universität wien

DISSERTATION

Titel der Dissertation

„Decomposition Strategies for Large Scale Multi Depot Vehicle Routing Problems“

Verfasser

Mag. Alexander Ostertag

angestrebter akademischer Grad

Doktor der Sozial- und Wirtschaftswissenschaften (Dr. rer. soc. oec.)

Wien, im Dezember 2008

Studienkennzahl lt. Studienblatt	A 084 157
Dissertationsgebiet lt. Studienblatt	Internationale Betriebswirtschaft
Betreuer	O.Univ.Prof. Dipl.-Ing. Dr. Richard F. Hartl

Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction

The need to transport goods from one location to another existed since the first human civilizations have emerged. Since then, transportation has always been one of the most important domains of social and economical activities. A society without proper means of transport was and still is strongly hampered in growth and efficiency. If we take a closer look on today's economical structure, which is centered around an international division of labor, it is obvious that efficient methods of transportation are crucial for success. Nearly every activity we perform in modern life, be it shopping in a store, travel to distant destinations or using a cellular phone, is supported by some kind of logistical operation. As a result the wealth of whole countries or economic blocks, as well as the success of individual companies is closely tied to optimizing their transportation activities.

According to the European Union (see Eurostat 49/2008), freight transport has increased by 5 % from 2005 to 2006 and 25 % from 2000 to 2006, reaching a total of 2600 billion tkm over all modes of transportation in the EU-27 member states. It is to note that road freight transport accounted for 73% of the total freight transported, and was growing by 3% from 2000 to 2006. The recent EU enlargements, the ongoing trend for decentralized production or other globalization trends will continue to support a steady rise of transportational activities. Similar figures can be seen in other economic blocks like the USA or Japan and even more in rising economies like India or China. This figures show the huge potential that is possible when optimizing transportation activities, especially in the field of freight transportation on road. From an organizational standpoint, the need to optimize distribution costs is crucial to stay competitive in highly decentralized economies. On average, almost half of the logistic costs are distribution costs, with industries like the food industry reaching nearly 70% of the value added costs of goods being distribution costs (cf. Bräysy and Gendreau 2005a). This also underlines the need for efficient logistics for corporations.

In the recent decades, a complete range of different research field that tackle in some way or another logistic problem over the whole Supply Chain have arisen. Operations Research and Mathematical Programming techniques have shown to be successful at handling the complex nature of transportation problems. Especially in the field of Vehicle Routing Problems (VRP) that deal with the efficient routing of vehicles between a recipient and a distributor of goods, this techniques are frequently applied. To reflect real world applications, the field of VRP comprises a wide range of different problems that vary in structural design or are defined by additional constraints. Time windows, in which customers need to be visited so that goods can be delivered or restrictions on capacity of trucks are examples for such constraints. Serving customers from more than one depot is another possible characteristic of the VRP.

Since VRP problems are generally NP-hard, special solution methods need to be developed. Exact methods like Linear Programming Methods (LP) or Mixed Integer Programming (MIP) methods are possible ways to tackle these kinds of problems, and were successfully applied to different types of VRPs. However, solving these problems to optimality usually takes a huge computational effort with these methods. Usually problems consisting of only a few dozen to a couple of hundred customer can be solved efficiently. Another approach than to solve them exactly, is to use some heuristic methods and especially metaheuristics. Metaheuristics can be described as general purpose methods aimed to guide an underlying heuristic. Metaheuristics prove to be of practical success when solving problems that can't be solved by traditional exact methods. There are different approaches developed in the field of metaheuristics that were all successfully applied to different transportation problems and especially VRPs. The most well known and most recently successful methods are for example Simulated Annealing (SA), Tabu Search (TS) or Ant Colony Optimization Methods (ACO). Solving strategies based on the ACO metaheuristic attempt to resemble the natural approach of ants that try to find their way from a foodsource to the nest. Another method that was successfully applied on a broad range of problems are Genetic Algorithms (GA) that are based on the neo-Darwinian theory of evolution, which features three main components; selection, recombination and mutation. Variable

Neighborhood Search (VNS) approaches try to improve a preconstructed solution through the use of different neighborhood structures that change over the search process.

This thesis focuses on solving the Large Scale Multi Depot Vehicle Routing Problem with Time Windows (MDVRPTW). The objective is to minimize the total distance traveled by the whole available vehicle fleet under the constraints of serving each customer exactly once in its corresponding time window. Furthermore load capacities of the vehicles may not be violated and each tour-length of the vehicles may not exceed a certain amount of time. Additionally, because of the multi depot nature of the problem, each vehicle tour has to start and end at the same depot. The MDVRPTW is a generalization of the VRP and is thus NP-hard. It therefore cannot be solved efficiently by exact methods when the problem size reaches a certain threshold.

The first contribution presented in this thesis is the developed Memetic Algorithm for the MDVRPTW for artificial standardized instances. Additionally the MA is hybridized with a specially developed ACO method to enhance solution quality and we show that the developed MA/ACO approach can obtain better results than the pure MA. The developed MA approaches are then compared to the most recent state of the art methods. We show that the results obtained are competitive for the small to medium sized test instances, that were also successfully tackled by a VNS in Polacek et al. (2004) and a TS in Cordeau et al. (2001b).

These standardized instances are however relatively small in size and may not be big enough to resemble a typical real world scenario with a couple of thousand customers. Because exact algorithms, as well as most developed metaheuristics can't handle problems of this size, decomposition approaches like POPMUSIC were developed to handle problems of very large scale.

Another contribution of this thesis is the development of two different algorithms that are able to solve real world MDVRPTWs of large scale. Both approaches are based on the POPMUSIC framework by Taillard and Voss (2001). This framework is a decomposition strategy that tries to overcome size restrictions, by intelligently splitting the problem into sub-problems and solving them separately. With the first approach, we demonstrate that population based approaches like MAs can successfully be integrated into the POPMUSIC framework to tackle real world problems of large scale. We present an efficient approach to generate sub-problems without destroying the valuable information stored in the population so that population based optimizer can easily be implemented. The efficiency of the approach is demonstrated by comparing the POPMUSIC-MA (PopMA) approach to the pure MA that tries to solve the problem without decomposition, as well as to a 2-phase approach where the problem is at first decomposed by a p-Median algorithm so that then each resulting sub-problem can be solved by the pure MA in the next phase. It is shown that the PopMA can outperform the same MA without decomposition as well as with an initial p-Median decomposition significantly. Further an accelerated version of the PopMA with tuned parameters is presented to demonstrate its flexibility with regards to finding high quality solutions in minimum time. Finalizing we compare our approach to a highly efficient VNS, and show that it can outperform it.

The second decomposition approach focuses on the integration of a VNS as an optimizer for the sub-problems. As the VNS only needs to manipulate one single solution, different ways to decompose the sub-problem are developed. Eight different approaches that use different measures how large scale MDVRPTWs can be decomposed are presented. Each of the strategies is tested and results are analyzed for the two depot, three depot and four depot case to further give insight how they behave with an increasing amount of depots. It is shown that the POPMUSIC-VNS (PopVNS) can further improve the solutions found by the PopMA significantly.

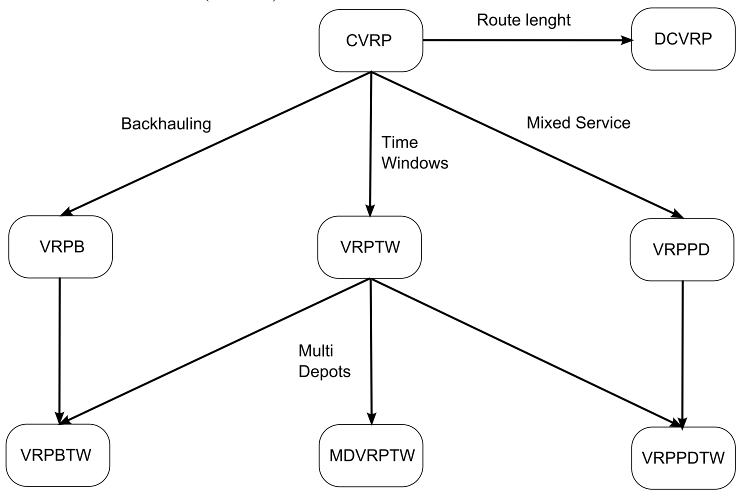
This thesis is organized as follows. The problem is explained in detail in Chapter 2. The POPMUSIC framework is described in Chapter 3. Chapters 4, 5 and 6 explain the metaheuristics used to solve the MDVRPTW. Chapter 7 presents the two developed solving strategies for the large scale real world MDVRPTW. In Chapter 8 the according numerical results are reported. In the same section different decomposition strategies are also analyzed in detail. In the conclusion (Chapter 9) we summarize the results and provide ideas for further research.

2. Multi Depot Vehicle Routing Problem with Time Windows

Carrier fleet operators are facing the following routing problem as their daily business: Goods dispatched from certain depots have to be delivered to a customer using a given vehicle fleet in a cost-effective and timely manner. In literature the VRP problem as well as its extensions have been studied in great detail. *"An overview of vehicle routing problems"* by Toth and Vigo (2001) is a comprehensive survey, that investigates the different VRPs with all its extensions. A short summary about the VRP classes is given in the following part of this section.

The most basic form in the VRP class is the Capacitated Vehicle Routing Problem (CVRP) where customers correspond to deliveries and have to be served a deterministic demand, by exactly one vehicle. The more complex variants all build on this basic concept of the CVRP, as can be seen in Figure 2.1. For a more realistic approach to the real world, time windows as well as a maximum allowed route length, are modeled as additional constraints and form the VRPTW and DCVRP variants. Backhauling constraints focus not only on delivering goods to a customer, but also on picking them up at certain backhaul-points. The critical assumption in this model is, that all deliveries have to be executed before any pickups can be made. This is in direct contrast to the VRP with mixed service (VRPPD), where pickups and deliveries can be executed at any time. All of the previously mentioned extensions deal only with one single depot, therefore the field of VRP was enriched by the MDVRP class, which focuses on efficient routing algorithms that can handle multiple depot setups. Other variants of the VRP are the VRP with split deliveries (SVRP), where customer orders can be carried out using more than one vehicle. The Periodic VRP (PVRP) deals with multiple periods in which customers can be served. This classification scheme divides the VRP

Figure 2.1.: The basic problems of the VRP class and their interconnections see Toth and Vigo (2001)



class by their main characteristics, but of course all of the mentioned constraints can be combined. This thesis focuses on the multiple depot case with the addition of time windows (MDVRPTW), so that customers must be serviced out of several depots, under the same constraints that apply for the VRPTW. Therefore the vehicle routes have to be determined in a way that:

- each route starts and ends at the same depot
- all customer requirements are met exactly once by a vehicle
- the time windows for both customers and the depots are respected
- the sum of all requirements satisfied by any vehicle does not exceed its capacity
- the total cost is minimized.

In the recent years most of the variants have been studied extensively. Especially the CVRP and VRPTW have been studied widely and many excellent approaches to solve them have been published in literature. The MDVRPTW however, is relatively new in origin and therefore hasn't attracted very much attention. Thus

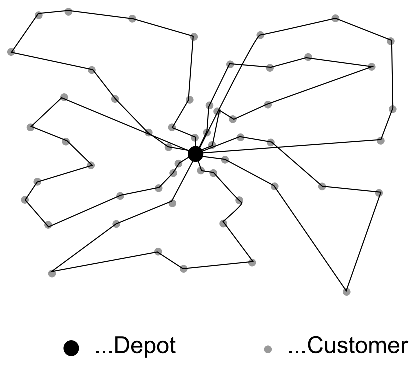


Figure 2.2.: VRPTW

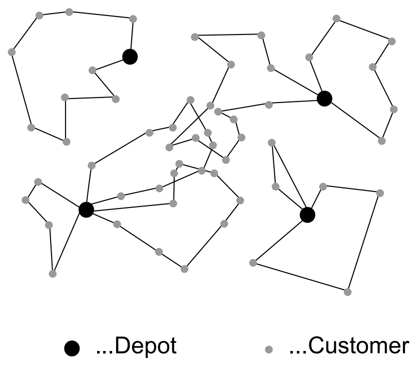


Figure 2.3.: MDVRPTW

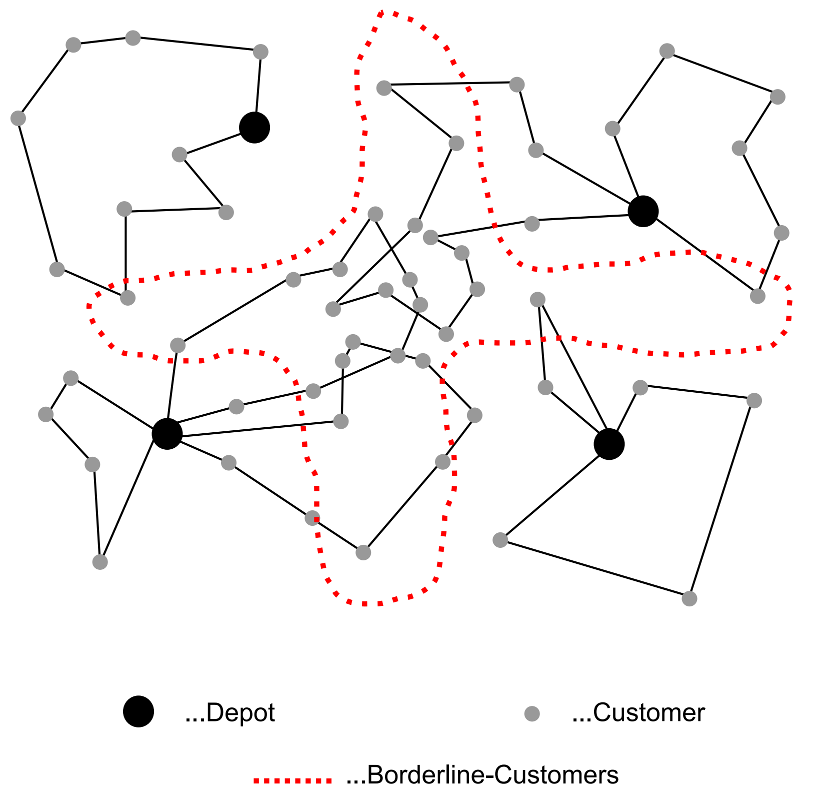
many high performing algorithms have been developed for the VRPTW, that may be the closest relative to the MDVRPTW, however if one takes a look in how routes are build in the multi depot environment (see Figures 2.2 and 2.3), and especially how customers are assigned from which depot they should be served, it is obvious that approaches that fully exploit the use of multiple depots, may yield a better performance. Especially the efficient rearrangement of borderline customers (customers that are in between different depots) to appropriate routes is of great importance to find highly competitive solutions. A typical MDVRPTW solution with its according borderline customers is depicted in Figure 2.4.

2.1. Model Formulation

The presented formulation of the MDVRPTW follows the models used in Polacek et al. (2004) and Cordeau et al. (2001b).

The MDVRPTW is defined on a complete graph $G = (V, A)$ where $V = \{v_1, ..., v_m, v_{m+1}, ..., v_{m+n}\}$ is the vertex set and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is the arc set. Vertices v_{m+1} to v_{m+n} represent the n customers. Vertices v_1 to v_m accord to the m depots to conclude all possible arcs that can be traversed. To each arc (v_i, v_j) a cost c_{ij} is given. To represent the constraints several weights are assigned to each vertex $v_i \in V, i = m+1, ..., m+n$. These weights are the demands d_i, the service times s_i, as well as the time windows $[e_i, l_i]$. The time windows are defined by the earliest e_i and latest l_i possible start times for the service. Not

Figure 2.4.: Borderline-Customers in the Multi Depot Environment



only are they applied to the customers, but also to to the depots ($i = 1, ..., m$) so that the opening hours of the depots can adequately be modeled. A non-negative travel-time or cost is linked to each arc (v_i, v_j). In the MDVRPTW case the vehicle fleet, consisting of K vehicles, is globally assigned to the m depots. The fleet is homogeneous and each vehicle is characterized by a non-negative capacity D and a non-negative maximum route duration T. The vehicle-fleet is evenly distributed amongst the m depots. The goal is to create K vehicle routes so that each customer i is serviced by exactly one vehicle in its corresponding time window $[e_i, l_i]$. The built routes have to start and end at the same depot and may not violate the maximal allowed tour length T or excess the vehicle capacity D. Routes have to be build with the objective to minimize the total time traveled c by all vehicles.

2.2. Related Work

The VRPTW has been extensively researched in literature with both exact and heuristic optimization approaches. An overview about exact approaches can be seen in Desrosiers et al. (1995) and Cordeau et al. (2001a). Other work that focuses on these exact methods in grater detail are Cook and Rich (1991) and Larsen (1999). Figure 2.1 shows that the MDVRPTW generalizes the VRPTW and is therefore also NP-hard. The most sophisticated exact methods for the VRPTW are only able to solve instances of very small size, so that they may be even less suitable to solve MDVRPTWs in reasonable time. In real world scenarios these size restrictions usually are surpassed relatively fast and therefore the most recent work focused mostly on developing metaheuristic approaches. Because this thesis focuses on solving real world problems, exact approaches are of relatively small interest and are therefor not further examined as they are by far not capable of solving them, if at all, in reasonable time. An overview about the VRPTW in the fields of heuristics and more sophisticated metaheuristics is presented in Bräysy and Gendreau (2005a,b).

To date, only two papers tackle the MDVRPTW efficiently. The first one to successfully tackle the MDVRPTW is the unified Tabu Search (TS) by Cordeau et al. (2001b). In this approach the authors use a very simple neighborhood structure, a move of one customer from one route to another, while allowing infeasible solutions. To guide the search from infeasible regions of the search space to feasible parts, a penalty function that adapts to the search history is used. The proposed TS approach was the first to be applied to the MDVRPTW, therefore the authors generated randomized data sets on which the algorithm was tested. To verify its general effectiveness it was tested on standardized VRPTW instances. The results achieved by the unified TS are competitive to other state of the art approaches. The second paper by Polacek et al. (2004) uses a VNS to further improve the solutions found by Cordeau et al. (2001b). The authors used the CROSS-Exchange operator for perturbing the solution, and a restricted 3-opt for bringing it to local optimality. The VNS was tested on the MDVRPTW instances introduced in Cordeau et al. (2001b), and it was shown that the results achieved outperform the results found by the unified TS approach. Another approache that tackles the MDVRPTW is the approach by Giosa et al. (2002), that tries to improve the solution finding process by applying a 2-phase approach. In the first phase customers are assigned to a certain depot through a specialized assignment algorithm. After all customers have been clustered, a version of the Clark and Wright heuristic solves the resulting single depot problems. The assignment of customers is fixed to a certain depot, therefore no interaction between the individual single depot VRPs is possible. This approach therefore is completely different to the previously described approaches (VNS, unified TS) were customers can flexibly be reassigned from one depot to another. More recently, in Tansini and Viera (2006), the authors improved their assignment of customers to depots by introducing a new measure of proximity. This measure not only uses distance but also the similarity of time windows as proximity and can further improve the solution finding process. However interaction between the different single depot VRPs is still not possible. To our best knowledge the mentioned approaches more or less exhaust the recent work done on MDVRPTWs.

2.3. Real World MDVRPTW Issues

Even though many different variations and extensions to the basic VRP problem are examined in literature, the typical real world problem can still feature some additional restrictions or constraints that are not considered so far. Split deliveries, stochastic demand, varying travel times, uncertainty or traffic jams are just some of these additional features that are hard to model, and where the resulting problems are even harder to solve. Additionally real world problems are usually considerably larger than the problems that are tackled in literature. In the case of VRPs exact algorithms can only solve a couple of hundred customers, while the state-of-the-art metaheuristic approaches are capable of handling VRP problems of up to 1000 customers (see Homberger and Gehring 2005; Mester and Bräysy 2007, 2005; Kytöjoki et al. 2007). This thesis focuses on developing strategies to solve the MDVRPTW in the real world. In literature however the most recent algorithms (Cordeau et al. 2001b; Polacek et al. 2004) only solve problems of up to 288 customers. Even though the mentioned algorithms work extremely well on larger problems, they are not specially designed to deal with a couple of thousands customers, like they can appear in the daily routine of medium sized carrier companies. Additionally real world problems might be different in the type of how customers and depots are distributed. In literature test instances are usually generated randomly, but they do not necessarily resemble a real world problem. For supraregional operators this means in detail that customers can be clustered in cities, with a very scarce distribution of customers in the country-side, where distances might be extraordinarily large. Even in cities where customers already are clustered, further clustering of customers, like in business districts can occur. Finally, a couple of customers can even be on the same spot like in shopping malls or large business structures. Big cities that are divided by highways rivers or other means can also be clustered. It can be seen that in the real world, instances generally are not homogenous. Algorithms that are developed to solve them should therefore try to exploit these geographical features to their advantage. This fact, combined with the large scale nature of real world problems lead in the direction that intelligently decomposing huge problems into smaller problems, can make originally impossible to solve problems feasible to solve in reasonable time. Further it is very likely that orders are only known partly in advance with the rest arising during the day. In this thesis however, we assume a deterministic demand that is known in advance, so that the resulting problems can be solved on a day to day basis. The time allowed to solve a problem is another restriction that is given in the real world. While algorithms are allowed to run for multiple days in a scientific setup, resources and time are limited in the real world. Especially for solving the VRP on the operational level, the run-times allowed may often not exceed the time between two working days, so that orders that arrive on the end of the day can be efficiently served the next morning. All these special characteristics of real world problems in the field of MDVRPTWs are often not accounted for in literature. This thesis focuses on developing decomposing strategies for real world MDVRPTWs so that they can be solved in reasonable time.

3. Decomposition Strategy - POPMUSIC

In this chapter a short introduction to decomposition strategies as well as the most related work is presented. Additionally we present the basic design of the POPMUSIC framework that was used as a decomposition strategy to solve the large scale MDVRPTW.

3.1. Introduction and Literature Review

Optimization problems like VRPs are usually of large scale when encountered in the real world. The need for transportation is steadily increasing, which results in the need for carrier fleet operators to handle a large amount of goods that need to be distributed to a huge customer-base. A couple of thousand customers to be served by a single company results in problems of considerable size that can hardly be solved even by the most advanced metaheuristic approaches. Decomposing strategies try to intelligently split these huge problems into manageable parts, so that they can be solved by efficient algorithms.

Examples for such strategies can be found in *"Parallel Iterative Search Methods for Vehicle Routing Problems"* by Taillard (1993), where two partitioning methods are presented that are applied to large scale VRP, with the intention to speed up the search of a TS. The first method tries to decompose into polar regions, while the second method is based on the arborescence built from the shortest paths from any city to the depot.

The Granular Tabu Search (GTS) by Toth and Vigo (2003) is a variant of the well known TS approach that uses a candidate-list strategy for solving the vehicle routing problems. The GTS works efficiently by guiding the search in a highly restricted neighborhood. This granular neighborhoods somehow decompose the problem into smaller parts, that can then be solved more efficiently by the TS.

Another example for a decomposition strategy is the D-Ants approach by Reimann et al. (2004) which divides problems of large size into smaller parts, so that the ants can construct solutions through a savings based procedure with reasonable effort. The authors demonstrate that decomposing the problem through the use of a sweep algorithm into sub-parts, and then solving these smaller parts, improves the effectiveness of the solving strategy so that standardized instances can be solved successfully.

Other approaches that successfully decompose large problems are *"A Tabu Search Approach for Delivering Pet Food and Flour in Switzerland"* by Rochat and Semet (1994), a *"Probabilistic diversification and intensification in local search for vehicle routing"* by Rochat and Taillard (1995). An approach that uses constraint programming as well as local search methods can be found in the work by Shaw (1998).

Some of these mentioned algorithms try to decompose a large problem into smaller sub-problems in a customized fashion that was specially designed for the problem at hand. The work of Taillard and Voss (2001) tries to standardize the procedure of decomposing a problem by introducing the POPMUSIC framework. In Ostertag et al. (2008a,b) we adapted the POPMUSIC framework to solve the MDVRPTW by using two different optimizers.

In the next part of this chapter the basic design of the POPMUSIC framework is presented.

3.2. Basic Design of the POPMUSIC Framework

In *"POPMUSIC: Partial Optimization Metaheuristic Under Special Intensification Conditions"* by Taillard and Voss (2001), the authors propose a framework for dealing with problems of large size. They defined the basic concept of the framework as shown in Algorithm 1.

To initialize the POPMUSIC strategy, a pre-calculated solution S is getting decomposed into parts $s_i; i = 1, ..., p$ through the use of some relatedness measure. There is no general way to define relatedness of one part to another as it highly

Algorithm 1 Basic POPMUSIC framework see Taillard and Voss (2001)

```
Input: Solution $S$ composed of parts $s_1, ..., s_p$, parameter $r$
Set $A \leftarrow \emptyset$
while $A \neq \{s_1, ..., s_p\}$ do
  Select seed part $s_i \notin A$
  Create a sub-problem $R_i$ composed of the $r$ parts $s_{i_1}, ..., s_{i_r}$ most related to $s_i$
  Optimize $R_i$
  if $R_i$ has been improved then
    Update $S$ (and corresponding parts)
    Set $A = A \setminus \{s_{i_1}, ..., s_{i_r}\}$
  else
    Set $A \leftarrow A \cup \{s_i\}$
  end if
end while
```

depends on the problem at hand. Usually in the context of VRP the relatedness measure is a distance based measure, taking into account travel time or travel costs, resulting more or less into clusters of customers that are geographically close together (see e.g. Rochat and Semet 1994; Rochat and Taillard 1995; Taillard 1993; Shaw 1998). Other approaches like "*New measures of proximity for the assignment algorithm in the MDVRPTW*" by Tansini and Viera (2006) additionally define relatedness of customers by their time windows. This is done in a way so that customers are highly related to each other when they are close together based on a distance measure and additionally also have to be serviced around the same time of the day.

In the next step of the POPMUSIC algorithm r of these parts are aggregated around the seed part s_i into a sub-problem. The parameter r therefore indirectly defines the size of the generated sub-problems, resulting in a higher decomposition of the problem the lower the parameter is set. The proper selection of parameter r and the relatedness measure are crucial for a working decomposition strategy. The better sub-problems are build that overlap and cover the whole solution S the harder it is to get stuck in local optimum. Each sub-problem generated is then improved with the help of an optimizer, that can be specially designed to solve the sub-problems at hand. Finalizing, if parts and sub-problems are well defined, every improvement of a sub-problem corresponds to an improvement of the whole solution S. The creation of new parts and their optimization is repeated as long as S can be improved.

This framework provides the guidelines for designing an efficient optimization method but also leaves certain design issues free to the decide for the developer of the metaheuristic. The points to can be freely decided on therefore are:

1. The definition of a part of a solution
2. The selection procedure of a part in A
3. The relatedness function between parts
4. The used sub-problem optimizer

In this thesis two different approaches to decompose a large scale MDVRPTW are presented. The used characteristics of the four mentioned points are explained in Chapter 7.

4. Variable Neighborhood Search

In the first section of this chapter we give an introduction to the VNS metaheuristic and present the most related and recent work found in literature. The basic design of the VNS, like it was implemented in our solving strategies, will also be presented in this chapter.

4.1. Introduction and Literature Review

Exact approaches to solve the MDVRPTW to optimality tend to be limited to rather small problem sizes. Additionally modern metaheuristics are often highly efficient and can provide solutions close to optimality or even solved to optimality in a fraction of the time. One of these highly efficient methods is the VNS which has demonstrated to be a very flexible and easily adaptable metaheuristic for various optimization problems. It has been developed by Mladenovic and Hansen (1997) and extended in Hansen and Mladenović (1999, 2001). The VNS they developed can be categorized as a stochastic improvement heuristic that manipulates one single solution. The VNS itself can't create a starting solution, therefor a solution has to be fed to the VNS; for example through the use of some construction heuristic. The not necessarily feasible solution, then functions as the starting point for further VNS iterations. After a shaking step, that perturbs a solution, some kind of Local Search (LS) operator then tries to enhance the resulting solution in each iteration. Because the VNS only works with one single solution and does not use any kind of memory system, where it can save pre-calculated solutions, it has to be specially designed so that it can overcome getting stuck in a local optimum. The two main parts of a VNS are therefore a working intensification phase, where solutions are improved through LS, and a diversification phase, were solutions are somehow randomly disturbed so that local optima can be overcome. In literature the VNS metaheuristic has proven to be very efficient in solving complex optimization problems in the field of VRP, a short overview of the most recent and successful papers is given in the next part of this section.

In *"Scheduling Periodic Customer Visits for a Traveling Salesperson"* by Polacek et al. (2007), a VNS was developed for a real world scenario, a similar VNS was then also applied to the Capacitated Arc Routing Problem with Intermediate Facilities by Polacek et al. (2008b) and for the MDVRPTW in *"A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows"* by Polacek et al. (2004) which was then further extended in *"A Cooperative and Adaptive Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows"* by Polacek et al. (2008a).

The approach by Kytöjoki et al. (2007) uses a VNS to solve the large scale CARP. The authors used a set of standard improvement heuristics that are embedded and guided by the VNS. To escape local optima a strategy similar to a guided local search metaheuristic is used. The VNS has proven to be fast and flexible in finding high-quality solutions for problems that can reach up to 20000 customers in size.

Another successful implementation of a VNS is presented in *"A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows"* by Bräysy (2003). In this paper the VNS is used to reduce the total distance traveled of the vehicles, so that new best know solutions can be found for standard data sets ranging up to 400 customers.

Hemmelmayr et al. (2009) propose a VNS to solve the PVRP, where a planning period of several days is considered and customers must be visited more than once. A Clark and Wright savings algorithm (see Clarke and Wright 1964) is used to construct the initial solution that is then iteratively improved by a VNS. The operators used in the Shaking Phase are based on the MOVE operator, where a customer can be moved to another position, and the CROSS Exchange Operator, where sequences of customers are switched with each other. Additionally they introduced a CHANGE Combination Operator that randomly changes the visit combinations for customers. A 3-opt operator (see Lin 1965) is then used to bring the newly generated solutions into local optima. The VNS was tested on standardized data sets and is able to outperform the formerly best known approaches.

The same authors used a similar VNS approach to solve a Real World Blood delivery problem (see Hemmelmayr et al. 2008) in a Vendor Managed Inventory setup.

Tricoire et al. (2007) developed a VNS for the Multi-Pile Vehicle Routing Problem (MP-VRP). The MP-VRP is a combination of two distinct problems. A packing problem, where items of certain size need to be packed in a limited amount of space in a vehicle, and a routing problem, where these vehicles need to deliver the items to their assigned customers at minimal cost. The VNS is applied in the routing-part, and also uses a Cross-Exchange Operator in the Shaking-Phase to perturb the incumbent solution so that local optima can be overcome. Their results show that the VNS is a fast and robust algorithm, that can improve 19 out of 21 best known solutions.

Another successful implementations of a VNS is for example found in *"Metaheuristics for the Team Orienteering Problem"* by Archetti et al. (2007). In this work the VNS is used to solve the Team Orienteering Problem (TOP), where a set of potentional customers needs to be selected out of available customers and a variable profit is collected when visiting them. The customers need to be visited in a given time limit through the use of a fixed fleet of vehicles with the objective to maximize profit. The paper shows that the VNS can successfully solve the TOP and beat the already known heuristics.

Fleszar et al. (2008) show that the VNS can also be successfully applied to the Open Vehicle Routing Problem (OVRP), were the objective is to minimize the number of vehicles and then to minimize the total distance traveled. The authors used a neighborhood structure that incorporated an exchanging of segments between routes, as well as a reversing of selected segments of routes.

In this thesis three different VNS based on the work by Polacek et al. (2004) are used on three occasions for different objectives. The first VNS used was implemented so that the generated solutions by it can serve as a valid point for comparison to other approaches. The second implementation of the VNS was used as some kind of mutation operator in a MA and the third VNS is used as a optimizer in a decomposing approach.

This chapter is organized as follows. The first part gives an overview about the basic design of the VNS. In the successive sections the primary components of the VNS, the construction of the initial solution, the shaking phase, the iterative improvement phase and the acceptance decision are presented.

4.2. Basic Design of the VNS

The VNS used was derived from the work done by Polacek et al. (2004) as it has been successfully applied to MDVRPTW problem instances created by Cordeau et al. (2001b). The instances range from 48 to 288 customers that need to be serviced by 4 to 30 vehicles from 4 or 6 depots. Additionally the data set is split in two parts, one where customers have very tight time windows in which they need to be served and the other where time windows are comparatively relaxed. The authors propose that the VNS is scaling very well with regards to the amount of customers in the problem, as runtimes are mostly dependant on the amount of customers in a route and not in the whole problem.

The Basic Steps of the VNS are shown in Algorithm 2.

Algorithm 2 Basic steps of the VNS see Hansen and Mladenović (2001)

```
Initialization. Select the set of neighborhood structures $N_\kappa(\kappa = 1, ..., \kappa_{max})$ to
be used in the search; find an initial solution $x$; choose a stopping condition;

Repeat the following steps until the stopping condition is met:

  1. Set $\kappa \leftarrow 1$;

  2. Repeat the following steps until $\kappa = \kappa_{max}$:
     a) Shaking. Generate a random point $x'$ from the $\kappa^{th}$ neighborhood of
        $x(x' \in N_\kappa(x))$;
     b) Iterative Improvement. Apply some iterative improvement method
        with $x'$ as initial solution; denote by $x''$ the so obtained local optimum;
     c) Move or not. If $x''$ is better than the incumbent or some acceptance
        criterion is met, accept $x''(x \leftarrow x'')$ and continue the search with
        $N_1(\kappa \leftarrow 1)$; otherwise, set $\kappa \leftarrow \kappa + 1$;
```

At first a Solution x is constructed, as the VNS needs an already existing solutions as starting-point. Furthermore a suitable neighborhood structure $N_\kappa(\kappa =$

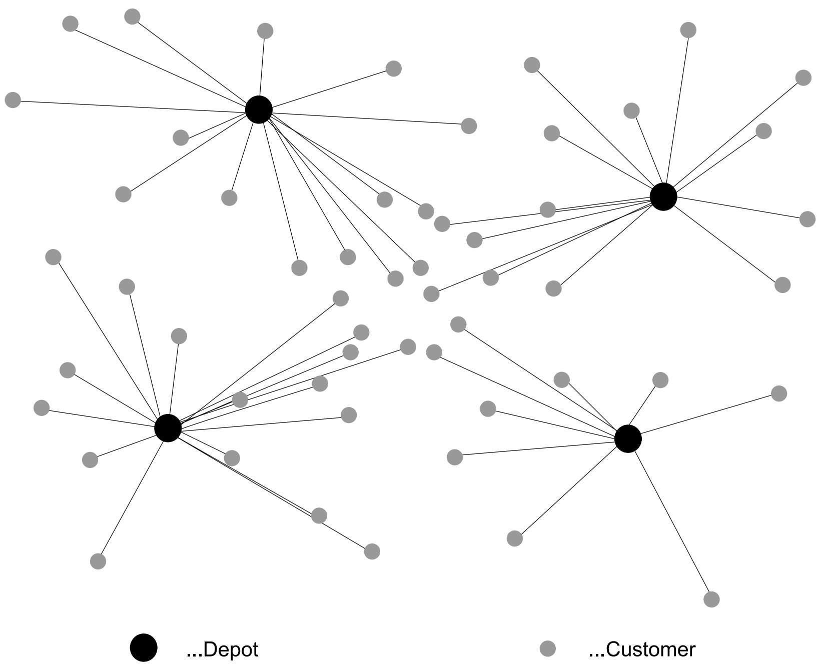
$1, ..., \kappa_{max})$ needs to be defined in which the VNS can perturb solutions. The VNS then repeats the following phases until a certain stopping criterion is met. In the shaking phase the incumbent solution x is changed at random to a solution x' in the κ^{th} neighborhood of $x(x' \in N_\kappa(x))$. The newly generated solution x' is brought in to local optimum by an iterative improvement method in the second phase and is denoted by x''. In the third phase, it is decided if the new found solution x'' is accepted for further calculation by some acceptance criterion. Usually better solutions are always accepted, but to break out of local optima, other acceptance criteria can be defined. If the new found solution is accepted it replaces x and the search resumes using the first neighborhood N_1. If no better solution can be found and x'' is not accepted, the search continues within the next neighborhood $N_{\kappa+1}$. The evaluation of the quality of a solution is done through the use of a fitness-function, that is explained in detail in Section 4.2.4.

4.2.1. Construct Initial Solution

The VNS is a very powerful and fast iterative improvement heuristic. Most authors therefor do not focus on developing very sophisticated construction algorithms, as tediously generated solutions may be found in a fraction of the time by the VNS that starts from relatively bad constructed initial solutions. The used construction heuristic is based on the highly efficient Clarke and Wright Savings algorithm (see Clarke and Wright 1964) and is therefore a more sophisticated method compared to the heuristics used for example by Polacek et al. (2004). However the developed VNS should be able to solve problems that are up to 50 times bigger than in comparable approaches and therefore each gain in solution quality benefits the decomposition of the respective problem into more precise sub-problems which will be presented in Chapter 3. Because the Clark and Wright algorithm is deterministic we enhanced it by a stochastic feature. This feature may not be needed for a VNS that only manipulates a single solution, but to establish a unified starting point for other population based approaches, a flexible heuristic that generates different solutions was needed.

In detail the Clark and Wright Savings algorithm is explained as follows; In the first stage, the clustering stage, all n customers are assigned to their geographically

Figure 4.1.: Clark and Wright - savings routes



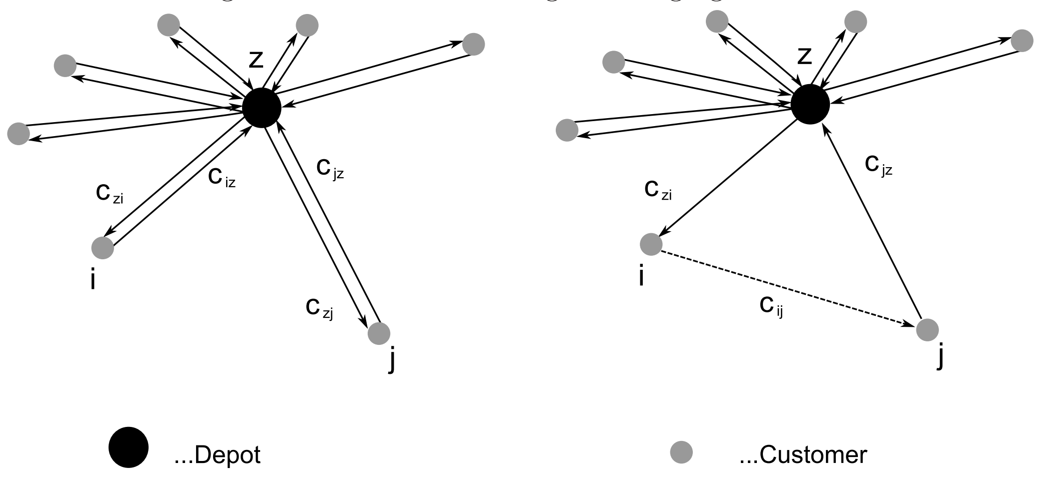
closest depot. We are aware that there exist more sophisticated methods for assigning customers to depots (see e.g. Salhi and Sari 1997), however a simpler approach was used because we didn't want to focus on developing a construction algorithms for the multi depot case. Afterwards routes are built with start- and end point at their assigned depot. Therefore n routes containing one customer only are generated. The procedure can be seen in Figure 4.1.

In the second stage, the savings stage, a list sorted according to the savings values s is generated. The merging of two routes is realized in the order of the greatest saving while prohibiting violations of the time-windows, route-length or load constrains. The savings value s_{ij} is calculated as follows.

$$s_{ij} = c_{iz} + c_{zj} - c_{ij} \tag{4.1}$$

where i, j denotes two possible customers that should be merged, and c_{ij} gives the travel cost from customer i to customer j. The depot is denoted with z. The savings procedure is depicted in Figure 4.2. The stochastic feature occurs

Figure 4.2.: Clark and Wright - merging of routes



exactly at this point, when a merging of two routes is theoretically allowed but rejected with a probability of 0.1 so that the algorithm continues with the next entry in the list. In the case of multi depots, customers serviced from two different depots can also be merged if their savings value would be the next in the list. In detail this means that the route of customer j switches the depot. In any case, regardless if the routes are serviced from the same depot or not, feasibility of the merging is checked. Violations of the tour-length, the time windows or the load are therefore not allowed. The algorithm then stops when no more routes can feasibly be merged. In the case when the number of routes exceed the maximum allowed number of routes K, routes are merged until K routes are reached with respect to generating the least violations of time windows, capacity or route duration according to Equation (4.2).

In the final step a 3-opt operator is applied to each route so that the constructed solution is the first incumbent solution within the VNS.

4.2.2. Shaking

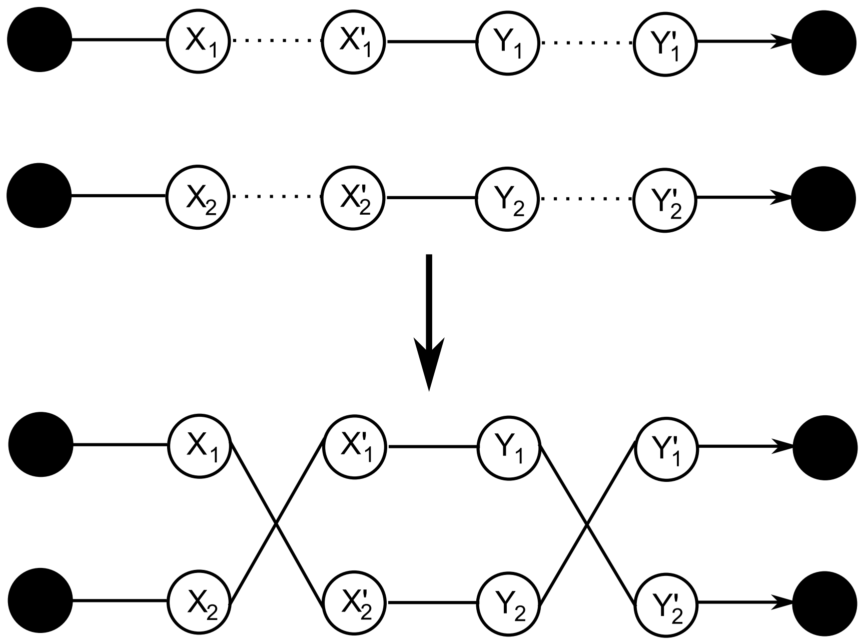
A very important design decision for the VNS is the selection of the right neighborhood structure in the shaking phase, so that the incumbent solution is sufficiently perturbed, while at the same time retaining the good parts of the solution. The

Table 4.1.: Set of neighbourhood structures

κ	Depots	maximum Sequence length
1	1	min(1, C_k)
2	1	min(2, C_k)
3	1	min(3, C_k)
4	1	min(4, C_k)
5	1	min(5, C_k)
6	1	C_k
7	2	min(1, C_k)
8	2	min(2, C_k)
9	2	min(3, C_k)
10	2	min(4, C_k)
11	2	min(5, C_k)
12	2	C_k

CROSS-Exchange Operator (see Taillard et al. 1997) is know to work well perturbing VRP Solutions and was used successfully in recent literature (see e.g. Tricoire et al. 2007; Fleszar et al. 2008; Polacek et al. 2004). The special features of the used shaking phase are explained as follows. The multi depot feature of the real world problem is considered by defining on which routes the CROSS-Operator is applied. Two different variants are realized depending on the active neighborhood. In the first variant only routes belonging to the same depot may be changed. In the second variant routes starting at different depots may be changed. In both of the mentioned variants the maximal allowed sequence length that may be changed by the CROSS operator depends on the active neigborhood and is defined between one and five. An additional case is considered where the maximum allowed length is equal to the number of customers in the route C_k with the smaller number of customers of the two considered routes. The actual sequence length that is then used for the CROSS-Exchange, is then randomly drawn between zero and the maximum allowed length. To avoid unproductive iterations, only one sequence length is allowed to be set to zero, which represents a move of a chain of customers from one route to another. The used neighborhood structure is shown in Table 4.1. We want to note that the two routes, on which the operator is applied,

Figure 4.3.: The CROSS-Exchange Operator



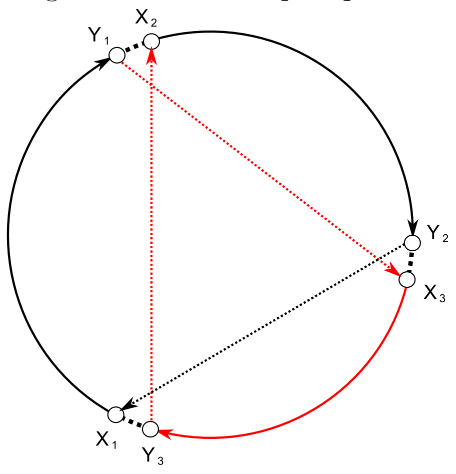
are selected at random. However, if the selected routes are the same, the operator is applied in a way so that it only moves sequences of customers in the route. The CROSS-Exchange Operator is shown in Figure 4.3 and works as follows. In the first step two edges X_1, X_1' and Y_1, Y_1' are removed from the first route while in the second route the edges X_2, X_2' and Y_2, Y_2' are removed. Afterwards the sequences $X_1' - Y_1$ and $X_2' - Y_2$ are swapped. The length of these two sequences may be arbitrary but the orientations of the sequences are preserved, but can be reversed on a route to route basis with a probability of $p_{iCross} = 0.001$.

4.2.3. Iterative Improvement

After the shaking phase each solution is improved by an iterative improvement procedure. The used procedure is a 3-opt (see Lin 1965) that is restricted to a maximum allowed sequence length of three customers. The 3-opt procedure is shown in 4.4 and explained as follows.

The route is split into three segments, $X_1 - Y_1$, $X_2 - Y_2$ and $X_3 - Y_3$ which are then

Figure 4.4.: The 3-opt Operator



reconnected in the following fashion; $X_1 - Y_1 - X_3 - Y_3 - X_2 - Y_2$. The sequences are therefore shuffled, without allowing a inversion of the customers in a sequence, so that time-window-constraints are violated as least as possible according to Equation (4.2). All possible sequence lengths for all three parts are then iteratively and systematically checked if they can be interchanged so that an improvement in the objective value occurs. We realized a first improvement strategy, which means the algorithm accepts the current solution as new incumbent solution as soon as an improvement is found. Afterwards the iterative improvement procedure restarts. A special feature of real world problems is that customers are often on the same geographic location (e. g. hospital, shopping mal, business centers,..) . For this fact a restrictive sequence length may hinder the 3-opt operator in improving the solution when more than the allowed amount of customers that can be shifted are located on the same position. When this happens we therefore allow within the 3-opt procedure to shift customers on the same location without any restrictions on the maximum allowed sequence length.

4.2.4. Acceptance decision

The fitness evaluation function of a solution S follows the implementation of Cordeau et al. (2001b) and Polacek et al. (2004). The total travel time of the routes is denoted by $c(S)$. The values $q(S)$, $t(S)$ and $w(S)$ respectively denote the total violation of load, duration and time window constraints. The arrival time a_i at each customer i is calculated and an arrival after the end of the time window $a_i > l_i$ is penalized while an arrival before the start of the time window $a_i < e_i$ is allowed but generates a waiting time. Each route is then checked for violations with respect to D and T as well as the total violation of the time window constraints $\sum_{i=1}^{n} max(0, a_i - l_i)$. The fitness function is show Equation (4.2).

$$f(S) = c(S) + \alpha q(S) + \beta t(S) + \gamma w(S) \tag{4.2}$$

We want to point out that α, β and γ are positive weights which are all set to 100 to strongly penalize infeasibility. Different values for the weights were tested, as well as some adaptive scheme that changes the weights according to the feasibility of the produced solutions. After some initial testing we came to the conclusion that fixing the weights at the according values, resulted in the best solutions. To measure the quality of a solution this evaluation function is used in all steps of the whole VNS procedure. It is therefore used to generate a starting solution, to improve a solution through the restricted 3-opt and in the *move or not* phase. The same fitness function as depicted in Equation (4.2) was used in all further approaches to evaluate individual solutions.

In the *move or not* phase, solutions x'' that are better than the incumbent solution x are always accepted. However this may lead to a fast convergence of the algorithm, and it may happen that it gets stuck in local optima. To overcome this restriction, we allow inferior solutions to get accepted as well, if both of the following criteria are met. The first criterion defines when inferior solutions are generally allowed to get accepted. It may not be desired to allow for deteriorating solutions to get accepted early on in the search, as it may hinder the improvement of the solution quality significantly. Ideally inferior solutions are only allowed to get accepted when the local optimum is reached. Since this is very hard to identify, we defined a number of unproductive iterations ($i_u = 10^5$) as the point when we allow for the acceptance of deteriorating solutions. Therefore after this limit is reached, the next inferior solution is allowed to get accepted as long as it meets the second criterion. This criterion defines a certain threshold value in which a inferior solution needs to be located. It is defined as a ratio ($p_t = 5\%$) of the so far best found solution to the current solution. As soon as a newly generated solution triggers the second criterion it is accepted for further iterations and the counter for unproductive iterations is reset, so that the convergence to better solutions is not hindered.

5. Memetic Algorithm

The basic design of the MA that was developed to solve the large scale MD-VRPTWs is presented in this chapter of the thesis. Additionally a short introduction to the field of MAs is given, with an overview about the most related work in recent literature.

5.1. Introduction and Literature Review

The term "Memetic Algorithm" is used to encompass a broad class of metaheuristics in the field of population based search. In population based search, a set of solutions is modified simultaneously so that individual solutions can interfere with each other. The development of MAs to solve complex combinatorial optimization problems relate to fields of Evolutionary Computing (EC) (see Fogel et al. 1966) and methods like Genetic Algorithms (GA) (see Reeves 2003; Goldberg 1989). The term genetic algorithm was coined by John Holland in 1975 and his book Adaption in Natural and Artificial Systems (Holland 1975), which lay the foundations for a flourishing field of research. Similar approaches by Rechenberg (1973) and Schwefel (1975) called Evolutionsstrategie (ES) were developed in the 60s and 70s. Both concepts are very similar to each other and revolve around the core of the Neo-Darwininan theory of evolution, which consists of the three main components; selection, recombination and mutation.

Moscato and Cotta (2003) describe the basic structure of a typical MA as similar to the one of a GA, but that is further enriched in a way that it can exploit all available knowledge about the problem that is under consideration. The basic structure of a MA is described as follows. To start the algorithm an initial population of solutions is needed. In the field of VRPs usually a fast construction heuristic is used to generate the first population. In the next step some kind of reproductive process is applied, which selects suitable individuals out of the population and generates offsprings with desired features. These offsprings are then used to update the population so that the average quality of the population is steadily increasing. Finally the population may converge, but not necessarily into the global optimum, as it may get stuck in a local one. Mutation operators are then used to disturb the current solutions so that new features may enter the population and the search can continue on another trajectory. The idea of MAs is to find a way to manipulate the search, so that it incorporates all available knowledge of the problem. This is usually done by adding additional operators to the search that enhance the quality of available solutions, or manipulates them so that solutions inherit some desired features. In literature the MA metaheuristic or similar GA metaheuristics were successfully applied to optimization problems in the field of VRP. An overview of the most relevant and recent work is given in the rest of this section.

Tan et al. (2001) developed a Hybrid Genetic Algorithm to solve the VRP. The developed GA uses a Partially Mapped Crossover (PMX) operator to recombine two parent solutions into new offsprings. The authors enhance traditional chromosome representation methods by adding information on the grouping of customers, which also represents the individual vehicle-routes. Local search operators are then applied to this grouping operation to find better solutions. After a grouping is fixed for a chromosome, traditional LS operators try to improve the offspring. The author use a λ-interchange procedure (see Christofides et al. 1979) in the LS step. The algorithm was tested on various standard benchmarks with satisfactory success.

"A simple and effective evolutionary algorithm for the vehicle routing problem" by Prins (2004) does not account for the constraints of time windows. However it is a very sophisticated approach that demonstrates how powerful properly implemented Evolutionary Algorithms can be. A general drawback of GAs in the field of VRP problems is the representation of chromosomes. If recombination operators are applied on chromosome representations without trip delimiters, the resulting offsprings are generally not feasible and need to be repaired. The authors propose a GA without trip delimeters that is hybridized with a local search procedure. A chromosome can be converted into a optimal VRP solution with regards to

the chromosome sequence at any time through the use of a specifically developed splitting procedure. This splitting procedure uses dynamic programming methods to find the optimal partitioning of the chromosome. An Order Crossover (OX) operator is used for recombination, and a set of nine different LS operators are then applied to improve the newly generated offsprings. This approach is one of the few GAs that is able to compete with other powerful approaches like TS in the field of VRP and especially on instances of very large size, where it was even able to outperform the existing best known results.

Another hybrid genetic algorithm for the VRPTW was developed by Berger and Barkaoui (2004) which focuses on the evolution on two different and parallel populations. While one populations objective is to minimize the total cost, the others objective is to minimize the violations. A master-slave message-passing paradigm coordinates the parallel procedure. The authors used an insertion-based crossover operator for recombination that was applied on a route to route basis. However they only allowed the recombination of routes in a certain neighborhood, which is defined as the maximum distance a centroid of a route may be away from another route. They then applied a suite of six mutation operators that try to modify, repair and improve the offsprings. The algorithm was tested on the Solomon problem instances and found six new best solutions.

The work *"Active guided evolution strategies for large-scale vehicle routing problems"* by Mester and Bräysy (2007) focuses especially on the development of an efficient approach to solve problems of very large size. This metaheuristic is a two-step procedure, where a Guided Local Search (GLS) Voudouris (see 1997) and Voudouris and Tsang (1999)) is used to regulate a composite LS in the first stage and the neighborhood of the evolution strategy in the second stage. The composite LS consists of the relocate (Savelsbergh 1992), the 1-interchange (Osman 1993) and the 2-opt* (Potvin and Rousseau 1995) improvement heuristics. The evolution strategy is implemented in a way so that a parent solution is purged of some customers. Afterwards the missing customers are inserted back into the problem by an insertion procedure. Finally, if the newly generated offspring is better than the parent, it replaces it. The method provided the best-known solutions to 86% of all of the 302 benchmark instances within reasonable computation time.

Figure 5.1.: Basic Steps of the Memetic Algorithm

1. *Initialization* Repeat *popsize* times
   a) Generate a solution with construction heuristic
   b) Improve solution with LS
   c) Insert solution in *pop*
2. Repeat until Stopping Criterion is met
   a) *Selection* Select two solutions from *pop* for recombination
   b) *Recombination* Generate offsprings O_1, O_2 through Crossover Procedure
   c) *Improvement Step*
      i. improve offspring O_1 and O_2 with Stochastic Local Search with probability p_1
      ii. improve *pop* with Stochastic Local Search with probability p_2
   d) *Population Management*
      i. insert the best offspring into *pop*
      ii. maintain *popsize* solutions in *pop*
   e) *Stopping Criterion* Stop algorithm when maximum allowed time or iterations is reached

5.2. Basic Design of the MA

The core components of every evolutionary algorithm are, selection, recombination and mutation, however other components like the initialization of the starting population, the updating strategy of the population and the evaluation of the quality of newly generated solutions are important when designing a MA for a combinatorial optimization problem. The Basic Steps of the developed and implemented MA can be seen in Figure 5.1 and are explained as follows.

In the Initialization Phase the population *pop* needs to be filled with newly generated solutions by a fast construction heuristic until the desired size *popsize* of the population is reached. Each of the individuals in the starting population is then brought into local optimum by a local search procedure, so that high quality solutions are at hand for the oncoming iterations of the MA. After the population has been initialized the following steps are repeated until the algorithm is stopped by some pre-defined criterion. At first two solutions need to be selected from the population through the use of a selection procedure. When two of them are selected a crossover-operator recombines them and generates two offsprings O_1 and O_2. In the improvement phase both of them are modified through a stochastic local search procedure. Additionally this procedure tries to improve the already existing solutions in the population. In the population management step the better of the two offsprings is allowed to enter the population. However because the size of the population is fixed it needs to be maintained. This is done by erasing duplicates in the population as well as by erasing one of the worst solutions in the population. Finally through this population management, the average quality of the population should increase, and therefore yield better offsprings in further iterations. The algorithm stops when either a time or iteration limit is reached, or when the population converges.

The individual parts of the MA are explained in detail in the rest of this chapter.

5.2.1. Initialization

The initial population is created through a modified I1 insertion heuristic (see Solomon 1987) which is explained as follows. The goal is to generate a population of size *popsize* with very distinct but high quality solutions. Therefore the I1 heuristic is enhanced with a stochastic insertion criterion so that the initialy deterministic heuristic can create different solutions. The modified I1 heuristic is composed of two stages. In the first stage, the clustering stage, all customers are assigned to their geographically closest depot. We are aware that there exist more sophisticated methods for assigning customers to depots (see e. g. Salhi and Sari 1997), however we opted for a much simpler and faster approach as we did not want to focus on construction algorithms. It is to note that the generation of the initial population is very fast and the MA can improve the quality of the population relatively fast, so that the additional time spend in building it does not necessarily improve the outcome. In the second stage, the routing stage, K empty routes are generated. Each depot is assigned K/m routes so that the vehicle fleet is distributed according to the problem-specification. Each customer is then tentatively inserted into a route at each possible position, and the resulting insertion costs are saved in a sorted list. After all customers have been tentatively inserted, one entry in the list is chosen by the stochastic insertion criterion out of the three highest entries in the list. The method for selecting an entry is as follows. The first entry is chosen with a probability of 0.5 and the second and third entry both share the remaining possibility for insertion. When a entry is selected, the customer is definitely inserted at the according place. Customers can be inserted at parallel in each route. A route is considered complete when no more customers can be feasibly inserted. Highly constrained problems might leave some customers unassigned, as they might not be inserted without generating violations. These customers are then inserted at the places where they generate the smallest violations. The heuristic stops if all routes have been completed or all customers have been assigned. As a final step the restricted 3-opt operator (see Section 4.2.3) is applied to the newly generated solution. We want to point out that the stochastic savings heuristic as described in Section 4.2.1 can also be used to generate the initial population. However while the stochastic savings heuristic is more efficient in generating solutions for problems of large size, the stochastic I1 insertion heuristic does perform better at solving instances that have very tight time windows. Therefore both initialization methods are used, depending on the problem at hand.

5.2.2. Selection

The fitness evaluation function of a solution S follows the implementation in Section 4.2.4. The selection procedure follows the idea of binary tournament, where two solutions S_1 and S_2 are randomly selected from the population *pop* and are evaluated by the fitness function. The better of the two individuals is then accepted as the first parent for the recombination. The selection procedure then restarts, for the selection of the second recombination partner, which has to be different to the primarily selected one.

Different methods, like a roulette-wheel selection (see Goldberg 1989), or a completely random selection were implemented and tested as well. However the binary tournament method proved to be the most efficient and successful method for selecting customers. We assume that selection methods like the roulette wheel, may not be appropriate when using Equation 4.2 for the evaluation. Using this equation a selection would mostly resemble a random selection if all individuals in the population are feasible, while being heavily biased towards feasible solutions if infeasible solutions are present in the population. The binary tournament method however always chooses the better of the two individuals, and therefore does not have these drawbacks.

5.2.3. Recombination

The reproductive process in evolutionary algorithms is simulated trough the use of specially designed recombination operators. In literature a couple of standard operators were developed (for an overview see Bräysy et al. 2004) that were successfully applied to different types of VRPs. These crossover operators usually work by recombining two selected solutions so that the offsprings hopefully inherit the good attributes of both parents. Sophisticated crossover operators like those presented by Prins (2004) are difficult and time expensive to implement due to the large problem size as well as the extensions like time windows and multiple depots. We therefore opted for a specially designed standard operator that is computationally inexpensive. The used operator is a route based two-point crossover operator (see Bräysy and Gendreau 2005b).

In detail the developed operator works as follows. The operator creates two offsprings O_1 and O_2 by combining, one at a time b out of a maximum of B pair of routes, R_{1b} of parent solution S_1 with R_{2b} of parent solution S_2. S_1 and S_2 are selected through the binary tournament selection method explained in Section 5.2.2. O_2 is generated by interchanging S_1 with S_2 and applying the crossover operator with the same parameters for a second time. The fitter of the two offsprings is then kept for entering the population. The number of routes B that are recombined, is randomly drawn between one and the maximum possible combination of routes, with a bias towards small values. The probabilities for the amount of routes selected are 0.99 for one pair of routes, 0.0075 for two pairs and the remaining probability is equally distributed between three and the maximum number of pairs. The bias for selecting only one route for recombination is set this high so that the offsprings that are generated are not highly infeasible. The probability to recombine two or more routes has to be existent to break out of degenerated populations.

After the pairs of routes are chosen the recombination is executed in the following way. The pairs of routes are randomly cut into three sequences, were the length of the middle sequence is at most the length of the smallest route diminished by the position of the first customer of the middle sequence. The sequence length stays the same for R_{1b} and R_{2b} as well as the starting positions of the sequences. This is done so that time window violations can be anticipatively minimized. After the exchange of the two middle sequences in all pairs of route is done, the solution is checked for missing or duplicated customers. The procedure then continues by erasing duplicate customers out of the routes where they appeared before the recombination step. All of the remaining missing customers are then inserted at the cheapest possible position, where cheapest is defined by the evaluation function. They are inserted by an I1 insertion heuristic proposed by Solomon (1987).

The used crossover operator is illustrated in the following example:

$$R_{1b}\ (1\ 2\ |\ 3\ 4\ 5\ |\ 6\ 7)$$
$$R_{2b}\ (5\ 2\ |\ 3\ 1\ 4\ |\ 9\ 6\ 7\ 8)$$

where R_{1b}, R_{2b} are the chosen routes for pair b from S_1 and S_2 that produce the following routes R_{O1b}, R_{O2b} of the offsprings O_1 and O_2 after swapping the middle sequence.

$$R_{O1b}\ (1\ 2\ |\ 3\ 1\ 4\ |\ 6\ 7)$$
$$R_{O2b}\ (5\ 2\ |\ 3\ 4\ 5\ |\ 9\ 6\ 7\ 8)$$

The new solutions need to be repaired in a way that no double or missing customers exist. The final routes of the offsprings can then look like this:

$$R_{O1b}\ (5\ 2\ |\ 3\ 1\ 4\ |\ 6\ 7)$$
$$R_{O2b}\ (2\ 1\ |\ 3\ 4\ 5\ |\ 9\ 6\ 7\ 8)$$

The best of the two offsprings is then used to update the population.

5.2.4. Mutation

A Stochastic Local Search procedure based on the VNS metaheuristic is applied to modify existing solutions as well as newly generated ones. The goal of this procedure is to better explore the search space as well as to overcome local optima. A reduced and much faster version of the VNS described in Chapter 4 was implemented as mutation operator. In detail the utilized VNS uses CROSS-Exchange neighborhoods in the shaking phase so that the shaking operator can swaps two sequences of customers belonging to two different routes. This leads to the possibility of perturbing the solution and reaching more distant neighborhoods. The maximum allowed sequence length is fixed as well as the number of depots involved in a move. The 12 different neighborhoods used in our VNS frame ($\kappa = 1, \dots, 12$) are shown in Table 4.1 where C_k denotes the number of customers assigned to route k. After the swapping of the sequences, a 3-opt (see Section 4.2.3) that is restricted to sequence length $sl = 3$, is used to bring the newly generated routes into local optimum. In comparison to the VNS described in Chapter 4 only better solutions are accepted. Another difference to the VNS described in Chapter 4 is the prohibiting of selecting the same route for doing a CROSS exchange, which therefore results in prohibiting intra-route moving of customers.

The described stochastic local search procedure is applied to each newly generated offspring as well as to solutions already in *pop* with different probabilities p_1 and p_2 respectively where $p_2 = p_1/10$. Because the focus of this step is a mutation of a solution in a desired direction the VNS stopping criterion is set to a small amount of iterations $it_{vns} = 100$, where one iteration is defined as a shaking step. Further, the mutation rate $p_1 = 0.1$ is also set relatively low so that the whole process is inexpensive with regard to computational time while at the same time allows to break out of local optima.

5.2.5. Population Management

After the generation of the initial population, the algorithm starts with the selection of individual solutions for recombination. The recombination operator then generates two offsprings, from which the better one is allowed to enter *pop*. This is done by updating *pop* in a steady state fashion (see Whitley 1987). A new solution is therefore allowed to enter *pop* if it is fitter than the worst solution in *pop*. The population is implemented as an array of chromosomes sorted by their fitness values. As *popsize* is fixed, when a new solution enters *pop* it has to replace an already existing solution. This is done by randomly replacing one of the $popsize/2$ worst solutions in *pop*. To save computational time, fitness values for whole solutions as well as for individual routes are stored in the chromosomes, and need only to be re-evaluated when a change in the chromosome occurs. Clones are detected by comparing the fitness values of the stored solutions, where identical solutions are defined by identical fitness values.

As the updating of the solution is driven by the fitness evaluation function, it can not be guaranteed that feasible solutions are present while the MA is running or when the algorithm stops. Therefore every time a new feasible solution is found, it is always saved if it is better then the previously stored one. In the case that at the end of the calculation, the population of the MA does not contain a feasible solution or the stored solution is better than the best solution in the population, the last saved solution then presents the final solution. We decided for a relatively small population ($popsize = 10$) to obtain a faster increase of solution quality, with the tradeoff, of a faster degeneration of the population.

6. Ant Colony Optimization

In this chapter an introduction as well as a literature review is given about ACO in the field of VRP. We also present the basic design of the ACO that was developed to solve the MDVRPTW, as well as the implementation of the ACO into the MA so that it can enhance the solution finding process.

6.1. Introduction and Literature Review

Ant Systems have received increasing attention by researchers since their development by Colorni et al. (1991). Different types of systems were developed for a broad range of different applications, spanning the fields of Graph Coloring Problems, the Quadratic Assignment Problem, the Traveling Salesman Problem or the Vehicle Routing Problem. The convergence proof by Gutjahr (2002) further underlines the importance of the Ant System metaheuristics in the field of optimization problems. The general principle of the Ant System approach, resembles the behavior of real ants that are searching for food. When ants are searching for food, they leave a certain aromatic essence called pheromone on the paths they traverse. If no pheromone is present at a certain location, ants perform a random walk when searching for food. However as soon as they reach a path were pheromone is present, they are more likely to stop the random walk and follow the pheromone trail. The tendency with which the ants decide which path to follow is directly related to the strength of the pheromone smell on the paths. If they now traverse on a already established path, further pheromone will be spread, so that the probability for selecting this part further increases. Because pheromone can vaporize, only the shortest and therefore the ones with the highest pheromone concentration will remain and be traversed so that the path of the ants from the food-source to the nest is minimized. In literature Ant Systems have proven to be efficient in solving different types of optimization problems.

Reimann et al. (2002a) developed an Ant System for the VRPBTW that uses an insertion procedure to construct solutions. The main contribution of this work was the changing of the Nearest Neighbor construction procedure (NN) as used in traditional Ant Systems (see e. g. Bullnheimer et al. 1999) to a more powerful insertion based procedure. The used insertion procedure is based on the Solomon (1987) I1 insertion heuristic. Un-routed customers are inserted into a route at all possible places and the according attractiveness values are stored. A roulette wheel selection method (see Goldberg 1989) then chooses a customer location out of all positive attractiveness values. After a solution is constructed a local search procedure tries to improve each solution.

Other approaches that uses a specialized route construction procedure are *"A Savings Based Ant System For The Vehicle Routing Problem"* by Reimann et al. (2002b). and *"D-Ants: Saving Based Ants divide and conquer the vehicle routing problem"* by Reimann et al. (2004). Both use a savings based procedure to construct new solutions. The algorithms are highly competitive in solving different standardized instances.

"An external partial permutations memory for ant colony optimization" by Acan (2005) shows that retrieving partial solutions with good features out of a external memory and then finalizing them through an Ant System approach results in significant performance achievements on terms of convergence speed and solution quality. This approach was then further developed into. *"A shared-memory ACO+GA hybrid for combinatorial optimization"* by Acan and Unveren (2007). This approach is hybridizing the search capabilities of a Genetic Algorithm with the capabilities of ant colony optimization algorithms. The two searching strategies work in parallel on two different populations of solutions and interact with each other through the use of a shared memory. This shared memory contains partially incomplete solutions that are of high quality. A new solution is generated by extracting a incomplete solution out of the shared memory which is then finalized through one of the two search strategies. The interaction of this two approaches through the use of a shared memory results in better solution quality then by using each of the approaches on its own when applied to TSP and QAP problems.

In the rest of this chapter the developed Ant Colony Optimization algorithm is described. It was applied to the standard sets of Cordeau et al. (2001b) to solve the MDVRPTW. However because of the size of the problem instances the obtained results could by far not compete to the TS by Cordeau et al. (2001b), and the VNS by Polacek et al. (2004) or the developed MA approach (see Section 5) because of high computational expenses. Nevertheless the solution building process of the ants worked satisfyingly, and the ACO approach was therefore modified and adapted so that it can enhance the developed MA approach (see Section 6.3).

6.2. Basic Design of the ACO

The basic design of the ACO is shown in Algorithm 3 and follows the idea of Colorni et al. (1991).

Algorithm 3 Basic design of the ACO

```
  Initialize pheromone information
  while $iterations < maxiterations$ and $time < maxtime$ do
    Generate $u$ solutions by ants according to heuristic and pheromone information
    Application of a local search to each of the ants' solutions
    Update of the pheromone information
  end while
```

To start the ACO process, the pheromone information needs to be initialized and assigned with values. After this is done, the algorithm starts to construct new solutions through the use of some attractiveness value η. This value inherits the pheromone information as well as the heuristic information, which is usually a distance measure. After the constructive heuristic has generated a solution, a local search procedure tries to improve it. If a newly constructed solution matches some specific criteria, like a certain solution quality, it is allowed to alter the pheromone information. At the same time pheromone globally evaporates, so that undesirable information eventually vanishes. Only allowing good solutions to lay pheromone, and the steadily evaporation of information, eventually will result in a convergence of the algorithm. Finally the algorithm stops, when a certain time or iteration limit is reached.

6.2.1. Pheromone Initialization

The pheromone information is stored in a matrix for each connection between customers and depots and is therefore of size $(n+m)\times(n+m)$. At the beginning the complete matrix is initialized with values of 1. The ACO algorithm was developed to solve the MDVRPTW instances and this is accounted for by introducing one pheromone matrix for each depot. The initial matrix is therefore copied m times resulting in a pheromone matrix of size $m \times (n+m) \times (n+m)$. The idea to generate multiple entries for each connection between customers comes from the fact, that there may be differences how customers are located in relation to depots. Two customers that are well connected by being near to each other may be ideally placed in a route starting from one certain depot, but this may not be the case if the route starts from another depot. This is often the fact when customers have matching time windows, where they can't easily be shifted to other positions in a route.

6.2.2. Solution Building Process

The work of Reimann et al. (2002b,a) clearly shows that specialized construction algorithms should be developed to solve vehicle routing problems. The authors mention that savings based procedures do not work very well with constraints like time windows. We therefore adapted the idea of developing a insertion procedure based on the I1 heuristic by Solomon (1987).

The Insertion Procedure is shown in algorithm 4.

Algorithm 4 ACO Insertion Heuristic

```
Initialize $m \times K$ routes with seed customer
for each un-routed customer do
    Calculate attractiveness value at each possible position
end for
Roulette wheel selection of a customer/position combination by attractiveness value
Insert the selected customer into the route
```

In detail the procedure works as follows. In the first step K empty routes are initialized for each of the m depots. A roulette wheel procedure, then selects a customer by its attractiveness value for each empty route. The attractiveness value η is calculated according to Equation (6.1).

$$\eta_{ijz} = \alpha \times \frac{1}{FV_{new} - FV_{old}} + \beta \times \frac{\tau_{jiz} + \tau_{is_jz}}{2 \times \tau_{js_jz}} \tag{6.1}$$

where FV_{old} is defined as the fitness-value before the insertion of the customer i after customer j in a route belonging to depot z, and the value of FV_{new} is calculated after this insertion. The fitness-values are calculated according to Equation (4.2). α and β are weights for the heuristic and pheromone values respectively. The heuristic value is therefor defined as the difference in distance and any penalties through violations when inserting a customer. The pheromone concentration τ_{jiz} contains the information how well the combination of a customer i and a customer j at depot z immediately after each other was in the previous iterations. s_j represents the customer that was immediately after j before the insertion. The second term in Equation (6.1) therefore is larger than one if the average pheromone of the arcs to be added, is higher then the pheromone of the arcs to be deleted. The seed customer for a empty route is chosen in a way that a the farthest yet un-routed customer to the depot is inserted. After all attractiveness values have been calculated, a roulette wheel procedure then randomly selects a value for the final insertion into the route out of the 10 highest values. Applying the selection procedure on a limited set of values guides the construction process in the direction of a faster decline in fittnessvalue.

6.2.3. Pheromone Update

A rank based scheme (see Bullnheimer et al. 1999) with $p = 3$ ranks, was implemented to update the pheromone information. After all u solutions have been generated the pheromone information is updated according to Equation (6.2) (cf. Reimann et al. 2002a).

$$\tau_{ijz} := \rho\tau_{ijz} + \sum_{\mu=1}^{p} \Delta\tau_{ijz}^{\mu} + \sigma\Delta\tau_{ijz}^{*} \tag{6.2}$$

Here ρ is defined as the trail persistence ($\rho = 0.95$) and $\sigma = p+1$ which amounts for the number of elitists. The equations shows two terms which represent the pheromone that is laid by the elitists. Where the first term is calculated as shown in Equation (6.3) and the second as shown in Equation (6.4).

$$\Delta\tau_{ijz}^{\mu} = \frac{p - \mu + 1}{FV^{\mu}} \tag{6.3}$$

$$\Delta\tau_{ijz}^{*} = \frac{1}{FV^{*}} \tag{6.4}$$

FV^* is the fitnessvalue of the best solution generated by the u ants. The second term therefor allows laying pheromone with strength σ into the pheromone matrix of the according depot. The first term then allows to lay pheromone according to the fitnessvalue FV^{μ}, where μ is defined as the rank of the solution. Better solution are therefore allowed to lay a higher pheromone concentration. Pheromone on the arcs that are in neither of the σ solutions finally evaporates at rate $1 - \rho$.

6.3. Implementation in the MA

The algorithm explained in Section 6.2 was applied to the standardized instances by Cordeau et al. (2001b). Precalculations have shown, that the developed ACO algorithm that uses a parallel insertion method can construct solutions with relatively good quality, but at extremely high computational expenses. In detail feasible solutions could only be calculated for the smallest problem instances, while calculation had to be aborted for the datasets containing a high number of customers. We therefor had to conclude that the developed algorithm is by far not competitive to the other approaches developed for the MDVRPTW. However the ACO algorithm was used to enhance the search of the MA described in Chapter 5. This was done in the following fashion.

As soon as the MA got stuck in local optimum, which we defined as a number of unproductive iterations $it_{stuck} = 10^5$ the solutions of the current population get replaced by newly generated ACO solutions. The 3-opt procedure explained in Section 4.2.3 is then applied to each newly generated solution. To follow the ideas of an external memory approach, pheromone information is generated by the MA. In detail every 10^4 iterations of the MA the pheromone values are updated and evaporated according to Equation (6.2) for the $p = 3$ ranks. After the reinitialization of the population the counter for unproductive iterations is set to zero, and the MA continues. More than one restart of the population is allowed until the stopping condition is meet.

7. Solving Strategies

In Chapters 3, 4 and 5 we have described all of the basic components used in the following two decomposition approaches. The first approach *"Popmusic for a Real World Large Scale Vehicle Routing Problem with Time Windows"* by Ostertag et al. (2008b) is a decomposition approach for a population based method. An MA is used as the optimizer in the POPMUSIC framework with the intention to store as much information gathered over the iterations in the population. A special design of the POPMUSIC framework therefore accounts for not destroying good population structures when generating new sub-problems. In the second strategy, *"A Variable Neighborhood Search Integrated in the POPMUSIC Framework for Solving Large Scale Vehicle Routing Problems"* by Ostertag et al. (2008a) a VNS is used as an optimizer that only manipulates a single solution. More sophisticated and different methods how relatedness is defined and how sub-problems are generated could therefore be developed and tested.

The design issues that needed to be addresses (see Section 3.2) are explained in the rest of this chapter.

7.1. Decomposition Strategies for Population Based Methods

For this strategies the MA algorithm as explained in Chapter 5 was used as an optimizer in the POPMUSIC framework. The principal components and design issues are explained in the following sections.

7.1.1. Obtaining an Initial Solution by Clustering

The effectiveness of the stochastic I1 heuristic explained in Section 5.2.1 is strongly influenced by the size of the problem. To overcome the problem of a long solution building process that may not yield high quality solutions, the customers are first clustered, so that each cluster builds a much smaller MDVRPTW from which the construction heuristic can then create an initial solution. The partitioning of customers is achieved by solving a relaxation of a capacitated p-Median problem (see Hakimi 1965; Taillard 2003; Waelti et al. 2002). This is done in the following fashion.

For each cluster, the distances between customers are modified with Lagrangian multipliers, so that routes can be built for which the overall demand is balanced. To initialize the algorithm, all multipliers are set to 0, so that a standard p-Median problem can be solved. In the next step, for each cluster c the overall demand Q_c is computed and compared to the global capacity V_c , where V_c is the sum of the capacity of all vehicles assigned to cluster c. In the case of $V_c < Q_c$, it is not possible to deliver all customers allotted to cluster c. If this happens, the Lagrangian multiplier λ_c associated to cluster c is increased by a certain amount that depends on the ratio Q_c/V_c. Then the distance c_{ij} between two customers, i and j is modified to create a new distance measure Π_{ij} shown in Equation (7.1).

$$\Pi_{ij} = c_{ij} + \lambda_c \cdot d_i \qquad \forall i, j \in c \tag{7.1}$$

It can be seen that the new distance measure incorporates length ($c_{i,j}$) and demand (d_i) units, therefore the Lagrangian coefficients λ_c must be multiplied by a factor that balances the influence of both units. According to Equation (7.1) all distances are calculated so that the p-Median solver can be restarted to decompose the problem again. The computation of new distances, and the p-Median solver process are repeated until a feasible decomposition can be found or an iteration limit is reached. It is to note that at this stage of the process, violations concerning the capacity constraints can be relaxed, as routes are not determined at this point. Customers therefore can shift between routes so that capacity constraints can be met. The main advantage of this relaxation is that constraints other than capacity (pick-up, time windows) can be added while using a common p-Median solver. The basic layout of the algorithm is presented in Algorithm 5.

Algorithm 5 p-Median decomposition algorithm

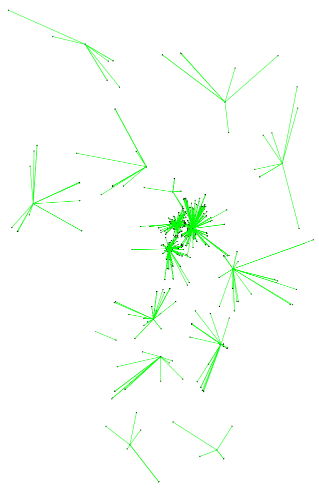
```
Input: MDVRPTW, number of clusters $p$, iteration limit $it_{dec}$
Build $p$-Median problem according to MDVRPTW customers
Allocate $K/p$ vehicles to each cluster $c$ and compute maximum capacity $V_c$
Set $Q_c \leftarrow \infty \;\; \forall c$, $\lambda_c \leftarrow 0$, $it \leftarrow 0$
while ($Q_c > V_c \;\; \forall c$) and ($it \leq it_{dec}$) do
    Solve $p$-Median problem with modified distances $\Pi_{ij}$ (see Taillard 2003)
    Compute overall capacity $Q_c$ of each cluster $c$
    Update $\lambda_c$ coefficients according to capacity constraint violation
    Set $it \leftarrow it + 1$
end while
```

7.1.2. Better Balancing Customers Between Clusters

The p-Median decomposition procedure assigned customers to clusters $s_1, ..., s_p$ in the initial phase. A typical solution of the p-Median decomposition can be seen in Figure 7.1 as an example of our real world problem instances that will be introduced in detail in the next chapter. Figure 7.1 gives an overview of the whole instance, while Figure 7.2 is a close-up of a densely populated area. In this special case the city of Vienna is shown which inhabits the major amount of customers that need to be served. It can be seen that most of the customers are located in a small geographic region in the center. The p-Median decomposition procedure therefore can pack a large amount of customers into the same cluster in highly populated regions, while only a handful of customers may be assigned to a cluster in low-density regions like the country-side. This feature may not be the perfect starting point for further calculations, therefore a preprocessing procedure, as described by Algorithm 6, tries to level out the number of customers inside clusters with the intention to destroy as less of the p-Median decomposition as possible.

In detail, the sub-problem optimizer (MA) is applied with it_{ini} iterations on each cluster to generate the first routes; which then build the first sub-solution. In the next step, clusters that exceed a certain size ($csize = 75$) are split with the help of a Sweep algorithm (Gillet and Miller 1974). In detail this is done in

Figure 7.1.: p-Median decomposition

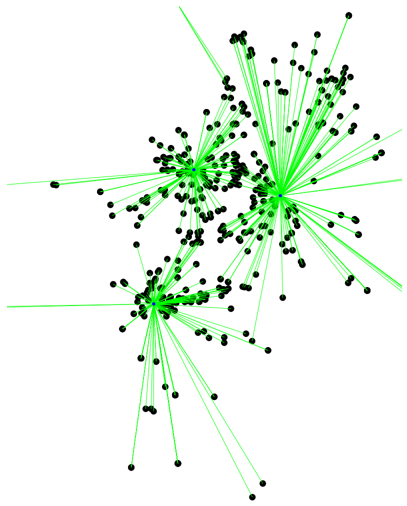


the following way. The center of gravity (cf. Reimann et al. 2004) is calculated for each route in the cluster to represent its aggregated customers. The Sweep algorithm then splits the clusters by the centers of its routes, with the starting point being randomly chosen. The algorithm then sequentially adds routes in a clock-wise fashion until the amount of customers in the added routes reaches $csize$ customers. If the limit is reached, the routes selected by the Sweep algorithm form a new cluster. The algorithm then restarts and tries to split the remaining cluster

Algorithm 6 POPMUSIC Initialization Phase

```
Assign customers to clusters by $p$-Median decomposition
Run $it_{ini}$ iterations of the MA on each cluster to build initial routes
**if** customers in $s_1, ..., s_p > csize$ **then**
    Start splitting procedure
**end if**
**if** customers $s_1, ..., s_p < csize$ **then**
    Start leveling procedure
**end if**
**if** amount of routes higher than $K \times m$ **then**
    Start repair procedure
**end if**
```

Figure 7.2.: Zoom in on p-Median clusters



as long as it still contains more than *csize* customers. If no more clusters can be split, the procedure stops.

The remaining clusters that were not split are then checked if they could be merged to form new clusters with a size smaller than *csize*. Clusters are merged by a greedy heuristic that uses the distance of the centers of gravity of each cluster; meaning that close clusters are merged first if both of them together contain less than *csize* customers.

When no more clusters can be merged, feasibility regarding the vehicle fleet is checked. If the solution is not feasible, the excess routes are randomly deleted, and the remaining customers are inserted by a I1 heuristic. Note, that the number of available vehicles in the real world problem tackled is far sufficient to perform the deliveries, so that the repair step was never executed over all conducted test runs.

7.1.3. POPMUSIC Customization

This section gives an overview on how the principal components of the POPMUSIC framework are customized to the problem at hand. In the case of the MDVRPTW, we defined a part as a route. The proximity measure, that puts parts into relation to each other, is defined as the distance between the centers of gravity of the entities. This entities can be a single route, or a set of routes (cluster of routes). The used distance measure therefore relates to the distance of aggregated customers. In the preceding steps, the creation of the initial solution, the customers were first clustered by solving a capacitated p-Median problem and are then balanced so that each cluster is around the same size. Each of them can be considered a small and independent MDVRPTW, where each route only visits customers that belong to the same cluster. In the POPMUSIC framework, a seed part, in a VRP environment, would usually be defined as a single route, however we opted to define a seep part as a cluster of routes. The center of gravity of the routes composing the cluster is computed and the seed part is extended by adding r new routes. Routes are chosen by their proximity to the cluster they will be added to in a greedy fashion, meaning nearest routes first. The reason for this modification is the possibility for using previously calculated information that is stored in the population. When talking about seed-parts or related routes, we always correspond to the best solution in the population. So after a seed-part (cluster of routes) is selected the corresponding individuals of the population are saved. Therefore when a cluster is extended by a route (related-part), this route can simply be added to each of the individuals in the population as it will always generate a feasible solution. In the next step, the route that left a cluster, has not only to be removed from the best solution but from the population as well. Since not all customers necessarily are located in the same route over all individuals in the population, the removed customers need to be purged at the according locations in the remaining individuals of the population. Reconnecting the routes guarantees formerly feasible solution to stay feasible without the need of extensive repair functions.

The generated sub-problem is therefore a subset of routes that can be treated as a small, independent MDVRPTW with an attached population, which is then solved by the optimizer. Different settings for parameter r were tested, but adding only one route ($r = 1$) to the seed part, resulted in the best solutions. The used optimizer is the MA described in Chapter 5. As shown in the results section (see Chapter 8) it turns out that the MA could find the best known solutions for all instances up to size 75 except one (where it only deviates by 0.08%) within reasonable computation time.

7.1.4. Different Strategies

While some work was done on developing methods to solve large scale VRP and VRPTW (see Homberger and Gehring 2005; Mester and Bräysy 2007, 2005; Kytöjoki et al. 2007), to our best knowledge no related work on real world MDVRPTW of large scale exists in literature. Therefore no data for comparison exists, and we decided to set up three different strategies to tackle the large real world problem to get a feel how good our decomposing approach can handle this type of problem. For comparison issues we used time as our stopping criterion for the following three strategies.

Strategy I (no decomposition)

This strategy is the most basic strategy. It solves the problem as a whole through the use of the pure MA until a certain amount of time is elapsed .

Strategy II (fixed decomposition)

This approach uses the initial clustering by the p-Median algorithm to solve the problem. Each of the generated clusters was treated as an individual MDVRPTW problem without further interaction between the clusters. Because the resulting problems vary in size the time allowed t_i for each problem s_i has to be shared in a fair manner. We decided to make the allowed time dependent on the square of the problem-size C_{s_i} in relation to the total problem-size C_{s_n} and the maximum time t_{max} allowed. This should help to put some bias on solving the larger clusters as they are significantly harder to solve. Equation (7.2) was used to calculate the corresponding times for each sub-problem.

$$t_i = t_{max} \cdot (C_{s_i})^2 / \sum_{n=0}^{p} (C_{s_n})^2 \tag{7.2}$$

After all sub-problems have used up their time limit, the complete solution was generated by simply merging the sub-solutions of the single clusters.

Strategy III (POPMUSIC)

This strategy was executed with two different parameter settings. The first setting $IIIa$ focuses on a longer search, while the second setting $IIIb$ was tuned for an "as-fast-as-possible" solution finding process. This was mainly done by giving strategy $IIIb$ less time to improve the sub-problems than in $IIIa$ to emphasize on a faster descent of solution quality, at the cost of a higher chance to miss the global optimum.

7.2. Decomposition Strategies for Individual Solution Methods

The optimizer used for this strategies, is the VNS explained in Chapter 4. The design issues and the principal components used are described in the rest of this section.

7.2.1. Construct Initial Solution

As an initial solution needs to be fed into the POPMUSIC framework, we used a modified Clarke and Wright Savings algorithm (Clarke and Wright 1964) to construct this initial solution as explained in Section 4.2.1.

7.2.2. POPMUSIC Customization

Like in the previous strategies a part $(s_1, ..., s_p)$ is defined as a specific route in the complete solution. The applied proximity measure (*relation - most related to* s_i) therefore needs to measure the distance between two routes. As only one solution is manipulated at a time, compared to the strategies in the previous section where a stored population needed to be handled, more choices how sub-problems could be build arise. We therefore examined two completely different ways to measure proximity that will be presented in detail in the following subsection (Section 7.2.3). A sub-problem is defined as a subset of $r = 10$ routes, each of which can be treated and solved like an independent MDVRPTW.

All resulting sub-problems are then optimized by the VNS, that terminates if a certain iteration limit is reached. Since the resulting sub-problems depend strongly on the selection of the seed customer or part, seed parts are chosen in a systematic way. This means each route in the solution has to be the seed part at least once, before another route can be chosen a second time.

7.2.3. Different Decomposition Strategies / Proximity Measures

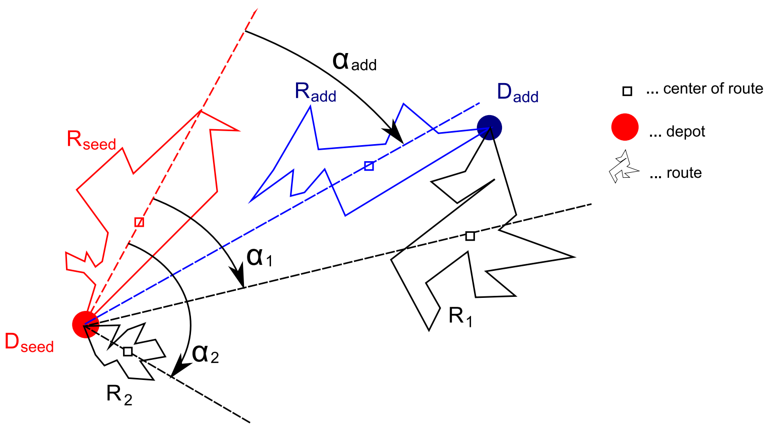
Two different relatedness measures for potential parts that can be added to a seed-part were examined and compared, to see which one will result into a better decomposition of the problem. The measures are different in the way how they define distance. While one measures distance by travel-time, the other does so through the use of trigonometric functions based on the Sweep idea (see Gillet and Miller 1974). Additionally a total of eight different strategies how to apply this measures of proximity were tested. Three of these measures are based on the Sweep idea, while the rest of the measures are building on a distance measure by travel-time.

Proximity Through Sweeping

This measure defines distance by the angle between the centers of two routes. The centers of two routes are defined as the centers of gravity of all customers in the according route. This concept of aggregating the customers was introduced in *"A tabu search heuristic for the vehicle routing problem with soft time windows"* by (Taillard et al. 1997) and built-on in *"D-Ants: Saving Based Ants divide and conquer the vehicle routing problem"* by Reimann et al. (2004). The Sweep algorithm then calculates the angle α between two centers, with the pivot point always being the depot D_{seed} of the seed route R_{seed}. The three Sweep measures are different in how they restrict the selection of route.

sweep with no restriction (S_I)
The most basic way to apply this measure is without any restriction on the selection of routes that can be added to create a sub-problem. A sub-problem is

Figure 7.3.: Measure S_I



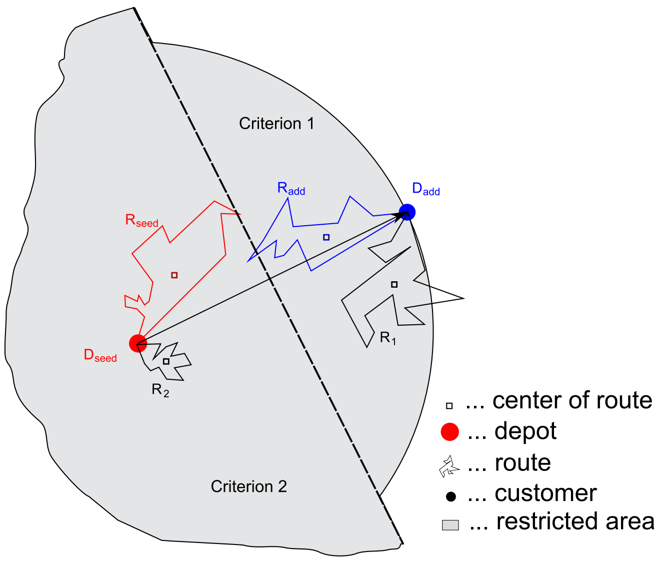
therefore created by adding the routes with the smallest angle up to the maximum allowed sub-problem size r. Any route can be selected for adding, even if most of the customers are far away from the center of gravity concerning the travel time. To avoid the creation of always the same sub-problems a diversification feature is introduced for the selection of a route. With probability of 0.1 a tentatively selected route is rejected to enter the sub-problem, and the algorithm continues with the next route that has the smallest angle to the seed route. The procedure is depicted in Figure 7.3.

sweep with tight restriction (S_{II})

This strategy resolves around the fact that even when angles between two routes are small, they don't necessarily have to be close to each other with regards to travel-time especially in the case when they belong to two different depots. We therefore restricted the selection of routes that belong to another depot than the seed part in the following way. Routes are still added with the smallest angle to the seed part, however they are only added when one of the two following criteria is fulfilled.

1. The distance between the route to be added and the seed depot is smaller than the distance between the two depots.
2. The route to be added is closer to the seed depot than to the original depot.

Figure 7.4.: Measure S_{II}



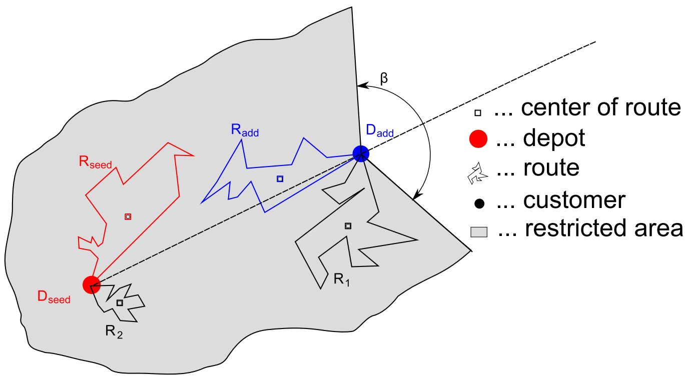
The sweep procedure with a tight restriction is depicted in Figure 7.4.

sweep with loose restriction (S_{III})

With this strategy we want to loosen the restriction when a route is allowed to enter a sub-problem. We did this by relaxing the criteria of S_{II} in a way that routes to be added that belong to another depot may be selected when their center of gravity does not lie behind the second depot D_{add} by a certain angle β. We therefore assume that routes in which customers lie behind another depot than the seed depot, should also be served from this depot, and therefore may not be closely related to the seed-part. However we did not simply prohibit the selection of routes that are "just-behind" the depot but allowed the selection of routes when they are not located in a certain sector. This sector is defined as the angle $\beta = 135$.

The whole procedure and the sector β is depicted in Figure 7.5.

Figure 7.5.: Measure S_{III}



Proximity by Smallest Distance

The second group of measures is different to the first one, as the travel time between two entities is used for proximity. The entities on which distance is measured are in this case single customers or all customers in a route. When a group of customers forms an entity the center of gravity of these customers is used to calculate the distance. Five different strategies were developed and tested. They basically differentiate in the aggregation level and on how customers are selected to join the sub-problems.

distance between aggregated customers of routes (D_I)

This strategy uses entities at the highes aggregation level. They are defined as the center of gravity for each route between which the distances are calculated in the following way. All distances between the seed route R_{seed} and all possible other routes are computed and stored in a sorted list. However like in the previous strategies based on the Sweep criterion, a mechanism to combat the creation of always the same sub-problems is set in place. It works in a way that only 75 % of the r routes with the shortest distance to the seed route R_{seed} may be added to the sub-problem. The missing routes are then selected with a roulette wheel procedure, where routes which are closer to the seed route have a higher probability of being selected. This method for selecting the routes is used in all of the remaining strategies.

Figure 7.6.: Measure D_I

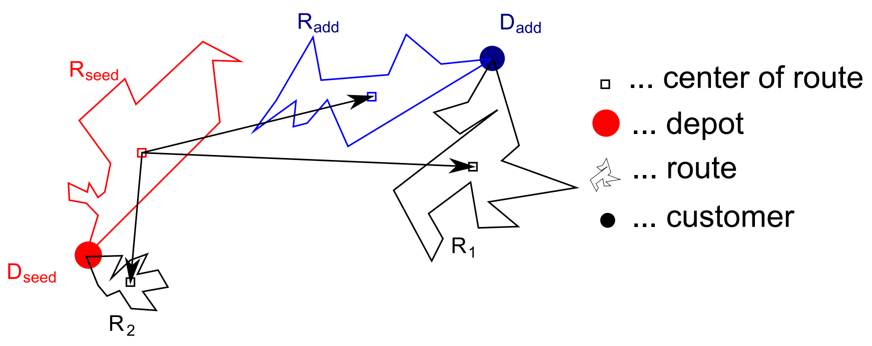
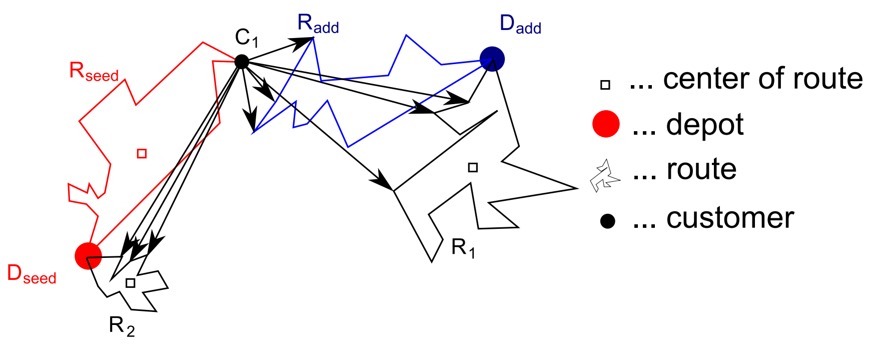


Figure 7.7.: Measure D_{II}



The strategy is illustrated in Figure 7.6.

distance between single customers of routes (D_{II})

Here, both aggregation levels are low, as entities are represented as single customers in the seed-route as well as in all remaining routes. A list containing the distances between the customers of the seed route and all other remaining customers is created. The customers with the smallest distance are then added to the sub-problem, however because a part is defined as a route, we have to add the complete route to the sub-problem. If a route enters the sub-problem the list is updated by removing all entries from customers belonging to already added routes. To conclude; $r-1$ routes are added to the sub-problem. The procedure is depicted in Figure 7.7.

distance between single customers of routes with restriction (D_{III})

This strategy is similar to strategy D_{II} concerning the aggregation level and the distance measure. However it extends the approach by adding a restriction on how customers can be selected. This is to counter the fact that strategy D_{II} mainly selects routes out of highly populated regions around the seed route. While this may be at first a desirable feature, it completely prevents tours that for example start in the city and serve customers in the hinterlands. This happens because distances are smaller on average in cities than in the country side. Therefore as soon as a route contains a customer in a city, the sub-problem is always extended with customers or routes in the same region. As a result; routes with customers in the country side cannot be combined reasonably with routes in cities. To overcome this structural drawback the selection procedure of D_{II} was modified.

In D_{III}, routes are still added by the smallest distance, however only one route per customer in the seed route may be added until all other customers in the seed route have added the same amount of routes. Therefore entries in the list are momentarily faded out if they contain a seed-customer that was selected more times than any other seed-customer. This restriction should help to minimize the bias towards highly populated regions.

distance between aggregated customers of the seed route and a single customer (D_{IV})

Here, we aggregated the customers in the seed part by calculating the center of gravity of the route. Distances are than calculated between these center and each remaining customer of the other routes. The aggregation level is therefore high in the seed-part and low in the remaining route. The sub-problems are then build in standard fashion. The procedure is depicted in Figure 7.8.

distance between a single customer in the seed route and aggregated customers of a remaining route (D_V)

Distances are calculated between a single customer in the seed route and the aggregated center of gravity of a remaining route. It is therefore the counterpart to D_{IV} concerning the aggregation levels. The procedure is depicted in Figure 7.9.

Figure 7.8.: Measure D_{IV}

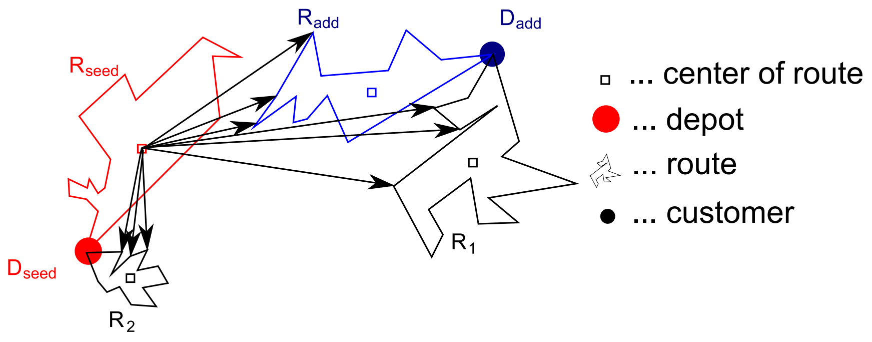
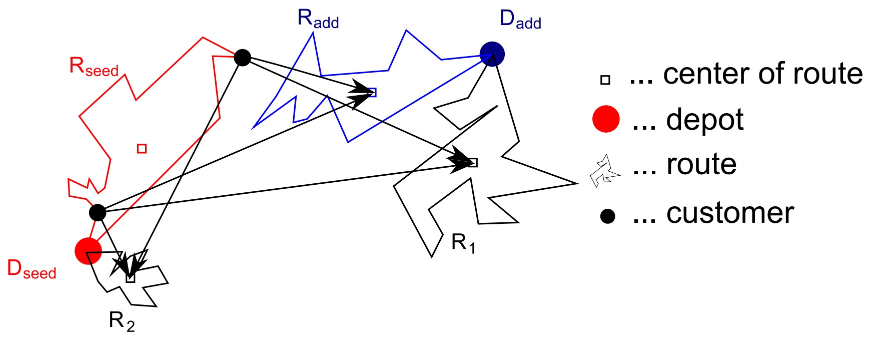


Figure 7.9.: Measure D_V



7.3. Implications - Population Based vs. Individual Solution

In this chapter we presented two general distinct decomposition approaches, that differentiate themselves through how the optimizer handles solutions. While the MA works in parallel on a population of solutions, the VNS only manipulates one solution at a time. The knowledge about the used optimizer therefore can be used to develop different decomposing strategies, that perfectly fit the algorithm used in the optimization step of the POPMUSIC approach. The idea of decomposing a problem into sub-problems, is to generate smaller solvable parts that consists of customers that are somehow close to each other. Compared to the VNS approach, where this creation of sub-problems is very flexible, the creating of sub-problems for the MA algorithm is slightly more complicated. Theoretically the same decomposing ideas can be used for the MA as for the VNS, however since the MA stores information not only in the singe-best-found solution, but also in the set of solutions in the population, other ideas to decompose can exploit this information so that it can be of further significance. The approach presented in the first part of this chapter, tries to do this, by incrementally changing the parts $s_1, ..., s_p$ of the solution S. Compared to the VNS approach where a seed part is defined as a route, the MA approach defines the seed part as a set of routes. Therefore the major advantage of this decomposition approach is that the information stored in the population can be reused, for the set of routes rather than dismissing it. The drawback however is, that sub-problems are relatively similar to each other. This may lead to a worse covering of the whole Solution space concerning the overlapping of sub-problems compared to the VNS approach. Figure 7.10 shows how a possible initial clustering can look like. It can be seen that the initial clustering does not overlap as each customer is assigned exactly once. Each of these initial clusters is then extended by r routes (parts) so that they can be optimized by the MA.

Figure 7.11 shows how each of the shown clusters can possibly be extended by related parts so that a new sub-problem can be generated. In this example there are four possible seed parts which can be chosen. This seed part is then extended with the most related (close) routes. This illustration shows that borderline cus-

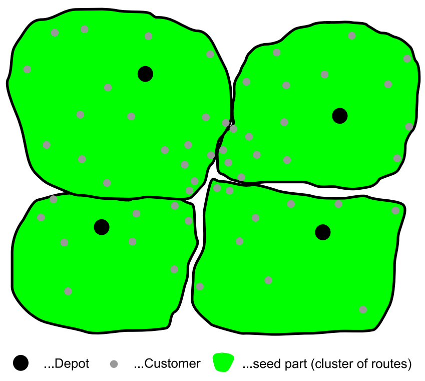


Figure 7.10.: cluster distribution - population-based

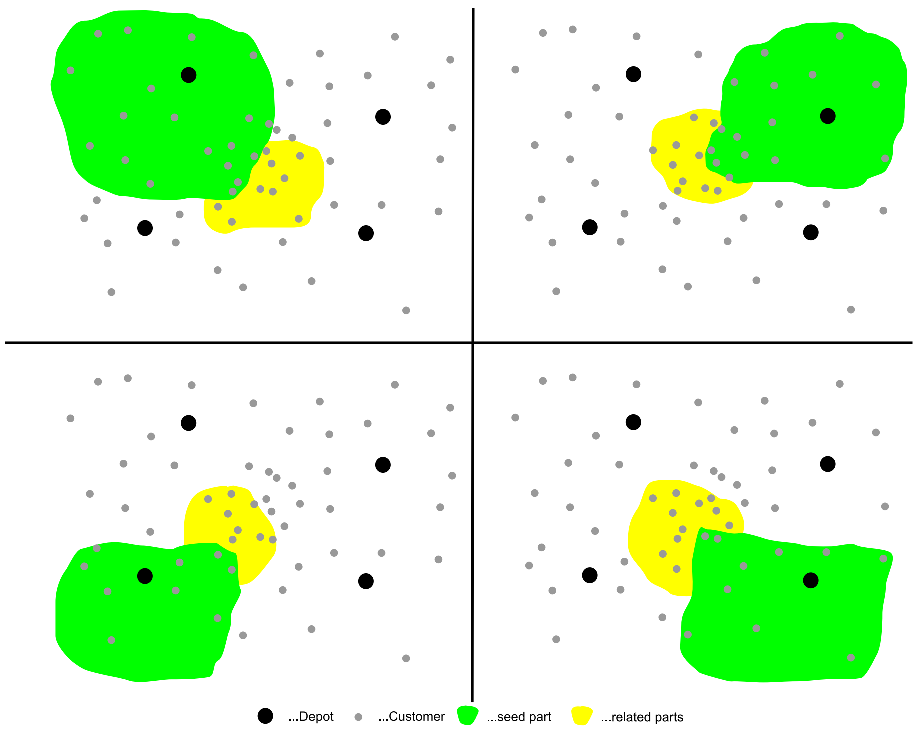


Figure 7.11.: decomposition population-based

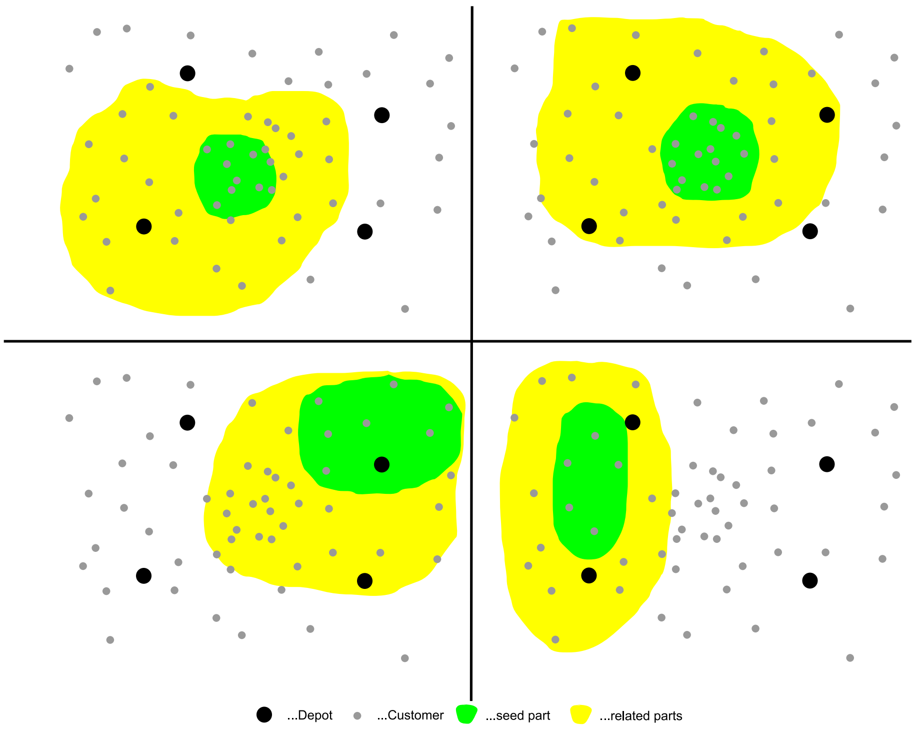


Figure 7.12.: decomposition single-solution-based

tomers can easily be transferred from one cluster to another so that at one time the best possible routing can be found. It can be seen that the seed-part, which contains a set of routes, is relatively large (we opted for a sub-problem size of 75 customers) compared to the related part, which contains $r = 1$ routes. The optimizer is then applied on the region of the solution that contains both parts. If the optimizer can improve the sub-problem, the related part enters the original seed part to form a new seed-part (cluster), therefore the routes in the related part need to leave another cluster. In each iteration a small portion of the region is therefore changing between clusters. This is somehow different to the mechanics of creating sub-problems for the decomposition approach for single-solution manipulation algorithms as shown in Figure 7.12.

For the single solution approach we decided to define a seed-part as a single route. Therefore it is obvious that more different seed-parts can be selected compared to the approach that defines a seed-part as a set of routes. As no population of solutions is existent, we are allowed to freely decompose the problem around the seed-part without any restrictions. Figure 7.12 shows four examples of how seed parts can be distributed among the whole problem. After a seed part is selected, $r = 9$ routes that are in proximity to the seed-part are selected to form the new sub-problem that is finally solved by the VNS-optimizer. In difference to the population based approach significantly more routes are added. However the resulting sub-problems are comparable in size for both algorithms. To conclude, the structural differences between the two approaches are the following. The population based approach tries to decompose the sub-problem in a way that information in the population can be re-used. This is mainly done by transferring the population of a previously optimized sub-problem to the new sub-problem and by modifying this population so that missing or double customers are erased. The idea is that the smaller the difference between one sub-problem to the next, the more information is useable. The incremental changing of sub-problems, therefore guides the search slowly around the complete problem. This is in contrast to the single solution approach, where a seed-route is randomly chosen, and the sub-problems are build around it. The resulting sub-problems can therefore by highly different from one iteration to another, which translates into a very erratic search in the complete problem.

8. Results

In this chapter, the results for the MA are presented for standardized instances and compared to the results of other state-of-the-art metaheuristics in Section 8.1. The results for the population based POPMUSIC that uses an MA as optimizer (see Section 7.1) are explained and discussed in detail in Section 8.2.1. The outcome of the single solution based POPMUSIC with an integrated VNS (see Section 7.2) as optimizer are presented in Section 8.2.2. Additionally the different proximity measures were tested, and the findings are analyzed in detail.

8.1. Standardized Instances

The data set was introduced by Cordeau et al. (2001b) and consists of 20 instances that are different in the amount of customers that need to be served as well as how tightly restricted the time windows to service them are distributed. Additionally the amount of depots m as well as the number of available vehicles K are different over the whole data set. The characteristics of the data set can be seen in Table 8.1.

The amount of customers that need to be served range between 48 and 288 customers. Only the 4 and 6 depot cases are considered, where between 8 and 30 vehicles serve the customers in the tightly restricted part of the data set, and only between 4 and 20 in the loosely restricted set. This decrease in vehicles should bring the second part of the data set on par concerning the difficulty to solve them. We decided to test the MA by doing 10 independent runs on each instance of the data set. To reflect real-world overnight calculations, each run had an 8 hour stopping-criterion ($t_{max} = 28800$) and was started with different seeds so that the robustness of the achieved results can be examined. To evaluate the effectiveness of the MA approach we decided to examine three different variations of the MA. One test run was done while prohibiting any restarts when the

Table 8.1.: Standardized data set (Cordeau et al. 2001b)

	narrow time windows				larger time windows		
No.	customers	depot	vehicles	No.	customers	depot	vehicles
1	48	4	8	11	48	4	4
2	96	4	12	12	96	4	8
3	144	4	16	13	144	4	12
4	192	4	20	14	192	4	16
5	240	4	24	15	240	4	20
6	288	4	28	16	288	4	24
7	72	6	12	17	72	6	6
8	144	6	18	18	144	6	12
9	216	6	24	19	216	6	18
10	288	6	30	20	288	6	20

population has degenerated. A degenerated population is characterized as a population in which no new features are introduced. For our algorithms we defined it as a maximum amount of unproductive iterations that may happen. We set the amount of iterations that may pass without replacing a solution in the population to $it_{unp} = 10^5$; where one iterations is defined by one recombination of two parent solutions. In the next setup we allowed a restarting of the population through the use of the modified I1 insertion procedure explained in Chapter 5.2.1. The last and most sophisticated approach uses the ACO-procedure described in Chapter 6 to inject new solutions into the degenerated population. The pheromone information is gathered and updated only every 10^4 iterations to make the whole process computationally less expensive.

We want to note that this instances do not qualify as large scale instances like encountered in the real world. Therefore we did not apply the decomposition approaches to them.

8.1.1. MA without Restarts

This section shows the obtained results by the MA (Chapter 5), when restarts of the population are prohibited.

Table 8.2.: Results MA without restarts

day	min	mean	max	stdev	$\%_{gap}$
1	1,074.12	1,079.29	1,083.53	4.48	0.48%
2	1,780.63	1,811.67	1,871.48	26.49	1.74%
3	2,433.83	2,467.21	2,514.01	24.81	1.37%
4	2,900.86	2,967.01	3,035.31	45.22	2.28%
5	3,052.57	3,163.03	3,269.60	74.96	3.62%
6	3,678.24	3,887.13	4,019.45	108.55	5.68%
7	1,425.29	1,444.82	1,483.47	17.82	1.37%
8	2,142.30	2,185.64	2,232.35	25.61	2.02%
9	2,797.89	2,817.16	2,844.00	15.47	0.69%
10	3,628.35	3,678.80	3,738.01	46.04	1.39%
11	1,005.73	1,020.73	1,054.30	20.71	1.49%
12	1,524.79	1,572.39	1,641.50	32.18	3.12%
13	2,049.26	2,112.39	2,152.20	33.44	3.08%
14	2,323.86	2,379.42	2,428.87	36.98	2.39%
15	2,589.54	2,706.07	2,821.64	74.12	4.50%
16	2,961.68	3,058.03	3,159.19	64.55	3.25%
17	1,290.01	1,339.08	1,441.69	48.98	3.80%
18	1,851.68	1,946.81	2,078.28	73.03	5.14%
19	2,358.79	2,427.39	2,556.50	54.44	2.91%
20	3,246.06	3,367.96	3,545.77	89.12	3.76%
avg.	2,305.77	2,371.60	2,448.56	45.85	2.70%

Table 8.2 shows the obtained results for the 10 runs for each instance. We report the objective-values for the best (min) the worst (max) and average (mean) solutions found over all runs. Additionally the standard deviation is calculated and reported in the stdev collum to gain further inside of the robustness of the approach. We report the gap between the average found solution and the best found solution of a run in collum $\%_{gap}$. In the last row of the table the averages over all instances are reported. The results show that the standard deviation is relatively low over all days in the majority of the cases. It can be seen that the gap between the best and average solution is high mostly for the harder to solve instances. Especially the results for the bigger instances (6, 15, 18) show a comparable higher deviation than for the smaller instances. We want to point out

that the average deviation for instances 1 to 10 is 2.06% and for instances 11 to 20 is 3.34% which may hint that the second part of the data set is harder to solve for the MA. Nevertheless the mean solution values over 10 runs are on average over all instances only 2.7% worse compared to the best found solution in these runs.

Table 8.3.: Comparison MA without restarts

day	MA	TS	VNS	RPD_{TS}	RPD_{VNS}
1	1074.12	1074.12	1074.12	0.00%	0.00%
2	1780.63	1762.21	1762.21	1.05%	1.05%
3	2433.83	2373.65	2373.65	2.54%	2.54%
4	2900.86	2852.29	2815.48	1.70%	3.03%
5	3052.57	3029.65	2993.94	0.76%	1.96%
6	3678.24	3627.18	3629.72	1.41%	1.34%
7	1425.29	1418.22	1418.22	0.50%	0.50%
8	2142.3	2102.61	2096.73	1.89%	2.17%
9	2797.89	2737.82	2730.54	2.19%	2.47%
10	3628.35	3505.27	3499.56	3.51%	3.68%
11	1005.73	1005.73	1005.73	0.00%	0.00%
12	1524.79	1478.51	1472.76	3.13%	3.53%
13	2049.26	2011.24	2001.83	1.89%	2.37%
14	2323.86	2202.08	2215.51	5.53%	4.89%
15	2589.54	2494.57	2465.25	3.81%	5.04%
16	2961.68	2901.02	2896.03	2.09%	2.27%
17	1290.01	1236.24	1236.24	4.35%	4.35%
18	1851.68	1792.61	1796.21	3.30%	3.09%
19	2358.79	2285.10	2292.45	3.22%	2.89%
20	3246.06	3079.16	3076.37	5.42%	5.52%
avg.	2305.774	2,248.46	2,242.63	2.41%	2.63%

Table 8.3 reports the results of the MA without restarts compared to the TS by Cordeau et al. (2001b) and the VNS by Polacek et al. (2004). We decided to not compare our results to the parallelized VNS version by Polacek et al. (2008a), because we wanted to focus on singe thread solution methods that can easily be implemented into the POPMUSIC framework. For the MA and the VNS the best found solution of the 10 runs is reported, while only one solution was obtained by the TS. The VNS solutions reported are found after 10^8 while the TS was

executed for 10^5 iterations. It is to note that our stopping criterion was time rather than iterations, but even though the authors give some insights on runtime we can hardly compare them concerning the computational effort as runtimes vary strongly between different hardware and implementations. The random percentage deviation (RPD) of the MA compared to the TS and VNS are reported in columns RPD_{TS} and RPD_{VNS}. It can be seen that both methods outperform the MA by 2.41% and 2.63% respectively. The MA can only find the best known results for the smallest two instances, with higher deviation directly related to the problem size. Additionally the random percentage deviation (RPD) of the results for the second part of the data set (time windows with loose restriction) are generally higher than for the first part, which resembles the assumption drawn before. The second part is therefore relatively harder to solve even though the time windows are not so restrictive. This is mostly due to the fact that the same amount of customers have to be serviced by a considerable smaller amount of vehicles.

8.1.2. MA with I1 Restarts

The results obtained by the MA shown in the previous section hint that the algorithm may get stuck in local optimum, from which it cant escape even when enough time is given. Therefore the MA was enhanced by allowing restarts as soon as the population might degenerate and therefore can't escape the local optimum. The restarting allows freshly generated solutions to enter the population so that they can help to lead the search into another direction so that it can overcome the local optima. The method used to generate the new solutions is the stochastic I1 heuristic presented in Section 5.2.1. Table 8.4 shows the obtained results for 10 runs on each instance when I1 restarts are allowed.

The objective-values for the best (min) the worst (max) and average (mean) solutions found as well as the standard deviation (stdv) over all runs are reported. The gap between the average found solution and the best found solution of a run is shown in collum $\%_{gap}$. It can be seen that standard deviation is slightly smaller compared to the MA without restarts. The gap between the best found solution and the average solution is 2.41% and is therefore smaller than when using no restarts. The average gap of the first part of the data set is 2.22% and 2.62% for

Table 8.4.: Results MA with I1 restarts

day	min	mean	max	stdev	$\%_{gap}$
1	1,074.12	1,084.80	1,099.68	10.00	0.99%
2	1,768.27	1,793.77	1,817.40	18.38	1.44%
3	2,402.99	2,442.05	2,509.98	31.76	1.63%
4	2,864.04	2,934.16	3,044.22	49.00	2.45%
5	3,031.80	3,147.44	3,341.93	91.41	3.81%
6	3,679.14	3,843.54	3,956.29	77.31	4.47%
7	1,425.29	1,436.38	1,459.58	13.19	0.78%
8	2,106.61	2,159.91	2,228.64	39.35	2.53%
9	2,786.82	2,840.63	2,909.03	40.17	1.93%
10	3,573.90	3,650.89	3,751.32	57.07	2.15%
11	1,005.73	1,010.89	1,045.08	12.62	0.51%
12	1,525.91	1,556.34	1,584.29	20.24	1.99%
13	2,027.48	2,084.00	2,191.56	58.01	2.79%
14	2,256.21	2,331.38	2,393.30	45.65	3.33%
15	2,600.48	2,663.77	2,742.99	41.83	2.43%
16	3,003.80	3,087.37	3,187.70	65.97	2.78%
17	1,269.09	1,318.93	1,469.03	63.28	3.93%
18	1,822.19	1,871.22	1,922.51	31.94	2.69%
19	2,346.85	2,378.02	2,405.15	20.73	1.33%
20	3,225.05	3,364.19	3,627.74	112.33	4.31%
avg.	2,289.79	2,349.98	2,434.37	45.01	2.41%

the second part respectively. Table 8.5 shows the results of the MA with I1 restarts compared to the TS and VNS. When looking at the random percentage deviation (RPD) of the MA compared to the TS (RPD_{TS}) and VNS (RPD_{VNS}) we can see that both of the algorithms beat the approach by 1.70% and 1.92% respectively. However the method can improve solution quality compared to using no restarts. Only for four instances the MA without restarts can provide slightly better results. The second part of the data set, keeps to be comparatively harder to solve which can be explained by the higher RPD to the best known solutions. We therefore conclude that restarting the population helps finding better solutions, however we assume that the I1 heuristic might not be the optimal choice for new solutions to enter the population. The I1 heuristic constructs relatively good solutions without

Table 8.5.: Comparison MA with I1 restarts

day	MA	TS	VNS	RPD_{TS}	RPD_{VNS}
1	1074.12	1074.12	1074.12	0.00%	0.00%
2	1768.27	1762.21	1762.21	0.34%	0.34%
3	2402.99	2373.65	2373.65	1.24%	1.24%
4	2864.04	2852.29	2815.48	0.41%	1.72%
5	3031.8	3029.65	2993.94	0.07%	1.26%
6	3679.14	3627.18	3629.72	1.43%	1.36%
7	1425.29	1418.22	1418.22	0.50%	0.50%
8	2106.61	2102.61	2096.73	0.19%	0.47%
9	2786.82	2737.82	2730.54	1.79%	2.06%
10	3573.9	3505.27	3499.56	1.96%	2.12%
11	1005.73	1005.73	1005.73	0.00%	0.00%
12	1525.91	1478.51	1472.76	3.21%	3.61%
13	2027.48	2011.24	2001.83	0.81%	1.28%
14	2256.21	2202.08	2215.51	2.46%	1.84%
15	2600.48	2494.57	2465.25	4.25%	5.49%
16	3003.8	2901.02	2896.03	3.54%	3.72%
17	1269.09	1236.24	1236.24	2.66%	2.66%
18	1822.19	1792.61	1796.21	1.65%	1.45%
19	2346.85	2285.10	2292.45	2.70%	2.37%
20	3225.05	3079.16	3076.37	4.74%	4.83%
avg.	2289.7885	2,248.46	2,242.63	1.70%	1.92%

many violations, however compared to solutions in the degenerated population they are worse in solution quality. While the new and poor solutions can help to get out of local optima when they are recombined with already good solutions, a lot of computation has to be done until the overall population reaches the quality of the previously degenerated population. We therefore decided to use a more sophisticated approach based on the ACO presented in Chapter 6. The results of the ACO restarts are shown in the next section.

8.1.3. MA with ACO Restarts

Table 8.7 shows the averaged obtained results when ACO solutions are injected into the population at a restart. It can be seen that standard deviation is slightly higher compared to the MA without restarts and the MA with I1 restarts. The gap between the best found solution and the average solution is 3.01% and is therefore just a little higher than for the other two approaches. The average gap of the first part of the data set is 2.32% and 3.69% for the second part respectively.

Table 8.6.: Comparison mean MA approaches

	mean	RPD
no restart	2,371.60	-
I1 restart	2,349.98	-0.27%
ACO restart	2,343.57	-1.18%

Table 8.6 reports the mean and RPD of the approaches compared to the ACO restart approach. Even though the standard deviation is highest for the ACO approach the average solution quality is lowest, with a difference of -1.18% compared to using no restarts and -0.27% to using I1-restarts.

Table 8.8 compares the best found solutions of the MA with ACO restarts to the MA without restarts (ACO/no), the MA with I1 restarts (ACO/I1) the TS (ACO/TS) and the VNS (ACO/VNS). The best found solutions are reported as well as the comparison of the ACO to all other approaches (RPD). It can be seen that the ACO restarts improve solution quality by -1.53% and -0.87% compared to the MA with no restarts and the MA with I1 restarts respectively. Furthermore the average deviation of solution quality is only around 1.05% worse to the best know solutions found by the VNS, and only 0.83% worse to the best known solutions found by the TS. The MA can find all but one of the best known results for instances up to size 72, with an only 0.08% worse solution for instance 17. It can be seen that the MA is most of the time only around 1% worse compared to the best known results, except for the biggest instances 6,10 and 20 where it deviates around 3% to the best known solutions. The MA can improve the solutions found by the TS by -0.05% in instance 8, but it cant improve the solutions found by the

Table 8.7.: Results MA with ACO-restarts

day	min	mean	max	stdev	$\%_{gap}$
1	1,074.12	1,087.36	1,099.68	10.23	1.23%
2	1,762.21	1,800.18	1,829.26	21.33	2.15%
3	2,393.64	2,446.03	2,480.24	28.13	2.19%
4	2,867.06	2,941.21	3,020.55	49.75	2.59%
5	3,069.38	3,176.93	3,331.92	72.72	3.50%
6	3,737.61	3,861.89	4,020.23	94.70	3.33%
7	1,418.22	1,443.30	1,458.11	12.94	1.77%
8	2,101.55	2,149.35	2,189.98	28.32	2.27%
9	2,761.25	2,797.79	2,853.53	27.27	1.32%
10	3,542.05	3,644.08	3,761.36	69.04	2.88%
11	1,005.73	1,012.58	1,047.33	14.84	0.68%
12	1,483.18	1,536.76	1,613.41	41.38	3.61%
13	2,016.54	2,097.15	2,206.65	58.85	4.00%
14	2,239.63	2,338.68	2,408.67	53.87	4.42%
15	2,510.94	2,638.79	2,738.47	74.76	5.09%
16	2,937.38	3,060.05	3,156.13	61.34	4.18%
17	1,237.18	1,295.92	1,354.25	42.77	4.75%
18	1,806.74	1,858.15	1,916.30	36.44	2.85%
19	2,300.73	2,375.28	2,460.67	48.88	3.24%
20	3,178.89	3,309.91	3,456.19	81.79	4.12%
avg.	2,272.20	2,343.57	2,420.15	46.47	3.01%

VNS. The results show that the MA can compete with other approaches when the problem size is relatively small. With the amount of customers rising, the MA is clearly dominated by the VNS that seems to scale better. The MA is therefore well suited to be implemented as an optimizer into the POPMUSIC framework as sub-problem sizes can flexibly be adapted so that the MA can work in its most efficient environment.

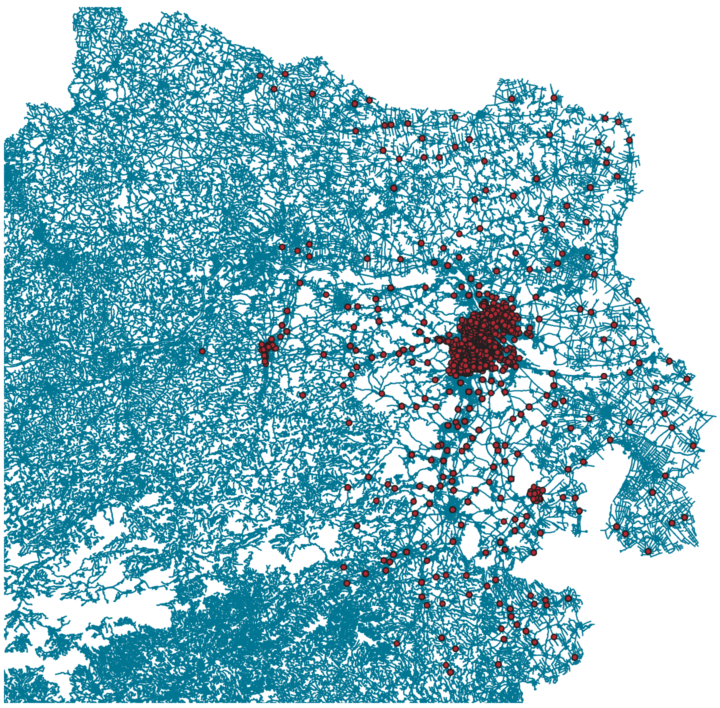
Table 8.8.: Comparison MA with ACO-restarts

				best found solutions			RPD		
Day	TS	VNS	no restart	I1 restart	ACO restart	ACO/no	AC0/I1	ACO/TS	ACO/VNS
1	1074.12	1074.12	1074.12	1074.12	1074.12	0.00%	0.00%	0.00%	0.00%
2	1762.21	1762.21	1780.63	1768.27	1762.21	-1.03%	-0.34%	0.00%	0.00%
3	2373.65	2373.65	2433.83	2402.99	2393.64	-1.65%	-0.39%	0.84%	0.84%
4	2852.29	2815.48	2900.86	2864.04	2867.06	-1.17%	0.11%	0.52%	1.83%
5	3029.65	2993.94	3052.57	3031.80	3069.38	0.55%	1.24%	1.31%	2.52%
6	3627.18	3629.72	3678.24	3679.14	3737.61	1.61%	1.59%	3.04%	2.97%
7	1418.22	1418.22	1425.29	1425.29	1418.22	-0.50%	-0.50%	0.00%	0.00%
8	2102.61	2096.73	2142.30	2106.61	2101.55	-1.90%	-0.24%	-0.05%	0.23%
9	2737.82	2730.54	2797.89	2786.82	2761.25	-1.31%	-0.92%	0.86%	1.12%
10	3505.27	3499.56	3628.35	3573.90	3542.05	-2.38%	-0.89%	1.05%	1.21%
11	1005.73	1005.73	1005.73	1005.73	1005.73	0.00%	0.00%	0.00%	0.00%
12	1478.51	1472.76	1524.79	1525.91	1483.18	-2.73%	-2.80%	0.32%	0.71%
13	2011.24	2001.83	2049.26	2027.48	2016.54	-1.60%	-0.54%	0.26%	0.73%
14	2202.08	2215.51	2323.86	2256.21	2239.63	-3.62%	-0.73%	1.71%	1.09%
15	2494.57	2465.25	2589.54	2600.48	2510.94	-3.04%	-3.44%	0.66%	1.85%
16	2901.02	2896.03	2961.68	3003.80	2937.38	-0.82%	-2.21%	1.25%	1.43%
17	1236.24	1236.24	1290.01	1269.09	1237.18	-4.10%	-2.51%	0.08%	0.08%
18	1792.61	1796.21	1851.68	1822.19	1806.74	-2.43%	-0.85%	0.79%	0.59%
19	2285.10	2292.45	2358.79	2346.85	2300.73	-2.46%	-1.97%	0.68%	0.36%
20	3079.16	3076.37	3246.06	3242.81	3178.89	-2.07%	-1.97%	3.24%	3.33%
avg	2248.46	2242.63	2305.77	2290.68	2272.20	-1.53%	-0.87%	0.83%	1.05%

8.2. Real World Problem

The problem considered, originates from a large real world problem of an Austrian logistics provider that operates two distribution centers (depots) $m = 2$ in or near the city of Vienna. The company serves from 700 to 2000 customers (n) every day with a total number of $K = 160$ vehicles. Both depots are of equal size and the vehicle fleet is equally split between them. The distribution of the customers that need to be served is shown in Figure 8.1. The graphic was created with the ArcMap software and realworld GIS data of the customers and the actual road network. It

Figure 8.1.: Distribution of customers



can be seen that the majority of customers is located in a small geographic region (Vienna). The remaining customers are spread out in the county side, with the occasional smaller towns and villages. Additionally the data set contained the time window $[e_i, l_i]$ for each customer, the service time s_i and the demand d_i that need to be served. The vehicle fleet is homogenous with a maximum capacity D

and is evenly distributed between the amount of m depots. The maximum route duration T is 8 hours, which represents a typical working day. We evaluated 2 weeks, each with 5 days and customers between 743 and 1848 per day. Table 8.9 shows the number of customers to be served for each day. The customers have large time windows, in detail some customers can be served in the morning between 8 a.m. and 12 a.m. , some customers can be served in the afternoon between 12 a.m. and 4 p.m. or during the whole day from 8 a.m. to 4 p.m. . The problem instances are merged into three classes (S,M,L) according to their size so that more concise results can be presented.

Table 8.9.: Problems size and class definitions

Day	1	2	3	4	5	6	7	8	9	10
Size	1201	1180	1284	1305	1175	743	889	1095	1848	1709
Class	M	M	M	M	M	S	S	S	L	L

8.2.1. POPMUSIC MA

In this section the MA was used as an optimizer in the POPMUSIC framework. As can be seen in Table 8.8, the MA can compete with the TS and VNS at problem sizes under 75 customers. We therefore restricted the POPMUSIC framework on constructing sub-problems around this size, so that the MA can efficiently solve them. The three solving strategies I, II and $IIIa$ were given $t_{max} = 28800$ seconds for each individual run, with 10 runs each. Additionally all of the eight p-Median clusterings ranging between 16 and 80 clusters were evaluated for both decomposing strategies. Table 8.10 shows the different parameters used for each strategy.

Strategy I

Strategy I is the most basic one, were the MA tries to solve the problem without any decomposition. The results achieved for each day are shown in Table 8.11. We report the objective-values for the best (min) the worst (max) and average (mean)

Table 8.10.: Parameter settings for the different strategies

	I	II	IIIa	IIIb
it_{pop}	∞	t_i	10	1
it_{ini}	-	-	200	1
t_{max}	28800	28800	28800	1800
Number of clusters	1	16	80	80

solutions found over all runs for each day. It can be seen that when the amount of customers to be serviced varies strongly in real world problems, it is directly reflected in the objective values. Therefore the vehicle fleet may be under-utilized at weak days, and overstrained at busy days. Additionally the standard deviation (stdev) is presented.

Table 8.11.: Results of Strategy *I* for each day

no decomposition				
Day	min	mean	max	stdv
1	4,560.34	4,747.76	4,938.84	106.30
2	4,829.90	4,975.89	5,138.89	102.01
3	4,860.68	5,070.26	5,315.12	125.95
4	5,348.45	5,490.42	5,628.42	97.85
5	5,008.66	5,143.97	5,238.79	73.10
6	2,872.69	3,046.42	3,253.43	119.44
7	4,030.47	4,103.11	4,159.57	38.54
8	3,865.22	4,007.95	4,164.30	99.37
9	6,861.64	7,004.53	7,216.25	106.26
10	7,269.68	7,527.55	7,726.65	145.73

The results by each class are shown in Table 8.12, with the objective values for the best found solution (min), the average solution (mean) and the worst solution (max). Up-to-date the MA was the first algorithm applied to this real-world problem. Therefore the results of the MA form the basis for further comparison.

Table 8.12.: Results of Strategy I by class

	no decomposition			
	min	mean	max	Stdv
Small	10,768.38	11,157.48	11,577.30	93.82
Medium	24,608.03	25,428.30	26,260.06	243.88
Large	14,131.32	14,532.08	14,942.90	127.30

Strategy II

Strategy II focuses on solving the problem sequentially. This is done by splitting the complete problem though the use of the p-Median procedure. Each of the resulting clusters, is then solved for the maximum allowed time t_i by the MA. As a result there is no interaction between the clusters, so that customers initially assigned to a cluster can never be relocated to another cluster. The complete solution, containing all of the customers, is finally generated by merging the routes of the sequentially generated solutions. Table 8.13 shows the results obtained for each class by this strategy.

The minimum, mean and average objective values as well as the standard deviation are calculated for each initial clustering, and are presented by class. Additionally the results for each clustering are ranked per class by their corresponding mean values. Compared to the results presented for the standardized instances, we mostly compare average results, as it may be more appropriate in a real-world scenario. The obtained results show that intelligently decomposing the problem into parts, and then solving them sequentially results in an improvement of about -12.83% compared to using the same optimizer without any pre-decomposition (see Table 8.16). In Table 8.13 it can be seen that with a rising amount of clusters the solution quality decreases which is true for all problem-classes. Like explained in Section 7.1.2 the nature of the p-Median procedure produces bigger clusters in highly populated areas and very small ones in regions in the country side (see Figure 7.1). As there is no interaction between clusters, the vehicle fleet can not be efficiently distributed to the clusters and borderline customers can not move to other better suited routes. Therefore the more the problem is decomposed

Table 8.13.: Results of Strategy *II* by class

	fixed decomposition					
	# clusters	min	mean	worst	stdv	rank
Small	16	10,476.20	10,569.07	10,679.76	33.95	2
	20	10,454.03	10,543.05	10,645.76	31.43	1
	22	10,826.24	10,905.97	11,068.00	29.49	3
	26	11,061.94	11,154.92	11,231.33	34.31	4
	32	11,453.13	11,546.82	11,627.81	32.36	5
	40	12,133.01	12,174.46	12,236.35	14.05	6
	53	13,253.88	13,328.14	13,377.80	27.40	7
	80	15,489.56	15,512.81	15,564.26	12.85	8
Medium	16	21,694.29	21,923.49	22,110.66	74.94	1
	20	21,759.04	21,988.95	22,305.77	73.24	2
	22	22,102.60	22,217.69	22,422.36	63.66	3
	26	22,410.12	22,546.74	22,799.75	72.60	4
	32	23,059.11	23,196.34	23,395.94	41.09	5
	40	23,900.03	24,031.28	24,160.76	36.74	6
	53	26,059.15	26,216.12	26,377.68	39.17	7
	80	30,452.64	30,547.83	30,653.02	25.66	8
Large	16	11,970.13	12,067.94	12,162.91	37.03	1
	20	12,165.35	12,238.52	12,343.51	36.75	2
	22	12,164.28	12,244.38	12,328.99	45.04	3
	26	12,324.99	12,417.89	12,507.92	34.22	4
	32	12,587.37	12,688.86	12,748.92	26.57	5
	40	12,960.88	13,012.02	13,083.13	16.11	6
	53	13,350.07	13,407.04	13,455.22	26.14	7
	80	14,402.30	14,487.32	14,594.94	23.97	8

the more inefficiencies can arise which is represented by solution values increasing with the amount of clusters. Looking at the results, one can see that there is a difference of about 50% when comparing the 16 cluster and 80 cluster solutions in class S, around 40% in class M and around 20% in class L. This figures show, that the more a problem is decomposed the more inefficiencies arise when no interaction between clusters is allowed. When comparing the results in tables 8.12 and 8.13 one can see that the best results obtained by strategy II for the class S are similar to the results of strategy I while at the same time the worst clusterings can't compete with the simple MA approach. However with increasing problem size, the decomposition approach greatly outperforms the simple approach, which is underlined when looking at the results of class L. It can be seen that even the worst clustering produces solutions that are on average better than the solutions obtained by strategy I.

Strategy III

Strategy III is the decomposing strategy based on the POPMUSIC framework. This strategy flexibly splits the large problem into smaller sub-problems that than can be easily solved by the MA. In contrast to strategy II this sub-problems can interact with each other, so that borderline customers can flexibly be assigned to the adequate sub-problems. Initially wrongly assigned customers can therefore easily be reassigned which should be reflected in lower solution values. The POPMUSIC strategy was tested with two different parameter settings. Table 8.14 shows the results with long runtime ($t = 28800$ seconds). The longer overall runtime coupled with a more intense search in the individual sub-problems should help to fully exploit the feature of reassigning customers that were initially assigned to the wrong routes. It can be seen that nearly -20% improvement can be achieved when using the developed decomposing strategy, compared to using a traditional solver without decomposition (see Table 8.16). In contrast to strategy II the initial clustering does not impact solution quality in the same way when using the POPMUSIC framework. Because sub-problems can easily interact with each other and are always around the size of 75 customers, the MA can operate in an optimal environment. This is not the case in strategy II were clusters can be far above the limit of 75 customers. Table 8.14 shows that the best results are achieved in the higher spectrum of initial clusters. The difference in solution quality is however relatively small (under 1%).

On the other hand Table 8.15 contains the results for the accelerated search, that focuses on finding feasible solutions as fast as possible. This approach was given much less time ($t = 1800$ seconds) to solve the problem, with reduced search time in the individual sub-problems. As the time given to optimize a sub-problem is highly limited in this approach, a complete cycle of optimizing all sub-problems uses a fraction of the time compared to strategy $IIIa$. The first feasible and good

Table 8.14.: Results of Strategy $IIIa$ by class

	POPMUSIC (long) $(10 \cdot 10 \cdot 8 \cdot 8h = 6400h)$					
	# clusters	Best	Mean	Worst	Stdv	Rank
Small	16	9,146.16	9,282.50	9,430.24	50.91	8
	20	9,102.46	9,249.05	9,391.41	56.45	6
	22	9,118.99	9,238.56	9,398.32	46.75	4
	26	9,127.43	9,266.33	9,408.25	49.89	7
	32	9,122.98	9,234.82	9,349.97	37.09	3
	40	9,053.41	9,218.38	9,398.19	50.84	1
	53	9,129.94	9,240.59	9,404.85	61.16	5
	80	9,089.32	9,227.38	9,397.18	39.62	2
Medium	16	20,437.59	20,686.65	20,993.94	104.61	8
	20	20,323.06	20,540.00	20,756.90	58.35	6
	22	20,268.55	20,581.12	20,848.29	85.18	7
	26	20,252.23	20,472.45	20,726.84	67.86	3
	32	20,221.06	20,483.68	20,770.48	92.92	4
	40	20,249.71	20,497.42	20,734.10	78.27	5
	53	20,269.93	20,471.12	20,810.73	102.07	2
	80	20,156.41	20,429.33	20,713.18	60.57	1
Large	16	11,532.49	11,648.60	11,759.85	50.09	7
	20	11,451.16	11,599.02	11,724.41	53.92	4
	22	11,525.01	11,662.78	11,763.10	54.16	8
	26	11,506.05	11,605.74	11,711.06	47.98	5
	32	11,395.66	11,553.61	11,688.02	47.80	3
	40	11,369.10	11,550.09	11,661.57	65.81	2
	53	11,406.17	11,545.81	11,685.24	60.97	1
	80	11,505.03	11,607.31	11,826.82	80.29	6

solutions are therefore generated much faster, at the cost of a worse exploration of the sub-problems. Nevertheless strategy $IIIa$ can improve the solutions found by the MA without decomposition by -13.46% by using only a fraction of the time (1/16).

Comparison of the Strategies

The results for three different approaches that all use the same optimizer were presented. In this section we further compare them against each other, and additionally give some insight on the speed of the solution finding process. Table 8.16

Table 8.15.: Results of Strategy $IIIb$ by class

	POPMUSIC (short) $(10 \cdot 10 \cdot 8 \cdot 0.5h = 400h)$					
	# clusters	Best	Mean	Worst	Stdv	Rank
Small	16	9,966.38	10,200.80	10,454.41	84.90	8
	20	9,866.84	10,085.59	10,253.86	54.51	6
	22	9,883.93	10,140.76	10,423.39	75.67	7
	26	9,830.67	10,020.04	10,228.69	78.48	5
	32	9,793.71	9,990.40	10,218.84	76.80	4
	40	9,651.10	9,865.66	10,098.00	72.80	1
	53	9,713.36	9,932.95	10,233.72	112.97	2
	80	9,687.67	9,966.59	10,289.52	112.26	3
Medium	16	22,352.24	22,743.57	23,090.57	136.76	8
	20	22,085.14	22,541.87	23,017.33	104.78	7
	22	22,138.83	22,514.31	22,976.41	74.45	6
	26	21,888.54	22,254.29	22,662.73	110.00	5
	32	21,741.50	22,100.39	22,518.78	70.69	4
	40	21,610.46	22,013.68	22,418.18	132.56	3
	53	21,648.42	22,003.30	22,353.57	103.83	2
	80	21,450.20	21,800.15	22,237.30	76.30	1
Large	16	12,636.64	12,829.57	13,121.98	63.39	8
	20	12,575.81	12,793.70	13,035.89	97.84	7
	22	12,585.25	12,784.62	12,952.19	73.43	6
	26	12,427.22	12,625.26	12,898.93	60.77	5
	32	12,314.27	12,557.46	12,756.27	90.84	3
	40	12,332.03	12,564.40	12,766.31	57.71	4
	53	12,227.43	12,434.03	12,627.31	86.63	1
	80	12,297.36	12,473.14	12,607.14	62.68	2

presents the obtained results of the best clustering for each strategy and class. The random percentage deviation between strategy I and all other strategies are presented as well as the RPD averaged over all classes.

It can be seen that solving the problem without any decomposition clearly resulted in the worst solution quality. A simple decomposing of the problem into intelligently chosen parts, like the p-Median procedure used in strategy II improves the results by roughly -13% compared to the most basic strategy. The average results obtained by strategy $IIIa$ are nearly -20% better than without any decomposition and prove that the POPMUSIC framework can solve problems

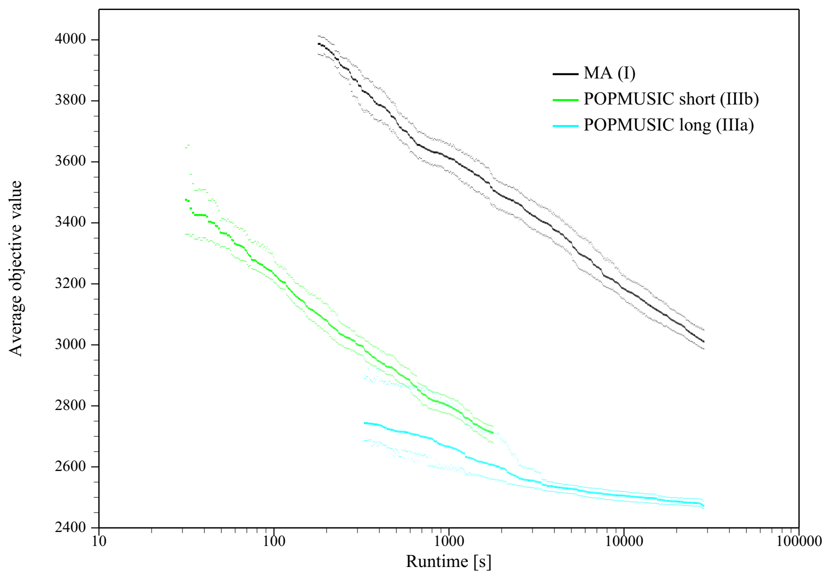
Table 8.16.: Comparison of the strategies

Class	RPD I/II	RPD I/IIIa	RPD I/IIIb
S–Small	-5.27%	-17.30%	-10.67%
M–Medium	-13.78%	-19.66%	-14.27%
L–Large	-16.96%	-20.13%	-14.17%
Average	-12.83%	-19.28%	-13.46%

of large scale efficiently. This is further underlined when looking at the results obtained by Strategy $IIIb$ Even though the algorithm has only about 6% of the time available it outperforms the simple Strategy I by nearly -14% and the decomposition Strategy II by around -0.75%. It also underlines how easy the framework can be adapted to the needs that arise in real world applications, when relatively good solutions are needed as fast as possible. As explained before we once again want to note that strategy II produced better results than strategy I when using the smallest amount of p-Median clusters, but is greatly outperformed when choosing the wrong initial clustering.

Figure 8.2 shows the objective values as well as a 95% confidence interval for Strategies I, $IIIa$ and $IIIb$ over the runtime for day 6. Strategy II is not represented in these figure because clusters are solved sequentially. This figure shows, that the MA without decomposition and the POPMUSIC long need around the same time to find a feasible solution. However the quality of the solutions is highly different and in favor to the POPMUSIC approach. The initial clustering by the p-Median procedure gives the POPMUSIC approach a significant head-start, which is then further developed by intelligently creating and solving the sub-problems. Nevertheless the MA can steadily improve over the whole runtime but never gets even close to the solution quality obtained early in the search by the POPMUSIC long approach. On the other hand the descent of the POPMUSIC long approach is not so steep, which may be a result of the already good solution quality. Looking at the results of POPMUSIC short, one can see that this approach is able to find solutions much faster then the other approaches. It can be seen that there is a trade-off between accelerating the search and finding good quality solutions, as strategy IIIb cannot reach the solution quality of strategy IIIa at the end of

Figure 8.2.: Average objective values and confidence intervals over runtime for day 6



the time-limit. This finding further underlines the flexibility of the POPMUSIC approach, as by simply tuning two parameters the search can easily be accelerated while maintaining good solution quality. Even though we can not directly compare strategy II concerning run-time, we want to point out that the final solution values of strategy II are roughly of the quality that can be found early in the search of strategy IIIb. Since both of the strategies start with the same initial clustering, it can be seen that flexibly re-arranging customers from one sub-problem to another, can yield high improvements relatively early in the search, while solving the sub-problems without any interaction results in a stagnating search. Furthermore we want to point out that, even though we only present a figure for day 6, similar results can be examined for all other days.

Finally we want to present the best known solutions in Table 8.17. It is to note that all solutions were found by strategy $IIIa$ which further underlines the effectiveness of the approach. The last column shows the initial p-Median clustering with which the solutions were obtained.

Table 8.17.: Best solution values found

day	best	# clusters
1	3,649.09	80
2	4,061.71	32
3	4,022.29	22
4	4,258.19	80
5	4,134.59	40
6	2,423.83	22
7	3,438.83	40
8	3,164.80	80
9	5,427.65	40
10	5,941.45	40

8.2.2. POPMUSIC VNS

In this section the results for the POPMUSIC that uses a VNS as optimizer are presented. Compared to the decomposition approach that uses an MA as an optimizer, the integration of a VNS into the POPMUSIC framework allows for a more flexible creation of sub-problems around a seed part. In the PopVNS approach a part is defined as a route, compared to a set of routes in the PopMA approach. This on the one hand allows for more different combinations of creating sub-problems, as well as for more flexibility in creating the actual proximity measure with which sub-problems can be created. The presented measures should in practice work for every method that only manipulates a single solution. To gain better inside of the performance of this approach, we compared it to the results obtained by the PopMA (see Section 7.1) and the VNS itself without any decomposition. It is to note that the results obtained in Section 8.2.1 are slightly different compared to the results in this section. This comes mainly from the fact, that a more recent road network was used for distance calculations, so that the emerging distance-matrix contains the eventual difference between two customers, compared to the original distances. Furthermore the 3-opt used in the MA was enhanced so that it can shift customers that are on the same location without any sequence restrictions. This was done to provide a unified basis for comparing the two algorithms only on basis of the decomposition approaches and the optimizers used and not because of structural design differences of the two approaches. The real-world instances include two depots from which customers need to be served and to which the vehicle fleet is evenly distributed to. To examine also the performance of the PopVNS approach for more than two depots, we extended the initial data set by introducing two additional depots. The extra depots were chosen out of 50 possible locations in the vicinity of Vienna. The possible locations were gathered by hand and are mostly located in business-parks, commercial areas or easily accessible areas. Two depot were selected by hand out of all the possible locations. No special selection algorithm was used, as we did not want to focus on the Facility Location Problem or even the Location Routing Problem (LRP). The POPMUSIC with a VNS as optimizer was therefore tested on the two, three and four depot case and over all instances. To gain some further insight on the performance of the eight different proximity measures, and how they behave in varying multi depot environments, all of them were tested on these newly generated instances. To compare the obtained results the VNS without any decomposition was applied to them as well. We decided to perform ten independent runs with a computation time of five hours for each run. The long time limit was chosen with the idea of possible overnight calculations, as they would be performed in the real world. Therefore each depot and each proximity measure combination was tested 10 times for each day for 5 hours each. These extensive calculations were performed on all of the 4 cores of identical Intel Pentium 640 'Prescott' 3.2GHZ, 800MHZ FSB, 2MB L2-Cache PC's with 4gb of memory.

Table 8.18 shows the results obtained for the two depot case. The results provided in this Table for the PopVNS are the results obtained through the use of the best proximity measure D_{IV}. The table presents the average costs obtained by the POPMUSIC that uses a MA as optimizer (PopMA) as explained in Section 8.2.1 or presented in the work of Ostertag et al. (2008b). Additional results are presented for the VNS without decomposition (VNS) as explained in Chapter 4 and the POPMUSIC that uses a VNS as optimizer with the D_{IV} proximity measure (PopVNS). The table reports the RPD between the PopVNS and the two other approaches. It can be seen that the PopVNS with the D_{IV} proximity measure, can improve the results obtained by the PopMA by -6.17%. This is an remarkable

Table 8.18.: Comparison of algorithms for the initial two depot setup

Day	PopMA	VNS	PopVNS	RPD PopMA / PopVns	RPD VNS / PopVns
10	3990.30	3900.22	3617.17	-9.35%	-7.26%
11	4248.84	4337.41	4035.87	-5.01%	-6.95%
12	4337.51	4308.20	4002.35	-7.73%	-7.10%
13	4526.04	4641.33	4258.72	-5.91%	-8.24%
14	4335.46	4383.96	4085.11	-5.77%	-6.82%
22	2531.97	2592.75	2441.07	-3.59%	-5.85%
23	3483.54	3627.22	3394.80	-2.55%	-6.41%
24	3483.54	3354.47	3128.26	-10.20%	-6.74%
25	6031.33	5855.59	5368.09	-11.00%	-8.33%
26	5948.30	6344.28	5911.63	-0.62%	-6.82%
avg.	4291.68	4334.54	4024.31	-6.17%	-7.05%

improvement as the PopMA already improved solution quality around -20% compared to the MA without any decomposition (see Section 8.2.1). Implementing the VNS into the POPMUSIC framework resulted in a -7.05% decrease of solution quality, compared to using the pure VNS without decomposition. In both cases, MA and VNS, the decomposition approach resulted in significant efficiency improvements. Even though the developed MA is clearly dominated by the VNS, the PopMA can improve the results obtained by the powerful pure VNS by roughly -1%.

Measure D_{IV} turned out to be the clear winner out of all the tested proximity measures. Nevertheless the results of all proximity measures are presented and discussed in great detail in the next section.

Analysis of Proximity Measure

In this section the results for the eight proximity measures are presented and analyzed. Three Sweep-based measures (D_I - D_{III}) are tested as well as five distance-based measures (D_I - D_V). All of them were tested on the initial 2-depot case as well as on the extended 3 and 4 depot case.

Table 8.19 presents the RPD to the VNS without decomposition and the rank-

Table 8.19.: Average results for two depots

measure	RPD to VNS	Rank
S_I	-6.62%	5
S_{II}	-6.62%	6
S_{III}	-6.56%	7
D_I	-6.87%	2
D_{II}	-6.71%	4
D_{III}	-6.85%	3
D_{IV}	-7.05%	1
D_V	-6.09%	8
$PopMA$	-0.80%	9
VNS	0.00%	10

ing of each measure for the initial 2 depot instances. It can be seen that D_{IV} provided the highest improvement with -7.05% compared to the VNS. The remaining strategies are performing at maximum only around 1% worse compared to the best proximity measure. Generally one can see that it seems like the distance based measure work better in a two depot environment than the measures based on the Sweep mechanic, with the only exception being measure D_V which provided by far the worst results. When looking at this table, one can see that each of the presented proximity measures outperforms the POPMUSIC approach that uses an MA as optimizer.

Table 8.20.: Route length and RPD for two depots and all days

	1	2	3	4	5	6	7	8	9	10	avg
S_I	3,630.68	4,053.56	4,022.77	4,280.37	4,105.72	2,409.96	3,393.67	3,173.86	5,462.52	5,934.92	4,046.80
S_{II}	3,650.70	4,062.70	3,988.77	4,262.64	4,104.92	2,426.27	3,382.31	3,179.13	5,416.32	5,931.18	4,040.49
S_{III}	3,663.69	4,052.95	4,023.78	4,273.93	4,124.61	2,412.99	3,378.98	3,165.56	5,458.83	5,943.68	4,049.90
D_I	3,625.62	4,059.36	3,999.05	4,261.26	4,107.01	2,430.87	3,387.13	3,159.87	5,412.76	5,894.32	4,033.73
D_{II}	3,617.17	4,035.87	4,002.35	4,258.72	4,085.11	2,441.07	3,394.80	3,128.26	5,368.09	5,911.63	4,024.31
D_{III}	3,651.46	4,064.19	4,018.03	4,302.37	4,145.65	2,462.58	3,423.51	3,204.13	5,415.27	5,953.94	4,064.11
D_{IV}	3,633.18	4,059.55	4,016.49	4,269.02	4,105.01	2,410.84	3,383.56	3,195.74	5,462.92	5,926.00	4,046.23
D_V	3,611.80	4,044.66	4,008.15	4,275.16	4,088.52	2,439.19	3,396.57	3,156.02	5,389.60	5,915.36	4,032.50
$PopMA$	3,990.30	4,248.84	4,337.51	4,526.04	4,335.46	2,531.97	3,483.54	3,496.30	6,031.33	5,948.30	4,292.96
VNS	3,900.22	4,337.41	4,308.20	4,641.33	4,383.96	2,592.75	3,627.22	3,354.47	5,855.59	6,344.28	4,334.54
S_I	-6.91%	-6.54%	-6.63%	-7.78%	-6.35%	-7.05%	-6.44%	-5.38%	-6.71%	-6.45%	-6.62%
S_{II}	-6.40%	-6.33%	-7.41%	-8.16%	-6.37%	-6.42%	-6.75%	-5.23%	-7.50%	-6.51%	-6.71%
S_{III}	-6.06%	-6.56%	-6.60%	-7.92%	-5.92%	-6.93%	-6.84%	-5.63%	-6.78%	-6.31%	-6.56%
D_I	-7.04%	-6.41%	-7.18%	-8.19%	-6.32%	-6.24%	-6.62%	-5.80%	-7.56%	-7.09%	-6.85%
D_{II}	-7.26%	-6.95%	-7.10%	-8.24%	-6.82%	-5.85%	-6.41%	-6.74%	-8.33%	-6.82%	-7.05%
D_{III}	-6.38%	-6.30%	-6.74%	-7.30%	-5.44%	-5.02%	-5.62%	-4.48%	-7.52%	-6.15%	-6.09%
D_{IV}	-6.85%	-6.41%	-6.77%	-8.02%	-6.36%	-7.02%	-6.72%	-4.73%	-6.71%	-6.59%	-6.62%
D_V	-7.39%	-6.75%	-6.96%	-7.89%	-6.74%	-5.92%	-6.36%	-5.92%	-7.96%	-6.76%	-6.87%
$PopMA$	2.31%	-2.04%	0.68%	-2.48%	-1.11%	-2.34%	-3.96%	4.23%	3.00%	-6.24%	-0.80%
VNS	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 8.21.: Results for three depots

measure	RPD to VNS	Rank
S_I	-6.38%	4
S_{II}	-5.97%	7
S_{III}	-6.45%	2
D_I	-6.14%	6
D_{II}	-6.17%	5
D_{III}	-6.41%	3
D_{IV}	-6.91%	1
D_V	-5.70%	8
VNS	0.00%	9

Table 8.20 gives further insight on how the developed proximity measures perform on the individual instances. The figures in the upper part of this table are again the average results obtained over all runs per instance, while the figures in the lower part represent the RPD to the VNS. Foremost on may notice that the PopMA performs at average around -0.8% better than the VNS, however it provides better solutions than the VNS only in 6 out of 10 instances. When looking at the individual proximity measures one can see that all of them provide better results than the VNS or PopMA for all instances, with improvements ranging between -4.48% and -8.33%. It can be seen that the smallest improvements can be gained for instances 6,7 and 8, which are the instances with the smallest amount of customers to be served.

Table 8.21 reports the results for the initial two-depot setup that is extended by one additional depot. It can be seen that D_{IV} is again the top ranking measure with an average improvement of -6.91% compared to the VNS. The second best ranking in the 3 depot setup was achieved by S_{III} closely followed by D_{II}. When adding one more depot the measures perform differently than in the two depot case. It seems that Sweep based measures can close their gap on the distance based measures, as they can more effectively assign the borderline customers to the best depot.

Table 8.22 presents the obtained results in great detail, by reporting the average values of ten runs, as well as the RPD to the VNS for each day and proximity measure combination. Like in the two depot case, all proximity measures provide better results than the VNS approach with improvements ranging between -3.69% and -8.09%. The observation, that the gain in solution quality is correlated to the instance size, can be supported as the average improvements when decomposing are relatively smaller for days 6,7 and 8 compared to the days that feature a larger amount of customers.

Table 8.22.: Route length and RPD for three depots and all days

	1	2	3	4	5	6	7	8	9	10	avg
S_I	3,475.91	3,907.79	3,851.27	4,147.14	3,966.47	2,358.49	3,251.19	3,009.56	5,218.71	5,796.79	3,898.33
S_{II}	3,470.65	3,940.16	3,851.23	4,151.07	3,973.47	2,371.36	3,263.31	3,036.65	5,218.05	5,768.62	3,904.46
S_{III}	3,482.05	3,920.49	3,847.14	4,112.11	3,948.49	2,352.23	3,265.23	3,013.22	5,233.53	5,772.76	3,894.73
D_I	3,459.60	3,923.96	3,830.19	4,136.50	3,971.80	2,363.79	3,263.64	3,026.81	5,202.14	5,768.20	3,894.66
D_{II}	3,450.54	3,895.84	3,813.91	4,121.25	3,939.82	2,350.96	3,245.18	3,011.14	5,166.37	5,746.64	3,874.17
D_{III}	3,476.18	3,915.61	3,880.61	4,164.33	3,982.23	2,408.56	3,299.15	3,057.57	5,236.79	5,795.13	3,921.62
D_{IV}	3,495.40	3,933.00	3,839.64	4,156.29	3,986.03	2,384.97	3,268.35	3,032.60	5,248.05	5,785.24	3,912.96
D_V	3,478.42	3,918.08	3,859.07	4,136.55	3,951.95	2,404.96	3,265.83	3,044.82	5,211.95	5,750.43	3,902.21
VNS	3,736.92	4,162.03	4,127.34	4,473.89	4,216.16	2,500.88	3,463.28	3,217.67	5,606.79	6,141.18	4,164.61
S_I	-6.98%	-6.11%	-6.69%	-7.30%	-5.92%	-5.69%	-6.12%	-6.47%	-6.92%	-5.61%	-6.38%
S_{II}	-7.13%	-5.33%	-6.69%	-7.22%	-5.76%	-5.18%	-5.77%	-5.63%	-6.93%	-6.07%	-6.17%
S_{III}	-6.82%	-5.80%	-6.79%	-8.09%	-6.35%	-5.94%	-5.72%	-6.35%	-6.66%	-6.00%	-6.45%
D_I	-7.42%	-5.72%	-7.20%	-7.54%	-5.80%	-5.48%	-5.76%	-5.93%	-7.22%	-6.07%	-6.41%
D_{II}	-7.66%	-6.40%	-7.59%	-7.88%	-6.55%	-5.99%	-6.30%	-6.42%	-7.86%	-6.42%	-6.91%
D_{III}	-6.98%	-5.92%	-5.98%	-6.92%	-5.55%	-3.69%	-4.74%	-4.98%	-6.60%	-5.63%	-5.70%
D_{IV}	-6.46%	-5.50%	-6.97%	-7.10%	-5.46%	-4.63%	-5.63%	-5.75%	-6.40%	-5.80%	-5.97%
D_V	-6.92%	-5.86%	-6.50%	-7.54%	-6.27%	-3.84%	-5.70%	-5.37%	-7.04%	-6.36%	-6.14%
VNS	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 8.23.: Results for four depots

measure	RPD to VNS	Rank
S_I	-5.86%	3
S_{II}	-5.17%	7
S_{III}	-5.92%	2
D_I	-5.54%	5
D_{II}	-5.42%	6
D_{III}	-5.78%	4
D_{IV}	-6.31%	1
D_V	-5.13%	8
VNS	0.00%	9

The results for the four depot case, the initial two depot case that was extended by two hand picked depots, are presented in Table 8.23. The D_{IV} measure seems to work well in all of the different variations of depot setups, as it is again the top performing measure with an average improvement of -6.31% compared to the VNS. S_{III} is again ranked as the second best measure, and S_I being the third best. Extending the problem by an additional depot again shifts the favor of good performing measures in the direction of the sweep-based measures. Even though measure D_{IV} seems generally very stable in all environments as it clearly outperforms all other measures in all conducted test runs.

Table 8.24.: Route length and RPD for four depots and all days

	1	2	3	4	5	6	7	8	9	10	avg
S_I	3,321.94	3,792.64	3,677.01	4,010.82	3,833.07	2,283.22	3,206.33	2,880.67	4,988.08	5,586.67	3,758.05
S_{II}	3,361.83	3,807.29	3,690.22	4,029.92	3,866.05	2,297.28	3,223.81	2,889.37	4,985.24	5,585.99	3,773.70
S_{III}	3,319.43	3,784.16	3,666.63	3,985.20	3,855.93	2,282.53	3,213.52	2,883.26	4,987.66	5,570.85	3,754.92
D_I	3,331.70	3,776.63	3,690.63	4,010.75	3,851.31	2,304.72	3,198.99	2,881.65	4,965.57	5,573.47	3,758.54
D_{II}	3,325.86	3,758.09	3,652.37	3,954.13	3,835.41	2,284.43	3,200.32	2,902.07	4,923.77	5,515.34	3,735.18
D_{III}	3,372.07	3,804.67	3,707.23	4,045.20	3,851.46	2,322.59	3,238.17	2,891.92	5,007.57	5,600.80	3,784.17
D_{IV}	3,353.62	3,802.89	3,698.50	4,038.57	3,856.44	2,304.09	3,219.94	2,906.94	5,046.17	5,629.83	3,785.70
D_V	3,338.16	3,796.25	3,695.33	4,028.09	3,860.95	2,299.14	3,215.99	2,890.60	4,990.06	5,576.39	3,769.10
VNS	3,554.77	4,009.16	3,954.96	4,278.35	4,058.78	2,389.15	3,419.67	3,030.56	5,346.42	5,910.37	3,995.22
S_I	-6.55%	-5.40%	-7.03%	-6.25%	-5.56%	-4.43%	-6.24%	-4.95%	-6.70%	-5.48%	-5.86%
S_{II}	-5.43%	-5.04%	-6.69%	-5.81%	-4.75%	-3.85%	-5.73%	-4.66%	-6.76%	-5.49%	-5.42%
S_{III}	-6.62%	-5.61%	-7.29%	-6.85%	-5.00%	-4.46%	-6.03%	-4.86%	-6.71%	-5.74%	-5.92%
D_I	-6.28%	-5.80%	-6.68%	-6.25%	-5.11%	-3.53%	-6.45%	-4.91%	-7.12%	-5.70%	-5.78%
D_{II}	-6.44%	-6.26%	-7.65%	-7.58%	-5.50%	-4.38%	-6.41%	-4.24%	-7.91%	-6.68%	-6.31%
D_{III}	-5.14%	-5.10%	-6.26%	-5.45%	-5.11%	-2.79%	-5.31%	-4.57%	-6.34%	-5.24%	-5.13%
D_{IV}	-5.66%	-5.14%	-6.48%	-5.60%	-4.99%	-3.56%	-5.84%	-4.08%	-5.62%	-4.75%	-5.17%
D_V	-6.09%	-5.31%	-6.56%	-5.85%	-4.87%	-3.77%	-5.96%	-4.62%	-6.67%	-5.65%	-5.54%
VNS	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 8.25.: Results averaged over all depots

measure	RPD to VNS	Rank
S_I	-6.29%	4
S_{II}	-5.92%	7
S_{III}	-6.31%	3
D_I	-6.18%	5
D_{II}	-6.10%	6
D_{III}	-6.35%	2
D_{IV}	-6.76%	1
D_V	-5.64%	8
VNS	0.00%	9

Table 8.24 reports the average values of ten runs, as well as the RPD to the VNS for each day and proximity measure combination for the four depot case. All of the eight tested proximity measures provide better results, than the VNS approach, with improvements between -2.79% and -7.91%, with smaller improvements for the smaller instances.

The average results over all three depot setups are shown in Table 8.25. It can clearly be seen that all of the tested measures perform significantly better than the VNS without decomposition. The improvement generated by the different strategies range between -5.64% and -6.76% and are therefore relatively close (around 1.12%) concerning solution quality. However measure D_{IV} seems to be the clear winner with an average improvement of -0.41% to the next best distance measure D_{III}. D_{IV} is also the best measure on average when looking at the detached results for the two, three and four depot case (see Tables 8.19, 8.21, 8.23), where it is always ranked number one with always a clear lead to the second best measure. Over all instances and depot setups, D_{IV} provided the best results, except for the small instances in the two depot and four depot setup. It is closely followed by D_{III} which generated the second best results as can be seen in the overview in Table 8.25.

Concerning the Sweep proximity measures S_{III} seems to work the best out of all tested Sweep measures. Furthermore it is interesting to point out that S_{III} seems to gain efficiency when dealing with more depots. For the three and four depot case S_{III} provides the second best results. As we have mentioned in Chapter 7, the major amount of customers is densely packed in a relatively small area of the region. We therefore introduced the restriction for selecting customers in D_{IV} to countervail the greedy selection of customers that are very closely together to enter the sub-problem. This restriction tries to level out the distribution around the seed-route, so that customers that are farther away and in the country-side do have a chance to enter the sub-problem. When looking at the results we can assume that this idea was fruitful, especially when one is aware of the fact that the second and third best measures (S_{III} and S_I) also don't allow this greedy selections. The Sweep measures do exactly the same, by allowing customers that are located in a beam that emits from a depot to enter the sub-problem regardless of how far away they are in relation to the seed-route. Further we want to point out that aggregating customers that are not in the seed route might not be a good idea, as can be seen by looking at the results for strategy D_V. This strategy performed worst in all of the different test-setups and even more with a huge gap to the leading measures.

9. Conclusion

In this thesis different solution methods for routing problems with multiple depots and time windows, with special focus on large scale real word problems, were developed. A Memetic Algorithm was developed to deal with standardized MDVRPTW instances and compared to the most recent state of the art solving strategies. We showed that restarting the MA can yield a noticeable improvement of around 0.87% even when restarting with a basic I1 construction heuristic. We also showed that restarting a population with highly competitive ACO solutions can further increase the MA efficiency by around 0.27% resulting in the up to now best known results obtained by an MA on the standardized test instances. When comparing the results of the MA with ACO restarts to other approaches like the VNS or TS the developed approach is performing relatively well. More precisely, the results of the MA are around 0.83% worse compared to the TS and around 1.05% to the up to now best known solution method, the VNS. However the MA can solve all small instances except one with up to 75 customers to the same solution quality as the two other approaches. The results obtained show that the MA does not scale well with regards to run times compared to the VNS. While the efficiency of the VNS is mostly related to the average number of customers in a route, the MA has to consider the complete problem. This special characteristic becomes obvious when looking at the recombination step which more or less resembles a shacking move in a VNS. While only two routes need to be considered in the VNS case, the complete problem needs to be analyzed when using a recombination of routes of two different solutions. The time to carry out one recombination is therefore directly related to the amount of customers not only in the considered routes, but in the whole problem. We can therefore conclude that the MA works reasonable well for problems of medium size, but loses efficiency for problems of large scale.

The huge size of typical real-world problems eventually lead to developing two decomposition approaches that were presented in this thesis. The two new decomposition approaches are both based on the POPMUSIC framework that was customized to the special real world requirements of MDVRPTWs. The first approach was specially designed so that population-based algorithms can be used to solve the problem. More precisely the developed MA for the MDVRPTW, that turned out to provide competitive results on the medium sized standardized instances, was implemented into the POPMUSIC framework and used as an optimizer for the resulting sub-problems. The results presented, have shown that a decomposition strategy that uses an MA as optimizer can be very efficient in solving large scale MDVRPTW. We achieved an average improvement of about -20% over all considered real world instances compared to the use of the same optimizer without decomposition. This figures reflect the ability of the decomposition approach to amplify the effectiveness of the MA approach when it can operate in an environment of reasonable problem sizes. The drawback of approaches that are scaling relatively bad with regards to problem size, can therefore be easily overcome trough the help of decomposition. Compared to the powerful VNS approach by Polacek et al. (2004) the POPMUSIC MA approach can generate solutions that are around 1% better. The POPMUSIC framework is therefore able to automatically and efficiently reassign customers that are equally distant from both depots. We also show that the framework customized for population-based approaches, can easily be adapted for a faster solution finding process, while at the same time maintaining high quality solutions.

In the second decomposition approach a VNS was used as optimizer in the POPMUSIC framework. We show that the results achieved by the population-based MA approach can easily be improved by 6.17% and can outperform the pure VNS approach by roughly 6.76% over all instances. The obtained results therefore again reflect the ability of the developed decomposition approaches to amplify the effectiveness of already good working metaheuristics like the already high performing VNS approach.

Another contribution of this thesis is the presentation of a number of different ways how proximity can be measured in an environment with a large amount of customers and more than one depot so that large routing problem can intelligently be decomposed into smaller problems. We analyzed eight different measures; five of them based on relatedness concerning distance measured in travel time, and three that use a Sweep mechanic to establish relatedness in trigonometric ways. The results show that the distance based proximity measures provide the best results (especially strategy D_{III} and D_{IV}), while properly implemented sweep based measures only work well when dealing with a higher number of depots (especially strategy S_{III}). In the real world, were carrier fleet operators have to service customers in highly populated regions as well as in more rural areas the choosing of a suitable proximity measure directly affects the quality of the resulting routing. The best performing measure D_{IV} is exploiting exactly this structural distribution of customers, by limiting the creation of the sub-problem into the direction of highly populated areas. The use of this measure therefore guides the search more or less evenly around the complete geographical region. A good inclusion of the rural customers in routes that are also serving customers in densely populated areas is highly important as they can have a significant impact when minimizing total distance traveled. Nevertheless we want to point out that using even very basic decomposing strategies can yield improvements around 5% better than when using no decomposition at all.

We therefore conclude that decomposition improves the solution quality significantly when tackling large scale problems with current state-of-the art methods and computers on the basis of the same runtime. Furthermore we showed that interaction between the decomposed parts is crucial for rearranging borderline customers, especially when dealing with restrictions like time windows. The POPMUSIC framework has proven to be easily developed and customized for large scale VRP instances so that sub-problem optimizers based on metaheuristic concepts can flexibly be integrated to further improve solution quality. We can conclude that local search based concepts that work with one incumbent solution, e.g. VNS or TS are high performing and suitable to be used within the POPMUSIC framework. Even though the approaches were specially developed for dealing with multiple depots, we assume that decomposing with the use of good proximity measures may result in better solution qualities for other problems in the VRP class.

A. Abbreviations

Abbreviation	**Description**
ACO	Ant Colony Optimization
CVRP	Capacitated Vehicle Routing Problem
DCVRP	Distance-constrained Capacitated Vehicle Routing Problem
ES	Evolutionsstrategie
GA	Genetic Algorithm
GIS	Geographic Information System
GLS	Guided Local Search
GTS	Granular Tabu Search
IP	Integer Programming
LS	Local Search
MA	Memetic Algorithm
MDVRP	Multi Depot Vehicle Routing Problem
MDVRPTW	Multi Depot Vehicle Routing Problem with Time Windows
MIP	Mixed Integer Programming
MP-VRP	Multi Pile Vehicle Routing Problem
NN	Nearest Neighbor
OX	Order Crossover
PMX	Partially Mapped Crossover
POPMUSIC	Partial Optimization Metaheuristic Under Special Intensification Conditions
PopMA	POPMUSIC with MA optimizer
PopVNS	POPMUSIC with VNS optimizer
PVRP	Periodic Vehicle Routing Problem
QAP	Quadratic Assignment Problem
RPD	Random Percentage Deviation
SA	Simulated Annealing
STDV	Standard Deviation
SVRP	Split Delivery Vehicle Routing Problem
TOP	Team Orienteering Problem
TS	Tabu Search
TSP	Traveling Salesman Problem
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPBTW	Vehicle Routing Problem with Backhauls and Time Windows
VRPPD	Vehicle Routing Problem with Pickups and Deliveries
VRPPDTW	Vehicle Routing Problem with Pickups and Deliveries and Time Windows
VRPTW	Vehicle Routing Problem with Time Windows

B. Notation

Table B.1.: Multi Depot Vehicle Routing Problem with Time Windows

Symbol	**Description**
G	Graph
V	Set of vertices
A	Set of arcs
i	Index of customers
v_i	Vertex of set V
n	Number of customers
m	Number of depots
c_{ij}	Cost from customer i to customer j
q_i	Demand of customer i
s_i	Service time at customer i
e_i	Earliest arrival time at customer i
l_i	Latest arrival time at customer i
K	Set of vehicles
D	Vehicel capacity
T	Maximum allowed tour duration
$c(x)$	Total travel time of solution x

Table B.2.: POPMUSIC framework

Symbol	Description
S	Solution
i	Index of part
s_i	Part i of solution S
A	Set of parts
R_i	Sub-problem related to part i
r	Number of parts to form a sub-problem

Table B.3.: Variable Neighborhood Search

Symbol	Description
κ	Index of a neighborhood
N_κ	Neighborhood
κ_{max}	Neighborhood delimiter
x, x', x''	Solutions
S	Solution space
X, X'	Node of a route
Y, Y'	Node of a route
i, k	Index of a route
C_k	Number of customers in route k
p_{iCross}	Probability to reverse sequence orientation
it_u	Unproductive iterations
p_t	Threshold for accepting worse solutions

Table B.4.: Clark and Wright Savings

Symbol	Description
i, j	Customers
z	Depot
$c_{i,j}$	Travel time from customer i to customer j
n	Number of customers

Table B.5.: Evaluation Function - Acceptance decision

Symbol	Description
S	Solution
i	Index of customer
a_i	Arrival time at customer i
$f(S)$	Evaluation Function
$l(S)$	Total Violation of Load of Solution x
$d(S)$	Total Violation of Duration of Solution x
$w(S)$	Total Violation of Time Window Constraints of Solution x
α, β, γ	Penalty Parameters

Table B.6.: Memetic Algorithm

Symbol	Description
pop	Population
$popsize$	Population size pop
O_1, O_2	Offsprings
S_1, S_2	Solutions for recombination
B	Number of pairs of routes to be recombined
b	Index of pair of routes to be recombined
R_{1b}, R_{2b}	Routes for recombination
R_{O1b}, R_{O2b}	Routes after recombination
p_1, p_2	Probabilities to apply Stochastic Local Search
it_{vns}	Iteration limit for the VNS

Table B.7.: Ant Colony Optimization

Symbol	**Description**
i, j	Customers
z	Depot
s_j	Customer immediately after j
u	Number of Solutions
η	Attractiveness value
FV	Fitnessvalue
α, β	Weights
τ	Pheromone concentration
p	Number of ranks
ρ	Trail persistence
σ	Number of elitists
μ	Rank of the Solution
it_{stuck}	Unproductive Iterations

Table B.8.: p-Median formulation

Symbol	**Description**
p	Number of Clusters
it	Iteration
it_{dec}	Iteration limit p-Median
c	Cluster
V_c	Maximum capacity of cluster c
Q_c	Overall demand
λ_c	Lagrangian coefficient
i, j	Customers
c_{ij}	Distance (cost) between i and j
d_i	Demand of customer i
Π_{ij}	New distance measure
it_{ini}	Iteration limit for optimizer to build initial routes
$csize$	Number of Customers

Table B.9.: Decomposition Strategies

Symbol	**Description**
s	Sub-problem
i	Index of Sub-problem
t_i	Assigned time to solve i
t_{max}	Time limit
C_{s_i}	Size of Problem s_i
C_{s_n}	Total problem size
p	Number of Clusters
D_{seed}	Seed depot
D_{add}	Depot added to sub-problem
R_{seed}	Seed route
R_{add}	Route added to sub-problem
α, β	Angles

C. Acknowledgment

Financial support from the Oesterreichische Nationalbank (OENB) under grant #11984 and from the Austrian Science Fund (FWF) under grant #P20342-N13 and #L362-N15 (Translational Research) is gratefully acknowledged.

Bibliography

Acan, A. (2005). An external partial permutations memory for ant colony optimization. *Proc. of the 5th European Conference on Evolutionary Computation in Combinatorial Optimization*, 3448:1–11.

Acan, A. and Unveren, A. (2007). A shared-memory ACO+ GA hybrid for combinatorial optimization. *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 2078–2085.

Archetti, C., Hertz, A., and Speranza, M. (2007). Metaheuristics for the Team Orienteering Problem. *Journal of Heuristics*, 13:49–76.

Berger, J. and Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 31:2037–2053.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15:347–368.

Bräysy, O., Dullaert, W., and Gendreau, M. (2004). Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows. *Journal of Heuristics*, 10(6):587–611.

Bräysy, O. and Gendreau, M. (2005a). Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39:104–118.

Bräysy, O. and Gendreau, M. (2005b). Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39:119–139.

Bullnheimer, B., Hartl, R., and Strauss, C. (1999). An improved Ant System algorithm for theVehicle Routing Problem. *Annals of Operations Research*, 89:319–328.

Christofides, N., Mingozzi, A., and Toth, P. (1979). The vehicle routing problem. *Combinatorial Optimization*, 11:315–338.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581.

Colorni, A., Dorigo, M., Maniezzo, V., et al. (1991). Distributed optimization by ant colonies. *Proceedings of the First European Conference on Artificial Life*, 142:134–142.

Cook, W. and Rich, J. L. (1991). *A Parallel Cutting-Plane Algorithm for the Vehicle Routung Problems with Time Windows.* Department of Computational and Applied Mathematics, Rice University. Technical Report TR99-04.

Cordeau, J.-F., Desrosiers, J., Solomon, M. M., and Soumis, F. (2001a). The VRP with Time Windows. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications*, pages 157–194. SIAM, Philadelphia.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001b). A unified tabu search heuristic for the vehicle routing problems with time windows. *Journal of the Operation Research Society*, 52:928–936.

Desrosiers, J., Dumas, Y., Solomon, M. M., and Soumis, F. (1995). Time Constrained Routing and Scheduling. In *Handbooks in Operations Research and Management Science 8: Network Routing*, pages 35–139. Elsevier Publishers, Amsterdam.

Fleszar, K., Osman, I., and Hindi, K. (2008). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research.* accepted for publication.

Fogel, L., Owens, A., and Walsh, M. (1966). *Artificial Intelligence Through Simulated Evolution.* John Wiley & Sons Inc.

Gillet, B. and Miller, L. (1974). A heuristic algorithm for the dispatch problem. *Operations Research*, 22:340–349.

Giosa, I., Tansini, I., and Viera, I. (2002). New assignment algorithms for the multi-depot vehicle routing problem. *Journal of the Operation Research Society*, 53:977–984.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Pub. Co.

Gutjahr, W. J. (2002). ACO algorithms with guaranteed convergence to the optimal solution. *Information Processing Letters*, 82:145–153.

Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13:462–475.

Hansen, P. and Mladenović, N. (1999). An introduction to variable neighborhood search. In Voss, S., Martello, S., Osman, I. H., and Roucairol, C., editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer Academic Publishers.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467.

Hemmelmayr, V., Doerner, K., Hartl, R., and Savelsbergh, M. (2008). Delivery strategies for blood products supplies. *OR Spectrum*. accepted for publication.

Hemmelmayr, V. C., Doerner, K. F., and Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791 – 802.

Holland, J. (1975). *Adaption in Natural and Artificial Systems*. University of Michigan Press. re-issued by MIT Press (1992).

Homberger, J. and Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162:220–238.

Kytöjoki, J., Nuortio, T., Bräysy, O., and Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34:2743–2757.

Larsen, J. (1999). Paralellization of the vehicle routing problem with time windows. In *Ph.D. thesis*. Institute of Mathematic Modelling, Technical University of Denmark.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269.

Mester, D. and Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, 32:1593–1614.

Mester, D. and Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 34:2964–2975.

Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100.

Moscato, P. and Cotta, C. (2003). A gentle introduction to memetic algorithms. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 105–144. Springer.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. *Annals of Operations Research*, 41:421–52.

Ostertag, A., Hartl, R. F., and Doerner, K. F. (2008a). A variable neighborhood search integrated in the popmusic framework for solving large scale vehicle routing problems. *Lecture Notes in Computer Science*, 5269:29–42.

Ostertag, A., Hartl, R. F., Doerner, K. F., Taillard, E. D., and Waelti, P. (2008b). Popmusic for a real world large scale vehicle routing problem with time windows. *Journal of the Operational Research Society*. to appear.

Polacek, M., , Benkner, M., Doerner, K., and Hartl, R. F. (2008a). A Cooperative and Adaptive Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *Business Research*, 1:1–13.

Polacek, M., Doerner, K., Hartl, R. F., Kiechle, G., and Reimann, M. (2007). Scheduling Periodic Customer Visits for a Traveling Salesperson. *European Journal of Operational Research*, 179:823–837.

Polacek, M., Doerner, K., Hartl, R. F., and Maniezzo, V. (2008b). A Variable Neigborhood Search for the Capacitated Arc Routing Problem with Intermediate Facilities. *Journal of Heuristics*, 14(5):405–423.

Polacek, M., Hartl, R. F., Doerner, K., and Reimann, M. (2004). A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *Journal of Heuristics*, 10:613–627.

Potvin, J. and Rousseau, J. (1995). An Exchange Heuristic for Routeing Problems with Time Windows. *Journal of the Operational Research Society*, 46:1433–1446.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31:1985–2002.

Rechenberg, I. (1973). Evolutionsstrategie-Optimierung technischer Systeme nach Prinzipien der biologischen Information. *Freuburg: Fromann.*

Reeves, C. (2003). Genetic algorithms. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 55–82. Springer.

Reimann, M., Doerner, K., and Hartl, R. (2002a). Insertion based ants for vehicle routing problems with backhauls and time windows. *Lecture Notes in Computer Science*, 2463:73–94.

Reimann, M., Doerner, K., and Hartl, R. F. (2004). D-ants: Saving based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31:563–591.

Reimann, M., Stummer, M., and Doerner, K. (2002b). A Savings Based Ant System For The Vehicle Routing Problem. *Proceedings of the Genetic and Evolutionary Computation Conference table of contents*, pages 1317–1326.

Rochat, Y. and Semet, F. (1994). A Tabu Search Approach for Delivering Pet Food and Flour in Switzerland. *Journal of the Operational Research Society*, 45(11):1233–1246.

Rochat, Y. and Taillard, É. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167.

Salhi, S. and Sari, M. (1997). A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. *European Journal of Operational Research*, 103:95–112.

Savelsbergh, M. (1992). The vehicle routing problem with time windows: minimizing route duration. *INFORMS Journal of Computing*, 4:146–54.

Schwefel, H. (1975). *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technische Universität Berlin.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. Technical report, ILOG SA.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 32(2):254–265.

Taillard, E. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks - New York-*, 23:661–661.

Taillard, E. D. (2003). Heuristic methods for large centroid clustering problems. *Journal of Heuristics*, 9:51–73.

Taillard, E. D., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J. Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31:170–186.

Taillard, E. D. and Voss, S. (2001). Popmusic: Partial optimization metaheuristic under special intensification conditions. In Ribeiro, C. and Hansen, P., editors, *Essays and surveys in metaheuristics*, pages 613–629. Kluwer Academic Publishers.

Tan, K., Lee, L., and Ou, K. (2001). Hybrid Genetic Algorithms in Solving Vehicle Routing Problems with Time Window Constraints. *Asia Pacific Journal of Operational Research*, 18(1):121–130.

Tansini, L. and Viera, O. (2006). New measures of proximit for the assignment algorithm in the MDVRPTW. *Journal of the Operation Research Society*, 57:241–249.

Toth, P. and Vigo, D. (2001). *An overview of vehicle routing problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Toth, P. and Vigo, D. (2003). The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15(4):333.

Tricoire, F., Doerner, K., Hartl, R. F., and Iori, M. (2007). Variable neighbourhood search for the multi-pile vehicle routing problem. Technical report, Universita di Bologna - D.E.I.S. - Operations Research.

Voudouris, C. (1997). *Guided Local Search for Combinatorial Optimisation Problems.* PhD thesis, University of Essex.

Voudouris, C. and Tsang, E. (1999). Guided local search. *Journal of Operations Research*, 113:80–119.

Waelti, P., Taillard, E. D., and Mautor, T. (2002). Cueillir du mimausa en écoutant de la popmusic. Technical report, MiS-TIC Institute, HEIG-Vd.

Whitley, D. (1987). Using reprodictive evaluation to improve genetic search and heuristic discovery. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, pages 108–115. Lawrence Erlbaum Associates.

Abstract

The optimization of transportation activities is of high importance for companies in today's economy. The Vehicle Routing Problem (VRP) class is dealing with the routing of vehicles so that the customer base of a company can be served in the most efficient way. One of the many variants in the VRP class is the Multi Depot Vehicle Routing Problem with Time Windows (MDVRPTW) which extends the VRP by additional depots from which customers can be served, as well as an individual time window for each customer in which he is allowed to be served. Modern carrier fleet operators often encounter these MDVRPTW in the real world, and usually they are of very large size so that exact approaches cannot solve them efficiently. This thesis presents two different approaches how this real world large scale MDVRPTWs can be solved. Both approaches are based on the POPMUSIC framework, which intelligently tries to decompose the large scale problem into much smaller sub-problems. The resulting sub-problems can then be solved more efficiently by specialized optimizers. The first approach in this thesis was developed for population based optimizers. A Memetic Algorithm (MA) was developed and used as an optimizer in the framework to solve a real world MDVPRTW from an Austrian carrier fleet operator. We show that decomposing the complete problem and solving the resulting sub-problems improves the solution quality by around 20% compared to using the MA without any decomposition. The second approach specially focuses on decomposition strategies for single solution methods. More precisely, a Variable Neighborhood Search (VNS) was implemented in the POPMUSIC framework to solve the real world instances. We show that decomposing the problem can yield improvements of around 7% compared to using the pure VNS method. Compared to the POPMUSIC MA approach the second approach can further improve the solution quality by around 6%. Another contribution in this thesis is the development of two generally different ways to measure proximity when creating sub-problems. In detail we tested eight different proximity measures and analyzed how good they decompose the problem in different environments. We tested the two, three and four depot case and present a clear winner that can outperform all other measures. Further we demonstrate that the POPMUSIC approach can flexibly be adjusted to real world demands, like a faster solution finding process, while at the same time maintaining high quality solutions. We show that a decomposition strategies combined with state of the art metaheuristic solvers are a very efficient and flexible tool to tackle real world problems with regards to solution quality as well as runtime.

Abstract in German

Das Umfeld in der heutigen Wirtschaft verlangt nach immer bessern Ansätzen, um Transportprobleme möglichst effizient zu lösen. Die Klasse der "Vehicle Routing Problems" (VRP) beschäftigt sich speziell mit der Optimierung von Tourenplanungsproblemen in dem ein Service-Leister seine Kunden möglichst effizient beliefern muss. Eine der VRP-Varianten ist das "Multi Depot Vehicle Routing Problem with Time Windows" (MDVRPTW), in dem Kunden von verschiedenen Depots in einem fix vorgegebenen Zeitintervall beliefert beliefert werden müssen. Das MDVRPTW ist im realen Leben dank seiner realitätsnahen Restriktionen sehr oft vertreten. Typische Transportprobleme, wie sie in der Wirklichkeit auftreten, sind jedoch oftmals so groß, dass sie von optimalen Lösungsansätzen nicht zufriedenstellend gelöst werden können.

In der vorliegenden Dissertation werden zwei Lösungsansätze präsentiert, wie diese riesigen, realitätsnahen Probleme zufriedenstellend bewältigt werden können. Beide Ansätze benutzen die POPMUSIC Grundstruktur, um das Problem möglichst intelligent zu dekomponieren. Die Dekomponierten und damit kleineren Subprobleme können dann von speziell entwickelten Algorithmen effizienter bearbeitet und letztendlich gelöst werden. Mit dem ersten Ansatz präsentieren wir eine Möglichkeit Transportprobleme zu dekomponieren, wenn populationsbasierte Algorithmen als Problemlöser eingesetzt werden. Dazu wurde ein maßgeschneiderter Memetischer Algorithmus (MA) entwickelt und in das Dekompositionsgerüst eingebaut um ein reales Problem eines österreichischen Transportunternehmens zu lösen. Wir zeigen, dass die Dekomponierung und Optimierung der resultierenden Subprobleme, im Vergleich zu den Ergebnissen des MA ohne Dekomposition, eine Verbesserung der Zielfunktion von rund 20% ermöglicht.

Der zweite Ansatz beschäftigt sich mit der Entwicklung einer Dekomponierungsmethode für Lösungsalgorithmen, die nur an einer einzigen Lösung arbeiten. Es wurde ein "Variable Neigborhood Search" (VNS) als Optimierer in das POPMUSIC Grundgerüst implementiert, um an das vorhandene Echtwelt-Problem heranzugehen. Wir zeigen, dass dieser Ansatz rund 7% bessere Ergebnisse liefert als der pure VNS Lösungsansatz. Außerdem präsentieren wir Ergebnisse des VNS Dekompositionsansatzes die um rund 6% besser sind als die des MA Dekompositionsansatzes.

Ein weiterer Beitrag dieser Arbeit ist das Vorstellen von zwei komplett verschiedenen Ansätzen um das Problem in kleinere Sub-Probleme zu zerteilen. Dazu wurden acht verschiedene Nähe-Maße definiert und betrachtet. Es wurde der 2,3 und 4 Depot Fall getestet und im Detail analysiert. Die Ergebnisse werden präsentiert und wir stellen einen eindeutigen Gewinner vor, der alle Testinstanzen am Besten lösen konnte. Wir weisen auch darauf hin, wie einfach die POPMUSIC Dekomponierung an reale Bedürfnisse, wie zum Beispiel eine möglichst schnelle Ergebnisgenerierung, angepasst werden kann. Wir zeigen damit, dass die vorgestellten Dekomponierungsstrategien sehr effizient und flexibel sind, wenn Transportprobleme, wie sie in der realen Welt vorkommen gelöst werden müssen.

Alexander Ostertag

Personal Data

date of birth	April 18, 1978 in St. Pölten
address	Helferstorferstr. 50, 2344 Maria Enzersdorf
email	alexander.ostertag@univie.ac.at
citizenship	Austrian

Education

since 2005	Ph.D. Program Business Administration, University of Vienna
2000-2005	Master Program in International Business Administration, University of Vienna
1997-1999	Master Program in International Business Administration, University of Vienna

Master thesis

Title	*Memetic Algorithms for the Multi Depot Vehicle Routing Problem with Time Windows*
Supervisor	O.Univ.Prof. Dipl.-Ing. Dr. Richard F. Hartl

Ph.D. thesis

Title	*Decomposition Strategies for Large Scale Multi Depot Vehicle Routing Problems*
Supervisor	O.Univ.Prof. Dipl.-Ing. Dr. Richard F. Hartl

Languages

German	Mother tongue
English	Advanced, fluent in written and spoken
Spanish	Good

Computer skills

Programming	C, C++
Technical	Artisan, Auto Cad
ERP	Navision

Interests

Research	Transportation, Metaheuristics, Vehicle Routing Problems
Sports	Snowboarding, Surfing, Mountainbiking
Traveling	Indonesia, Spain, Portugal, France

Honors and Awards

Honor for diploma thesis	“Prämierung ausgezeichneter Diplomarbeiten des Instituts für BWL”, University of Vienna

Presentations at Conferences

June 2007	**MIC 2007: The Seventh Metaheuristics International Conference**, *Montreal (Canada)*, POPMUSIC for a Real World Large Scale Vehicle Routing Problem with Time Windows, Ostertag, A., Doerner, K.D., Hartl, R.F., Taillard, E.D. and Waelti, P..
October 2008	**HM 2008: International Workshop on Hybrid Metaheuristics**, *Malaga (Spain)*, A Variable Neighborhood Search Integrated in the POPMUSIC Framework for Solving Large Scale Vehicle Routing Problems, Ostertag, A., Doerner, K.D., Hartl, R.F..

Publications

A. Ostertag, R. F. Hartl, and K. F. Doerner. A variable neighborhood search integrated in the popmusic framework for solving large scale vehicle routing problems. *Lecture Notes in Computer Science*, 5269:29–42, 2008.

A. Ostertag, R. F. Hartl, K. F. Doerner, E. D. Taillard, and P. Waelti. Popmusic for a real world large scale vehicle routing problem with time windows. *Journal of the Operational Research Society*, 2008. to appear.