr at de ikke kan optimaliseres på samme tid.

Det er fortsatt stort potensial for videre forskning på den multi-objektive varianten av MCGRP, og vi håper at denne oppgaven vil motivere til fremtidige studier.

Acknowledgements

This thesis was written in cooperation with SINTEF, and is my completion of a Master's degree in Applied Mathematics at the Norwegian University of Science and Technology.

I would like to thank my supervisors at SINTEF Dr. Geir Hasle and Dr. Elin Halvorsen-Weare, and my supervisor at NTNU, Dr. Markus Grasmair.

Especially, I would like to thank Geir and Elin for giving me the opportunity to write the thesis at SINTEF and present our work at the VeRoLog conference in Oslo, June 2014.

Finally, I would like to thank my good friends Kjetil and Ingrid for feedback on the thesis.

Ingvild Lyckander Trondheim, July 2014

Abbreviations

MCGRP	Mixed Capacitated General Routing Problem
CVRP	Capacitated General Routing Problem
CARP	Capacitated Arc Routing Problem
MOP	Multi-objective optimization problem
PPS	Potential Pareto optimal solution
EA	Evolutionary algorithm
GA	Genetic algorithm
LS	Local search
LNS	Large Neighborhood Search
VRP	Vehicle Routing Problem
TSP	Travelling Salesman Problem
GRP	General Routing Problem
CGRP	Capacitated General Routing Problem
CGRP-m	Capacitated General Routing Problem on Mixed Graphs
OX	Order crossover
NSGA	Nondominant Sorting Genetic Algorithm
RBX	Route based crossover
w-DAG	Weighted directed acyclic graph
ALNS	An Adaptive Large Neighborhood Search

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Routing problems have been well-studied since first introduced in combinatorial optimization theory by Dantzig and Ramser (1959, [1]), mainly because of the wide range of applications in industry, such as logistics, transportation and distribution, but also due to scientific interest. For all capacitated routing problems, capacity constrained vehicles seek to service a set of customers. The customers are located on a network that defines the path and travel cost between them. The aim is to generate a set of routes for the vehicles, where a route is the vehicle's path including all the customers serviced by the same vehicle. The set of routes must fulfil the criterion that every customer is serviced exactly once and that the total demand serviced by each vehicle does not exceed the vehicle capacity. At the same time the travel cost should minimized.

Routing problems are often divided into two classes according to the type of the components of the graph that are required to be serviced. When a subset of the nodes is required to be serviced, the problem is categorized as a Capacitated Vehicle Routing Problem (CVRP). On the other hand, problems containing a set of links required to be serviced are referred to as Capacitated Arc Routing Problems (CARP). However, for many real life problems there is no clear distinction between node and arc routing problems. Because of that, Pandi and Muralidharan (1995, [2]) introduced a general routing problem on a mixed graph, which later was the inspiration for the problem studied in this thesis, called the Mixed Capacitated General Routing Problem (MCGRP). Here, both links and nodes are required to be serviced. Hence, the MCGRP is a generalization of both the CVRP and the CARP. In contrast to the latter two problems, the MCGRP is defined on a mixed graph, i.e., the network may contain both directed and undirected links. The general definition of the problem makes MCGRP more suited than node or arc routing problems for modelling certain real-life problems. Two relevant examples are waste collection and newspaper delivery. Here, demand located along streets of single households can be aggregated and represented as a link required to be serviced. On the other hand, isolated demand locations, such as hospitals and apartment buildings, are more adequately represented by nodes. The problem is explained in more detail in Chapter 3. Due to the differences between the problems, previously proposed solution methods for CVRPs and CARPs cannot be used to solve the MCGRP without transforming the problem or modifying the method. Hence, in the study of the MCGRP new solutions methods are constructed, but they are often inspired by existing solution methods for other routing problems.

Despite its scientific interest and the large number of fields of applications, there are few previous studies on the MCGRP. Moreover, in the existing literature, the problems usually consider only a single cost objective. However, in many applications, additional objectives may be of interest. Relevant objectives can be route balancing, load balancing, soft time windows, minimizing the number of vehicles in the solution, and others. In this thesis we study multi-objective variants of the MCGRP, where total route cost is still minimized, but route balancing must be optimized at the same time. The objective components are given in more detail in Chapter 4.

The goal of a multi-objective optimization problem (MOP) is to obtain a set of Pareto optimal solutions. Here, a solution is said to be Pareto optimal if none of the objective function values can be improved without deteriorating at least one other objective value. The set of Pareto optimal solutions is called a Pareto front. To get a good approximation to the whole Pareto front, diversity among the many Pareto optimal solutions is needed. Thus, for a MOP we are aiming to achieve diversity along the Pareto front and convergence towards optimal solutions and at the same time want to keep the run-time of the algorithm at a reasonable level.

For real-life applications, the size of the instances is likely to cause run-time issues. This is a result of the complex structure of the MCGRP, combined with the extremely high number of possible solutions. Due to its complexity, heuristics and metaheuristics have been applied to find approximations of the optimal solutions to the MCGRP. In this thesis a hybrid metaheuristic is proposed to find a high quality approximation to the Pareto front for the multi-objective MCGRP. The metaheuristic uses a genetic algorithm to provide a set of diversified solutions, on which a local search based heuristic is applied with aim of finding improved solutions closer to the frontier.

To increase the efficiency of the search, the metaheuristic uses a non-dominated ranking sort mechanism. In addition, elitism among the population is applied in the GA with the intention of increasing the diversity among the solutions. Both principles are explained in more detail in Chapter 5, followed by the description of the metaheuristic in Chapter 6.

In order for the reader to fully appreciate the methods proposed in this thesis, some background on MOPs is needed. Hence, a brief description of MOP and an overview of solution methods to multi-objective problems are given in Chapter 2.

This is the first study on multi-objective MCGRP, so the results cannot be compared with results from other methods. Instead, the results are evaluated by studing the shape of the Pareto front and the minimal cost found is compared with the total route cost of the best known solutions to the single-objective MCGRP. The results and evaluation are presented in Chapter 7 and Chapter 8.

There is still a lot of research potential for the multi-objective MCGRP and improvements to the proposed solution method that can be done. Some suggestions are presented in Chapter 8.2.

Chapter 2

Background

Before the metaheuristic method proposed in this thesis is described, some background on multi-objective problems needs to be given. This chapter covers a brief description of Multi-Objective Problems (MOP) in general and of some methods that can be used to solve them.

A MOP is a combinatorial optimization problem with multiple objectives that should be minimized given a set of constraints. In Ehrgott and Gandibleux (2000, [3]), the general form of a linear MOP is defined as follows:

$$\underset{x}{\text{minimize}}\, Fx \tag{2.1a}$$

subject to

$$x \in X, \tag{2.1b}$$

where X is the feasible domain in $\mathbb{R}^N$ defined by the set of constraints. The constraints are assumed to be linear, hence the domain is convex. Here F is a $Q \times N$ objective matrix, where Q is the number of objectives. The matrix F contains Q $1 \times N$ row vectors F_q, defining the parameters for each objective q.

An optimal solution to a MOP is a set of non-dominated solutions, i.e. solutions not dominated by other solutions. Jozefowiez et al. (2007, [4]) define dominance as follows:

Definition 2.0.1 *A solution $x = (x_1, x_2, ..., x_N)$ dominates (denoted $\prec$) a solution $z = (z_1, z_2, ..., z_N)$ if and only if for all $q \in \{1, ..., Q\}$ we have $F_q(x) \leq F_q(z)$, and there exists a $q \in \{1, ..., Q\}$ such that $F_q(x) < F_q(z)$.*

The set of non-dominated solutions is called a *Pareto set.*

Definition 2.0.2 *A Pareto set consists of all non-dominated solutions to a MOP. The solutions in a Pareto set are called Pareto optimal solutions. The continuous line of all possible non-dominated solutions to a MOP is referred to as the Pareto front.*

It may be of interest to observe that the Pareto optimal solutions to a linear MOP on a convex domain are always located on the boundary of the feasible domain (Faggian 2007, [5]).

When a heuristic search method is used to find the Pareto set, the solutions are not guaranteed to be optimal. Therefore, it will be useful to define a solution to be potential Pareto optimal when discussing heuristic search methods:

Definition 2.0.3 *A potential Pareto optimal solution (PPS) relative to an algorithm $\mathcal{A}$ is a solution found by $\mathcal{A}$, which is not dominated by any other solutions found by $\mathcal{A}$. The line between all the PPS found by algorithm $\mathcal{A}$ is referred to as the potential Pareto front relative to $\mathcal{A}$.*

In this thesis, the goal is to find an approximation to the Pareto front. However, if the preferences of the objectives are known, techniques can be used to seek a single solution on the frontier. In the following, techniques searching for a single solution are presented, before techniques searching for an approximation to the Pareto front are discussed.

Methods searching for a single solution
Variations of weighted method, bounded constraint methods and goal programming are some examples. In the *weighted method*, the objectives are combined by a foreknown preference, forming a single-objective problem. The objectives can be combined linearly or with a higher degree. A solution found by the weighted method is Pareto optimal, and the weight parameters define the location of the solution on the Pareto front. However, there are no correlations between the weight and the Pareto front. Or in other words, an uniform spread of weight parameters does not necessarily produce a uniform spread of points on the Pareto front (Caramia and Dell'Olmo, 2008, [6]). Hence, the variation of weights may be a computational burden.

In *bounded constraints methods*, only the single most important objective function is minimized. The other objectives form additional constraints, where given upper and lower bounds decide the feasible interval for each objective value.

The *ϵ-constraint method* is an example of the bounded constraint method on a minimization problem, where a lower bound is not needed. The selection of the bound parameters is the main challenge. With appropriate values for the bound parameters, the method will find Pareto optimal solutions. However, when the constraints defined by the bound parameters are strict, the problem may become infeasible.

In *goal programming methods*, a target value is defined for each objective and the total deviation from the goals is minimized in one single objective. The method offers a Pareto slack optimum, which means the objective values are at most a set distance from the optimum. The allowed maximum distance from the optimum is defined by the decision-maker, and by reducing this distance, the method generates a Pareto optimal solution. Common for all these methods is that the problem is turned into a single-objective problem.

Another approach is the *Lexicographic method*, where single-objective problems are solved multiple times. In this method, the objective functions are arranged in order of importance, and the problems are solved in order of decreasing priority. At each iteration, the objectives with higher priority than the one to be minimized form additional constraints. These constraints ensure the corresponding objective value to be as least as good as the optimum found when that objective was minimized. This method is useful when a continuous trade-off among the objectives is not of interest. The shortcoming is that optimization problems that are solved near the end of the hierarchy may become infeasible or do not influence the optimal solution, because of strict constraints.

A drawback for all of these methods is the requirement of an a priori articulation of preference, which means that the decision-maker has to choose appropriate parameters for the value of the objective components or the relative importance of each objective.

Methods searching for Pareto front
A priori articulation of preference is not needed when searching for the Pareto front. Heuristic search algorithms are popular methods to find an approximation to the Pareto front. Because the goal of these methods is to find an approximation to the whole Pareto front, we have to choose a heuristic search method which seeks diversity along the Pareto front and convergence towards Pareto optimal solutions. Diversity may be achieved by using a global optimization technique or by including specially designed diversity techniques. The diversified solutions can be improved by a local search (LS) based method to find local optimums in the areas close to the solutions.

The most common heuristics used on MOPs are LS strategies and evolutionary algorithms (EA), introduced by Fogel (1966, [7]), or a hybrid of these. In EA, new solutions are created by combining existing solutions and only the best solutions are kept in the search.

Since the new solutions are created by crossing two solutions, the new solutions may differ to a great extend from the existing solutions at each step. This enables the search to cover a large area of possible solutions in a few steps, and the search converges more likely to the global optimal solutions than to local optimal solutions (Marler, 2004 [8]). These methods are discussed in detail by Marler and Arora (2004, [8]).

The genetic algorithm (GA) is an evolutionary method, first introduced by Holland (1975, [9]). The algorithm is based on the principle of natural selection and evolution. A population of candidate solutions evolves towards better solutions in an iterated process through genetic operators. At each iteration, the current solutions in the population are called a generation. A new generation of the population is selected from the current generation, based on the solutions *fitness value*. The fitness value is a measure of how good a solution is relative to the current generation. A crossover operator is then applied on pairs of selected solutions in the population, called *parent solutions*, to generate new solutions, called *child solutions*. The crossover operator decides how the two parent solutions are combined to generate child solutions. A mutation operator makes changes on a random selection of the child solutions with a certain probability, before the child solutions are included in the population.

As mentioned, evolutionary algorithms search a large area of possible solutions and do not easily get trapped in local optima. However, when one of the global optimums is almost reached, EA may not be sufficient to find the precise location of the optimal solution because of the possibly big changes in the solutions at each step. LS based strategies, on the other hand, are designed to locate the local optimum. Hence, it would be useful to combine the GA with a LS based method to obtain both the local and the global property.

In a LS based heuristic, an initial solution gradually improves by making changes within a single solution as long as an improved solution is found. Two LS based heuristics proposed in this thesis, is the Large Neighborhood Search (LNS) and Cross.

The LNS was first proposed by Shaw (1998, [10]). When the LNS is applied on a solution in a routing problem where the aim is to distribute tasks to vehicles, the solution is first destroyed by removing a set of tasks. Then, the tasks are reinserted in the solution through a repair process.

When the Cross algorithm (1997, [11]) is applied on a solution in a routing problem, a new solution is created by swapping a set of tasks between two vehicles within the same solution.

Chapter 3

The MCGRP

Capacitated routing problems were first introduced by Dantzig and Ramser (1959, [1]). They called the problem the Vehicle Routing Problem (VRP). The VRP is similar to the Travelling Salesman Problem (TSP), but the total demand serviced by each vehicle is constrained by an upper capacity. The problem is defined on an undirected graph and a homogeneous fleet of vehicles is used to service the subset of the nodes that are required to be serviced.

15 years later, Orloff (1974, [12]) needed a formulation to model real transportation problems, where the VRP was not sufficient. He defined the General Routing Problem (GRP), where both a subset of the edges and the nodes are required.

Both the VRP and the GRP are defined on an undirected graph. However, when modelling routing problems in urban areas, it is useful to introduce the possibility of one-way streets, modelled as arcs in the network. A routing problem defined on a directed graph was first introduced by Golden and Wong (1981, [13]): The Capacitated Arc Routing Problem (CARP). Here, as for the VRP, a subset of arcs is required to be serviced. The VRP is also referred to as the Capacitated Vehicle Routing Problem (CVRP).

The MCGRP is a generalization of all the latter problems. In contrast to the other problems, the MCGRP is defined on a mixed graph containing both edges and arcs at the same time. In addition, a subset of the nodes, edges and arcs may be required to be serviced, as in CVRP, GRP and CARP respectively.

Prior work on the MCGRP with a single objective is presented in this chapter. In Chapter 3.2, the MCGRP is described in more detail.

3.1 Prior work on the MCGRP

Pandi and Muralidharan (1995, [2]) were the first to introduce the General Routing Problem (GRP) on a mixed graph under constraints, defined as the *Capacitated General Routing Problem* (CGRP). For the CGRP defined by Pandi and Muralidharan, a heterogeneous fleet of vehicles is fixed over specified segments and nodes of a street network and the routes are under maximum duration constraints. A homogeneous fleet version of the CGRP was introduced by Gutierrez et al. (2002, [14]), called *Capactiated General Routing Problem on Mixed Graphs* (CGRP-m). Prins and Bouchenoua (2004, [15]) formulated the MCGRP as we will use in this thesis. They constructed a set of benchmark instances which will be used in the analysis in this thesis.

Several heuristic solution methods have been used to generate upper and lower bounds on instances of the MCGRP. The most common approach for generating upper bounds for the MCGRP is LS based heuristics, such as Simulated Annealing (Kokubugata et al., 2007 [16]), an Adaptive Iterated Local Search algorithm (Dell'Amico et al., 2012 [17]), and SINTEF's VRP solver Spider (Hasle et al., 2012 [18]). Kokubugata et al. (2007, [16]) use three LS based operators to generate neighbor solutions in the Simulated Annealing algorithm: exchange of required tasks between two routes, move of one task from one route to another, and exchange of tasks within the same route. The proposed method proved to be advantageous in handling complicated variants of the MCGRP such as MCGRP with time windows or multiple depots. In the Adaptive Iterated Local Search algorithm by Dell'Amico et al. (2012, [17]), the LS based operators or-opt, 2-opt, 3-opt, and swap of tasks are utilized. A set of Destructor and Constructor operators is used to remove and re-insert a randomly drawn number of tasks at each iteration. A Destructor and Constructor operator pair is chosen based on the previous effectiveness of the operators. Prins and Bouchenoua (2005, [15]) use a memetic algorithm to solve the MCGRP. An initial solution is constructed by joining the nearest free task to a route until the vehicle's capacity is exhausted. The routes in an initial solution are then merged if possible. To generate new generations, the order crossover (OX) is applied on two parent solutions. The child individual forms a large TSP tour, on which a split operator is used to separate the tour into feasible routes, based on the shortest path in a direct auxiliary graph. Based on the solutions of a matching problem, Bach et al. (2013, [19]) generated lower bounds on the MCGRP.

Bosco et al. (2013, [20]) were the first to formulate the problem as an Integer Program and used a *Branch-and-Cut* algorithm to find exact optimal solutions for small instances. Exact solution methods were also explored by Gaze (2014, [21]), where more optimal solutions were found.

3.2 Problem description

The aim of the MCGRP is to generate a set of vehicle routes starting and ending at a depot, such that every task is serviced exactly once and the total demand serviced by each vehicle does not exceed the vehicle capacity. In this thesis, a multi-objective variant of the MCGRP is studied. Here, the total travel cost is minimized, and at the same time the route balance is optimized.

Before we define vehicles route, a description of the network follows. The MCGRP is defined on a mixed weighted graph $G = (V, A, E)$ of vertices, directed links, and undirected links. In order to simplify the notation we assume that the set V of vertices has the form $V = \{1, ..., N\}$, where N is the number of vertices. The set $A \subseteq \{(i, j) \in V \times V\}$ contains the directed links, referred to as *arcs*. The undirected links, referred to as *edges*, are elements of the subset $E \subseteq \{(i, j) : i, j \in V, i < j\}$. To easily keep the properties of arcs and edges in the implementation of the problem in Chapter 5, edges are handled as two opposite directed arcs. Therefore, an additional set of links E^T is introduced. The set E^T is defined as $E^T = \{(j, i) : (i, j) \in E\}$, meaning that for every element (i, j) in E, there is a corresponding element with opposite direction (j, i) in E^T. Because a link cannot be both an arc and an edge, the sets A and E must be disjoint, i.e., $A \cap E = \emptyset$. Similarly, $A \cap E^T = \emptyset$. The union of A and E forms the set of all links in the graph. Every link has a non-negative weight c, denoting the cost of traversing the given link. There is no weight associated with the nodes. The graph G is assumed to be strongly connected. That is, there exists a path between any two vertices. By our definition of the problem, loops and parallel links may occur in the graph. However, the weighted graph need not obey the triangle inequality.

Some components in the graph are required to be serviced, defined by the subsets $V_R \subseteq V$, $A_R \subseteq A$, and $E_R \subseteq E$. Every required component, later referred to as *task*, must be serviced once by a single vehicle. Every task has a non-negative demand, given by the demand function d: $(V_R \cup E_R \cup A_R) \to \mathbb{R}_{\geq 0}$. In addition, every task in the graph has a corresponding non-negative service cost, s: $(V_R \cup E_R \cup A_R) \to \mathbb{R}_{\geq 0}$.

Now, as the network is defined, the description of a vehicle route follows. A homogeneous fleet of vehicles K is based in one distinguished vertex, referred to as the *depot*. The fleet consists of a set of at most M vehicles with maximum load capacity C. A vehicle route, later referred to as *route*, is a sequence of connected links and nodes, starting and ending at the depot. A route forms a closed walk, defined in ([22]) as follows:

Definition 3.2.1 *In a graph, a walk is a sequence of vertices* $(v_0, v_1, v_2, ..., v_k)$ *such that* v_i *is adjacent with a link to* v_{i+1} *for all* $1 \leq i \leq k - 1$*. In a walk,* *vertices may be repeated. We say that a walk is closed if the first and last vertices are the same.*

Thus, a route R_k has the form

$$R_k = (v_{k,0}, e_{k,1}, v_{k,1}, e_{k,2}, ..., v_{k,n_k-1}, e_{k,n_k}, v_{k,n_k}), \quad n_k \in \mathbb{N}, \tag{3.1}$$

where $v_{k,0} = v_{k,n_k}$ is the depot, $v_{k,l} \in V$ for all $l \leq n_k$, $e_{k,l} \in (E \cup E^T \cup A)$ for all $l \leq n_k$ and $e_{k,l}$ is a link from node $v_{k,l-1}$ to node $v_{k,l}$. Note that a link or a node, including the depot, may occur multiple times in a route. By the definition of a closed walk, there cannot be any disconnected subtours within a route.

A solution to the MCGRP is a set of routes, such that every task is serviced exactly once. A set of routes together with a function $\Psi : (E_R \cup A_R \cup V_R) \rightarrow k$ form a *routing plan* $P := \{\{(R_k)_{k \in K}\}, \Psi\}$. The function Ψ defines which tasks are serviced by vehicle k. It has the property that $v \in R_{\Psi(v)}$ for all $v \in V_R$ and $e \in R_{\Psi(e)}$ for all $e \in (E_R \cup A_R)$ in such way that all tasks are contained precisely once in one route in the routing plan. This is one out of two criteria for a routing plan to be feasible. The second criterion says that the total demand serviced on a route cannot exceed the vehicle maximum load capacity C, i.e.,

$$\sum_{\substack{e \in (E_R \cup A_R) \\ \Psi(e)=k}} d(e) + \sum_{\substack{v \in V_R \\ \Psi(v)=k}} d(v) \leq C, \quad k \in K. \tag{3.2}$$

The number of vehicles in use in the problem is defined by the instance, and may be either fixed, bounded by the size of the fleet, or unbounded if the fleet size is unlimited. Due to the criterion of non-splitting demand, there cannot be solutions with more vehicles than tasks in the graph.

When a link is traversed without being serviced, it is said to be *deadheaded*. The cost of the route is defined as the sum of traversal cost on the deadheaded links and service cost on the required links and nodes included in the route. There is no cost in terms of additional vehicles in use. If the number of vehicles in use is not fixed, the number of vehicles is decided by minimization of the total route cost, but cannot exceed the fleet size for the bounded case.

Figure 3.1 shows a simple example of a network used in the MCGRP, from Lyckander (2014, [23]). Required links and nodes are drawn with a solid line, non-required links and nodes with a dashed line. The costs and demands are shown on the corresponding link or node. In this example, the vehicle capacity is 400 and node 1 represents the depot. A solution to the example shown in Figure 3.1 is illustrated in Figure 3.2.

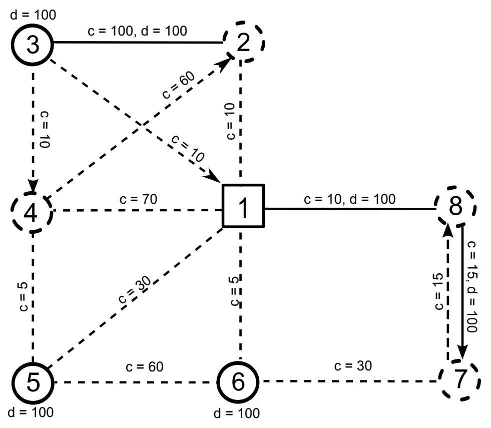


Figure 3.1: A simple MCGRP example.

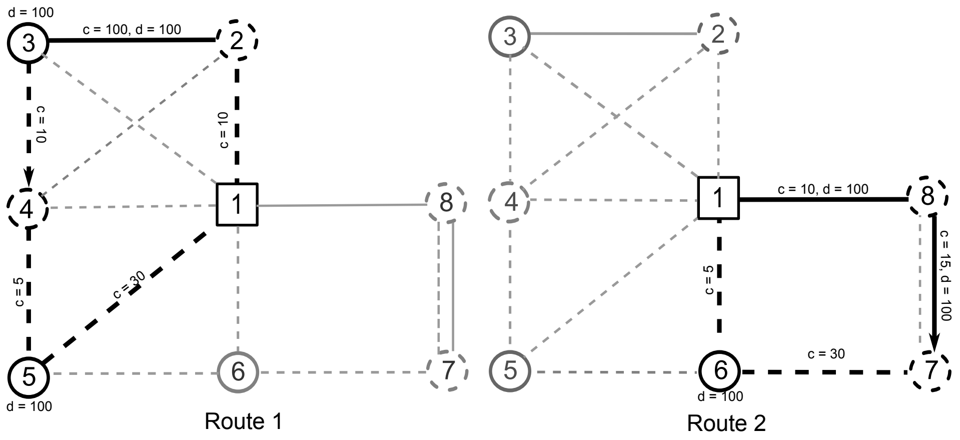


Figure 3.2: A two-route solution to the example shown in Figure 3.1. Route 1: cost = 155, load = 300; route 2: cost = 60, load = 300. Total routing cost is 215.

Run-time issues may occur when solving a MCGRP, due to the problem's complex structure. In fact, the TSP is a special case of the MCGRP.

The TSP was proven to be NP-hard in the mid-1960s, discussed in e.g. Garey and Johnson and (1979, [24]). Hence, MCGRP is NP-hard.

Exact methods can often solve MCGRP instances of smaller size, but due to the exponential running time, heuristic procedures are preferred as solution methods for many real life problems.

Chapter 4

Multi-objective MCGRP

In most studies on routing problems, the aim of the problem is to minimize the total route cost including service costs and use of vehicles. However, a majority of real-life problems, e.g. in distribution and logistics, have additional objective components that should be optimized. In this chapter, an overview of multi-objective routing problems which previously have been addressed in the literature is given. Relevant objectives for the MCGRP and how they may influence the problem are described in Chapter 4.2.

4.1 Prior work on multi-objective routing problems

Multi-objective vehicle routing problems have received increased attention, due their its many real life applications and scientific interest. Jozefowiez et al. (2008, [25]) survey the different multi-objective vehicle routing problems addressed in the literature. Marler and Arora (2004, [8]) present an overview of multi-objective optimization methods in general. In this chapter, an overview of the studies based on techniques that can be used to find a single point on the frontier follows, before an overview of studies of heuristic search methods used to find an approximation to the Pareto front is given.

Different techniques to find a single Pareto optimal solution have been proposed in the literature. Norouzi et al. (2009, [26]) solve an open vehicle routing problem, where travel cost is minimized and the obtained sale is maximized. They use an ϵ-constraint method to compare the results from a multi-objective particle swarm optimization method proposed in the paper. Giannikos (1995, [27]) uses goal programming to solve a location and routing problem for hazardous waste transportation and treatment. In addition to minimizing the total operating cost and perceived risk, the aim is the equitable distribution of risk among population centres and of the disutility caused by the operation of the treatment facilities. Keller and Goodchild (1987, [28]) propose a lexicographic method for minimizing the length and maximizing the profit of a TSP with profit.

Usually, when solving a multi-objective routing problem, the aim is to find a set of solutions representing a good approximation to the Pareto front instead of presenting a single solution. That is the reason why the most used heuristic methods are population based approaches, such as genetic or memetic algorithms. For instance, Jozefowiez et al. (2009, [29]) use a genetic algorithm involving classical multi-objective operators to solve a bi-objective VRP with route balancing. The aim is to minimize total route cost and the difference between the maximal and minimal route cost. Additionally, they introduce two mechanisms which favour the diversification of the search. Mei et al. (2011, [30]) use a Decomposition-Based Memetic Algorithm for a multi-objective CARP. As mentioned in Chapter 2, different LS based methods are applied on MOPs. Hansen (1997, [31]) gives a good description of how Tabu Search could be implemented to solve MOP in general, and Caballero et al. (2007, [32]) apply Tabu Search on a location routing problem. Banos et al. (2013, [33]) implement a Simulated Annealing algorithm to solve a multi-objective VRP with time window. Other approaches are also addressed in the literature, such as Ant Colony system, by e.g. Baran and Schaerer (2003, [34]), and Scatter Search, by e.g. Corberan et al. (2002, [35]). To ensure both diversification and convergence toward an optimal solution, hybrid algorithms have been a preferred method. Jozefowiez et al. (2007, [4]) use a multi-objective combined EA and LS to solve the VRP with route balancing. They use a genetic algorithm to generate a set of solutions spread out close to the Pareto front, and use Tabu Search to search for solutions closer to the Pareto front. Banos et al. (2013, [36]) introduce a combined multi-start multi-objective EA with Simulated Annealing to solve the multi-objective VRP with time windows.

Different techniques can be used to achieve diversity in the search. Srinivas and Deb (1995, [37]) propose a Nondominant Sorting Genetic Algorithm (NSGA) to solve multi-objective optimization problems in general. The non-dominated sorting ranks the solutions in a population based on their non-domination. A sharing function is used to locate high populated areas. The fitness function favours low rank and low density areas, with the aims of archiving better diversity and convergence than regular genetic algorithms. An improved version of the NSGA is proposed by Deb et al. (2002, [38]), called NSGA-II. The main improvements were the lowering of the run-time of the non-dominated sorting from $O(MN^3)$ to $O(MN^2)$, inclusion of elitism techniques which speeds up the performance of the algorithm, and reduce the number of parameteres set by the decision maker. An elitist technique was also proposed in Jozefowiez et al. (2009, [29]). The diversification technique was an expansion of the sharing strategy, see e.g. Goldberg and Richardson (1987, [39]), inspired from the elitism strategy. In the search, potential Pareto optimal solutions are stored in an archive A_0. In addition to A_0, archives containing the non-dominated solutions to the problem where one objective function is maximized instead of minimized are considered. At each iteration, some solutions in the archives are included in the population.

4.2 Relevant objectives for the multi-objective MCGRP

Depending on the application, many objectives may be of interest when considered a multi-objective MCGRP. Minimizing a risk or the waiting time for the costumers, or maximize a profit are some examples. Balancing of the routes is also a relevant aspect, since the drivers of the routes in delivery problems are often motivated by equally distributed workload.

In this thesis we have studied two objectives: *routing cost* and *route balance*. When only two objectives are considered, the problem may be referred to as a bi-objective MCGRP.

In this thesis, the cost of a link denotes the length of the link. Additionally, we assume the vehicles to traverse the links with constant speed. Thus, the time required to traverse a link is proportional to the length. We inform the reader that cost, time and length of a link may be used interchangeably in this thesis. In the same way, the service cost of a task indicates the time required to service the task.

Routing cost

The aim of the routing cost objective is to minimize the total cost of a solution in terms of travel time. The cost objective includes the costs of traversing the routes and servicing the tasks. Two cost functions are defined for the problem:

The function

$$c : (E \cup A) \to \mathbb{R}_{\geq 0} \tag{4.1}$$

defines the cost of *deadheading* links, i.e., when a link is traversed, and the function

$$s : (V_R \cup E_R \cup A_R) \to \mathbb{R}_{\geq 0} \tag{4.2}$$

defines the cost of servicing a task. The cost of route k is defined as

$$c_k(R_k) = \sum_{e \in R_k \cap (E \cup A)} c(e) + \sum_{\substack{e \in (E_R \cup A_R) \\ \Psi(e)=k}} s(e) + \sum_{\substack{v \in V_R \\ \Psi(v)=k}} s(v), \quad \forall k \in K. \tag{4.3}$$

The first sum denotes the total traversal cost for route R_k. The second and third sum are the total service cost of tasks serviced by vehicle k located on links and nodes respectively.

The cost objective function F_1 for a routing plan is the total cost of the routes in the routing plan, i.e.,

$$F_1(x) = \sum_{k \in K} c_k(R_k). \tag{4.4}$$

Route balance

The aim of the route balance objective is to equalize the route lengths within a solution. There are several ways to formulate route balance to obtain this. One approach is to minimize the length difference between the longest and shortest route in the routing plan. In this formulation, the length of a route is equal to the route cost, defined in the previous objective. Hence, the objective function F_2 is defined as

$$F_2(x) = \max_{k \in K}(c_k(R_k)) - \min_{k \in K}(c_k(R_k)). \tag{4.5}$$

Another approach is to minimize the spread of route lengths, i.e. variance of the sets of routes in a solution. This variance can be formulated as

$$Var(x) = \frac{1}{|K|} \sum_{k \in K} (c_k(R_k) - \mu_{c_k})^2, \tag{4.6}$$

where μ_{c_k} is the mean route length. When inserting the formula for the mean and letting the objective value be equal to the variance, equation (4.6) becomes

$$F_2'(x) = \frac{1}{2|K|} \sum_{k_1 \in K} \sum_{k_2 \in K} (c_k(R_{k1}) - c_k(R_{k2}))^2. \tag{4.7}$$

The route balance objective may cause complications to the solutions in terms of *artificially balanced routes.* The equalization of route lengths can easily be obtained by making the shortest routes longer by including detours. A solution is said to be artificially balanced if at least one route includes detours. However, this contradicts the route cost objective and is unacceptable in solutions for real life cases. Hence, artificially balanced routes must be avoided. Thus, when the route balance objective is considered, an additional criterion must be included, ensuring every route is optimal in terms of route cost given the set of distributed tasks to service. This criterion is later referred to as *route optimality criterion*. Due to the route cost objective, this criterion is automatically fulfilled when the route balance objective is not considered.

Chapter 5

Genetic algorithm for a multi-objective problem

The hybrid metaheuristic proposed in this thesis is a variant of a genetic algorithm. Therefore, a description of how the genetic algorithm can be applied on a general multi-objective problem is given in this chapter. A description of problem specific parts of the hybrid metaheuristic proposed is given in Chapter 6.

5.1 The genetic algorithm

In a genetic algorithm, a set of solutions, also called individuals, form a population which is improved gradually in an evolutionary process. Each iteration in the search represents a generation of the population. Every unique solution has a unique chromosome, consisting of values for a set of parameters that define a potential solution to the problem. Properties of the chromosome can be inherited or mutated. This leads to the creation of new solutions and the population is evolved toward better solutions.

The basic steps of a genetic algorithm is described in Algorithm 1 at page 23.

At first, the data set is initialized. How we initialize the data is explained in detail in Chapter 6.2. Then a construction algorithm, see Chapter 6.4, is used to create a set of initial solutions. Each iteration in the algorithm consists then of four steps: *assessment, selection, recombination,* and *improvement.*

Assessment
To evaluate the individuals, every individual in the population is assigned a fitness value measuring the success of the individual. The fitness value depends

on the number and size of potential Pareto fronts dominating the individual and the distance from other individuals in the current generation measured in term of the ojective values. Non-dominated individuals, distanced from other individuals, are favoured by this fitness value. The fitness value is defined in Chapter 5.2.

As a part of the assessment step, the method executes an elitism diversification technique to achieve a population well spread along the potential Pareto front. Here, archives containing members of the population with special properties are created. The individuals in the archives are later added back in the population. This technique is explained in Chapter 5.3.

Selection
The selection step consists of building a new generation based on the individuals fitness value and the diversification techniques, and choosing the parent individuals to be recombined to create new child individuals. This is explained in more detail in Chapter 5.4.

Recombination
The recombination step is divided into two parts: crossover and mutation. In the crossover operator, two child individuals are created, inheriting different parts of the chromosomes of each of the parent individuals. The mutation operators are applied on the child individual with a certain probability, making a small change in the child chromosome.

The basic layout of this step is discussed in Chapter 5.5 and a detailed explanation of the implementation of this step in our concrete problem is given in Chapter 6.5 and 6.6.

Improvement
Finally, an improvement function is applied on the child's chromosome. This function systematical makes changes to specific parts of the chromosome with an aim to improve the individual. What kind of improvement methods used in the proposed solution method is described in Chapter 6.7.

The generation iterations in line 3 to 8 in Algorithm 1 continue until the search converges or a maximum number of iterations is reached. This is explained in more detail in Chapter 5.6.

Algorithm 1 `Genetic Algorithm`(*instance*)

```
1: Initialization(instance)
2: P := Initial Solution
3: repeat
4:     Assessment(P)
5:     P ← Selection(P)
6:     Recombine(P)
7:     Improve(P)
8: until Stopping criterion
9: return non-dominated elements of P
```

5.2 Assessment

The genetic algorithm evaluates an individual s in generation P_i by a fitness value φ_s^i. The fitness value depends on the chromosome of the individual and the rest of the individuals in generation P_i. First, a *rank value* κ_s^i is assigned to each individual, denoting how many potential Pareto fronts dominates the individual. The rank value is used for defining a preliminary fitness value ψ_i^s, which is then modified by means of a *niche counter* ϕ_s^i, denoting the density of other solutions in the area around the solution s. The values of the rank, fitness and niche counter are explained in this chapter.

Rank

The rank κ_s^i of solution s in generation P_1 is set by an elitism mechanism. This mechanism uses the dominance rule given in Definition 2.0.1 to assign a rank for each individual. The rank of an individual denotes how many potential Pareto fronts dominate the individual s.

At first, the non-dominated individuals in P_i are assigned the rank 1. In other words, $\kappa_s^i = 1$, if there exists no individual s' in generation P_i with $s' \prec s$. The individual with rank 1 constitute the subset $S_i^1 \subset P_i$, and form the first potential Pareto front in generation P_i.

Then, the non-dominated individuals in $P_i \backslash S_i^1$ are assigned the rank 2. Hence, these individuals are only dominated by some of the individuals in the first potential Pareto front in generation P_i. These individuals constitute the subset $S_i^2 \subset P_i$, and they form the second potential Pareto front.

Inductively, the non-dominated individual in $P_i \setminus \bigcup_{k=1}^m S_i^k$ are assigned the rank $m + 1$.

In other words, for every individual s in P_i, the rank κ_s^i equals $m+1$ if:

1. There exists no solution s' in $P_i \setminus \bigcup_{k=1}^{m} S_i^k$ such that $s' \prec s$.

2. There exist a solution s'' in $P_i \setminus \bigcup_{k=1}^{m-1} S_i^k$, such that $s'' \prec s$.

A naïve approach to sompute the rank for all the individuals, is first to find all non-dominated individuals, assign them the rank 1 and remove them from the set. Then, find all non-dominated individuals, assign the rank 2, and so on. This naïve approach requires $O(N^N)$ time, where N is the number of individuals in the population, but the most common implementation, introduced by Deb et al. (2002, [38]), requires $O(N^2)$ time. We propose a faster approach, $O(N \log(N))$ time, assuming the problem is bi-objective.

At first, the population is sorted in lexicographic order, i.e., in ascending order with respect to the first objective value, and then with respect to the second objective. The rank is assigned to the individuals in that order. This means, when assigning a rank to an individual, referred to as the *current individual*, the individuals with lower first objective value than the current individual are already assigned a rank.

Since the rank is the number of potential Pareto fronts dominating the current individual, studying only individuals dominating the current one is sufficient when assigning a rank value. In other words, it is sufficient to compare the current individual with the individuals with lower first objective value or with the same first, but lower second objective value, to decide a rank for the current individual.

To decide how many potential Pareto fronts dominating the current individual, we need to find an individual with the highest rank dominating the current individual. Thus, the rank of the current individual is one higher than the rank of these individuals. Moreover, these individuals can be found by searching through the individuals with lower first objective value, i.e., the individuals that are already assigned a rank value.

Instead of searching through all the individuals with lower first objective value, it is sufficient to compare the current individual with one individual representing each rank. Due to the ordering of individuals, we know the first objective of the individuals are equal or lower than the first objective value of the current individual. Hence, if a potential Pareto front dominates the current individual, then also the individual with the lowest second objective value in that potential Pareto front, called the *corner individual*, dominates the current individual.

Therefore, we only need to compare the current individual with the corner individual for each rank assigned.

To do this, a list containing the second objective value for the corner individuals is created. The list is sorted in ascending order with respect to the corner individuals' ranks, and due to the dominance rule, the objective values are in ascending order too.

When the second objective of an individual lies between the values at positions i and $i+1$ in the list, the individual is dominated by at least one individual in the i^{th} potential Pareto front, but not by any solutions in the potential Pareto front number $i+1$. Hence, the individual is assigned the rank $i+1$. In addition, it becomes the new corner individual for rank $i+1$.

A special case occurs when the second objective is lower than the first element in the list. Then, the solution is not dominated by any other solution, and it is assigned rank 1. On the other hand, if the second objective is higher than the last element in the list, the solution is dominated by every other already initialized potential Pareto front. The rank is the length of the list plus 1, and an element containing the second objective value is inserted at the end of the list.

The pseudocode for the rank assignment is given in Algorithm 2. In the algorithm, $F_q(s)$ means the value of the q^{th} objective of the individual s.

Algorithm 2 `Rank`(*Current Generation*)

```
1:  $RankIndex \leftarrow \emptyset$
2:  $Mergesort_1(Current\ Generation)$
3:  for all $s \in Current\ Generation$ do
4:      if $F_q(s)$ is equal $F_q(s')$ for all objectives $q$ then
5:          $s \leftarrow$ set same rank as $s'$
6:      else if $F_2(s) <$ first element in RankIndex then
7:          Set $F_2(s)$ to be the first element in RankIndex
8:          $s \leftarrow$ set rank to be 1
9:      else if $F_2(s) >$ last element in RankIndex then
10:         Add $F_2(s)$ to the end of RankIndex
11:         $s \leftarrow$ set rank to be the number of elements in Rank Index
12:     else
13:         Find $j$ such that $Rank\ Index(j) \leq F_2(s) < Rank\ Index(j+1)$.
14:         $RankIndex(j+1) \leftarrow F_2(s)$
15:         $s \leftarrow$ set rank $j+1$
16:     end if
17:     $s' \leftarrow s$
18: end for
```

In Algorithm 2, the sorting in line 2 requires $O(N\log(N))$ time. Then, for every N solutions, the look up and insertion in lines 13 to 15 requires $O(\log(N))$ time. The other cases in lines 4 to 11 are computed in constant time. Hence, the complexity of the algorithm is $O(N\log(N))$.

The preliminary fitness value
The preliminary fitness of the individual s is given as:

$$\psi_s^i = \frac{S(N + R_{\kappa_s^i}) - R_{\kappa_s^i}}{N} \tag{5.1}$$

where N is the population size, κ_s^i is the rank of solution s in population P_i and $R_{\kappa_s^i}$ is

$$R_{\kappa_s^i} = 1 + \sum_{r > \kappa_s^i} |E_r^i|, \tag{5.2}$$

where E_r^i is the size of the set of all individuals in population P_i with rank r. Moreover, S is a selection pressure, decided by the decision maker. Higher preliminary fitness is favored if S is greater than 1.

Hence, the preliminary fitness value does not depend on the objective values themself, but only on the rank. As the preliminary fitness value is defined, every individual with the same rank has the same preliminary fitness value. But, up until now, the density around an individual is not considered. The density is calculated by a sharing function, explained in the next section.

Sharing function
The individuals in low density areas are preferred compared to individuals in crowded areas. Therefore, to enhance diversity in the population, the fitness value is divided by a *niche counter*. The niche counter increases with the number of solutions within a given distance from the solution, and a smaller distance gives a higher value.

The niche counter at generation P_i is defined as

$$\phi_s^i = \sum_{s' \in P_i} sh(s, s'), \tag{5.3}$$

where $sh(s, s')$ is the sharing function of the current individual s and another individual s' in the same generation. The sharing function is defined as

$$sh(s, s') = \begin{cases} 1 - \frac{d(s,s')}{\gamma}, & \text{if} \quad d(s, s') \leq \gamma, \\ 0, & \text{otherwise,} \end{cases} \tag{5.4}$$

where $d(s, s')$ is the Manhattan distance between the objective values of the individuals s and s'. γ is the distance of which two solutions are defined to be neighbors, given by the decision maker.

At last, a solution s in generation P_i is evaluated by a new fitness value, which is divided by the niche counter:

$$\varphi_s^i = \frac{\psi_s^i}{\phi_s^i}. \tag{5.5}$$

5.3 Diversification techniques

An elitism diversification technique proposed in Jozefowiez et al. (2007, [4]), is included in the solution method to achieve population spread out along the potential Pareto front.

All new non-dominated individuals are kept in a separate archive A_0. At each generation, a number of individuals stored in the archive are included in the population. In this way, the population will contain a higher number of high-quality individuals.

Additionally, to increase the diversity of the population, a number of individuals are included in the population from separate *elitism archives*. There is a corresponding elitism archive for each objective function, containing the non-dominated individuals according to a new dominance rule $\prec_q$, where one objective is maximized instead of minimized. The dominance rule is defined as follows:

$$\begin{aligned} &\forall y, z \in P_i, y \prec_q z \Leftrightarrow \\ &\qquad \big(\forall l \in \{1, ..., Q\} \setminus \{q\} F_l(y) \leq F_l(z)\big) \wedge \big(F_q(y) \geq F_q(z)\big) \wedge \\ &\qquad \big((\exists l \in \{1, ..., Q\} F_l(y) < F_l(z)) \vee (F_q(y) > F_q(z))\big), \end{aligned} \tag{5.6}$$

where P_i is the current population. The archive A_q contains the non-dominated individuals according to dominance rule $\prec_q$, where the q^{th} objective component is maximized and the rest of the components are minimized. A few individuals from each archive will be added to the new population at each generation, which will be explained in Chapter 5.4.

In the following, the method to find the non-dominated individuals in the population with respect to the new dominance rule $\prec_q$ is described. Here, we assume the problem to be bi-objective.

At first, the population is sorted in increasing order by the value of the objective to be minimized, later referred to as $\overline{q}$, and then by the other objective value.

The first individual in the sorted population is non-dominated. Then, an individual is non-dominated if the $\overline{q}$th objective value is as least as good as the $\overline{q}$th objective values for all the individuals denoted as non-dominated.

An example of the Pareto fronts for the dominance rules $\prec_1$ and $\prec_2$ is given in Figure 5.1.

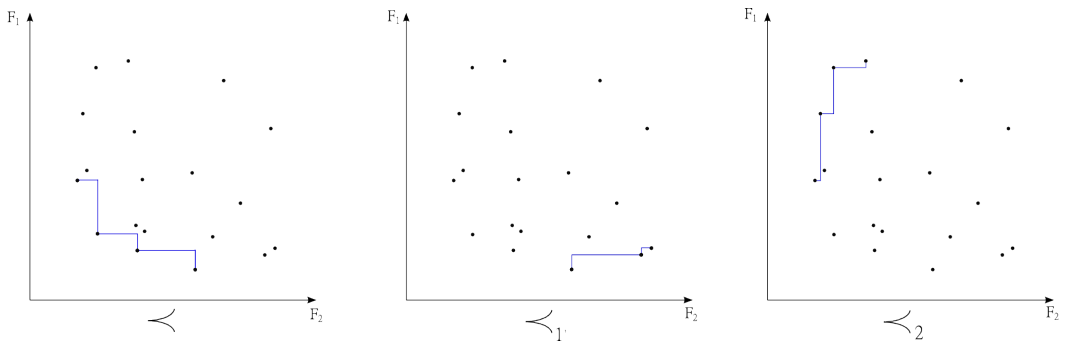


Figure 5.1: An example of the Pareto fronts for the dominance rules $\prec$, $\prec_1$, and $\prec_2$.

Every non-dominated individual found in the search with respect to the dominance rules $\prec$, $\prec_1$ and $\prec_2$ is stored in the respective archive. In this way, even if the individual is replaced in the population, it is not lost. However, new individuals created may dominate individuals in the archives with respect to the relevant dominance rule. In order to prevent this, the dominated solutions are removed from the archives after a number of generations, set by the decision maker.

5.4 Selection

In the selection step, a new generation is created and a set of individuals from the new generation are chosen as parent solutions for the recombination step.

A new generation P_{i+1} consists of the following:

1. M randomly chosen individuals from each archive,
2. $\frac{N-(Q+1)M}{2}$ best solutions with respect to fitness in P_i, where N is the population size, Q is the number of objectives and $(Q+1)M$ is the number of individuals included from the archives,
3. $\frac{N-(Q+1)M}{2}$ new child individuals from the recombination phase, to be described in Chapter 5.5.

The latter are created by first selecting $\frac{N-(Q+1)M}{2}$ parents from generation P_i by independent binary tournaments. In a binary tournament, two individuals are randomly selected from the population. The individual with the highest fitness will become a parent individual. Then, the parent individuals are paired. Finally, from each pair of parents a pair of child individuals is created.

5.5 Recombination and improvement

As a final and essential contribution to the new generation, $\frac{N-(Q+1)M}{2}$ new individuals are created by recombining the pairs of parent individuals. The recombination phase consists of two processes: crossover and mutation.

The chromosomes of two parent individuals are recombined to create two new child individuals. The crossover is a function $f(s_{p1}, s_{p2}) = s_{o1}, s_{o2}$. Here, the individuals s_{p1}, s_{p2} are the parent individuals, combined to create the child individuals s_{o1}, s_{o2}. If the parents are in generation number i, i.e., $s_{p1}, s_{p2} \in P_i$, then the child individuals are in generation number $(i+1)$, i.e., $s_{o1}, s_{o2} \in P_{i+1}$.

For each child, a mutation operator may be applied with a certain probability. A mutation function m makes a change in a single individual's chromosome: $m(s_{o1}) = s'_{o1}$. When the mutation function is applied, s_{o1} is no longer in the population, but is replaced by s'_{o1}.

The concrete operations of the recombination step are strongly problem dependent. For instance, what qualities the child individual aims to inherit from the parent individuals is decided by the problem to be solved and which operations that can be applied on individuals is confined by the encoding of the chromosomes. The crossover and mutations operators proposed in this thesis are explained in more detail in Chapter 6.5 and 6.6 respectively.

Finally, all the child individuals are improved, if possible, by a function: $i(s_{o1}) = s^*_{o1}$. How this function is defined, is explained in Chapter 6.7.

5.6 Stopping criterion

A maximum and a minimum number of generations for the search are set by the decision maker. The aim of the minimum number of generation is to prevent premature stopping of the algorithm before the Pareto is reached. The maximum number of generations prohibits the search to continue in infinite time. At the generations between the minimum and maximum number of generations, the search may stop due to a convergence criterion, determined by an error measure.

The error measure is defined as the area of the non-dominated domain, bounded by a given upper value for each of the objectives. The domain is illustrated in Figure 5.2. The area of the domain will decrease as long as new stricly non-dominated solutions are found. The search stops if the error measure is not changed in a given number of generations, also set by the decision maker.

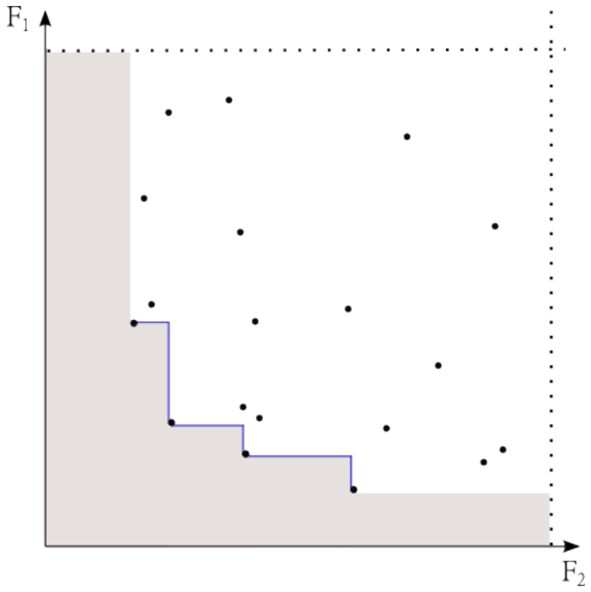


Figure 5.2: The area (grey) of the non-dominated domain. The upper values for each objective are illustrated by the dotted lines.

Chapter 6

Metaheuristic procedure

In this chapter, the parts of the genetic algorithm presented in Chapter 5 which are strongly dependent on the problem to be solved are presented. First, the encoding of the problem is given. Then, the proposed operations applied on the individuals are described.

6.1 A second look at the problem

We recall from Chapter 3.2 that a solution to the MCGRP is a routing plan, containing information about which task is assigned to which route and the tasks' ordering in the route.

The route problem can be divided into two problems: finding the optimal distribution of tasks among the vehicles and creating the optimal order of the given task in one route with respect to the set of objectives.

However, for instance when a solution to a delivery problem is applied, an employee would always choose the shortest path including the given set of tasks. This is consistent with the *route optimality criterion* discussed in Chapter 4.2. As a consequence, the second part of the problem is considered a single-objective problem; when the tasks are distributed among the vehicles, the route is decided as the most cost-efficient way to service all tasks. This includes both a cost optimal ordering of tasks, as well as requirs the path between the tasks to be the shortest path with respect to route cost.

Thus we can reformulate the problem as a two-stage problem:

1. Assign the tasks to different vehicles.
2. For each vehicle, given the assigned set of tasks, find the shortest route that serves all these tasks.

6.2 The network representation

The second part of the problem consists of solving a TSP for each vehicle. Due to the *route optimality criterion*, the path between two tasks is set to be the shortest path between the tasks. Hence, finding the shortest route servicing a set of tasks is equivalent to finding the best possible order of the tasks. In order to obtain this, different sequences of servicing the tasks are studied. Assume a sequence is proposed, the optimal route is simply the shortest path from one task to the next. The distance between two tasks is defined as follows:

1. The distance between two tasks located on nodes is the length of the shortest path from the first node to the second node.
2. The distance from a task located on a node n_1 to a task located on an arc a_1 is the length of the shortest path from the node n_1 to a_1's start node. For the opposite case, the distance is the length of the shortest path from a_1's end node to node n_1.
3. The distance from a task on arc a_1 to a task on arc a_2, is the length of the shortest path from a_1's end node to a_2's start node.
4. The tasks located on edges are handled as two tasks located on opposite directed arcs, referred to as α and β arcs, where only one of the two tasks needs to be serviced. The distances are calculated accordingly.

The length of the shortest path between two nodes is generated by the Floyd-Warshall algorithm.

However, finding an optimal TSP path when tasks are located on edges is not straight forward. If both possible traversal directions of a required edge are studied, this may cause an additional computational burden, as the number of possible solutions doubles for every task located on an edge.

Tasks located on edges
Instead of trying to service a task from either direction for each edge task in a route, the traversal direction of an edge task is already decided when assigning it in the first stage of the problem.

This is done by replacing every edge with two arcs in opposite directions, having the same cost and demand, but the criterion that just one of the arcs needs to be serviced. These arcs are called α and β arcs, illustrated in Figure 6.1. When an edge task is distributed to a vehicle, it is either the α arc or the β arc that is distributed. During the search, an α arc may be changed to a β arc and vice versa. This will be explained in more detail in Chapter 6.6.

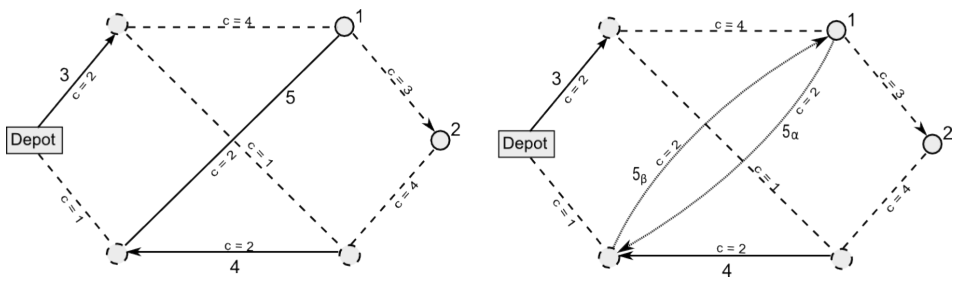


Figure 6.1: Replace the required edges with an α and a β arc.

We find it suitable to encode the network as a distance matrix, denoting the distance of the shortest path between any pair of tasks. The distance matrix for the network in Figure 6.1 is given as follows:

Tasks	1	2	3	4	5_α	5_β
1	0	3	3	5	0	2
2	8	0	7	4	8	6
3	4	5	0	1	4	3
4	2	5	1	0	2	0
5_α	2	5	1	4	0	2
5_β	0	5	3	5	2	0

Properties of the distance matrix worth noticing is that it may be asymmetric, which is a consequence of directed links in the network, and the zeros outside the diagonal is a result of calculating the distance from task t_m to t_n if

1. t_m is located on a node, and t_n is located on an outgoing arc from that node,

2. t_m is located on an arc, and t_n is located on the endnode of that arc, or

3. t_m and t_n are located on arc a_m and a_n respectively, and the endpoint of a_m if connected to the initial point of a_n.

The shortest paths from task 1 to 4 and from task 4 to 1 are illustrated in Figure 6.2.

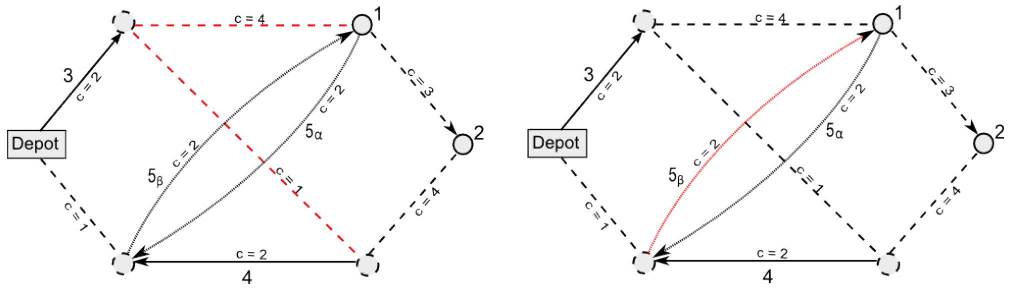


Figure 6.2: The shortest path from task 1 to task 4 (left) and from task 4 to task 1 (right).

According to the definition of the two-stage problem, it is sufficient to represent the individuals by:

1. which tasks are assigned to each route in the solution,
2. the ordering of the tasks within each route,
3. for each task located on an edge, define whether the α or the β arc is serviced.

With this information, the objective values for an individual can be evaluated immediately given the distance matrix and the service costs for the tasks.

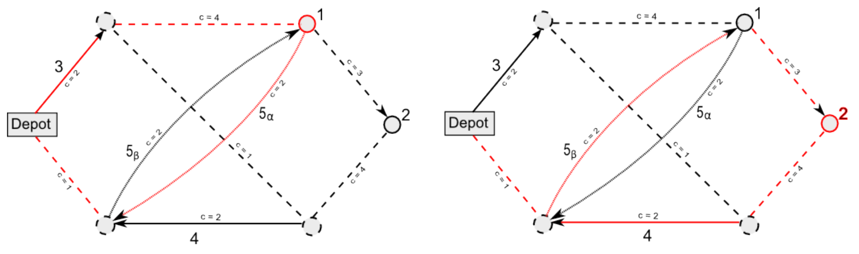
6.3 Chromosome encoding

As we have seen above, a solution, i.e., a routing plan, can be easily constructed from the distribution of the tasks to the different vehicles and the order in which the vehicles service their respective tasks. Hence, it is suitable to encode the chromosome as a list of routes, where a route is represented as a list of tasks in the order they appear in the route, encoded with their number. Using this encoding, there is an one-to-one correlation between the chromosome and the solution, hence a chromosome is unambiguous.

In this way, a chromosome defines a routing plan if and only if every task appear precisely once. But there is no guarantee that the routing plan is feasible according to the capacity constraint given this encoding. However, the operators proposed in the solution method are designed to ensure feasibility, by preventing actions resulting in a solution becoming infeasible. The operators are described in more detail in Chapter 6.5 and 6.6.

Given this encoding, operations such as extraction and deleting of tasks or routes are possible, as well as inserting tasks or routes if they are disjoint with the existing routes. But, some post-processing is usually necessary in order to obtain a valid routing plan, such as re-distribution of tasks for obtaining feasibility or re-ordering of tasks to fulfil the *route optimality criterion*.

A possible solution and its chromosome representation is illustrated in Figure 6.3. In this example, the tasks 1, 3 and 5_α are serviced in this order in the first route, and tasks 2 and 4 in the second route.



3	1	5_α
2	4	

Figure 6.3: Example of an individual representation

6.4 Initial population

An initial population of N individuals is created by a construction heuristic. The aim of the heuristic is to create a set of reasonable high quality individuals with high diversity.

To evaluate an individual, a scalarizing objective function is used, defined as

$$\widetilde{F}_\nu = \nu F_1 + (1-\nu)F_2. \tag{6.1}$$

Here, F_1 and F_2 are the objective values, and the variable ν is the weighting of the objectives. For each ν uniformly distributed in [0,1], an initial individual is constructed, evaluated with respect to $\widetilde{F}_\nu$.

When constructing an initial individual, tasks are chosen in random order among the tasks not yet assigned and inserted in the individual's chromosome. When a task located on an edge is chosen, the direction of servicing the edge, i.e., the α or β arc, is chosen randomly.

The task is placed in the first route in the chromosome that remains feasible after insertion of the task.

The placement of the task in the route is determined by $\widetilde{F}_\nu$. Here, the task is first tentatively inserted at all possible positions in the route and evaluated with respect to $\widetilde{F}_\nu$. The task is then inserted at the first position where $\widetilde{F}_\nu$ is minimized.

The procedure continues until there are no more tasks to assign.

Since the tasks are placed in random order, the distribution of tasks among the vehicles is a result of a stochastic process only. The ordering of the tasks, on the other hand, is a result of the scalarizing objective function. Here, the aim of an unique objective weight for each individual is to obtain a diversified population.

As a consequence of ordering the tasks based on the scalarizing objective function, the individuals may not fulfil the route optimality criterion discussed in Chapter 4.2. However, the main purpose of the initial population is to create a set of solutions with high diversity, and therefore this criterion is neglected for the initial population.

As a final remark; since tasks are only added to routes which remained feasible after inserting the task, and all tasks are added exactly once, all individuals in the initial population are feasible.

6.5 Crossover

In the crossover process, two parent individuals are combined and generate two child individuals. Two crossover operators are proposed in this thesis: `RBX` (Potvin and Bengio 1996, [40]) and a `Cross-and-Split` (Prins 2004, [41]). Which operator to use is chosen randomly, but for each pair of parents, the same crossover is applied to generate the two children.

RBX

The route based crossover (`RBX`) operator creates a child solution by extracting a set of routes from each parent individual. An example of the operator applied on two parent solutions are illustrated in Figure 6.4. The operator will be described in more detail in the following.

Firstly, a random number of routes is extracted from the chromosome of the first parent. The routes to be extracted are chosen randomly. This is illustrated in step 2 in Figure 6.4, and is represented as lines 3-6 in Algorithm 3.

To avoid duplicates of tasks when extracting parts from the chromosome of the second parent, the tasks currently constituting the child's chromosome are removed from the second parent's chromosome. Step 3 in Figure 6.4 gives an example of this operation, forming the lines 7-9 in Algorithm 3.

Finally, the non-empty routes in the second parent's chromosome are extracted and added to the child's chromosome. The result is illustrated in the final step in Figure 6.4.

For the second child individual created, the first parent individual become the second parent individual and vice versa.

Algorithm 3 `RBX`($Parent_1, Parent_2$)

```
1: $number\ of\ routes \leftarrow random\ number$
2: $Routes \leftarrow \emptyset$
3: for all $i = 1, ..., number\ of\ routes$ do
4:     $Routes \leftarrow$ add random route in $Parent_1$
5: end for
6: $Child \leftarrow Routes$
7: for all $task \in Routes$ do
8:     $Parent_2 \leftarrow Parent_2 \setminus \{task\}$
9: end for
10: $Parent_2 \leftarrow$ remove empty routes
11: for all $route \in Parent_2$ do
12:     $Child \leftarrow$ add $route$
13: end for
14: return $Child$
```

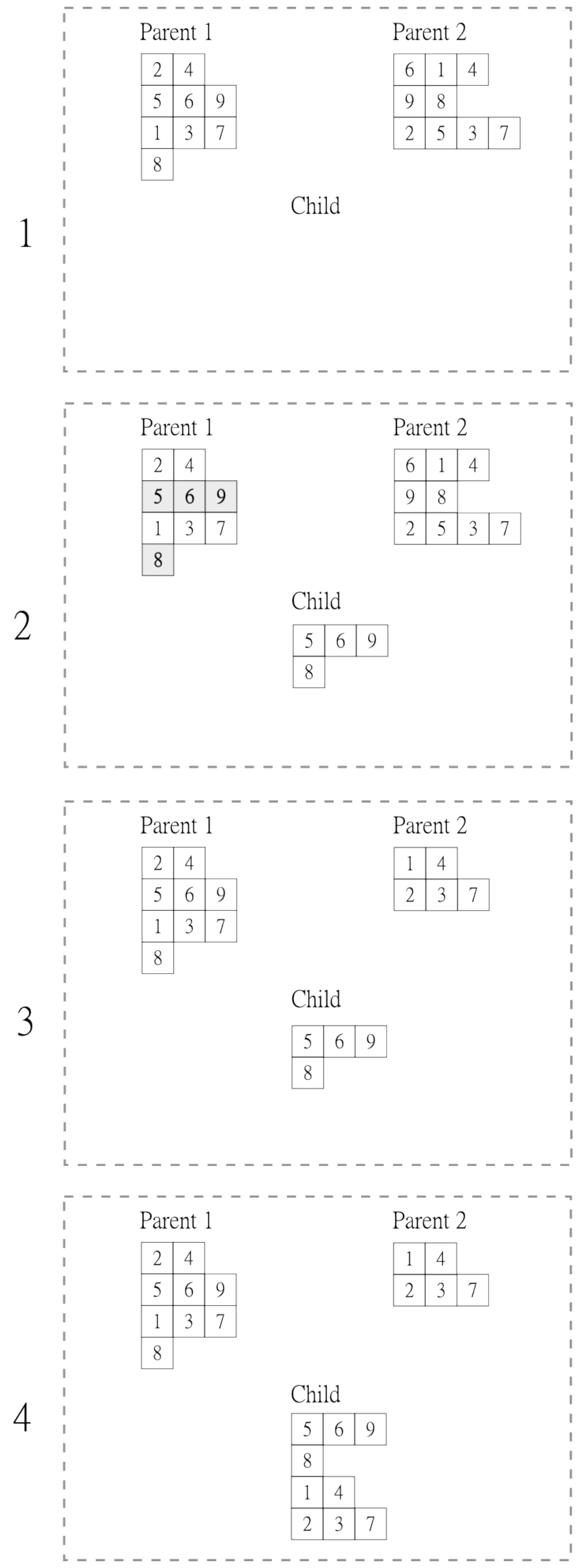


Figure 6.4: An example of the RBX crossover

Cross-and-Split

The `Cross-and-Split` operator creates a child individual by extracting a sequence of tasks from each parent individual. First, the routes in each parent chromosome are merged, and a sequence of both chromosomes is inherited by the child individual. Then, a tour splitting operator is applied on the child in order to split the single route into feasible routes. An example of the operator applied on two parent solutions are illustrated in Figure 6.6. The operator will be described in more detail in the following, and the pseudocode is given in Algorithm 4.

First, the routes in the chromosome for each of the parents are merged, such that each chromosome consists of a single route servicing all the tasks. An example is given in the second step in Figure 6.6.

To extract a sequence from the first parent's chromosome, two breaking points in the chromosome are randomly chosen. The sequence between the breaking points is extracted and added to the child chromosome, placed at the same position as it occurs in the parent's chromosome. This operation is illustrated in step 3 in Figure 6.6.

As for the RBX operator, the tasks which currently constitute the child's chromosome are removed from the second parent's chromosome, in order to avoid duplication of tasks.

The remaining tasks in the second parent's chromosome are added to the child's chromosome. These tasks are extracted in the ordering they occur in the parent chromosome, starting from a randomly chosen position. The tasks are inserted in the first available position in the child chromosome.

Now, as illustrated in the last step in Figure 6.6, the child chromosome contains all the tasks in a single route. However, the single route is most likely infeasible according to the capacity constraint and needs to be split into feasible routes. To find a cost efficient way to split the route without making a change in the task ordering, the chromosome is represented as a weighted directed acyclic graph (w-DAG), where the shortest path in the graph with respect to route cost defines where to split the routes.

An arc in the w-DAG corresponds to the route servicing the tasks defined by the intitial and ending point of the arc in the w-DAG. In the following t_i denotes the task at position i in the child's chromosome. Then, the w-DAG representing the chromosome is built in the following way:

1. For every task t_i in the chromosome, there is a corresponding arc a_i in the w-DAG. The endpoint of a_i is then connected to the initial point of a_{i+1}. The arc a_i corresponds to the route servicing task t_i only.
2. Additional arcs are added for every sequence of tasks $t_i, t_{i+1}, ..., t_k$ in the chromosome for which the total demand of the route given by $[t_i, ..., t_k]$ does not exceed the capacity C. These arcs connect the initial point of link a_i and the endpoint of a_k, and corresponds to the route given by the tasks $[t_i, ..., t_k]$, serviced in the given order.
3. The cost of an arc is the route cost of the route corresponding to the arc.

The single route in the chromosome is split into the routes corresponding to the arcs which constitute the shortest path in the w-DAG.

An example of a graph representing a large route, is illustrated in Figure 6.5. Here, the capacity is 10 and the tasks are serviced in the order $3-4-5^{\beta}-1-2$. The red links in the w-DAG represent the shortest path and split the route into three routes defined by the tasks $[3,4]$, $[5^{\beta}]$, and $[1,2]$.

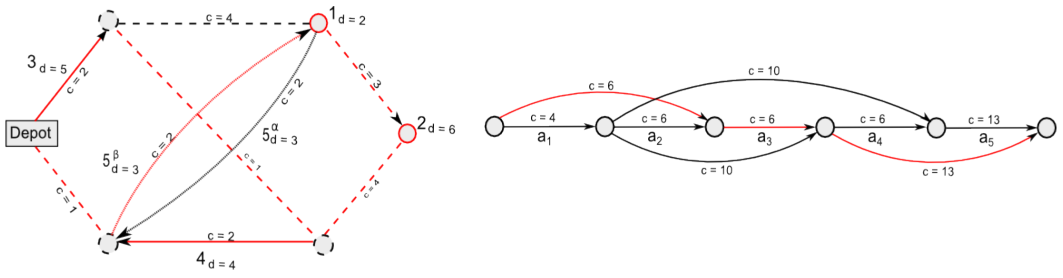


Figure 6.5: An example of how a route (left) is represented as a w-DAG (right). $C = 10$.

For the second child individual created, the first parent individual become the second parent individual and vice versa. The same splitting points in the chromosomes of the parents are used for both child solutions.

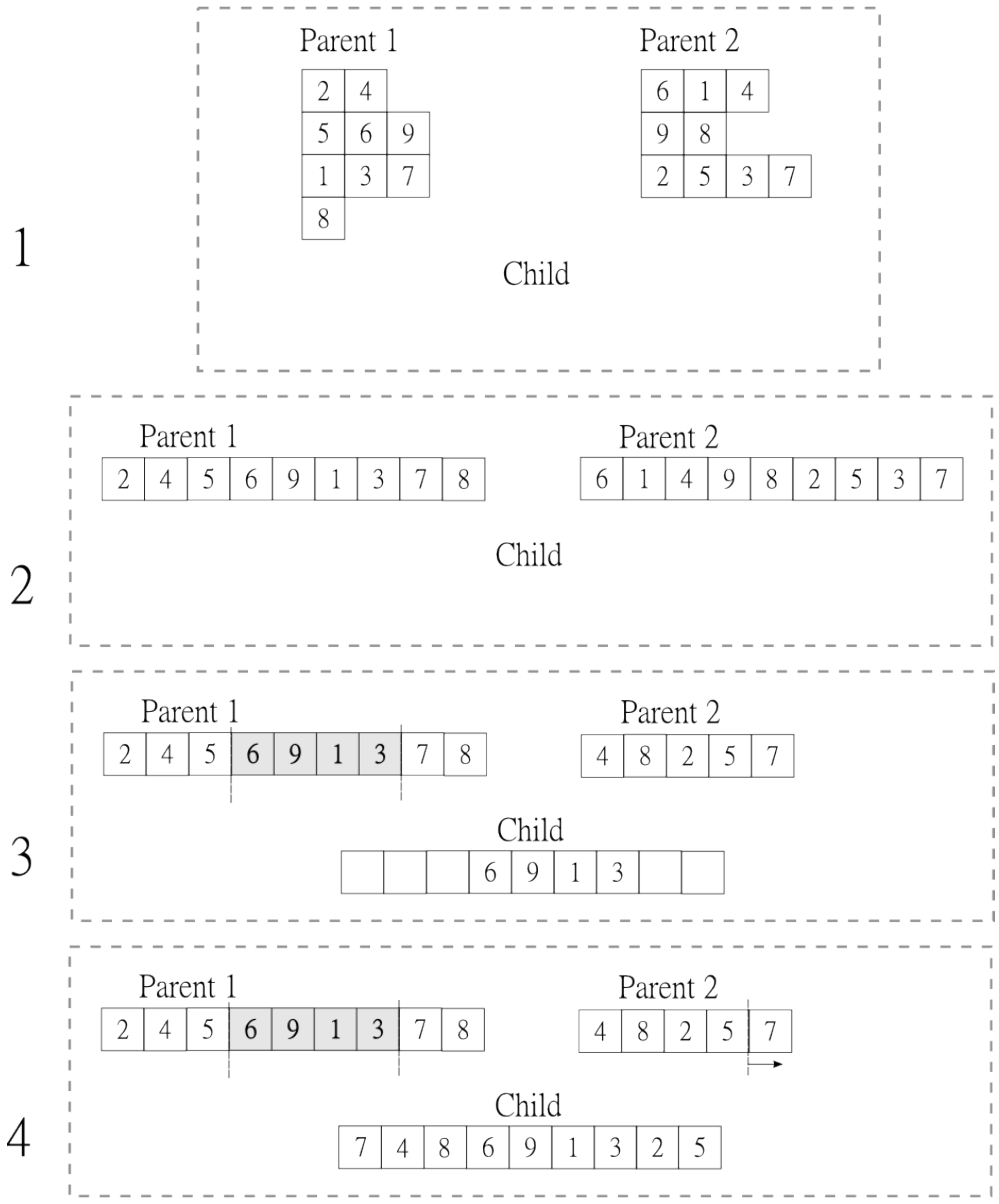


Figure 6.6: An example of the Cross-and-Split crossover

Algorithm 4 `Cross-and-Split`($Parent_1, Parent_2$)

```
1: mergeRoutes($Parent_1, Parent_2$)
2: $i, j \leftarrow$ random indices in $Parent_1$
3: $S \leftarrow \emptyset$
4: $S$ add $Parent_1[i \rightarrow j]$
5: $Child[i \rightarrow j] \leftarrow S$
6: for all $task \in S$ do
7:     $Parent_2 \rightarrow$ remove task
8: end for
9: $k \leftarrow$ random index in $Parent_2$
10: $l_{P_1} \leftarrow$ length of $Parent_1$
11: $l_{P_2} \leftarrow$ length of $Parent_2$
12: for all $index \leq$ size of $Parent_2$ do
13:     $Child[(j+index+1)\ modulo\ l_{P_1}] = Parent_2[(k+index)\ modulo\ l_{P_2}]$
14: end for
15: seperateRoute($Child$ )
16: return $Child$
```

6.6 Mutation

After the crossover operator is applied, one of several mutation operators will be applied with a given probability. In this thesis, five mutation operators are applied to improve individuals: two simple operators, one large neighorhood search, a cross exchange and a greedy approach.

Simple mutation operators
Two simple mutation operators are implemented: *Swap tasks* and *Swap edge.*

The swap task operator swaps two randomly chosen tasks between two randomly chosen, distinct routes in a solution such that it remains feasible after the swap of tasks. A task is reinserted at the position from where the other task was removed.

The swap edge operator randomly chooses an edge in one of the routes in the solution and changes the traversal direction. For the chosen edge, this means the α arc is transformed into a β arc and vice versa. Since the edge remains at the same position, feasibility is not an issue.

An Adaptive Large Neighborhood Search
An Adaptive Large Neighborhood Search (ALNS), inspired by Pisinger and Ropke (2010, [42]), is introduced with the aim of making more radical changes in the solution by replacing individual tasks in different routes in an iterative process. At each iteration, a neighborhood of the current solution is constructed. For each neighbor constructed, a destroy operator is first applied on the current solution to remove a set of tasks. Then, a repair operator is applied to reinsert the tasks.

If an improvement with respect to route cost is found in the neighborhood, the search continues with the solution in the neighborhood with best route cost objective value. Else, the search stops and returns the last improved solution.

Destroy operators:
The destroy operators choose which tasks to be removed from the solution. The highest number of tasks to be removed is decided by the decision maker. As the iteration number increases, the number of tasks to be removed reduces with a degree decided by the decision maker. The following destroy operators are applied:

- `Delete random`: The tasks to be replaced are chosen randomly among all routes in the solution.
- `Delete max cost`: The tasks are chosen randomly among the 2σ tasks with the highest service cost, where σ is the number of tasks to be replaced.
- `Delete max reduction in cost`: The tasks are chosen as the tasks with the highest reduction in total route cost when removed from the solution.

Repair operator:
After a destroy operator is applied, a repair operator is applied to reinsert the tasks. The following repair operators are proposed:

- `Repair Hamilton Cycle`: For each route that would remain feasible after insertion of the task and each position in that route, the cost of placing the task at that point is computed. The task is placed at a position that would result in the minimal increase in total route cost. The tasks are inserted in descending order with respect to demand. In this way, conflicts due to infeasible routes are reduced to a minimum.

- `Repair fast insert`: A task is inserted in the shortest route that would remain feasible after insertion. As for the `Repair Hamilton Cycle` operator, the task is placed at the most cost-efficient position in the route and the task are inserted in descending order with respect to demand.

If the load of the vehicles is close to their capacity, it is possible that the tasks are inserted in the routes in such a way that no routes remain feasible before the final task is inserted. When that happens, the neighbor solution is not added in the neighborhood.

On each neighbor to construct, one destroy and one repair operator is applied. The probability of an operator to be chosen is affected by the success earlier in the search. The success rate of the i^{th} destroy operator is denoted as ρ_i^- and ρ_i^+ is the success rate of the i^{th} repair operator. When a neighbor is created and added to the neighborhood, the values are updated in the following way:

$$\rho_i^- = \lambda^- \rho_i^- + (1 - \lambda^-)\Phi \tag{6.2}$$

and

$$\rho_i^+ = \lambda^+ \rho_i^+ + (1 - \lambda^+)\Phi, \tag{6.3}$$

where $\lambda^-, \lambda^+ \in [0, 1]$ are decay parameters controlling the change in ρ_i^- and ρ_i^+. Φ is assigned the highest value of the following parameters:

- ω_4 if the solution is feasible and the best solution in the neighborhood,
- ω_3 if the solution is feasible and improved compared to current solution,
- ω_2 if the solution is feasible
- ω_1 if no feasible solutions are found.

Here, $\omega_4 \geq \omega_3 \geq \omega_2 \geq \omega_1$ are assigned values by the decision maker. Hence, operators which have generated good solutions have higher probability to be chosen than operators generating worse solutions or none at all.

Cross

In the `Cross` operator, sequences of tasks are swapped between two routes in an individual in an iterative process. At each iteration, the last part of the longest route is swapped with the last part of one of the other routes in the same individual. The search continues as long as an improvement with respect to the route balance objective is found.

The longest route is defined as the route in the current solution with the highest route cost. The length of the sequence of tasks to be removed from the longest route is chosen randomly between 0 and half the number of tasks in this route.

The other route included in the swap is decided by the route balance objective. Here, the route which causes the best route balance value as a result of the swap of tasks is chosen.

However, the length of the sequence to be removed from this route and inserted in the end of the longest route is decided by the route cost objective. Here, the most cost-efficient swap in terms of different lengths of the sequence is chosen, given all routes remain feasible.

Greedy swap of task

This mutation operator uses a greedy approach of moving tasks with the aim of improving the route balance objective. The idea is to iteratively move tasks from the longest route to the most cost-optimal route as long as an improvement is found.

At each iteration, a randomly chosen task is removed from the longest route.

The task is inserted by the `Repair Hamilton Cycle` operator in the ALNS, i.e., it is inserted in the route with lowest increased total route cost after insertion, when the task is placed at the most cost-efficient position in the route.

To evaluate the individual by the correct objective values, a `2-OPT` TSP-solver is applied on both routes where a change is made.

If the value of the route balance objective is improved, the search continues. Otherwise, the last improved individual is returned.

The pseudocode is given in Algorithm 5. Here, we assume the route balance objective to be the second objective function, i.e. $F_2(\cdot)$.

Algorithm 5 `Greedy swap of task`(*Child*)

```
1: $improved \leftarrow$ TRUE
2: while $improved$ do
3:     $s_{temp} \leftarrow$ copy $Child$
4:     $r_l \leftarrow$ longest route in $s_{temp}$
5:     $task \leftarrow$ random task in $r_l$
6:     $IncreaseCost \leftarrow \emptyset$
7:     for all $route$ in $s_{temp}$, except $r_l$, do
8:         $IncreaseCost(route) \leftarrow$ the increased cost if $task$ is included in $route$
9:     end for
10:    $r_{min} \leftarrow$ the $route$ with lowest $IncreaseCost$
11:    $r_{min}$ in $s_{temp} \leftarrow$ add $task$
12:    2-OPT$(r_l, r_{min})$
13:    if $F_2(s_{temp}) \leq F_2(Child)$ then
14:        $Child \leftarrow s_{temp}$
15:    else
16:        $improved \leftarrow$ FALSE
17:    end if
18: end while
19: return $Child$
```

6.7 Improvement

The improvement function in this solution method, is the heuristic TSP solver *2-OPT* (Croes, 1958, [43]).

The function is applied on every route individually in the child's chromosome. The ordering of tasks in a route is changed if the function finds a new ordering with lower route cost than the original route. As well as improving one objective value, the main purpose of this function is to fulfil the *route optimality criterion* discussed in Chapter 4.2, which is the reason why the route balance objective is neglected.

As a result of using a heuristic TSP solver, the ordering of tasks is not guaranteed to be optimal. But since the solution method proposed in this thesis is a heuristic, we define it is sufficient to fulfil the *route optimality criterion* if the ordering of tasks is determined by the value of route cost only.

Ideally, the function would be applied on every child. However, the function is relatively time consuming compared to rest of the method. Hence, this TSP solver is applied on every new child individual with a generation frequency, set by the decision maker.

Chapter 7

Computational study

The proposed solution method is applied on the bi-objective variant of the MCGRP, where the total route cost and route balanced is optimized. The route cost objective is defined in equation (4.3). The route balance objective is formulated as minimizing the length difference between the longest and shortest route in the routing plan, defined in equation (4.5).

How the proposed solution method is implemented and information about the instances studied are explained in Chapter 7.1. The settings for the parameters introduced in Chapter 5 and 6 are given in Chapter 7.2. Finally, the results are presented in Chapter 7.3 and discussed in Chapter 7.4.

7.1 Implementation and instances

The proposed solution method was coded in C++. The computational experiments were conducted on a PC with 2.67 GHz 4×6-core Xeon 24 Intel CPUs.

The evaluation was conducted on the 23 CBMix instances, from Prins and Bouchenoua (2005, [15]). These are the most studied instances for the MCGRP. For the CBMix benchmark instances, the number of nodes varies from 11 to 150, and number of tasks from 20 to 212. The problem size for each instance is given in Table 7.1. The number of vehicles is not fixed, and for each solution the number of vehicles is optimized in terms of the objectives.

Table 7.1: The instances

Instance	Nodes	Links	Tasks
CBMix1	21	66	48
CBMix2	68	246	185
CBMix3	31	88	79
CBMix4	53	111	98
CBMix5	32	61	65
CBMix6	49	96	108
CBMix7	75	158	168
CBMix8	77	168	177
CBMix9	29	47	50
CBMix10	56	110	107
CBMix11	69	235	82
CBMix12	38	71	53
CBMix13	150	292	141
CBMix14	94	332	93
CBMix15	52	91	91
CBMix16	71	138	169
CBMix17	42	118	63
CBMix18	117	212	127
CBMix19	126	302	212
CBMix20	43	131	73
CBMix21	60	138	180
CBMix22	25	58	42
CBMix23	11	27	20

Service cost was added to the required entities for the instances. Since the service cost is a constant term in the total route cost for a solution, this term is usually neglected when solving a single-objective minimize cost problem. However, when considering route balance, the service costs influence the solutions to a great extent, and need to be considered.

The quality of the route cost objective function value is evaluated by the distance to the best known solution of the single-objective problem where only route cost is considered. Here, the service cost is neglected.

The CBMix instances and the best known solutions are found on the MCGRP subpage to SINTEF's TOP webpage ([44]), where the best known results of the CBMix instances from the studies by Prins and Bouchenoua (2005, [15]), Kokubugata et al. (2007, [16]), Hasle et al. (2012, [17]) and Bosco et al. (2013, [20]) are presented.

Each instance was solved 30 times. A number of instances were additionally solved with a stricter convergence criterion, resulting a higher running time.

7.1.1 Assumptions for the input data

The following assumptions about the input data are made:

1. The network cannot have any links with negative cost, i.e. infinite cycles to reduce cost cannot occur.

2. The network is not a multi-graph, i.e. there is no parallel links. This is to avoid implementation issues when different objectives do not prefer the same link in a pair of parallel links.

7.1.2 Neglecting service cost

An interesting observation made while testing the proposed solution method is that the service cost had a great impact on the operators' actions which are affected by the route cost. The reason for this is probably that the service cost dominates the decision of determining the most cost efficient route to remove or add a task. By neglecting the service cost for these operators, the quality of both objectives were improved. The service cost was neglected in the following operations:

- Crossover `Cross-and-Split`: In the search for the shortest path of the w-DAG.
- Mutation `ALNS`: In the destroy operator `Delete max reduce in cost` and both the repair operators `Repair Hamilton Cycle` and `Repair fast insert`.

7.2 Parameter settings

The population consists of 80 individuals. At each generation, 8 randomly chosen solutions are added to the population from each of the three archives A_0, A_1, and A_2 stroring the non-dominated individuals with respect to the dominance rules described in Chapter 5.3. At every fifth generation, the TSP solver 2-OPT is applied on every route in all the child solutions.

Fitness
The solution method was experimentally tuned, but we observed no great variation of the results by changing in the values of the parameters. The parameters defined in the fitness value given in equation (5.1) and (5.4) are assigned the following values:

Parameter	Notation	Value
Selection pressure	S	1.7
Sharing distance	γ	0.01

Crossover and mutation
The probability for the operators to be applied is decided as follows:

- Each of the two crossover operators is applied with a probability of 50%.
- For each child solution, a mutation operator is applied with a probability of 50%.
- Each of the five mutation operators is chosen with equal probability, i.e., with a probability of 20%.

ALNS
At the beginning of the search, 15% of the tasks are to be removed and reinserted. At each iteration, the number of tasks to be removed by a destroy operator is reduced with 5%.

The following success rate and decay parameters were used:

Parameter	Notation	Value
First success rate	ω_1	0
Second success rate	ω_2	$\frac{1}{3}$
Third success rate	ω_3	$\frac{2}{3}$
Fourth success rate	ω_4	1
Repair decay parameter	λ^+	0.7
Destroy decay parameter	λ^-	0.5

Stopping criterion
The minimum and maximum number of generations were set to 2000 and 25000 respectively.

After the minimum number of generations is reached, the search stops when the area of the non-dominated domain is unchanged for 100 consecutive generations.

7.3 Results

Since this is the first study on multi-objective MCGRP, there exists no work which can be used to compare the results from the solution method proposed in this thesis. Hence, it is difficult to evaluate the performance of the method. The method is evaluated with respect to the diversity of the potential Pareto front and the quality of the objectives. The qualities of the objectives are compared with the solutions to the single-objective MCGRP. We are also interested to see whether the objectives are conflicting. The objectives are conflicting if improvement in one objective requires a reduction in the quality of the other objective, thus both objectives cannot be simultaneously optimized. At last, the solution method is evaluated by its consistency and running time.

The instances CBMix5, CBMix13, CBMix17, CBMix19, and CBMix21 were in addition solved with a stricter stopping criterion. Here, the minimum and maximum number of generations were 10 000 and 100 000 respectively, the convergence criterion was applied when the area of the non-dominated domain is unchanged for 500 consecutive generations, and the population size was increased to 150. The results are presented in Appendix A, by a plot of the potential Pareto front and all the non-dominated solutions found by every 500th generations for both a standard and a longer run. Additionally, the objective values for the solutions in the potential Pareto front are given as a list.

In the results presented below, the service cost is excluded from the value of the cost objective. Thus the value can then be compared with the best known solution for each instance. However, the service cost is included when evaluating in terms of the route balance objective.

The potential Pareto fronts of the instances had different shapes; *plateau, curved*, and potential Pareto fronts with *corner points*. In Figure 7.1 to 7.6, potential Pareto fronts illustrating the different shapes are presented. For each case, the first figure is the union of the final set of non-dominated solutions for all 30 runs of the corresponding instance, which most likely gives the best picture of the behaviour of the Pareto front. The second figure illustrates two sample potential Pareto fronts found by the solution method for the same instance. The total route cost of the best known solution to the single-objective minimizing cost MCGRP is represented as a dotted line in the figures.

We want to make the reader awair of that the x-axis, which denotes the total route cost, does not start at zero in the following figures.

Plateau
The shape of the Pareto front contains a plateau if there is a saddle point in it.

For Pareto fronts of this shape, there is an interval where one objective can be improved without requiring a reduction in the other objective. Hence, for an interval for one objective value, the objectives are not conflicting. In this interval, it is necessary with a high density of solutions in the potential Pareto front to obtain a good approximation of the optimal Pareto front, and preclude the case if it is a part of the potential Pareto front dominating an area where it is hard to find new solutions, instead of an actual plateau.

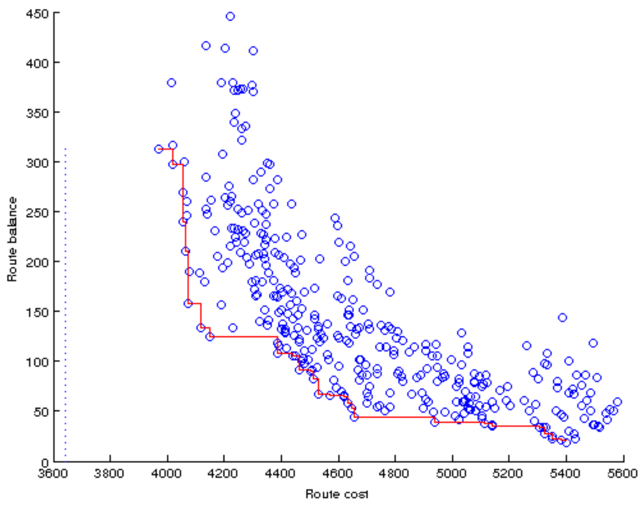


Figure 7.1: The potential Pareto front of CBMix3 has a plateau. The total route cost of the best known solution is 3643.

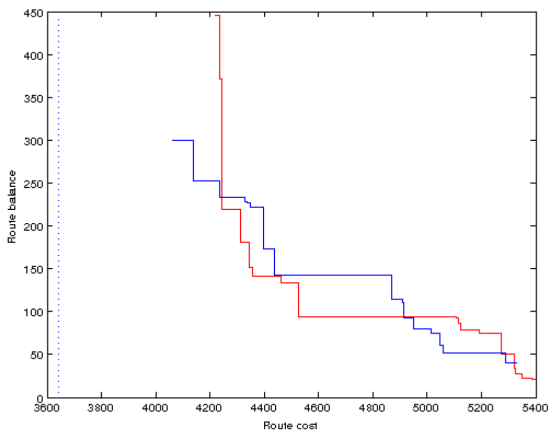


Figure 7.2: Two sample potential Pareto fronts of CBMix3

Curved

The Pareto front is gently curved if the slope of the tangent to the polynomial interpolation of the non-dominated solutions is not-increasing and there is no single solution where both objectives are optimized simultaneously.

For a *gently curved* Pareto front, an improvement of a high quality objective value requires a higher reduction in the other objective value than the same improvement in a objective value when it is of lower quality. Pareto fronts of this shape require many and well spread solutions in the potential Pareto front to obtain a good approximation.

The objectives are conflicting for all objective values in a *gently curved* Pareto front, in contrast to the Pareto fronts containing a plateau.

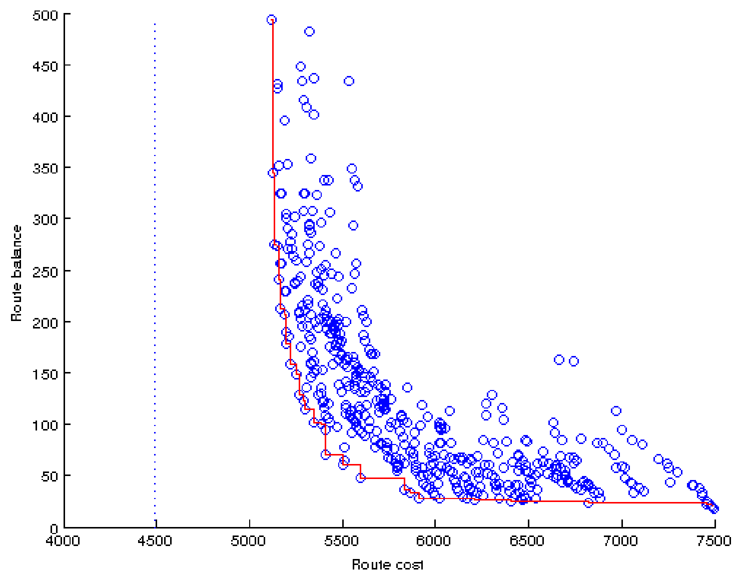


Figure 7.3: The potential Pareto front of CBMix11 is gently curved. The total route cost of the best known solution is 4494.

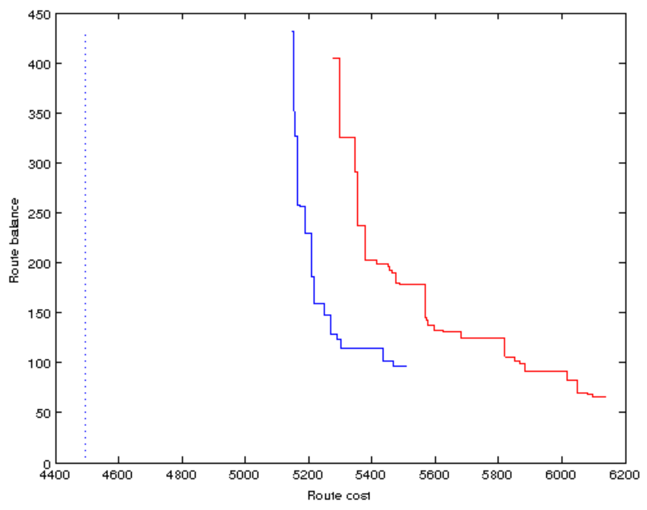


Figure 7.4: Two sample potential Pareto fronts of CBMix11

Corner point

If the Pareto front consists of one almost vertical and one almost horizontal line, meeting at a corner solution of high quality, the Pareto front is said to have a *corner point* shape.

Here, one objective may be improved, without requiring much reduction in quality of the other objective. The objectives are in this case less conflicting than for the Pareto fronts that are *plateau* and *gently curved* shaped.

A solution among the corner point(s) is a natural choice as a single solution of the problem.

Since the Pareto front consists mostly of straight lines between the non-dominated solutions, Pareto fronts of this shape requires a lower density of solutions in the potential Pareto front to obtain a good approximation of the Pareto front compared to the two other shapes.

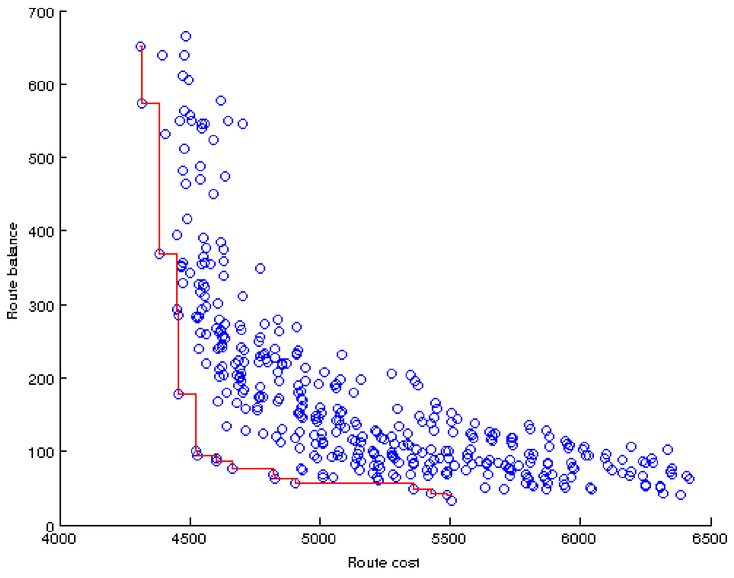


Figure 7.5: The potential Pareto front of CBMix17 has a *corner point* shape. The total route cost of the best known solution is 4037.

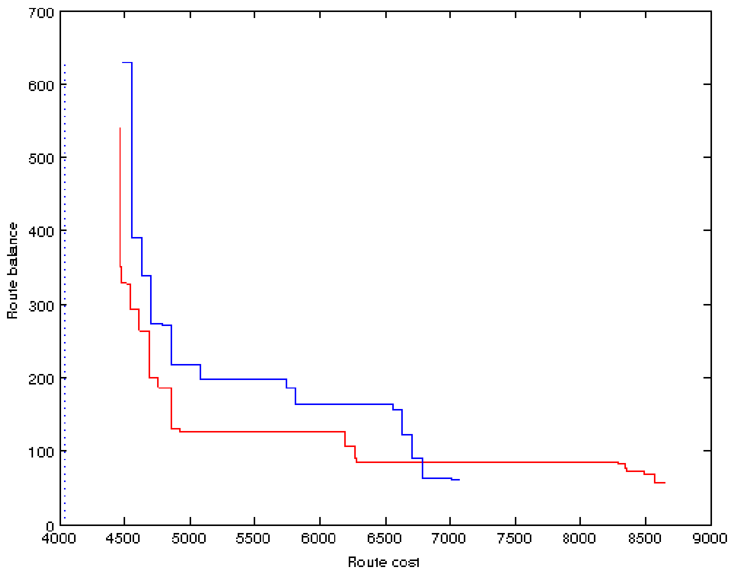


Figure 7.6: Two sample potential Pareto fronts of CBMix17

7.3.1 Diversity

The diversity of the potential Pareto front is evaluated by a quality measure b_p, which is the average distance from every integer point on the potential Pareto front to the nearest solution, weighted by the relative importance of having a close solution to that point at the potential Pareto front. The measure b_p is between 0 and 1, and low values are favoured. It is defined as

$$b_p = \sum_{m_j \in P_f} \frac{d_{m_j}}{|P_F|} w_{m_j}, \tag{7.1}$$

where P_f is the set of points in the potential Pareto front with integer objective values and m_j is such a point on the potential Pareto front located between the non-dominated solutions s_j and s_{j+1}. The distance d_{m_j} is the Manhattan distance from point m_j to the nearest solution, weighted by w_{m_j}. The value of w_{m_j} is defined in the following.

The importance of finding a close solution to a point m_j is given by the area of the rectangle defined by the s_j in the left, upper corner and the s_{j+1} in the right, lower corner. Hence, relative importance w_{m_j} is defined as

$$w_{m_j} = \frac{D_j}{\sum_{l=1,..,|A_0|-1} D_l}. \tag{7.2}$$

Here, D_j is the the area defined by the solutions s_j and s_{j+1} and A_0 is the set of the non-dominated solutions, i.e., the solutions located on the potential Pareto front. The distance d_{m_j} and the area D_j are illustrated in Figure 7.7.

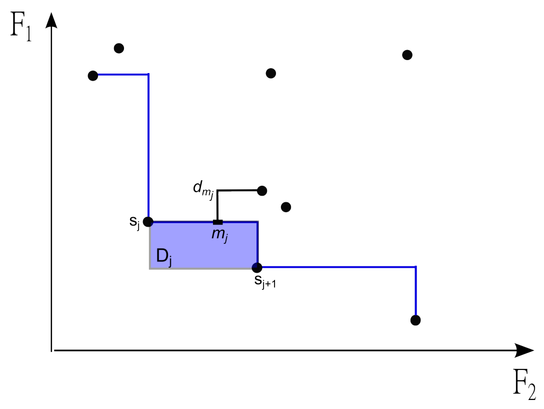


Figure 7.7: The distance d_{m_j} from point m_j in the potential Pareto front to the nearest solution, weighted by the area D_j (blue).

The reason why the importance of a close point is determined by the area D_j is that the desire for a high density of solutions in an area depends on the shape of the potential Pareto front. A high density of solutions is more important in the parts of the potential Pareto front where the improvement in one objective value requires a large reduction of the other objective value (large area), than in parts of the potential Pareto front where one objective is improving while the other is unchanged (small area). This is because we want detailed information about how much reduction is required in an objective value to achieve an improvement in the other objective value. If there are many close solutions, the potential Pareto front between the two non-dominated solutions is a good representation of possible solutions to the problem. On the other hand, if there are no close solutions, a long distance between the two non-dominated solutions can be due to poor diversification properties of the solution method, and the potential Pareto front may be unreliable. It may also be that no solutions exist in this area, as a consequence of solving an integer problem.

The solutions studied in the calculation of b_p, are the solutions in the potential Pareto front and the union of the set of non-dominated solutions found by every 500th generation. The sets of non-dominated solutions for a consecutive number of generations are usually not disjoint, hence only the union of non-dominated solution set for some generations is needed. We found it appropiate to use the set of non-dominated solutions by every 500th generation as a good representation for the total set of solutions found by the solution method.

Table 7.2 gives the best and average value of b_p over 30 runs for each instance.

7.3.2 The quality of the objectives

The quality of the objective function values is measured for each instance by the following values:

1. The total route cost of the minimal total route cost objective found in 30 runs.
2. The relative gap between the minimal total route cost and the total route cost in the best known solution.
3. The average total route cost of the minimal total route cost objective found in 30 runs.
4. The relative gap between the average minimal total route cost and the total route cost in the best known solution.

5. The minimal route balance objective value found in 30 runs, divided by the total route cost in the best known solution.

6. The average minimal route balance objective value found in 30 runs, divided by the total route cost in the best known solution.

Table 7.3 shows the results of the evaluation of the quality of the objectives for all the 23 instances in the CBMix benchmark. Best known value marked with $*$ are known to be optimal.

Table 7.2: The b_p-value for the CBMix instances.

Instance	Best b_p	Average b_p
CBMix1	0.0048	0.0192
CBMix2	0.0051	0.0280
CBMix3	0.0057	0.0187
CBMix4	0.0022	0.0148
CBMix5	0.0034	0.0170
CBMix6	0.0029	0.0183
CBMix7	0.0038	0.0249
CBMix8	0.0034	0.0289
CBMix9	0.0030	0.0131
CBMix10	0.0036	0.0173
CBMix11	0.0022	0.0167
CBMix12	0.0019	0.0306
CBMix13	0.0037	0.0188
CBMix14	0.0029	0.0164
CBMix15	0.0012	0.0210
CBMix16	0.0024	0.0263
CBMix17	0.0037	0.0147
CBMix18	0.0057	0.0210
CBMix19	0.0023	0.0277
CBMix20	0.0042	0.0159
CBMix21	0.0028	0.0287
CBMix22	0.0056	0.0147
CBMix23	0.0014	0.0359

Table 7.3: The quality of the solutions.

Instance	Best known	Minimal cost	Gap [%]	Average cost	Gap [%]	Optimal balance [%]	Average balance [%]
CBMix1	2589	2631	1.62	2737	5.72	0.39	0.79
CBMix2	12220	13757	12.58	14090	15.30	0.47	0.75
CBMix3	3643	3971	9.00	4215	15.71	0.52	1.35
CBMix4	7583	8857	16.80	9330	23.04	0.95	1.63
CBMix5	4531	4879	7.68	5103	12.63	0.51	1.05
CBMix6	7087	7749	9.34	7960	12.32	0.62	0.92
CBMix7	9607	10783	12.24	11112	15.67	0.60	0.88
CBMix8	10524	11928	13.34	12442	18.23	0.71	1.09
CBMix9	4038	4298	6.44	4482	10.98	0.62	1.23
CBMix10	7582	8741	15.29	9025	19.03	1.00	1.77
CBMix11	4494	5095	13.37	5321	18.40	0.36	0.87
CBMix12	3138*	3235	3.09	3271	4.25	0.16	1.02
CBMix13	9110	10195	11.91	10642	16.82	0.90	1.37
CBMix14	8566	9452	10.34	9812	14.54	0.54	0.99
CBMix15	8280	8960	8.21	9409	13.63	1.15	2.36
CBMix16	8886	9817	10.48	10178	14.54	0.81	1.52
CBMix17	4037	4309	6.74	4528	12.17	0.82	1.49
CBMix18	7089	7675	8.27	7973	12.48	0.92	1.31
CBMix19	16347	19075	16.69	19587	19.82	0.46	0.73
CBMix20	4844	5297	9.35	5533	14.23	0.39	0.90
CBMix21	18069	21143	17.01	21987	21.68	0.79	1.24
CBMix22	1941	**1941**	0.00	2046	5.43	0.21	0.66
CBMix23	780*	**780**	0.00	781	0.17	0.13	0.51

7.3.3 Related objectives

To study whether the objectives are conflicting, the minimal total route cost and the associated route balance objective value are compared with the minimal route balance objective value and the associated total route cost found in 30 runs, for each instance.

Results are provided in Table 7.4.

Table 7.4: The total route cost and route balance associated with the routes of minimal total route cost and optimal balance found in the search. Relative balance is the value of the route balance objective function value relative to the associated minimal total route cost.

Instance	Cost optimality	Balance	Relative balance [%]	Cost	Balance optimality
CBMix1	2631	273	10.38	3416	10
CBMix2	13757	372	2.70	14834	58
CBMix3	3971	313	7.88	5399	19
CBMix4	8857	707	7.98	10864	72
CBMix5	4879	2113	43.31	7416	23
CBMix6	7749	407	5.25	8283	44
CBMix7	10783	587	5.44	12555	58
CBMix8	11928	316	2.65	14882	75
CBMix9	4298	488	11.35	5066	25
CBMix10	8741	350	4.00	9931	76
CBMix11	5095	643	12.62	7502	16
CBMix12	3235	59	1.82	4131	5
CBMix13	10195	436	4.28	13462	82
CBMix14	9452	643	6.80	11651	46
CBMix15	8960	497	5.55	11707	95
CBMix16	9817	557	5.67	12652	72
CBMix17	4309	651	15.11	5503	33
CBMix18	7675	365	4.76	13162	65
CBMix19	19075	527	2.76	20606	75
CBMix20	5297	674	12.72	6183	19
CBMix21	21143	801	3.79	24945	143
CBMix22	1941	139	7.16	2570	4
CBMix23	780	42	5.38	947	1

7.3.4 Consistency

To evaluate the consistency of the search, the variance in the non-dominated area A and the variance of b_p (see Chapter 7.3.1) in 30 runs are studied for each instance. The first measure indicates the consistency of the quality of the solutions, and the latter indicates the consistency of diversity.

Here, the objective values are normalized by the 1-norm and the reference point for calculating non-dominated area is determined by the highest value found for each objective in the 30 runs.

The results are presented in Table 7.5.

Table 7.5: Consistency of the solution method

Instance	Variance in A	Variance in b_P
CBMix1	1.741871e-03	3.917344e-04
CBMix2	5.598549e-04	8.811634e-04
CBMix3	4.944684e-04	3.319523e-04
CBMix4	4.503000e-04	4.316225e-04
CBMix5	9.402769e-04	1.368447e-03
CBMix6	4.546529e-04	7.387032e-04
CBMix7	1.298392e-03	6.837383e-04
CBMix8	1.340938e-03	7.384597e-04
CBMix9	6.099346e-04	3.142817e-04
CBMix10	3.941335e-04	4.043126e-04
CBMix11	3.815741e-04	4.699770e-04
CBMix12	4.098985e-03	3.919886e-03
CBMix13	6.129167e-04	5.165376e-04
CBMix14	2.306751e-04	7.095727e-04
CBMix15	1.570979e-03	1.809534e-03
CBMix16	4.147505e-04	2.496749e-03
CBMix17	4.192875e-04	3.014201e-04
CBMix18	6.023753e-04	6.353965e-04
CBMix19	4.019810e-04	1.009852e-03
CBMix20	3.176083e-04	4.440617e-04
CBMix21	1.906364e-03	1.507752e-03
CBMix22	4.449079e-04	3.887032e-04
CBMix23	9.400558e-04	1.022579e-03

7.3.5 Running time

Finally, we consider how the search converges and the CPU time required, and evaluate the following:

1. Number of runs stopped because the maximum number of iterations was reached.
2. The average number of iterations required for the search to finish.
3. The average time required for the search to finish.

Table 7.6 shows the results of the evaluation of convergence and running time, for all the 23 instances in the CBMix benchmark.

Table 7.6: Convergence, number of iterations, and running time

Instance	Non-converged	Iterations required	Average time [s]
CBMix1	0	2281.9	209.46
CBMix2	0	2306.9	1870.43
CBMix3	0	2823.8	429.41
CBMix4	0	3187.0	771.15
CBMix5	0	3272.6	353.39
CBMix6	0	2479.9	614.38
CBMix7	0	2893.9	1417.52
CBMix8	0	2325.5	1317.65
CBMix9	0	2349.7	182.56
CBMix10	0	2306.4	623.83
CBMix11	0	2762.9	455.47
CBMix12	4	4849.3	823.23
CBMix13	0	2169.8	834.12
CBMix14	0	2391.9	575.75
CBMix15	0	2770.3	640.82
CBMix16	0	2695.7	1601.13
CBMix17	0	2683.1	332.09
CBMix18	0	2557.7	1036.35
CBMix19	0	2536.1	2087.65
CBMix20	0	2360.2	293.30
CBMix21	0	2389.3	1494.15
CBMix22	0	2946.1	170.62
CBMix23	5	9415.2	351.33

7.4 Discussion

The distance between the solutions in the potential Pareto front differs among the instances. For example, the solutions in the potential Pareto front of instance CBMix17 in Figure A.4 was found to be well diversified. The non-dominated solutions found for the CBMix19 instance on the other hand, are not as well spread out along the potential Pareto front, indicated by potential Pareto fronts presented in Figure A.5. Here, the density of solutions is extremely low in the area close to what looks like a plateau in the potential Pareto front between the total route cost values <20 100,21 200> in Figure A.5(b) compared to other parts of the potential Pareto front. Hence, we cannot verify whether the plateau dominates an area where no improved feasible solutions exist, or if the solution method was unable to find solutions in this area. The difference in the quality of the diversification properties in the result from these two instances is supported by the average b_p value in Table 7.2. The value was found to be 0.0147 and 0.0277 for the CBMix17 and CBMix19.

Additionally, we see from Table 7.2 that there is a large gap between the best and average b_p value. Hence, one run of the solution method is probably not sufficient to obtain a well diversified result.

However, as mentioned in Chapter 7.3, the aim is not to obtain a potential Pareto front where the solutions are equally distributed along the potential Pareto front, but rather to find a good approximation of the Pareto front. In the abscence of prior work on the problem or exact solution method, we do not have any prior knowledge of how the Pareto fronts for the instances looks like. As a result of working with descrete problems, there may not exist solutions well spread along the Pareto front. Hence, for parts of the potential Pareto front with low density of solutions, we cannot conlcude if this is representative for the Pareto front or due to a weakness in the soluton method. On the other hand, for the areas of high density of solutions, we can study the correlation of the objectives by the shape of the potential Pareto front.

For most of the instances, the potential Pareto fronts were either curved or shaped as a curve with one or multiple plateaus in the potential Pareto front. Hence, in most of the potential Pareto fronts, the objectives are conflicting, but for certain intervals on the potential Pareto front one objective could be improved without requiring a reduction in the other objective. These intervals are of great interest, since both objectives can be optimized simultaneously, which makes the choice of a single best solution within the interval straightforward. An example of such an interval, where the density of solutions is sufficiently high as to verify plateaus in the Pareto front, appears for the CB-Mix15 instance in Figure A.3(b), for total route cost between <9400, 9600> and <9600, 9800>.

We observed that when a new non-dominated solution was found in a generation, such that the area of the non-dominated domain decreased, the solution usually dominated multiple non-dominated solutions from the last generation. Hence, the total number of non-dominated solutions decreased. However, in the following generations, new non-dominated solutions on the potential Pareto front were usually found. As a consequence, the number of generations the method could search for new solutions with an unchanged non-dominated area had a great impact on the number of solutions in the potential Pareto front.

Solutions with total route cost as good as the total route cost in the best known solution was found in relation to two instances (CBMix22, CBMix23), where one of them is the proven optimal solution, as shown in Table 7.3. On average for all instances the gap between the minimal total route cost found and the total route cost in the best known solution was 13.8%.

From Table 7.3, we see that the relative value of the route balance objective does depend on the problem size, rather than the value of the total route cost. This could be explained by the increased difficulty to find balanced routes when the problem size is large, or that a larger population size may be necessary to explore a higher number of solutions.

The mutation operators based on a local search (`ALNS`, `Cross`, and `Greedy swap of tasks`) had a great effect on improving the quality of the objectives. If only the crossover operators and the simple mutation operators are applied, only a small number of all possible solutions were found. The main reason for this is that in these operators, only a consecutive sequence of tasks in the routes are inherited or a single task is replaced, which is not sufficient operators for finding all possible solutions. In the local search based mutation operators multiple independent tasks are removed and inserted in more efficient positions. Due to this property, solutions which otherwise would not have been explored are found, preventing premature convergence.

After all, premature convergence was an issue in the beginning of the search in some instances. This was, however, usually a temporary state where no non-dominated solutions were found, and the population resumed evolving after a number of generations. Thus, the problem was reduced by introducing a minimum number of generations.

Despite a sufficiently high value for the minimum number of generations to prevent premature convergence, most of the runs converged in a short number of generations after the minimum number of generations was reached. This is shown in Table 7.6.

For five of the six instances presented in Appendix A, we observe a small improvement in the quality of the objectives by letting the search continue for a longer time. Here, the required CPU time was approximately ten times higher than for the standard runs. The gap between the total route cost of the best known solution and the average minimal total route cost found by the proposed solution method was reduced from 16.13% to 13.56% on average for these six instances. Hence, the value for the minimum number of generations and number of generations for the non-dominated area to be unchanged before the convergence criterion is applied is decided as a trade off between the quality of the objectives and the required running time.

However, a longer run does not guarantee an improved solution compared to the standard run. For one of the instances (CBMix17), the minimal total route cost found in the standard run is lower than for the long run. This is a consequence of that the method is stochastic, hence the difference in the quality of the objectives is a result of coincidences in the operators, as well as running time.

The variances in b_p and in the area of the non-dominated domain are low and consistent for all of the instances, as shown in Table 7.5. However, even though the variances are low, Figures 7.1 to 7.6 indicate that a single run of the solution method may not be sufficient to obtain a good approximation of the Pareto front. Instead, a union of multiple potential Pareto fronts for the same instance would make a more precise approximation.

Chapter 8

Conclusion and further research

8.1 Conclusion

Routing problems in general and the MCGRP in particular are of great interest, both scientifically and in terms of applications for delivery problems and logistics. Prior work on the MCGRP has only been focusing on minimizing the total route cost. However, the addition of other aspect may be interesting when modeling real life problems. For instance, the drivers of the vehicles in a delivery case are motivated by an equal distribution of work load. Hence, balancing of routes is of likely to be an important factor in practicle applications.

In this thesis, we have studied a bi-objective variant of the MCGRP that is optimized with respect to total route cost and route balance. For the latter objective, the length difference between the longest and shorterst route in a solution is minimized. We propose a hybrid metaheuristic solution method based on the genetic algorithm with local search based heuristic methods as mutation operators. The aim of the search is to find a set of high quality solutions with great diversity.

To our knowledge, this is the first study of a multi-objective variant of the MCGRP; hence it is difficult to evaluate the performance of the solution method. To evaluate the quality of the objectives, the results are compared with the best known solutions to the single-objective minimize total cost MCGRP. The diversity of the solutions is evaluated by studying the location of the solutions in the potential Pareto front, and properties of the objectives are determined by the shape of the potential Pareto front.

The best known total route cost was found for two instances (CBMix22, CBMix23), where one of them is the proven optimal solution. The maximum gap between the minimal total route cost found and the total route cost in the best known solution was 17.01% (CBMix21). On average over all instances the gap was 13.8%.

The results indicate a good diversity of the solutions along the potential Pareto front. However, a larger population size than proposed in this thesis or multiple runs of the same instance may be necessary to obtain a good approximation of the Pareto front. As expected, to achieve a well diversified Pareto set of high quality solutions is more difficult for instances of larger sizes than for instances of smaller sizes.

Based on the shape of the potential Pareto fronts, the objective values seem to be conflicting in most of the instances. However, the precise correlations of the objectives differ among the instances.

Our contribution to the research consists of providing the first multi-objective variant of the MCGRP. Additionally, we propose an approach to assign ranks to the solutions in $O(N \log(N))$ time, which can easily be expanded to general bi-objective problems based on elitism.

8.2 Further research

There is potential for improvement in the proposed solution method. First of all, as mentioned in Chapter 6.3, the operators proposed in this method are designed to ensure feasibility in all the solutions. However, it may be the case that improved feasible solutions are found by allowing the solution to be infeasible as a temporary state in the search.

Secondly, a potential drawback of the method is that the operators' behavior does not depend on the previous success or failure. Here, we may achieve improved results if the operators only did small changes in the solutions as the search converged, or the operators were adaptive, i.e. which operator to choose was based on previous success.

The CPU time required for the search to converge will decrease significantly if the solution method is implemented using parallel programming. At each generation, it is only the assessment and selection steps which require information about the rest of the population. The two remaining steps, which are also the most time consuming, are independent for each pair of parents chosen in the selection step, and could be implemented in as many parallels as there are pairs of parents.

Since this is the first study on a multi-objective variant of the MCGRP, the performance of the solution method cannot be evaluated by comparing the results with results from other methods. It would be interesting to construct an exact solution method and compare the results with the Pareto front for a small instance.

It may also be interesting to extend the problem. We have only studied two objectives in this thesis. But as we mentioned in Chapter 4.2, other objectives may be of interest in applications, such as time objectives, profits or number of overlapping routes.

The proposed solution method can easily be generalized to problems with more or other objectives than the two objectives considered in this thesis. The main change needed to handle more objectives is the method for assigning a rank value, described in Chapter 5.2. Additionally, an evaluation measure of the objectives must be defined, and it is strongly recommended to redesign the operators which are optimizing a particular objective if this objective is no longer considered. An example is the repair operator in the `ALNS`, see Chapter 6.6, where tasks are reinserted in the shortest route, with the aim of improving the route balance objective.

Bibliography

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, pp. 80–91, Oct. 1959.

[2] R. Pandi and B. Muralidharan, "A capacitated general routing problem on mixed network," *Computers and Operations Research*, vol. 22, pp. 465–478, May 1995.

[3] M. Ehrgott and X. Gandibleux, "A survey and annotated bibliography of multiobjective combinatorial optimization," *OR Spektrum*, vol. 22, pp. 425–460, 2000.

[4] N. Jozefowiez, F. Semet, and E.-G. Talbi, "Target aiming pareto search and its application to the vehicle routing problem with route balancing," *Journal of Heuristics*, vol. 135, no. 5, pp. 455–469, 2007.

[5] S. Faggian, *Maximum Principle for Linear-Convex Boundary Control Problems applied to Optimal Investment with Vintage Capital.* math.OC, 2007.

[6] M. Caramia and P. Dell'Olmo, "Multi-objective optimization," in *Multi-objective Management in Freight Logistics Increasing Capacity, Service Level and Safety with Optimization Algorithms*, ch. 2, Springer, 2008.

[7] L. Fogel, A. Owens, and M. Walsh, *Artificial Intelligence Through Simulated Evolution.* New York: John Wiley & Sons, 1966.

[8] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[9] J. H. Holland, *Adaptation in Natural and Artifical Systems.* University of Michigan Press, 1975.

[10] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," *Lecture Notes in Computer Science*, vol. 1520, pp. 417–431, 1998.

[11] E. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J. Y. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *IEEE Transacion on evolutionary computation*, vol. 31, pp. 170–186, 1997.

[12] C. S. Orloff, "A fundamental problem in vehicle routing," *Networks*, vol. 6, no. 3, pp. 281–284, 1974.

[13] B. Golden and R. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, pp. 305–315, 1981.

[14] J. Gutiérrez, D. Soler, and A. Hervás, "The capacitated general routing problem on mixed graphs," *Revista Investigacion Operacional*, vol. 23, no. 1, pp. 15–26, 2002.

[15] C. Prins and S. Bouchenoua, "A memetic algorithm solving the vrp, the carp and general routing problems with nodes, edges and arcs," *Recent Advances in Memetic Algorithms*, vol. 166, pp. 65–85, 2004.

[16] H. Kokubugata, A. Moriyama, and H. Kawashima, "A practical solution using simulated annealing for general routing problems with nodes, edges, and arcs," *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, vol. 4638, pp. 136–149, 2007.

[17] M. Dell'Amico, J. Díaz, G. Hasle, and M. Iori, "An adaptive iterated local search for the node, edge, and arc routing problem." Working paper submitted, 2012.

[18] G. Hasle, O. Kloster, M. Smedsrud, and K. Gaze, "Experiments on the node, edge, and arc routing problem," *SINTEF*, May 2012. ISBN 978-82-14-05288-6.

[19] L. Bach, G. Hasle, and S. Wøhlk, "A lower bound for the node, edge, and arc routing problem," *Computers & Operations Research*, vol. 40, no. 4, pp. 943–952, 2013.

[20] A. Bosco, D. Laganà, R. Musmanno, and F. Vocaturo, "Modeling and solving the mixed capacitated general routing problem," *Optimization Letters*, vol. 7, no. 7, pp. 1451–1469, 2013.

[21] K. A. Gaze, E. E. Halvorsen-Weare, G. Hasle, and C. Mannino, "A branch-and-price method for the mixed capacitated general routing problem." Working paper, 2014.

[22] P. Soberon, "Graph theory," in *Problem-Solving Methods in Combinatorics*, ch. 4, pp. 43–57, Springer Basel, 2013.

[23] I. Lyckander, M. Grasmair, E. Halvorsen-Weare, and G. Hasle, "Route balance for the mixed capacitated general routing problem." Project thesis, 2014.

[24] M. R. Garey and D. S. Johnson, *Computers and Interactability; A Guide to the Theory of NP-Completeness*. W.H.Freeman and Company, 1979.

[25] N. Jozefowiez, F. Semet, and E.-G. Talbi, "Multi-objective vehicle routing problems," *European Journal of Operational Research*, vol. 189, no. 2, pp. 293–309, 2008.

[26] N. Norouzi, R. Tavakkoli-Moghaddam, A. Salamatbakhsh, and M. Alinaghian, "Solving a novel bi-objective open vehicle routing problem in a competitive situation by multi-objective particle swarm optimization," *Journal of Applied Operational Research*, vol. 1, no. 1, pp. 15–29, 2009.

[27] I. Giannikos, "A multiobjective programming model for locating treatment sites and routing hazardous wastes," *European Journal of Operational Research (1998)*, vol. 104, pp. 333–342, 1996.

[28] C. P. Keller and M. F. Goodchild, "The multiobjective vending problem: a generalization of the travelling salesman problem," *Environment and Planning B: Planning and Design*, vol. 15, pp. 447–460, 1988.

[29] N. Jozefowiez, F. Semet, and E.-G. Talbi, "An evolutionary algorithm for the vehicle routing problem with route balancing," *European Journal of Operational Research*, vol. 195, no. 3, pp. 761–769, 2009.

[30] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 151–165, 2011.

[31] M. P. Hansen and M. P. Hansen, "Tabu search for multiobjective optimization: Mots," in *Proceedings of the Thirteenth International Conference on Multiple Criteria Decision Making*, pp. 6–10, Springer-Verlag, 1997.

[32] R. Caballero, M. Gonzale, F. Guerrero, J. Molina, and C. Paralera, "Solving a multiobjective location routing problem with a metaheuristic based on tabu search. applocation to a real case in andalusia.," *European Journal of Operational Research*, vol. 177, pp. 1751–1763, 2007.

[33] R. Banos, J. Ortega, C. Gil, A. Fernandez, and F. de Toro, "A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows," *Expert Systems with Applications*, vol. 40, pp. 1696–1707, 2013.

[34] B. Baran and M. Schaerer, “A multiobjective ant colony system for vehicle routing problem with time windows,” *Proceedings of the Twenty-first IASTED International Conference on Applied Informatics*, pp. 97–102, 2003.

[35] A. Corberan, E. Fernandez, M. Laguna, and R. Marti, “Heuristic solutions to the problem of routing school buses with multiple objectives,” *Journal of the Operational Research Society*, vol. 53, pp. 427–435, 2002.

[36] R. Banos, J. Ortega, G. Consolacion, M. A. L., and F. de Toro, “A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows,” *Computer and Industrial Engineering*, vol. 65, pp. 286–296, 2013.

[37] N. Srinivas and K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms,” *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1995.

[38] K. Deb, A. Pratab, S. Agarwal, and T. Meyarival, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE Transacion on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[39] D. Goldberg and J. Richardson, “Genetic algorithms with sharing for multi-model function optimization,” *Second International Conference on Genetic Algorithms*, pp. 41–49, 1987.

[40] J.-Y. Potvin and S. Bengio, “The vehicle routing problem with time windows part ii: Genetic search,” *INFORMS Journal on Computing*, vol. 8, pp. 165–172, 1996.

[41] C. Prins, “A simple and effective evolutionary algorithm for the vehicle routing problem,” *Computers and Operations Research*, vol. 31, pp. 1985–2002, 2004.

[42] D. Pisinger and S. Ropke, “Large neighborhood search,” *Intelligent Information Management*, vol. 4, pp. 66–74, May 2012.

[43] G. A. Croes, “A method for solving traveling salesman problems,” *Operations Res. 6*, pp. 791–812, 1958.

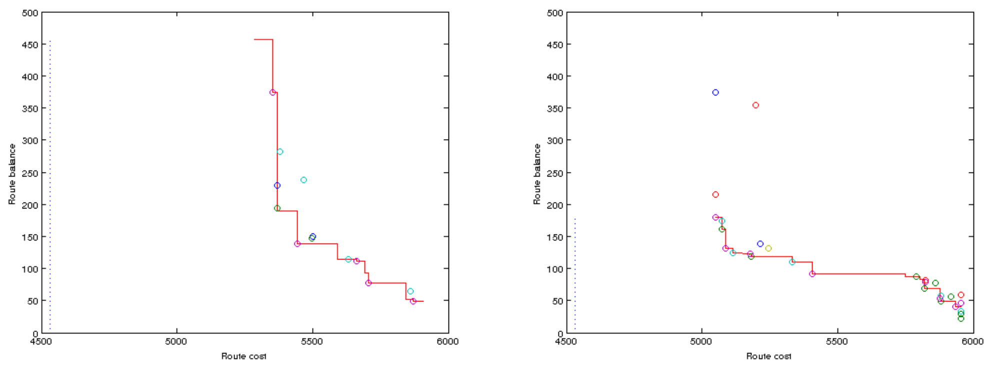
[44] SINTEF, “Top web page.” Accessed: October 2013.

Appendix A

Sample potential Pareto fronts

In this appendix, a sample potential Pareto front of a standard and a longer run of the instances CBMix5, CBMix13, CBMix17, CBMix19, and CBMix21 are presented. For each instance, a plot of the potential Pareto front and all the non-dominated solutions found by every $500th$ generations for both a standard (left figure) and a longer run (right figure) is given. Additionally, the objective values for the solutions in both potential Pareto fronts are stated in a table below the figures.

A.1 CBMix5



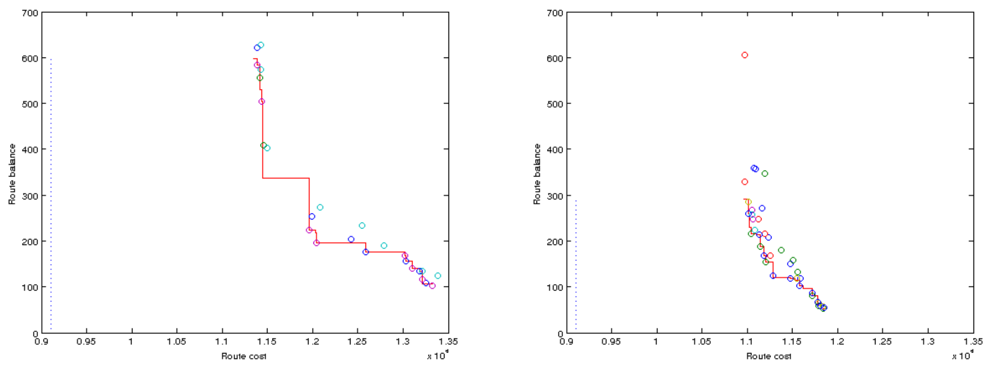
(a) Standard run (b) Long run

Figure A.1: Sample potential Pareto front for CBMix5

Table A.1: The objective values of the solutions in Figure A.1

(a) Total route cost	(a) Route balance	(b) Total route cost	(b) Route balance
5277	1021	4954	1837
5286	457	5050	179
5353	374	5075	161
5370	189	5088	132
5444	138	5113	125
5591	115	5178	123
5663	111	5182	119
5692	93	5333	110
5706	77	5407	91
5842	52	5749	88
5869	49	5789	87
5909	43	5803	83
		5821	69
		5877	53
		5879	49
		5936	40
		5953	22

A.2 CBMix13



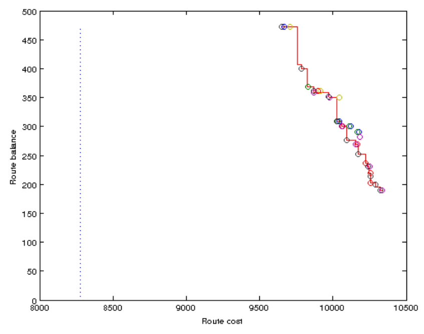
(a) Standard run (b) Long run

Figure A.2: Sample potential Pareto front for CBMix13

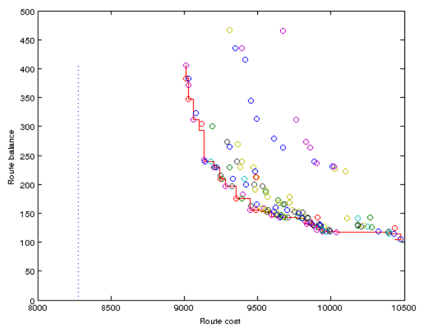
Table A.2: The objective values of the solutions in Figure A.2

(a) Total route cost	(a) Route balance	(b) Total route cost	(b) Route balance
11312	1740	10950	709
11341	598	10960	292
11387	584	11014	260
11421	530	11025	229
11436	504	11039	215
11444	338	11130	213
11963	224	11144	188
12027	217	11181	168
12037	196	11208	155
12591	176	11283	120
13025	168	11481	119
13032	157	11523	114
13103	140	11577	102
13181	134	11619	97
13208	117	11720	81
13217	106	11784	61
13326	103	11786	59
		11805	54
		11841	52
		11846	51

A.3 CBMix15



(a) Standard run



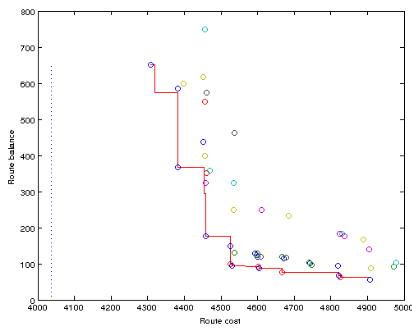
(b) Long run

Figure A.3: Sample potential Pareto front for CBMix15

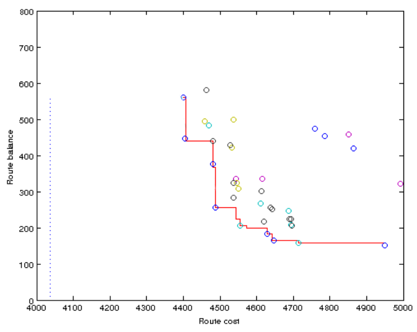
Table A.3: The objective values of the solutions in Figure A.3

(a) Total route cost	(a) Route balance	(b) Total route cost	(b) Route balance
9651	473	9011	406
9755	407	9013	383
9787	400	9026	371
9825	368	9028	348
9869	362	9065	312
9872	359	9104	293
9971	351	9135	242
9977	350	9139	240
10029	309	9200	229
10062	301	9239	210
10092	276	9282	197
10157	269	9356	176
10170	252	9450	156
10220	237	9524	153
10245	219	9597	144
10258	202	9769	141
10288	199	9823	135
10291	196	9827	133
10323	189	9833	132
10336	172	9839	130
		9879	129
		9903	121
		10039	117
		10396	114
		11227	105

A.4 CBMix17



(a) Standard run



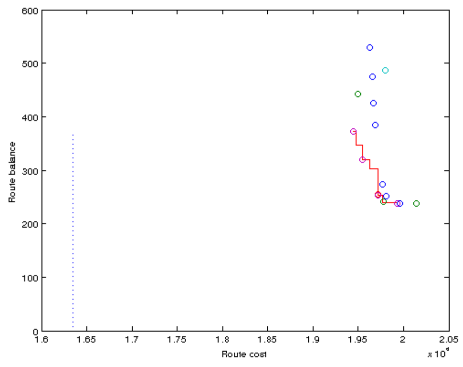
(b) Long run

Figure A.4: Sample potential Pareto front for CBMix17

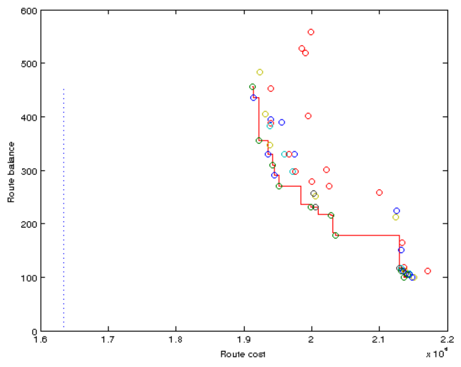
Table A.4: The objective values of the solutions in Figure A.4

(a) Total route cost	(a) Route balance	(b) Total route cost	(b) Route balance
4309	651	4410	561
4319	574	4417	440
4382	368	4480	377
4454	294	4481	366
4458	177	4487	256
4526	100	4544	225
4531	95	4555	206
4601	91	4572	198
4605	87	4629	182
4667	77	4642	164
4821	68	4714	158
4827	62	4949	152
4907	56	5003	145
		5074	135
		5148	129
		5219	113
		5288	95
		5362	92
		5370	85
		5444	77
		5510	74
		5513	71
		5519	56
		5581	46
		5656	43

A.5 CBMix19



(a) Standard run



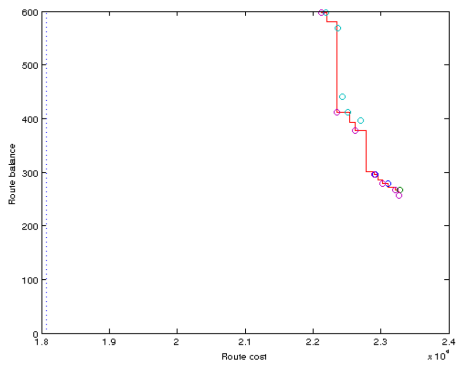
(b) Long run

Figure A.5: Sample potential Pareto front for CBMix19

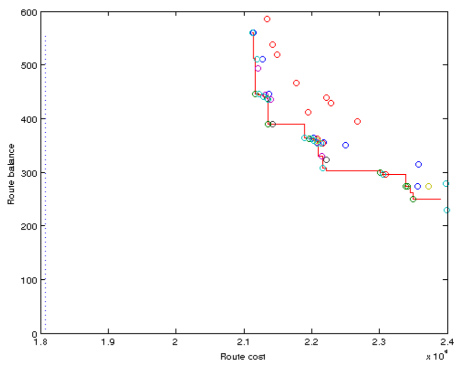
Table A.5: The objective values of the solutions in Figure A.5

(a) Total route cost	(a) Route balance	(b) Total route cost	(b) Route balance
19444	372	19123	456
19474	347	19147	436
19546	320	19223	356
19626	303	19360	330
19716	253	19426	310
19764	240	19455	290
19926	238	19521	270
		19837	236
		19993	231
		20094	217
		20281	216
		20313	184
		20353	178
		21292	117
		21315	112
		21349	107
		21361	100
		21441	96

A.6 CBMix21



(a) Standard run



(b) Long run

Figure A.6: Sample potential Pareto front for CBMix21

Table A.6: The objective values of the solutions in Figure A.6

(a) Total route cost	(a) Route balance	(b) Total route cost	(b) Route balance
22027	1796	21134	560
22038	812	21139	512
22127	598	21164	446
22201	580	21220	444
22349	412	21294	441
22537	393	21347	438
22624	377	21350	435
22778	301	21359	390
22908	295	21900	364
22964	286	21956	362
23029	279	22028	358
23104	272	22083	356
23215	266	22086	353
23253	263	22095	329
23263	257	22165	308
		22223	302
		23011	299
		23049	296
		23390	273
		23451	261
		23495	250
		23900	229
		24897	229
		24821	229