Scheduling in the Water Business

Diploma Thesis

by

Fabian König

November 30, 2008

Supervisor:	Prof. Dr. Dorothea Wagner	Faculty of Informatics
Advisor:	Dipl.-Math. Reinhard Bauer	Faculty of Informatics

Universität Karlsruhe
Faculty of Informatics

Acknowledgments I would like to thank Prof. Dr. Dorothea Wagner for the opportunity to work on an interesting topic and for allowing a high level of freedom in the definition of the problem statement as well as the embodiment of this work. Special thanks go to my supervisor Reinhard Bauer for his steady support, his excellent guidance and his friendly and helpful way; Denise Torpey and Karen Dence for their readiness to shed light onto the bits and pieces of the real-life scheduling processes; Scott Eastwood and Mark Ridgley for granting access to the company data and providing the contacts, networks and system access that were necessary to gain a thorough insight of the real-world problems; Katie Carter, Linda Glennon, Brent Lowe and Mike Kowalski for their great support in obtaining the handwritten copies that were the basis for our test instances. I finally want to thank Felix Schuttack, Sung-Gon Choe and Erik Leukart for supporting me in the tedious work of manual data entry.

Declaration

I declare that I have written this thesis by myself and have not used any sources or assistance other than those listed.

Karlsruhe, November 30, 2008 .....

Fabian König

Contents

1 Introduction **7**

2 Problem Description **9**
2.1 Water Scheduling in Practice 10
2.2 Model 12
2.3 ILP Formulations 14
2.3.1 Water Scheduling Routing Problem 15
2.3.2 Water Scheduling Makespan Routing Problem 16
2.3.3 Water Scheduling Combined Routing Problem 16
2.4 Related Problems 17

3 Heuristic Approaches to Water Scheduling **19**
3.1 Greedy Construction 19
3.2 Decomposition Strategy 20
3.2.1 Assignment Phase 21
3.2.1.1 k-means 21
3.2.1.2 Greedy Assignment 23
3.2.2 Tour Construction Phase 26
3.2.2.1 ILP 26
3.2.2.2 Greedy Roundtour Optimization 26
3.3 Post Optimization 27
3.3.1 2-Opt 28
3.3.2 Greedy 2-Opt 28

4 Experiments **30**
4.1 Naming Conventions and Units Of Measure 30
4.2 Strengths and Weaknesses of the *ILP* Formulations 33
4.2.1 Water Scheduling Makespan Routing Problem vs. WSRP 33
4.2.2 Water Scheduling Combined Routing Problem vs. WSMRP & WSRP 34
4.3 Instances of Test Data 34
4.3.1 Origin and Obtaining of Test Instance Data 36
4.3.2 Characteristics of the Test Instances 37
4.4 Experimental Setups and Lead Through 39
4.4.1 Hard- and Software Used for Calculations 39

4.4.2 Experimental Configurations and Lead Through 39
4.4.2.1 Key Questions and Lead Through 40
4.4.2.2 Combinations of Scheduling Approaches 41
4.5 Experimental Results 42
4.5.1 Runtime Analysis 42
4.5.1.1 ILP Performance 42
4.5.1.2 Heuristic Scheduling Performance 48
4.5.1.3 Conclusions on Performance 49
4.5.2 Analysis of Scheduling Quality 53
4.5.2.1 Overall Cost of Schedules 53
4.5.2.2 Overall Technician Cost 58
4.5.2.3 Conclusions on Scheduling Quality 59

5 Final Remarks **61**
5.1 Conclusions 61
5.2 Further optimization prospects 62

Index **64**

6 Acronyms **66**

Bibliography **67**

Zusammenfassung

In der vorliegenden Arbeit befassen wir uns mit einem Schedulingproblem aus dem Bereich der wasserverarbeitenden Industrie, die gewisse Anforderungen an ihre Schedules stellt, welche in bisherigen *Traveling Salesman Problem*, *Multiple Traveling Salesman Problem* oder *Vehicle Routing Problem* Formulierungen so nicht berücksichtigt sind. Die Problemstellung wurde motiviert durch die Zusammenarbeit mit der US-Depandance eines weltweit operierenden Konzerns, dessen Kerngeschäft die Aufbereitung, Filtration und Reinigung von Wasser in allen Bereichen des kommunalen und industriellen Umfelds darstellt.

In dieser Arbeit untersuchen wir verschiedene Ansätze zur Erzeugung von Techniker-Kunden-Zuordnungen und Techniker Routings. Den in der Praxis auftretenden Nebenbedingungen wird in allen Schritten Beachtung geschenkt. Wir werden uns auf die Nebenbedingungen für Skill Requirements (Anforderungen Seitens des Kunden an den Ausbildungsstand des Technikers) sowie maximale Schicht-Arbeitszeiten konzentrieren, da diesen bei der Erstellung eines realistischen Schedules die größte Bedeutung zukommt und sie bereits einen nennenswerten Beitrag zur Komplexität des Gesamtproblems leisten. Einige weitere Nebenbedingungen, wie z.B. Teileverfügbarkeit oder eine Priorisierung der Kundenbesuche, können durch eine Vorverarbeitung und entsprechende Einschränkung der Eingabedaten in unsere Formulierungen realisiert werden.

Wir führen eine experimentelle Studie unserer vorgeschlagenen Ansätze durch, basierend auf Realweltdaten, die dem oben genannten Unternehmen entstammen. Wir legen dabei großen Wert auf die Realitätsnähe der Szenarien und haben den Anspruch, dass die präsentierten Lösungsansätze direkt auf die Realweltanwendung beim Unternehmen angewendet werden können.

1 Introduction

Companies operating a large field service fleet are often challenged to maximize the efficiency and throughput of their field service technicians while satisfying a number of side-constraints that greatly differ between different sectors of industries. In the example of the water purification and treatment business some of the most important constraints are skill requirements for a technician to provide service to certain customers including customer-specific trainings, preferred technicians, high-priority service visits as well as short-notice emergencies, maximum working cost, limited truck load capacity, spare parts availability and time windows of service. Finding an optimal schedule and routing while satisfying all these constraints is a challenging task and the diversity in requirements from company to company make it hard to find an out-of-the-box scheduling solution that meets all requirements.

In this work we examine different approaches for creating customer-technician assignments and technician routings while always taking the side-constraints that occur in practice into account. We will focus on the constraints for skill requirements and maximum working cost since these are most important when obtaining a realistic schedule and they already add a considerable level of complexity to the problem. Some of the other constraints, such as spare parts availability and high-priority service visits, can be modeled by limiting the input data to our problem formulation. We perform a case study based on real-world data for service requests as well as customers and technician configurations, which stem from a large company with operations all over the US and Canada. We attached great importance to maintaining realistic scenarios that can be applied back to the real-world application.

Overview

In Chapter 2 of this work we introduce the problem including the different side constraints in detail. We also present a formal model and three *Integer Linear Program* (*ILP*) formulations. Furthermore we provide an overview of the different variants of *TSPs* used in this work and present related work.

Chapter 3 describes different heuristic approaches to the problem. We examine a *Greedy Construction* approach as well as a *Decomposition Strategy* consisting of two phases: an assignment phase and a tour construction phase. We also take post optimization

techniques into account which further improve the quality of the heuristic solutions.

In Chapter 4 we present experimental results obtained by our implementations and compare the different approaches according to performance and solution quality. We also compare our solutions to the real-world manual scheduling practices of a large water purification company in the US.

The last Chapter 5 summarizes the results and insights and points out further prospects for optimization.

2 Problem Description

In this chapter we get acquainted with the problem behind this thesis. We describe the problem which stems from a real-world application in a major water treatment company in detail and present models that formally describe the problem. We briefly introduce related problems and provide references to related work.

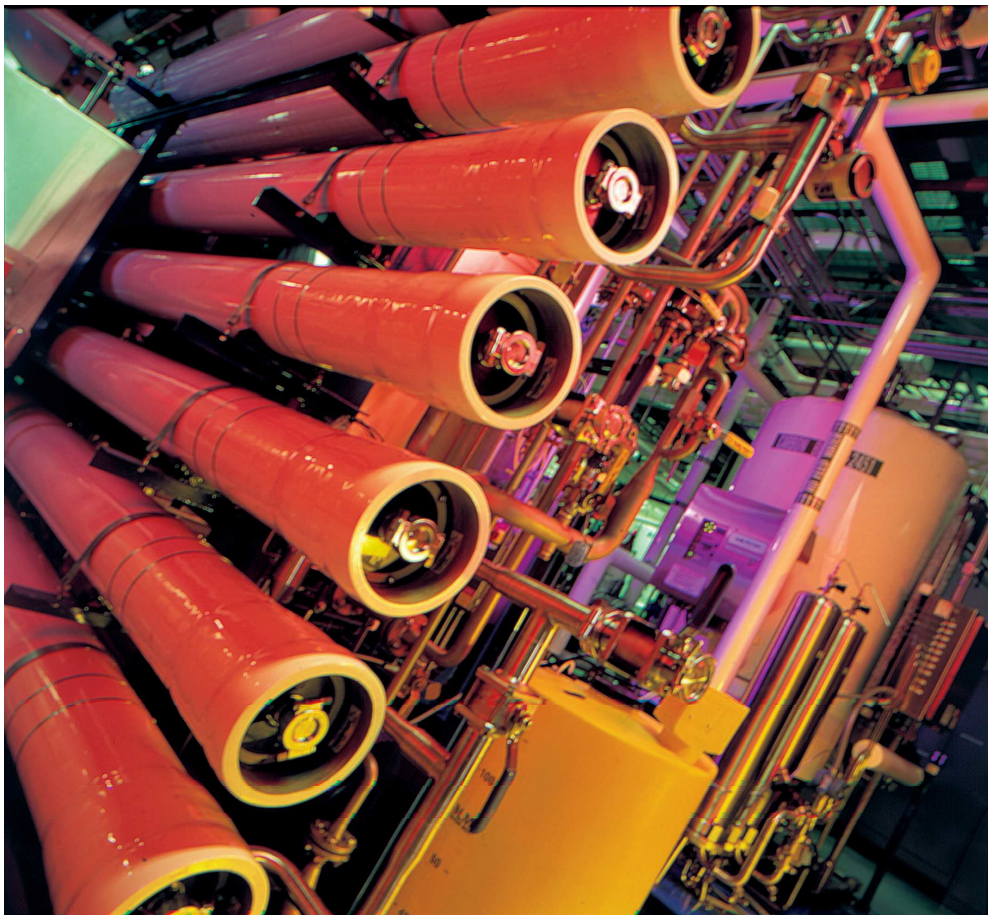


Figure 2.1: Reverse-osmosis facility for production of high purity water for pharmaceutical products

2.1 Water Scheduling in Practice

The problem of Scheduling in the Water Business arose in a major water purification and treatment company with more than 90 branches all over the Unites States and Canada.

Company Operations

The company services a wide range of customers from different backgrounds. Municipal products and services include drinking water treatment, ground water remediation and contamination removal, biosolids reduction such as sludge dewatering and drying, waste water treatment and water desalination. The Industrial applications range from Biopharmaceutical to Food and Beverage, from Automotive to Environmental Remediation, Mining, Metals and Metal Finishing, from Oil and Gas to Power, Semiconductor and Solar, just to name a few. Special solutions are offered for laboratory applications such as hemodialysis which require highest purity and guaranteed quality standards. Also aquatics and leisure products and services for fountains, swimming pools or theme and amusement parks are contained in the portfolio.

Figure 2.2: Fibers of a membrane filter

The employed technologies vary greatly. Diffused aeration, biological treatment, granular activated carbon filtration, UV disinfection, membrane filtration, ion exchange and polymeric micro filtration are just a short list of examples. The installations at the customer sites come in all imaginable sizes and variations. There are small under-sink cartridge filters and there are multi-story resin filters for large scale filtrations and deionization, advanced high-purity systems using reverse-osmosis and so forth. Some installations supply entire cities up to the size of New York City with drinking water and take care of non-drinking water that goes down the pipe. If there is one thing that is true all over the company, it is that no two installations are equal.

Methods of Operation

Almost as different as the customer base are the branches and their ways of operation, in order to optimally meet the requirements of the customers in their region. Service dispatchers in each branch are typically manually dispatching and scheduling service requests to their technicians, basing their decisions of who to send where and in which sequence purely on what we have come to refer to as “tribal knowledge”. Long years of experience in the business and in their local area are key in order to create efficient and cost-saving schedules for their fleet of service technicians. Besides the experience of the

service dispatchers, also the technicians themselves are required to have good knowledge of their typical service areas, particularly when driving big trucks through downtown areas where narrow alleys with parking cars and one-way streets make it hard to come through.

With the increasing number of customers, partly in metropolitan areas, partly in rural regions, and a growing technician base, the task of scheduling and organizing the service fleet becomes harder and harder. Rising prices for gas put an additional pressure on the quality of the schedules, extra tours have to be prevented and the available service fleet has to be optimally utilized in order to reduce cost. Short response times to customer call-ins do not allow for long and extensive planning.

When it comes to the crunch and sewage pipes break, filter tanks leak or the quality control lights on laboratory equipment turn red the technician is required on-site within a few hours. The technicians load their trucks, grab a bunch of service requests from the "preventative maintenance" pile to fill up their day and hit the road. The sequence of those "fill-in" stops is then usually decided while gulping down a cup of coffee and heading towards the ramp.

Our Modeling Approach

In order to facilitate the scheduling activities and improve the efficiency of the daily routes, an automated scheduling system is desirable. This system has to be capable of scheduling routes out of thousands of service requests within seconds, because the emergency calls typically come in in the morning, short before the technicians head out towards the customers. The time between the call and the requirement for a new route is very limited. In peak times, up to 30 branches request schedules for several hundred technicians within 30 to 60 minutes. Because spare parts, filter tanks and vessels are big and heavy, only those parts that are absolutely required during the day can go on the truck. Often the number of stops a technician can perform during a day is not only limited by the drive time and on-site working time, but also by the number of tanks that can be loaded on the trucks. This imposes the requirement to have the schedule for the entire day available in the morning, before the technicians start to load their trucks and leave the depot.

The great variety in offered services and installed products makes standardized customer service an impossible task. All technicians are trained and specialized on a set of products and technologies they know by heart, while there are other things they won't touch. Certain customers such as nuclear plants and military bases apply strict rules on who gets access to the sites, some of them require special training and certification exclusively for their systems. Skill levels and categories for both technicians and customers are therefore an indispensable requirement for a scheduling and routing solution. Different skill levels of technicians and skill requirements of customer installations have to be matched up and obeyed in the schedule. The system has to pick, out of several thousand preventative maintenance requests that can be serviced on any day during the entire month, the right ones which are located in the vicinity around the emergency stops to minimize drive time

and mileage. The runtime of the scheduling algorithms therefore has to be minimal.

In this thesis we compare manual schedules obtained from historic data of different branches with those created by our scheduling algorithms and examine how the different scheduling approaches perform compared to each other, where they have strengths and weaknesses. Due to the fact that we are using real-world data to test and benchmark our approaches, and by comparing the results with historic data from actual branches, we are able to maintain a high level of applicability to the real world.

Figure 2.3: Waste water treatment plant in Ruhleben (modernized by the German subsidiary) and comparison of a water sample before and after filtration. Ruhleben is the largest and most important waste water treatment plant in the greater Berlin area and processes 240,000 cubic meters of waste water daily.

2.2 Model

Given is a set $C' = \{c_i \mid i = 1, \ldots, N\}$ of N customers and the depot c_0. $C = C' \cup \{c_0\}$. Also given is a set $T = \{t_k \mid k = 1, \ldots, M\}$ of M technicians and a set $\Theta = \{S_l \in \mathbb{N}^+ \mid l = 1, \ldots, \Lambda\}$ of Λ skill categories. Furthermore, a cost function $\text{dist}(c_i, c_j) : C \times C \rightarrow \mathbb{R}^+$ is given which defines the cost associated with a technician traveling from customer c_i to c_j.

For each customer c_i the following properties and requirements are provided:

1. an on-site working cost $a_i \in \mathbb{R}^+$ (e.g. "time") required to provide service to customer c_i,
2. a set $\text{Skillset}(c_i)$ of skill requirements $\text{req}_s(c_i) \in \mathbb{N}^+, s \in \Theta$ a technician has to fulfill in order to provide service to c_i,

For each technician t_k the following properties and restrictions apply:

1. a set $\text{Skillset}(t_k)$ of skill levels $\text{skill}_s(t_k) \in \mathbb{N}^+, s \in \Theta$,

2. an amount of maximum working cost $W_k \in \mathbb{R}^+$ per day.

A *feasible routing solution* ξ_{ijk} is an assignment of technicians to customers and a sequencing of the individual customer visits within the technician routes which forms a round tour for each technician (i.e. starting location equals ending location), starting at the depot, that ensures visiting every customer exactly once and furthermore obeys the constraints stated below.

The ξ_{ijk} are defined as:

$$\xi_{ijk} = \begin{cases} 1 & \text{technician } t_k \text{ travels from customer } c_i \text{ directly to customer } c_j, \\ 0 & \text{otherwise} \end{cases} \tag{2.2.1}$$

The following constraints have to be obeyed by a *feasible routing solution*:

1. The travel cost $\text{dist}(c_i, c_j)$ plus on-site working cost a_i of all customers c_i, c_j on a technicians route must not exceed the maximum working cost W_k of a technician t_k on any given day. More formally:

$$\sum_{i=0}^{N} \sum_{j=0}^{N} \text{dist}(c_i, c_j) \cdot \xi_{ijk} + \sum_{i=1}^{N} \left(a_i \sum_{j=0}^{N} \xi_{ijk} \right) \leq W_k, \quad k = 1, \ldots, M \tag{2.2.2}$$

2. A technician t_k must have the same or a higher skill level $\text{skill}_s(t_k)$ in all skill categories $\text{req}_s(c_i)$ the customer c_i requires. More formally:

$$\text{req}_s(c_i) \cdot \xi_{ijk} \leq \text{skill}_s(t_k) \quad \forall i, j, k, s \tag{2.2.3}$$

3. Every customer c_i has to be visited exactly once. More formally:

$$\sum_{i=0}^{N} \sum_{k=1}^{M} \xi_{ijk} = 1, \quad j = 1, \ldots, N \tag{2.2.4}$$

4. The depot c_0 is start and endpoint for every technician route. More formally, in conjunction with (2.2.4):

$$\sum_{i=0}^{N} \xi_{ipk} - \sum_{j=0}^{N} \xi_{pjk} = 0, \quad k = 1, \ldots, M, \quad p = 0, \ldots, N \tag{2.2.5}$$

We examine three different problem formulations which have evolved throughout the course of our work.

Definition 1 The *Water Scheduling Routing Problem* (*WSRP*) is that of finding a *feasible routing solution* such that the sum Z of travel costs over all technician routes is minimized. More formally:

$$\text{Minimize } Z := \sum_{i=0}^{N} \sum_{j=0}^{N} \left(\text{dist}(c_i, c_j) \sum_{k=1}^{M} \xi_{ijk} \right) \tag{2.2.6}$$

Because of shortcomings of the *WSRP* formulation with regard to balancing of technician utilization, we have come up with the following alternative formulation:

Definition 2 The *Water Scheduling Makespan Routing Problem* (*WSMRP*) is that of finding a *feasible routing solution* such that the makespan $\overline{Z}$, i.e. the maximum over all technician routes of the sum of travel costs plus on-site working costs over all customers in a route, is minimized. More formally:

$$\text{Minimize } \overline{Z} \tag{2.2.7}$$

$$\text{subject to } \sum_{i=0}^{N} \sum_{j=0}^{N} (\text{dist}(c_i, c_j) + a_i) \cdot \xi_{ijk} \leq \overline{Z}, \quad k = 1, \ldots, M \tag{2.2.8}$$

The *WSMRP* formulation introduces an unwanted side effect for certain test instances, particularly when the resulting tours vary greatly in overall cost between the technicians. To compensate this effect we developed the following formulation:

Definition 3 The *Water Scheduling Combined Routing Problem* (*WSCRP*) is that of finding a *feasible routing solution* such that a new objective function $\widetilde{Z}$, consisting of the sum Z of travel costs over all technician routes and the makespan $\overline{Z}$ from *WSMRP*, is minimized. The influence of Z and $\overline{Z}$ is weighted by a factor α. More formally:

$$\text{Minimize } \widetilde{Z} := \alpha \cdot Z + (1 - \alpha) \cdot \overline{Z} \tag{2.2.9}$$
$$\text{subject to (2.2.8)}$$

We will learn more about the strengths and weaknesses of these formulations in Chapter 4 on page 30.

2.3 ILP Formulations

In the following chapter we present three slightly differing ILP formulations which have evolved during the course of our work. The effect of these differences will be further discussed in Chapter 4 on page 30.

2.3.1 Water Scheduling Routing Problem

The ILP formulation of the *Water Scheduling Routing Problem* (*WSRP*) is based on the *Christofides flow-based model* as stated in [NC81], which is basically a *Vehicle Routing Problem* (*VRP*) including constraints for vehicle capacity and unload cost as well as maximum cost per vehicle route.

$$\text{Minimize} \quad Z = \sum_{i=0}^{N} \sum_{j=0}^{N} \left(\text{dist}(c_i, c_j) \sum_{k=1}^{M} \xi_{ijk} \right), \tag{2.3.1}$$

$$\text{subject to} \quad \sum_{i=0}^{N} \sum_{k=1}^{M} \xi_{ijk} = 1, \quad j = 1, \ldots, N \tag{2.3.2}$$

$$\sum_{i=0}^{N} \xi_{ipk} - \sum_{j=0}^{N} \xi_{pjk} = 0, \quad k = 1, \ldots, M, \quad p = 0, \ldots, N \tag{2.3.3}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} \text{dist}(c_i, c_j) \cdot \xi_{ijk} + \sum_{i=1}^{N} \left(a_i \sum_{j=0}^{N} \xi_{ijk} \right) \leq W_k, \quad k = 1, \ldots, M \tag{2.3.4}$$

$$\sum_{j=1}^{N} \xi_{0jk} = 1, \quad k = 1, \ldots, M \tag{2.3.5}$$

$$u_i - u_j + N \sum_{k=1}^{M} \xi_{ijk} \leq N - 1, \quad i \neq j = 2, \ldots, N \tag{2.3.6}$$

$$\xi_{ijk} \in \{0, 1\}, \quad \forall i, j, k \tag{2.3.7}$$

$$\text{req}_s(c_i) \cdot \xi_{ijk} \leq \text{skill}_s(t_k) \quad \forall i, j, k, s \tag{2.3.8}$$

Expression (2.3.2) states that a customer must be visited exactly once. Expression (2.3.3) states that technician t_k has to depart from a customer c_p after he has visited him. Expression (2.3.4) ensures that the maximum working cost W_k is not exceeded; and we introduce support for varying maximum working costs for different technicians. Expression (2.3.5) forces every technician t_k to leave the depot and provide service to at least one customer while expression (2.3.6) eliminates subtours. Expression (2.3.7) finally enforces the ξ_{ijk} to be binary. With our added expression (2.3.8) we ensure that the assigned technicians t_k have the required skills for the customers c_i.

All constraints for a *feasible routing solution* are postulated in the *ILP*, i.e. every *ILP* solution corresponds to a *feasible routing solution*, and every *feasible routing solution* corresponds to a feasible solution of the *ILP*, thus they are equal.

2.3.2 Water Scheduling Makespan Routing Problem

The *WSMRP* can be formulated as an *ILP* as follows:

$$\text{Minimize} \qquad \overline{Z} \tag{2.3.9}$$

$$\text{subject to} \quad \sum_{i=0}^{N}\sum_{j=0}^{N}(\text{dist}(c_i, c_j) + a_i) \cdot \xi_{ijk} \leq \overline{Z}, \quad k = 1, \ldots, M \tag{2.3.10}$$

$$(2.3.2) \ldots (2.3.8)$$

In conjunction with the new objective function (2.3.9), the new constraint (2.3.10) minimizes the overall makespan $\overline{Z}$. The remaining side constraints (2.3.2)...(2.3.8) apply in the same way as in the *WSRP* formulation.

2.3.3 Water Scheduling Combined Routing Problem

The *WSCRP* can be formulated as an ILP as follows:

$$\begin{aligned} &\text{Minimize} && \widetilde{Z} = \alpha \cdot Z + (1 - \alpha) \cdot \overline{Z} \\ &\text{subject to} && (2.3.1) \ldots (2.3.8), (2.3.10) \end{aligned} \tag{2.3.11}$$

With the new objective function $\widetilde{Z}$, the *WSCRP* minimizes the overall makespan $\overline{Z}$ plus the sum of travel costs over all technician routes. The factor α allows a weighting of the influence that each of the aspects Z and $\overline{Z}$ shall have on the objective function and the resulting schedule. It is thus a combination of the *WSRP* and the *WSMRP*.

The *WSCRP* therefore finds a good compromise between the following properties:

- the technician workload is evenly distributed,
- the round tour of each technician is in itself efficiently sequenced,
- the overall makespan of the schedule is reduced.

2.4 Related Problems

In this section we will outline related problems that are commonly known and present their key aspects.

The Traveling Salesman Problem

The *Traveling Salesman Problem* (*TSP*) is the problem of finding a least-cost route through a given number of customer locations. Traveling from one customer to the next involves travel cost, this is typically either travel time or mileage. Each of the customers shall be visited exactly once and the salesman has to return to his starting location at the end of the roundtrip.

The fact that the *TSP* is NP-hard and thus not solvable in polynomial time makes it difficult to optimally solve the problem for large numbers of customers. Even in restricted cases, e.g. where all customer locations lie in a plane with Euclidean distances, the problem remains NP-hard. Dropping the constraint of visiting each customer exactly once, i.e. allowing multiple visits, does not simplify the problem, as an optimal round tour does not include multiple visits. Assuming the road network connecting the customers is a complete graph, the triangle inequality proves in a planar case that the tour length can be decreased by skipping an already visited customer.

In order to efficiently solve large *TSPs* the use of heuristics is often inevitable. Based on the side constraints of the particular problem, different heuristic approaches have been developed. We will examine several heuristics that are of interest for our application in the following chapters.

The Multiple Traveling Salesman Problem

The *Multiple Traveling Salesman Problem* (*mTSP*) is an extension of the *TSP* by the *Assignment Problem* where multiple salesmen or technicians are available to provide service to the customers. The problem now is two fold: besides finding an optimal route through a given number of cities, it needs to be determined which technician will provide service to which customers. It is obvious that the technician-customer assignments have a significant influence on the overall travel cost.

The Vehicle Routing Problem

The *VRPs* differ from the *mTSP* such that the *VRPs* entail maximum vehicle capacity. The task of delivering goods to customers is an important aspect of the *VRPs* and often a cost factor for unloading the demanded goods at the customer location is part of the problem definition. Typically the fleet of vehicles is fairly big compared to, for example, the *Job Shop Scheduling Problems (JSPs)*, where only a limited number of alternative serving entities is available. Vehicles are typically also considered equal to each other with regard to capacity and travel speed.

In our real-world application constraints like *skill requirements* and *preferred technician* limit the number of alternative serving vehicles respectively technicians. We also have to deal with a variety of different service vehicle types, each with a different capacity.

3 Heuristic Approaches to Water Scheduling

In this chapter we will explore different heuristic approaches to our problem. The first is a *Greedy Construction* approach, which creates a *feasible routing solution* by iteratively finding the next closest customer to any of the technicians and assigning it to this technicians route. The second approach is a *Decomposition Strategy* which aims at splitting the problem into two subproblems which can then be solved at lower cost. The first step is to find a technician-customer assignment without worrying about the sequence in which the customers are visited. The second step will then put the technician-customer assignments obtained in Step 1 into sequence to form proper round tours. Finally we will examine a post-optimization technique to further optimize the resulting round tours.

Feasibility Issues

These heuristic approaches do not guarantee to find an initial solution if one exists. The problem of guaranteeing to find such an initial solution is very complex and itself NP complete which can easily be seen:

The decision version of the *TSP* is known to be NP complete, which is equivalent to our *WSRP* formulation when only one technician is regarded and all side constraints except the round tour constraint are dropped.

During our tests, however, finding an initial solution was not a problem and we expect this heuristic to perform reasonably well on any given set of natural data as we will see in Chapter 4 on page 30.

3.1 Greedy Construction

The *Greedy Construction* algorithm is probably the simplest algorithm we examine in this paper. The approach is straight-forward: all technicians are located in the depot at the beginning. Based on the current location of all technicians the algorithm determines the pair of a technician t' and an unvisited customer c' that are closest to each other and for which t' has all required skills to service c'. It then assigns customer c' to the route of t' and sets the current location of t' to be at the location of c'. This is repeated until all customers are served. This approach is presented in Algorithm 1 on the next page.

The definition of the "$\leq$" relation on skill sets is given in Definition 4 on page 22.

Algorithm 1: Greedy Construction Algorithm

```
1 foreach c ∈ C do
2     isVisited(c) := false
3 foreach t ∈ T do
4     currentLocation(t) := Depot
5 while there are unvisited customers do
      (t', c') = arg min_{c ∈ unvisited C, t ∈ T | Skillset(c) ≤ Skillset(t),
6                 workload(t) + dist(currentLocation(t),c) + a_c ≤ W_t}  {dist(currentLocation(t),c)}
7     append c' to tour of t'
8     currentLocation(t') := c'
9     isVisited(c') := true
```

A typical effect of the *Greedy Construction* approach are crossing edges in the resulting round tour as we can see in Figure 3.1 on page 24. The *Greedy Construction* algorithm is not capable of preventing this because it does not take the overall tour length into account but only looks for the next closest customer that can be reached from the current location of any of the vehicles. Depending on the geographic distribution of the customer locations, this can have a smaller (yellow tour in the bottom left) or a stronger effect (green tour in the top right) on the resulting round tour. A subsequent optimization step is therefore advised, for example by using the *2-Opt* algorithm or our *Greedy 2-Opt* approach which we will describe in Section 3.3 on page 27.

3.2 Decomposition Strategy

A different approach is the so-called *Decomposition Strategy*. The first step of a decomposition is to generate smaller sub-problems of the original problem that can then be further optimized at lower cost. The first step of the *Decomposition Strategy* determines the technician-customer assignments without worrying about the sequence in which the customers will be visited by the technicians. In our case, we explore two alternatives, a *k-means* approach and a *Greedy Assignment* approach. The second step is then to find optimal sequences in which each of the technicians visit their assigned customers. Depending on how complex these remaining sub-problems are, they can either be solved to optimality using an ILP, or a different set of heuristic algorithms can be applied, such as the *Greedy Roundtour Optimization* algorithm. An example for an iterative improvement during the first decomposition step is depicted in Figure 3.2 on page 25.

3.2.1 Assignment Phase

As mentioned above, the first phase of the *Decomposition Strategy* is to create the assignments between technicians and customers. The sequence in which the customers will be visited is not of interest at this point. In this paper we explore two approaches; the first is similar to a *k-means* algorithm which applies an adapted *Voronoi Partitioning* to the given set of customers in order to partition them into smaller sets of neighboring customers that can each be serviced by a single technician. The second is a *Greedy* approach, which assigns the next closest customer to the next closest technician. We designed the algorithms so that our side-constraints are always obeyed.

3.2.1.1 k-means

Our *k-means* approach consists of three steps. In the first step we calculate a random partitioning of our customer base. Each partition will later represent a set of customers that can be serviced by a single technician. In the second step, we create the technician-customer assignments based on the partitioning we obtained in the previous step while taking our constraints for maximum working cost and customer skill requirements come into consideration. In a third and final step, the partitioning is iteratively improved in order to obtain a better fitting schedule.

Partitioning of Customer Base

The initial step in our *k-means* approach is to randomly choose so-called *Voronoi Centers*. A *Voronoi Center* center(V_k) can be any customer location that is not the depot, and it forms the center point of a so-called *Voronoi Cell* $V_k \in \Xi$. A *Voronoi Cell* in turn is distinctly associated with a technician. All other customer locations that are neither the depot nor a *Voronoi Center* are associated to the next closest *Voronoi Cell*. They will later form a partition of customers, that can all be visited by a single technician. While choosing the *Voronoi Centers* as well as during the assignment of customers to a *Voronoi Cell*, all side constraints are obeyed. This has the effect that we slightly deviate from the general definition of a *Voronoi Cell*. Customers, which for example require special skills, may therefore in fact not be associated to the closest *Voronoi Cell* available but to the closest one that satisfies all constraints. Another reason for assigning a customer to a more distant *Voronoi Cell* could be that the technician who his assigned to the cells center would exceed his maximum working cost constraint if he was also going to service this customer. "Closeness" in this context means our cost function dist(center(V_k), c_i) takes a comparably small value. We assume that there is no customer that cannot be served by any of the technicians due to higher skill requirements than the highest available technician skill level.

Creation of Technician-Customer Assignments

Following the initial partitioning into *Voronoi Centers*, we perform a first full assignment of all available customers to their respective *Voronoi Cells* to gauge the quality of our

partitioning. In order to prevent deadlock situations where highly skilled technicians get filled up with low-skill service requests, which would impose the risk that they don't have any work shift capacity left when the high-skill service requests are being assigned, we sort the service requests in descending order according to their skill requirements.

Because we have multiple categories of skills where an inter-category order is not intuitively given, we proceed as follows:

We assume that a manual sorting for the skill categories $S = \{1, \ldots, \Lambda\}$ according to their "importance" is given such that the skill categories $1, \ldots, \Lambda$ are sorted in descending order of their importance.

A "more important" skill category overrides possibly present skill levels of "less important" skill categories when sorting according to skills. Assuming a customer c_1 has a high skill requirement level in a less important category, while customer c_2 has a low skill requirement level in a more important category, the skill requirements of customer c_2 are still considered higher than the skill requirements of customer c_1. The same applies to the skill levels of technicians.

Definition 4 (Order on skill sets) Given two skill sets

$$\text{Skillset}(c_1) = \{\text{req}_1(c_1), \ldots, \text{req}_\Lambda(c_1)\} \text{ and } \text{Skillset}(c_2) = \{\text{req}_1(c_2), \ldots, \text{req}_\Lambda(c_2)\}$$

we define

$$\text{Skillset}(c_1) \leq \text{Skillset}(c_2) :\Leftrightarrow \begin{cases} \text{req}_i(c_1) = \text{req}_i(c_2) \quad \forall\, i = 1, \ldots, \Lambda \\ \text{or there is a } l \in 1, \ldots, \Lambda \text{ for which} \\ \text{req}_l(c_1) < \text{req}_l(c_2) \text{ and } \text{req}_i(c_1) = \text{req}_i(c_2)\ \forall\, i < l \end{cases}$$

Following this logic of sorting customers in descending order according to skill levels, we calculate the assignments of customers to *Voronoi Centers* and thus technicians one by one. We iterate through all customers, starting with those with high skill requirements and search for the closest *Voronoi Center*. We then check whether the technician belonging to this center possesses all skills that are required to service this customer and ensure that he would not exceed his maximum working cost. If both of these constraints are fulfilled, the customer is assigned to this technician. If any of these constraints is not fulfilled, we continue searching for the next closest *Voronoi Center* until we find a matching pair. We do so until all customers are assigned to a technician.

Improving the Partitioning

This first "guess" of *Voronoi Centers* is typically suboptimal and the partitioning is improved in subsequent steps. These improvements are accomplished iteratively, improving

each *Voronoi Cell* at a time. In each iteration step we scan through all customer locations belonging to a *Voronoi Cell* and determine, whether the cell becomes more "harmonic" when a different customer location is made the *Voronoi Center* for this cell. With "harmonic" we refer to the sum of all distances from the *Voronoi Center* to all members of its cell being as small as possible (the "disharmony" is small), which is the case when the *Voronoi Center* lies in the middle of a cell. More formally:

Given a *Voronoi Cell* V_k with *Voronoi Center* center(V_k) the "disharmony" of V_k is given as

$$\text{disharmony}(V_k) := \sum_{c \in V_k} \text{dist}(\text{center}(V_k), c) \tag{3.2.1}$$

If the "disharmony" of a center is smaller with a new choice for the *Voronoi Center* than it is with the current center, the new candidate for the center is stored. In each iteration step the best candidate for the new *Voronoi Center* is determined before the center is actually changed.

After the new centers have been determined for all cells, the assignments of customers to *Voronoi Centers* are recalculated as we have already described above. This has the consequence that the "harmony" in the cells changes again. Customers that were previously assigned to a cell may now be assigned to a neighboring cell, because the *Voronoi Centers* moved and a different center is now closer to the customer. We therefore have to repeat our improvement step until the *Voronoi Centers* have found their final position and the technician-customer assignments no longer change. We can visually interpret this as the *Voronoi Centers* moving away from each other, forming "hot spots" of customer service demand that are satisfied by one technician each. The result is (hopefully) a near-optimal assignment of technicians to customers, such that all customers are assigned to their closest *Voronoi Cell* and neighboring cells don't overlap. Algorithm 2 on the following page presents this proceeding and Figure 3.2 on page 25 displays an initial *Voronoi Partitioning* together with three improvement steps.

To form a proper technician schedule out of this partitioning, we have to apply additional algorithms which put the customer visits in sequence. These algorithms are described in Section 3.2.2 on page 26.

3.2.1.2 Greedy Assignment

The *Greedy Assignment* approach is almost identical to the *Greedy Construction* algorithm described in Section 3.1 on page 19. The only difference is how the results of the algorithm are interpreted and further processed. In the *Greedy Assignment* approach, we discard the sequence which was obtained as a side-product of the algorithm and only care about the technician-customer assignments.

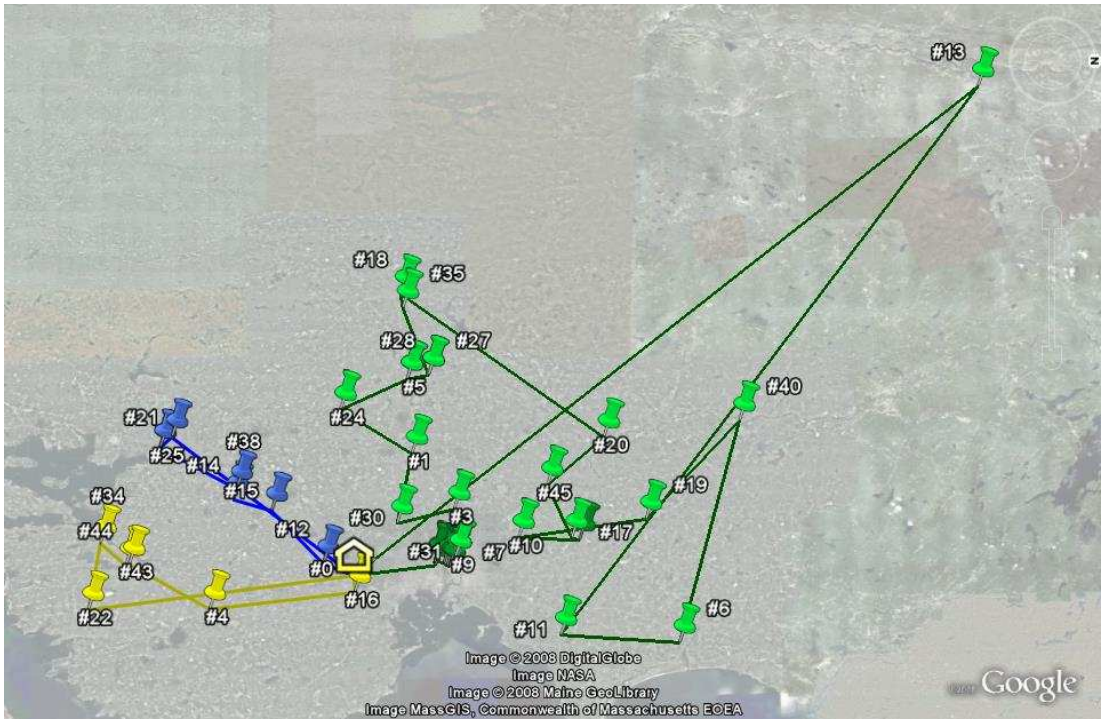
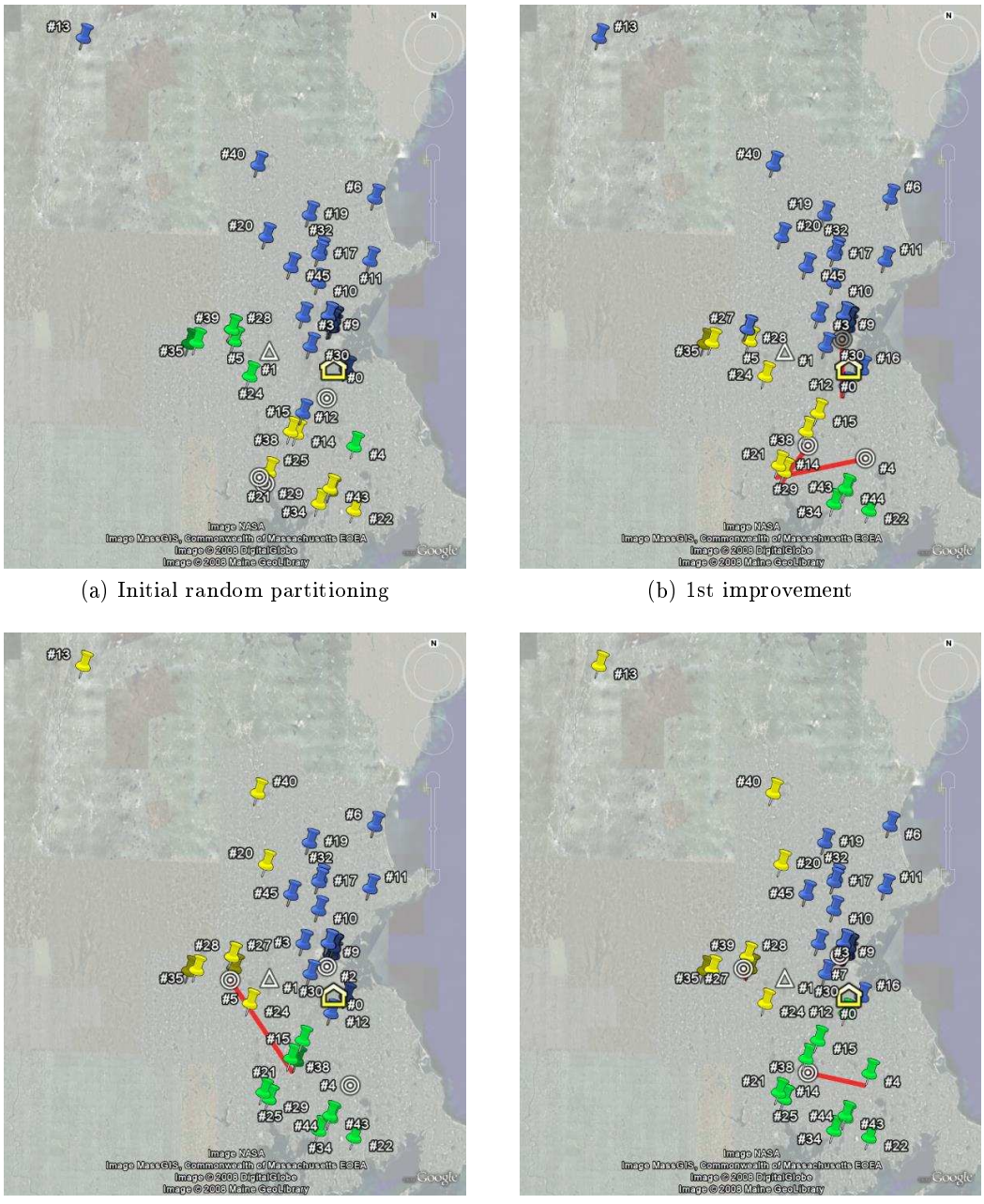


Figure 3.1: Crossing edges in the *Greedy* approach

Algorithm 2: Improve Voronoi Partitioning

repeat
  improved := false
  foreach *Voronoi Cell* $V = \{c_1, \ldots, c_k\}$ **do**
    $currentDisharmony(V) := \sum_{c_j \in V} \text{dist}(\text{center}(V), c_j)$
    $minDisharmony(V) := \min_{c_i \in V} \sum_{c_j \in V} \text{dist}(c_i, c_j)$
    $c'_i := \arg\min_{c_i \in V} \{\sum_{c_j \in V} \text{dist}(c_i, c_j)\}$
    if $currentDisharmony(V) > minDisharmony(V)$ **then**
      $\text{center}(V) := c_{i'}$
      improved := true
  recalculate Voronoi Cells and technician-customer assignments
until *improved* = *false*



(a) Initial random partitioning

(b) 1st improvement

(c) 2nd improvement

(d) 3rd improvement

Figure 3.2: Iterative improvements of *Voronoi Partitioning*. The bullseye symbols represent the *Voronoi Centers* and the red lines show how they move between the individual improvement steps.

3.2.2 Tour Construction Phase

In this section we examine different ways of creating routes based on technician-customer assignments we have obtained using one of the heuristic approaches in Section 3.2.1 on page 21. In other words, we already know which technician will provide service to which customer and now have to figure out in which sequence the customers shall be serviced by solving the remaining sub-problems.

We examine two alternatives: the first is using *ILPs* which provide an optimal solution to each of the remaining sub-problems. The second approach is a recap of our *Greedy Construction* approach which always drives to the next customer location that is closest to the current location of a technician.

3.2.2.1 ILP

Our first approach is to optimally solve the remaining problems using an *ILP*. The remaining problems are similar to the *TSP* in that they only take a single technician into account at a time. The *ILPs* presented in Sections 2.3.1 thru 2.3.3 on page 16 can all be utilized, however because we already have determined which technicians can service which customers, we no longer need the constraints for maximum working cost and technician skills. Constraints (2.3.4) and (2.3.8) can therefore be taken out. Per technician an individual instance of the *ILP* will be computed and the *ILPs* will automatically resize to a single technician because M will be set to 1.

3.2.2.2 Greedy Roundtour Optimization

Algorithm 3: Greedy Roundtour Optimization Algorithm

input : for each technician t a set of technician-customer assignments $currentRoute(t)$

output: for each technician t a routing solution $newRoute(t)$ with sequenced round tours

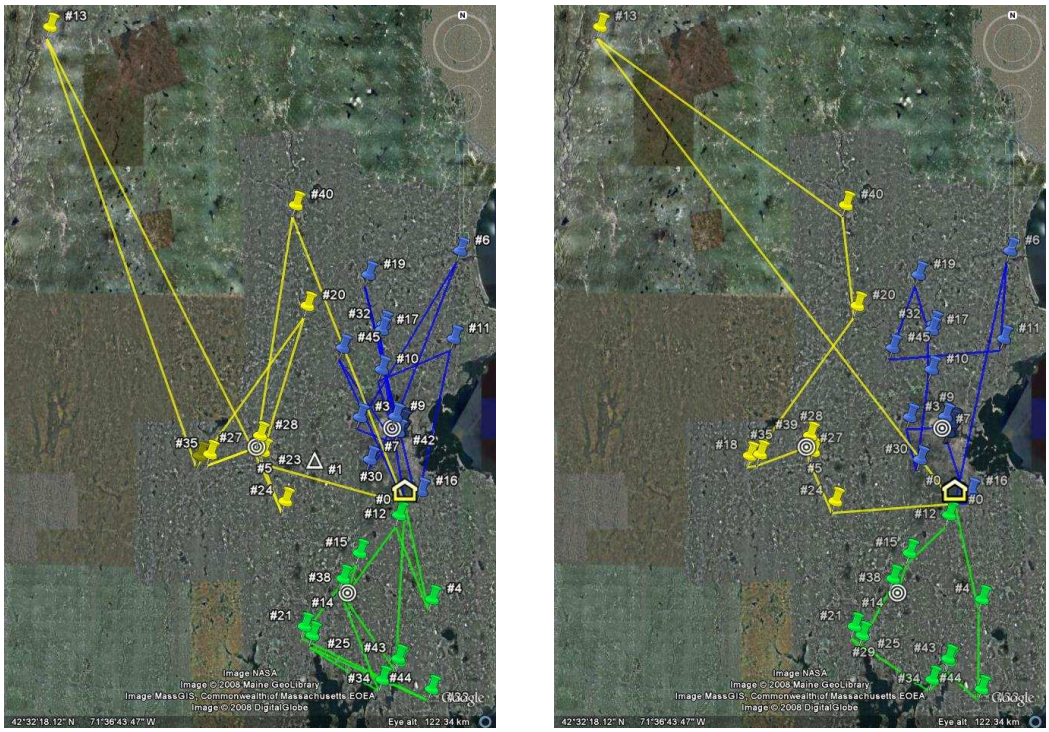
1 **foreach** $t \in T$ **do**
2 $currentLocation(t)$:= Depot
3 **while** *there are customers in* $currentRoute(t)$ **do**
4 $c' = \underset{c \in currentRoute(t)}{\arg\min} \{\text{dist}(currentLocation(t), c)\}$
5 append c' to $newRoute(t)$
6 $currentLocation(t) := c'$
7 remove c' from $currentRoute(t)$

The *Greedy Roundtour Optimization* algorithm takes a set of technician-customer assignments with unsequenced round tours as input and sequences the individual stops technician by technician. For each technician, it puts the assigned customers into sequence by starting at the current location of the technician, which is the depot in the

beginning, and iterating through the list of his assigned customers, adding the next closest customer as the next stop, until all customers in the list are put in sequence. This proceeding is given in Algorithm 3 on the facing page.

The result is a routing with sequenced technician-customer assignments that can be further optimized using one of the approaches presented in Section 3.3.

In Figure 3.3 we see the effect of the *Greedy Roundtour Optimization* algorithm on a partitioning that was obtained using the *k-means* approach described in Section 3.2.1.1 on page 21. We can clearly see several crossing edges in the tours, which the *Greedy Roundtour Optimization* algorithm is not capable of preventing, because it simply picks the next closest customer location from the current technician position and does not take the overall round tour cost into account.



(a) Partitioning obtained by k-means

(b) After Greedy Roundtour Optimization

Figure 3.3: Effect of *Greedy Roundtour Optimization* on a partitioning obtained by the *k-means* decomposition approach

3.3 Post Optimization

We observe two slightly differing formulations of the *2-Opt* algorithm for the Post Optimization phase. Besides the original version, we also present a *Greedy 2-Opt* approach.

3.3.1 2-Opt

The *2-Opt* algorithm is a simple local search algorithm which eliminates crossing edges in a single tour by picking two edges and reconnecting the four adjacent vertices, in our case these are the customer locations. We chose to use the following variant:

Given a round tour $c_0, c_1, c_2, \ldots, c_k, c_0$ we check for each combination of customers c_i, c_j, $i < j$, $i, j \in \{1, \ldots, k-1\}$, whether driving from c_{i-1} to c_j and from c_i to c_{j+1} is actually better than driving in the original sequence, which is c_{i-1} to c_i and from c_j to c_{j+1}. If this is the case, we store the two indices i and j together with the improvement that this modification would yield and continue searching for even better improvements among the remaining combinations of c_i and c_j. After all possible combinations have been checked, the sequence is modified according to the best alternative found. The sequence before (3.3.1) and after improvement (3.3.2) are given below:

$$c_0, c_1, c_2, \ldots, c_{i-1}, \quad c_i, c_{i+1}, c_{i+2}, \ldots, c_{j-1}, c_j, \quad c_{j+1}, \ldots, c_k, c_0 \tag{3.3.1}$$

$$c_0, c_1, c_2, \ldots, c_{i-1}, \quad c_j, c_{j-1}, c_{j-2}, \ldots, c_{i+1}, c_i, \quad c_{j+1}, c_{j+2}, \ldots, c_k, c_0 \tag{3.3.2}$$

If we have improved, we follow this same proceeding again until we don't find any further candidates for optimization.

Figure 3.4a on the facing page illustrates this by reversing the travel direction between vertices 2 and 7, which represent the invariant way points c_{i-1} and c_{j+1}. Figures 3.4b and 3.4c on the next page show the improvement of *2-Opt* over a technician-customer assignment that was obtained using the *k-means* approach presented in Section 3.2.1.1 on page 21, followed by a *Greedy Roundtour Optimization* as presented in Section 3.2.2.2 on page 26.

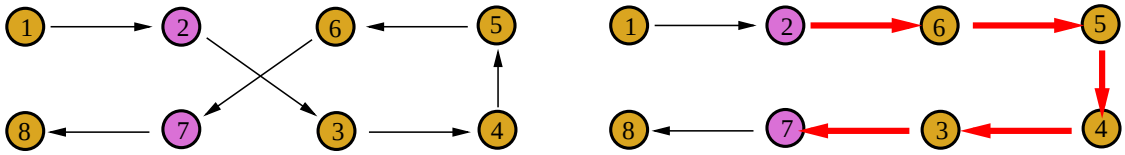
3.3.2 Greedy 2-Opt

The *Greedy 2-Opt* algorithm is an algorithm inspired by the original *2-Opt*, with the difference that it performs the modifications of the travel sequence immediately when it finds them and does not look for the best possible modification in the current iteration first. This proceeding is given in Algorithm 4 on the facing page.

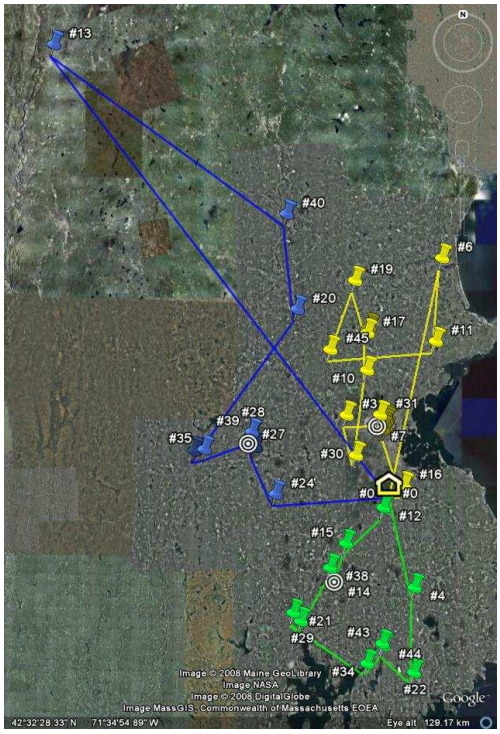
Algorithm 4: Greedy 2-Opt Algorithm

input : A single round tour of customer visits
output: An optimized round tour of customer visits

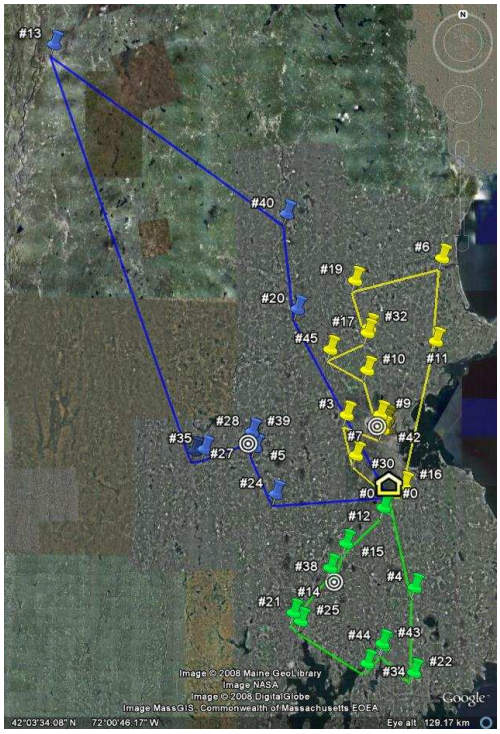
```
1 repeat
2     improved := false
3     foreach Customer c_i ∈ C do
4         foreach Customer c_j ∈ C,    j > i do
5             if dist(c_{i-1}, c_i) + dist(c_j, c_{j+1}) − dist(c_{i-1}, c_j) − dist(c_i, c_{j+1}) > 0 then
6                 reorder to drive from c_{i-1} to c_j and from c_i to c_{j+1}
7                 improved := true
8 until improved = false
```



(a) Schematic 2-Opt. Reversing travel direction between vertices 2 and 7.



(b) Before 2-Opt



(c) After 2-Opt

Figure 3.4: 2-Opt algorithm removing crossing edges in tours.

4 Experiments

In this chapter we outline our experiments and present the experimental results we obtained throughout the course of our work. We start with an introductory section on naming conventions and units of measure that we will use throughout this chapter. In the second section we provide a comparison of the strengths and weaknesses of our different *ILP* formulations and provide motivations for their existence. In Section 4.3 we then present some details on our instances of test data. Besides a description about where the data stems from and how it was obtained, we also outline some characteristics of the individual instances which will allow the inclined reader to get a feeling for the underlying data. In the following section we present the experimental setups together with some details on the lead through of our experiments. We state present hard- and software involved in the tests, outline the key questions for our experiments and provide an overview of the different combinations of scheduling algorithms we used. In Section 4.5 we then present our experimental results, starting with the runtime analysis of the different *ILP* formulations for different sizes of input data, followed by a runtime analysis of our heuristic scheduling approaches. Finally, we compare the qualities of the schedules obtained by the different approaches in detail.

4.1 Naming Conventions and Units Of Measure

Let's start with a couple naming conventions and some guidance on how to interpret the units of measure used in our data analyses.

In order to save real estate in the tables and figures that will appear in the upcoming sections, we will use short forms instead of the full names of algorithms and algorithm combinations. These short forms and their related full names are given in Table 4.1 on the next page.

Costs and Hours

Furthermore, in the original real-world problem, we have two different types of cost that we need to distinguish. First, we have the on-site working cost which is measured in hours and indicates, how long a technician has to remain at the customers site in order to fulfill his duty. Second, we have the travel cost which is typically measured in miles and indicates the distance between two locations. In order to optimize according to

Short Form	**Algorithm**		**Post-Opt**
S	Real-World Schedule as performed in the field		
I1	*WSRP* ILP		
I2	*WSCRP* ILP		
I3	*WSMRP* ILP		
GO	*Greedy Construction*		Greedy 2-Opt
GR	*Greedy Construction*		2-Opt
Decomp. ↓	**Assignment Phase**	**Tour Construction Phase**	**Post-Opt**
G1	*Greedy Assignment*	*WSRP* ILP	
G2	*Greedy Assignment*	*WSCRP* ILP	
G3	*Greedy Assignment*	*WSMRP* ILP	
S1	Real-World assignments	*WSRP* ILP	
S2	Real-World assignments	*WSCRP* ILP	
S3	Real-World assignments	*WSMRP* ILP	
SGO	Real-World assignments	*Greedy Roundtour Opt.*	Greedy 2-Opt
SGR	Real-World assignments	*Greedy Roundtour Opt.*	2-Opt
V1	*k-means*	*WSRP* ILP	
V2	*k-means*	*WSCRP* ILP	
V3	*k-means*	*WSMRP* ILP	
VGO	*k-means*	*Greedy Roundtour Opt.*	Greedy 2-Opt
VGR	*k-means*	*Greedy Roundtour Opt.*	2-Opt

Table 4.1: Naming Conventions for Chapter 4

both on-site working cost and travel cost at the same time, we need to combine them into one measure, which we call "overall cost". This is the sum of on-site working cost and a fraction of travel cost that estimates the time that is required to travel the given distance. We chose to estimate the travel time by sampling some of the distances in our test instances, determining the actual drive times using popular routing tools like Google Maps, and thus obtaining an approximate conversion factor for our test instances. The most intuitive unit of measure for the overall cost is hours, although we have to take into consideration that the travel cost part is really just a good guess of the time required to drive from A to B. The reason why we point this out is that for some technicians the total work day may have 30 "hours" and more, which obviously does not mean they were servicing customers for 30 hours straight. Instead, it means they had to drive long ways on the highway and therefore actually required less time than indicated when assuming travel cost to be measured hours.

The reason why we chose this proceeding is three-fold. On the one hand the requirement from the practice was to reduce mileage rather than travel time. For this reason using travel time as the cost factor for travel cost was not an option and thus the distance between two locations was the cost factor of choice in the objective functions.

On the other hand we had to model the maximum working cost constraint and somehow marry up on-site working cost with travel cost. We therefore obtained a constant factor to convert the actual mileage into the measure of hours. An alternative approach, which is worth taking into consideration when using our approaches for a real-world implementation, would be to use two separate cost functions, one for travel mileage in the objective function and one for travel time in the maximum working cost constraint. For the conceptional aspect of our work, however, the choice of the cost functions does not have an influence on correctness or expressiveness.

Finally, we had the requirement to service all customers that were selected for optimization, which is a typical requirement in the practice, where over hours are exceptional cost and need to be prevented, but customer satisfaction still has a higher priority than cost reduction.

In the following data analysis we will concentrate on the overall cost as this is used in the objective functions and provides an intuitive way of comparing the schedules of two technicians.

Instance Names

The Instance Names like *BM5* or *FA7* are derived from the company internal code of the branch in the first character, followed by "M" for the month of March, respectively "A" for the month of April. The number in the end denotes the day of month. For *BM5*, this means it's Branch "B" on March 5^{th}, whereas *FA7* refers to Branch "F" on April 7^{th}.

In grouped Figures, the instance names are typically ordered in sequence with the associated date, while branch "B" typically precedes branch "F". Where it is applicable, the instances may be grouped according to matching scales rather than branch or month to

allow a maximum level of detail for each display group. Scales may therefore vary from group to group.

4.2 Strengths and Weaknesses of the *ILP* Formulations

We will now foreclose some insights about the strengths and weaknesses of the different *ILP* formulations that we observed during the experimental phase of our work. These insights are the motivations for the existence of the three *ILP* formulations and are important in order to understand their differences. We start with a comparison of the *WSMRP* with the *WSRP* and conclude with a comparison of the *WSCRP* with both the *WSMRP* and the *WSRP*.

4.2.1 Water Scheduling Makespan Routing Problem vs. WSRP

The results obtained using the plain *WSRP* formulation reveal one significant shortcoming: the different technicians are not necessarily utilized to a comparable extent, if the maximum working cost W_k is not set close to the actual resulting working cost of the technicians. When the gap is large enough, or W_k has to be set high to allow visiting a remote customer location that alone would exceed the otherwise typically required working cost, few technicians are typically assigned very long tours serving many customers, while others get very short tours.

The reason for this behavior of the *WSRP* lies in the objective function of the formulation. It minimizes the overall cost of a schedule and does not include any constraint to balance technician utilization. In our application, however, an even distribution of the workload among the different technicians is desired and the value for W_k should not have to be adjusted for every run.

With the makespan approach, we introduce a new objective function and a set of additional constraints to evenly distribute the technician workloads regardless of the value of W_k. The effect of this approach can be seen in Figure 4.1 on page 35 where 4.1a displays the solution of the plain *WSRP* with a high value of W_k and 4.1c shows the solution of a formulation minimizing the makespan over all travel costs with the same value of W_k. In Figure 4.1a the red route is significantly longer than all the others while in 4.1c we see a more realistic schedule. With a value of W_k set closer to the actually required working costs, also the *WSRP* approach presents a good looking solution as shown in Figure 4.1b on page 35.

We also introduce on-site working cost into the objective function of our *WSMRP* formulation, which reflects the time the technician requires to perform the job at the customer location.

The makespan $\overline{Z}$ of a schedule is defined as the maximum over all technicians of the sum of travel cost to all assigned customers plus on-site working cost required to provide service each of the customers.

Even if the on-site working cost is equal for all customers, its absence influences the workload balancing in such a way that travel cost-wise shorter routes typically receive more stops while travel cost-wise longer routes receive less. Adding the on-site working cost is an effective counter measure. The resulting schedule is more balanced and thus more realistic than without considering on-site working cost as we can see in Figure 4.1 on the facing page. The red route in 4.1c has 6 stops while the blue route only has one stop. Taking on-site cost into account, in 4.1d the red route has 4 stops while the blue route has 3 stops. The green route remains unchanged in both situations.

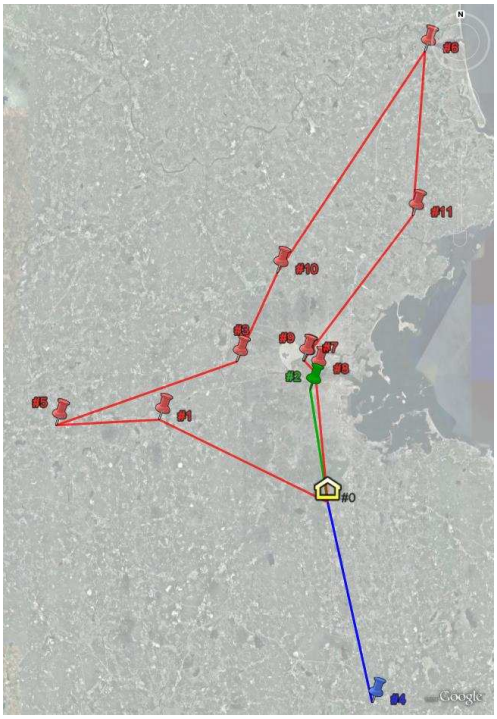
4.2.2 Water Scheduling Combined Routing Problem vs. WSMRP & WSRP

The *WSMRP* has effectively solved our problem around balancing the workload among the technicians. However, using only the makespan in the objective function can develop an unwanted side-effect if the different tours vary strongly in length due to other constraints. The *WSMRP* does not enforce an optimal sequencing of service visits for all technician routes, but only for the one making up the makespan. A scenario where this becomes a problem could be a remote customer location, which alone fills up the entire day of a technician. If the other technicians are not fully utilized, they can make detours and still return to the branch before the poor technician who is still on the road. Another common scenario is when a high skilled technician calls in sick and his equally skilled colleagues have to fill in for him. They will end up with longer days than their low skilled colleagues, who could well have a second cup of coffee in the morning without risking to return last.

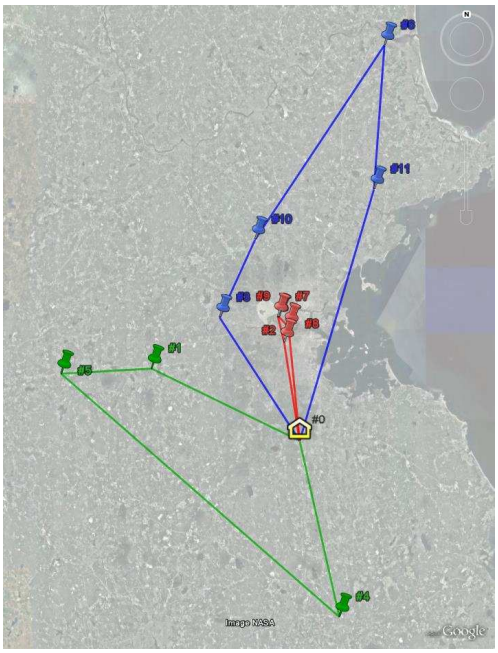
In the *WSRP* formulation, the optimal sequencing for each route is obtained by the objective function, which takes every single mile any technician drives into account. The makespan approach, however, only cares about the longest round tour in the schedule. All other tours don't count for the objective function. By including the travel cost of all technician routes in the objective function, we can effectively enforce an optimal sequencing in every single technician route. The weighting factor α allows an adjustment of the influence that each scheduling aspect has on the solution, i.e. we can gradually shift from *WSRP* to *WSCRP* to *WSMRP*. For the *WSMRP* test runs we have chosen a value of $\alpha = 0.5$.

4.3 Instances of Test Data

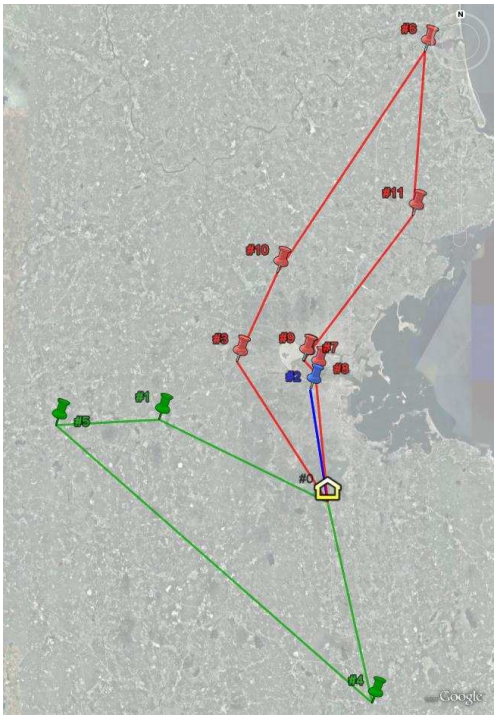
The core of our experimental study, besides the presented scheduling approaches, are the experiments on test instances with real-world data. In this section we further describe the origin of our test instances and explain how the data was obtained. We then characterize the different instances to get a better impression of the underlying data.



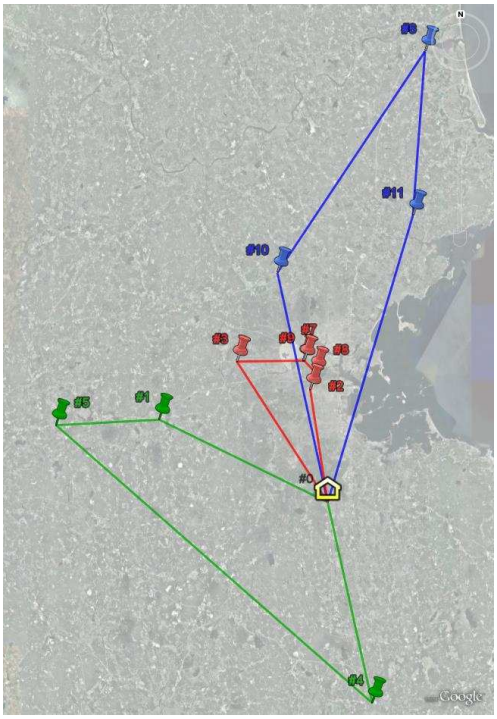
(a) *WSRP* solution with high W_k



(b) *WSRP* solution with minimal W_k



(c) *WSMRP* solution without on-site cost



(d) *WSMRP* solution with on-site cost

Figure 4.1: Effect of makespan and on-site cost on workload balancing in the *Water Scheduling Makespan Routing Problem*. All on-site costs are equal.

4.3.1 Origin and Obtaining of Test Instance Data

The test instances that were used in our experiments stem from a large water purification enterprise with operations in the entire US and Canada. The scheduling practices vary significantly among the over 90 branches due to differing clientele, different kinds of predominant service requests and also different levels of experience and knowledge of both the technician workforce as well as the service supervisors about the vicinity of their branches and the locations of their customers. While some experienced technicians know the rat runs to avoid traffic jams and other obstacles, especially younger and less experienced technicians at times get stuck in narrow one-way streets where they can barely maneuver with their big trucks.

The instances were obtained in such a way that a technicians work day could be fully reconstructed and compared to the solutions provided by our scheduling algorithms, including the exact sequence of travel throughout the day and the time it took them to get from A to B. Side constraints like technician skills, customer skill requirements and maximum working cost were also obtained and taken into account.

Obtaining the Test Instance Data

Some of the information could be easily obtained from the company's *Enterprise Resource Planning* (*ERP*) system, other data had to be manually acquired. The data from the *ERP* system included address information of the customers to generate GeoCodes in order to perform the routing and calculate distances between the individual customers. The *ERP* system also provided basic information about the tasks that needed to be performed at the customer location.

Because it was not possible to readily obtain the skill requirements for specific customers, but only for specific technicians, we assigned skill requirements to every customer visit based on the skills of the technician that had serviced this customer in reality. We acknowledge the fact that these skill requirements are an upper bound to the actual skill requirements of the specific customer in the sense that the customer could actually require lower skills than provided by the technician who actually served him. This implicates that the resulting schedule qualities are an upper bound to the schedule qualities that could be obtained with actual skill requirements. In other words, there may be even better schedules once skill requirements are fully maintained in the company's feeding system, e.g. the *ERP* system.

It was at first also not possible to reconstruct the exact sequence in which customers were served in a given day purely by using data from the *ERP* system. The information about the date of service and the technician-customer assignment were frequently outdated because service requests were kept open in the system to schedule follow-up work or there had been errors during data entry - which were either caused by inadvertence of the entering party in the back-office or by illegible scribblings of the service technicians. Due to a paper based process which is at present being replaced by a paperless system company wide, service orders had to be filled out by hand and then keyed into the *ERP*

system by a different person in the back-office.

In order to obtain a proper reconstruction of each technicians day, we obtained copies of all *Service Activity Reports (SARs)* for two full months in two different branches - which makes a total of four months worth of *SARs*. The *SARs* are hand written reports that indicate the exact sequence and time of service that each technician is required to fill out during his work day. In order to prevent us from making the same mistakes again that lead to the problematic data in the *ERP* system in the first place, we manually entered 16 full days of service, distributed across the four months of *SAR* data, a total of 583 technician-customer assignments, plus the sequence in which the services were performed, the travel time and mileage that were actually driven, as well as the maximum working cost for each service visit, which makes 3,498 hand-keyed data points. Following the data entry we compared all records against the records in the *ERP* system and manually resolved all conflicting records. We also used the information regarding actual drive time and mileage for validity checks. All this information together forms our 16 test instances.

For presentation in this paper, the data points were obfuscated in such a way that it is not possible to trace back individual customers, while the explanatory power and correctness of the tests and results was conserved.

4.3.2 Characteristics of the Test Instances

In order to allow a better understanding of the underlying data, we will now describe some characteristics of our test instances. An explanation of the instance names can be found in Section 4.1 on page 30.

Our set of test instances *FMx* comes from one of the larger branches on the west coast of the US. The maximum number of technicians in this test instance is 10; the total number of technicians in this branch is significantly higher though, but only a part of the technicians are working in parallel on any given day and some of the technicians are purely working on long-term installations, i.e. they don't participate in the daily scheduled service. Table 4.2 on the following page displays the number of technicians and number of customers in a given test instance along with how many technicians possess a certain maximum skill level, respectively how many customers require a certain minimum skill level. Each of these test instances represent a full day within the same month at this branch. Every technician has a minimum skill level of REP:1 and every customer requires a minimum skill level of REP:1. Per definition of our skill levels, a technician with a higher skill level also possesses all lower skill levels of the same category. Additionally, for our test instances, a TECH possesses all REP levels and an FSE possesses all TECH and REP levels. For easier readability only the maximum skills of a technician are displayed in the tables, i.e. a technician with level FSE:1 will only be counted in the FSE:1 row and not in the TECH and REP rows.

Similarly, our set of test instances *FAx* stems from the same branch but a different month in the same year. Table 4.3 on the next page provides the details on this test instances.

Our third and fourth set of test instances, *BMx* and *BAx*, stem from a mid-sized branch

Instance:	**FM5**	**FM6**	**FM7**	**FM12**	**FM13**
Technicians:	8	7	7	10	8
Customers:	43	38	45	48	40
Skills ↓	# Technicians / # Customers				
Skill REP:1	1 / 4	1 / 2	1 / 4	1 / 2	1 / 1
Skill TECH:1	6 / 38	5 / 35	6 / 41	8 / 45	7 / 39
Skill TECH:2	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Skill TECH:3	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Skill FSE:1	1 / 1	1 / 1	0 / 0	1 / 1	0 / 0

Table 4.2: Real-World Instances *FMx*

Instance:	**FA2**	**FA3**	**FA4**	**FA7**	**FA8**
Technicians:	7	8	10	7	8
Customers:	35	47	39	27	50
Skills ↓	# Technicians / # Customers				
Skill REP:1	1 / 3	1 / 6	1 / 4	0 / 0	1 / 5
Skill TECH:1	5 / 30	6 / 38	6 / 31	6 / 24	6 / 42
Skill TECH:2	1 / 2	1 / 3	1 / 2	1 / 3	1 / 3
Skill TECH:3	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Skill FSE:1	0 / 0	0 / 0	2 / 2	0 / 0	0 / 0

Table 4.3: Real-World Instances *FAx*

on the east coast of the US with a maximum of 9 active technicians throughout our chosen test days. The details on these sets of test instances are stated in Table 4.4.

Instance:	***BM5***	***BM6***	***BM7***	***BM12***	***BA10***	***BA11***
Technicians:	7	7	8	7	9	7
Customers:	30	33	33	31	26	18
Skills ↓	# Technicians / # Customers					
Skill REP:1	1 / 2	1 / 3	1 / 2	0 / 0	1 / 2	1 / 2
Skill TECH:1	4 / 24	3 / 22	5 / 27	4 / 24	5 / 16	4 / 14
Skill TECH:2	1 / 1	1 / 3	0 / 0	1 / 3	0 / 0	0 / 0
Skill TECH:3	1 / 3	2 / 5	2 / 4	2 / 4	2 / 7	2 / 2
Skill FSE:1	0 / 0	0 / 0	0 / 0	0 / 0	1 / 1	0 / 0

Table 4.4: Real-World Instances *BMx* and *BAx*

4.4 Experimental Setups and Lead Through

In this section we present our experimental setups and describe in detail which configurations and combinations of the scheduling approaches that were introduced in Section 2.3 on page 14 and Chapter 3 on page 19 we used. We first give a brief description of the Soft- and Hardware involved, before we go into the details on our experimental configurations. We then present our key questions for the lead through of our experiments and the interpretation of our experimental data.

4.4.1 Hard- and Software Used for Calculations

All calculations were performed on 1 core of an Intel Xeon machine with 8 cores total at 2.66 GHz each and 32 Gigabytes of RAM running SuSE Linux 11.0. The *ILPs* were solved using ILOG CPLEX 11.1. The implementation of our heuristic algorithms was done in Java 5.0.

The calculations were partly performed in parallel, thus utilizing multiple cores of the same machine at the same time. The memory consumption of our runs was low enough, however, so that there were no noteworthy runtime impacts.

4.4.2 Experimental Configurations and Lead Through

We will now take a close look at the experimental configurations and combinations of scheduling approaches that were involved in our tests. In the following paragraphs we first present our key questions during lead through. Subsequent to this we describe the combinations of scheduling approaches that were examined during our experiments.

4.4.2.1 Key Questions and Lead Through

One important question during the course of this work was how "big" can an instance be in order to be solvable by the presented *ILP* formulations to optimality in a reasonable amount of time with a reasonable amount of computation hardware. A "reasonable amount of time" is clearly in an order of seconds up to possibly a couple minutes per entire branch if the solution shall be feasible for a real-world application. A "reasonable amount of computation hardware" is a bit harder to define. Let's say it is what a typical company can afford without having to tune the financial statements at the end of the year. A hardware setup similar to the one we used for our testing should better be sufficient in order to find solutions to our scheduling problems.

1. How big can an instance be to be solved with our *ILPs*?
2. What has a stronger influence on runtime - number customers or technicians?
3. How quick are our Heuristic Approaches?
4. How good are the resulting heuristic schedules compared to real-world data?
5. What shortcomings and drawbacks do they have?
6. How do our Decomposition Strategies behave and perform on different test instances?
7. Can *ILPs* be used to sequence single-technician tours in reasonable time?
8. How good does our *Greedy Roundtour Optimization* approach perform compared to the optimal *ILP* solutions?
9. What are the reasons for all of the above and how can things potentially be improved?

To answer question 1 "How big can an instance be?" we ran the *ILP* formulations presented in Section 2.3 on page 14 with varying sizes of subsets of our actual test instances. We started with 3 technicians and 10 customers and increased technicians and customers step by step until the runtime of our *ILPs* exceeded 8 hours. At this point the runs were canceled and a subsequent, even bigger configuration was started. It is clearly evident and perspicuous that both the number of customers and the number of technicians had a significant influence on the runtime of the *ILPs*. However, it was also noticeable that some bigger configurations solved significantly quicker than smaller ones. An explanation for this initially odd appearing behavior are the variations in customer skill requirements, technician skills, as well as the geographic distribution of the customers and maximum working cost constraints with every new configuration. While the problem set of a "smaller" configuration may originally be nicely partitioned, adding a single customer can spoil this partitioning and make even just the problem of finding a feasible, let alone an optimal schedule a brainteaser.

This leads us to question 2. It is evident, both the number of customers and the number of technicians influence runtime. But which parameter has a stronger influence? By analyzing the run times of all configurations, we will find our answer to both questions

in Section 4.5.1 on the following page, which also entails the answer to question number 3.

Questions number 4, 5 and 6 are answered in Section 4.5.2 on page 53 by comparing the schedules obtained by all different scheduling combinations and identifying strengths and weaknesses of each of the approaches.

For question 7 we will investigate the run time behavior of the *ILPs* in the single-technician case in Section 4.5.1 and question 8 is closer examined in Section 4.5.2.

Finally, we will draw our conclusions and outline potential further improvements, thus answering question 9 in Chapter 5 on page 61.

4.4.2.2 Combinations of Scheduling Approaches

In order to find conclusive answers to our key questions above, we have constructed several combinations of scheduling approaches that allow us to identify strengths and weaknesses of each of the partial approaches and trace peculiar behaviors back to their root cause.

An overview of all combinations is given in Table 4.1 on page 31.

Our first "combination" are the three *ILP* formulations by themselves. We then formed a combination consisting of the *Greedy Construction* algorithm followed by the *2-Opt* post optimization approach. This combination is deterministic because both parts are implemented as deterministic algorithms, which means they come to the same conclusion in every run. Our third approach is a *Decomposition Strategy* which consists of three phases. The first is the *Assignment Phase* in which technician-customer assignments are created without worrying about the exact sequence in which the services are performed. The second phase is called the *Tour Construction Phase* in which the service visits of each technician are put in sequence. The third and final phase is the *Post Optimization* which further improves the round tours obtained by our heuristic tour construction algorithm.

The *Assignment Phase* of our *Decomposition Strategy* consists of two alternative approaches. One is a *k-means* based approach, the other is a *Greedy Assignment* algorithm. While the *Greedy Assignment* is deterministic, the *k-means* approach is indeterministic and may therefore find a different solution in every run.

The second phase, the *Tour Construction*, also consists of two alternatives: solving the remaining single-technician routing problems using any one of our three *ILP* formulations, or running our *Greedy Roundtour Optimization* algorithm.

The third phase, *Post Optimization*, is only invoked following the *Greedy Roundtour Optimization* because the sequencing of service visits within each of the routes obtained by the *ILPs* is already optimal.

The alternatives for both the *Tour Construction* and *Post Optimization* phase are deterministic. The only indeterministic approaches are therefore the combinations containing the *k-means* approach.

4.5 Experimental Results

In this section we present our experimental results and draw first conclusions on different behaviors that we observed. We start with a presentation of the runtime analyses for the *ILP* runs as well as for our heuristic scheduling approaches. We then compare the quality of the scheduling solutions obtained by our different approaches.

4.5.1 Runtime Analysis

In the first part of our runtime analysis will see what parameters have an influence on the runtime behavior of our *ILPs* and how strong their respective effect is. We will also see for what input sizes the solution performance of our *ILP* approaches is no longer feasible.

In the second part we take a close look at the runtime behavior of our heuristic and combined scheduling approaches. We will again see which factors have an influence on solution times and will try to trace them back to properties in the design of the approaches as well as characteristics of the test instances.

Finally, we will find answers to some of our questions from Section 4.4.2.1 on page 40.

4.5.1.1 ILP Performance

The runtime of our 3 *ILP* formulations is clearly dependent on the number of customers and technicians in the test instances. While small instances with 3 technicians and 10 customers solved to optimality in less than a second, a moderately sized problem set of 7 technicians and 45 customers ran for almost three weeks - until it was finally canceled. We would like to better understand what factors are influencing the runtime of the *ILPs*.

Figure 4.2a on page 44 shows the relation between the number of customers and the runtime. It suggests a strong correlation between the two measures: the more customers we have, the harder it is to find a schedule. When looking at the relation between runtime and number of customers even closer in Figure 4.3 on page 45 by keeping the number of technicians constant in every frame and splitting the data additionally by test instance, we see a clear correlation in the example of instance *BM6* (Figure 4.3b). For the other test instances a correlation is not directly visible.

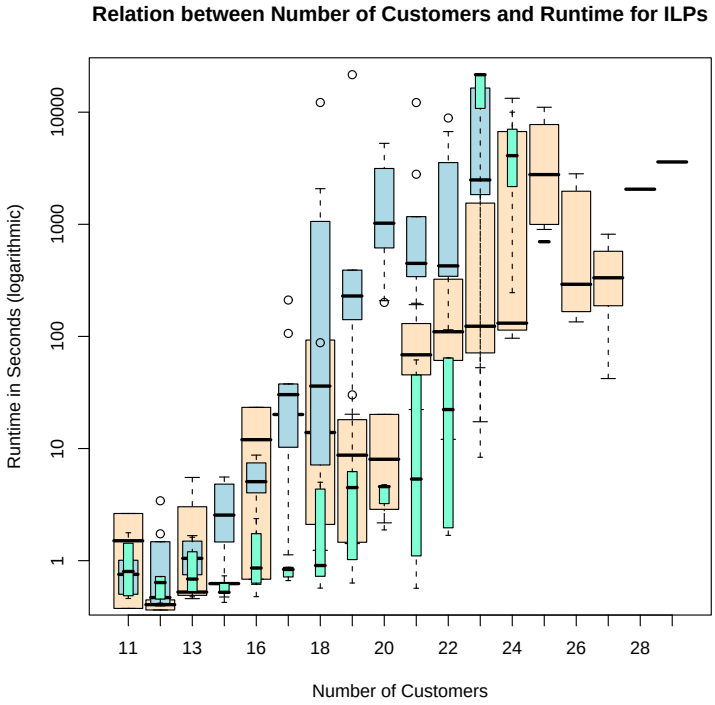
Figure 4.2b on page 44 puts the run time in relation to the number of available technicians. We would expect to see a similar sign of correlation as in case of the customers, but it tells a different story. It appears as if the number of technicians had a relaxing or at least invariant effect on the solution finding process. Well, to some extend this may be true and one could argue that if there are more technicians the algorithm has to choose from, the likelihood to find a matching technician-customer pair is higher than if there were only a hand full technicians in the pool.

Let's look at this relation a little closer. In Figure 4.2b we compared runtime and technicians. However, during the tests also the number of customers has been altered from run to run. All these runs with different numbers of customers are aggregated

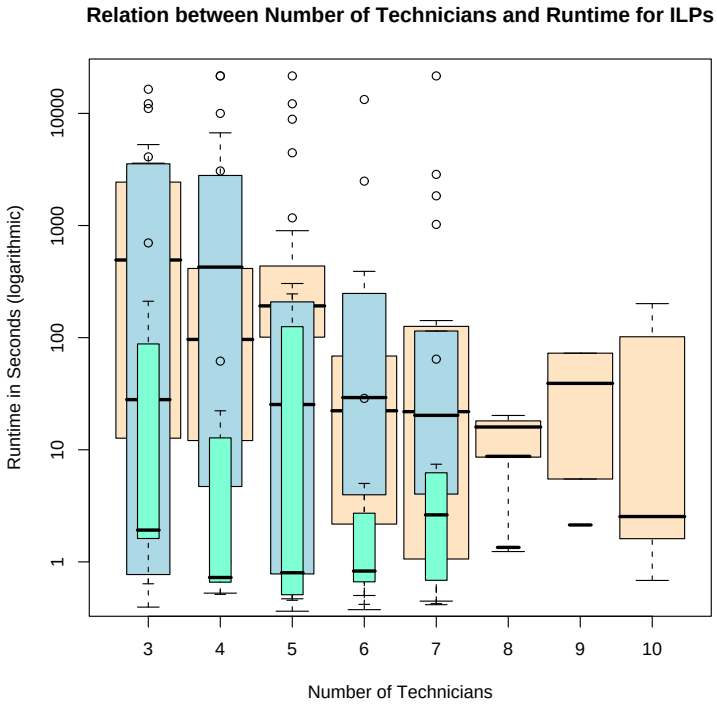
in Figure 4.2b, which blurs the view on the actual variable of interest and its potential influence on the runtime. In Figure 4.4 on page 46 we take our examination a level deeper and keep the number of customers constant in each frame. The number of customers is given in the frame captions. While varying the number of technicians, we see that the first assumption of a relaxing effect of a higher number of technicians can actually not be supported. One would expect to see the typical signs of linear correlation, an ascending or descending trend, similar to what we saw for the number of customers and their influence on the runtime in Figure 4.2a.

While we are certain that there is a noticeable effect associated to the number of technicians in a problem set for sufficiently large numbers of technicians, we believe that this effect is in our cases overlayed by the noise that is introduced by the different test instances and their response to our constraints. This theory is supported by Figure 4.5 on page 47 where we drill a level deeper into Figure 4.4 on page 46 and impose a further split on the level of the test instances. We see how there is no visible linear correlation between the number of technicians and the runtime for any of the test instances.

The same algorithms behave very differently depending on which instance they currently process. We conclude that this is due to the differing conditions the solver is presented with every new customer that is added to the schedule. Besides a new geographic arrangement of the target area, there are typically also new skill requirements associated to this customer. It is likely that technicians that were initially planned for this region don't possess these skills. The entire solution base can be torn apart by a single new customer. This is one of the main reasons why this combination of problems is difficult to control.

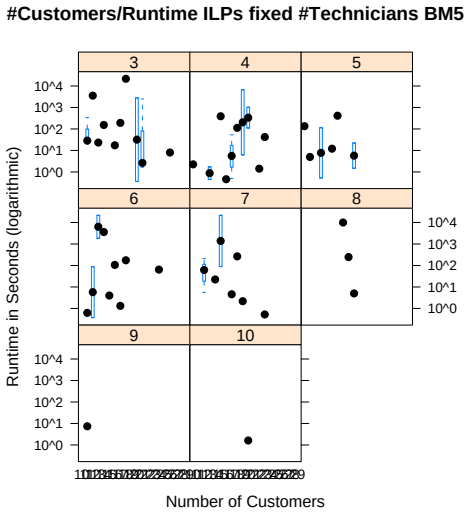


(a) Customers vs. Runtime

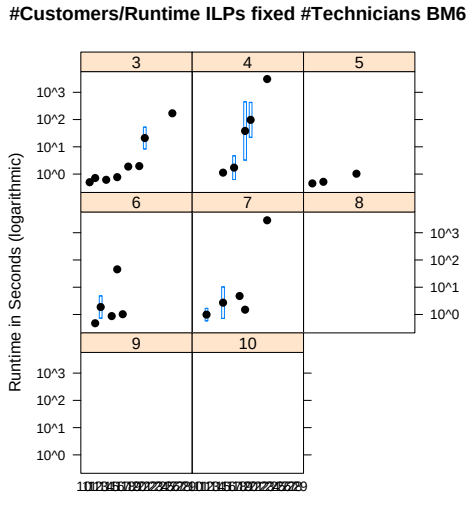


(b) Technicians vs. Runtime

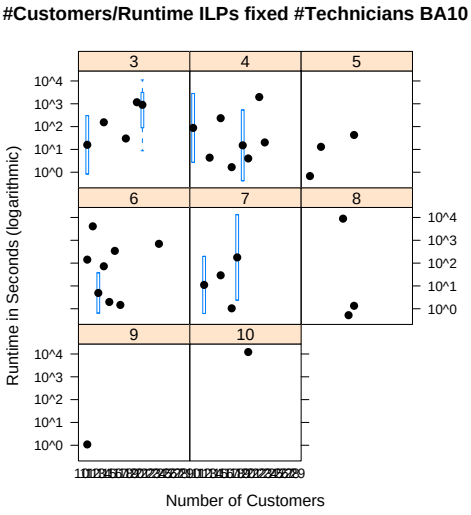
Figure 4.2: Relation between Number of Customers respsectively Number of Technicians and the Runtime for ILPs. The wide pale red boxes indicate *I1*, the medium pale blue boxes *I2* and the narrow pale green boxes *I3*



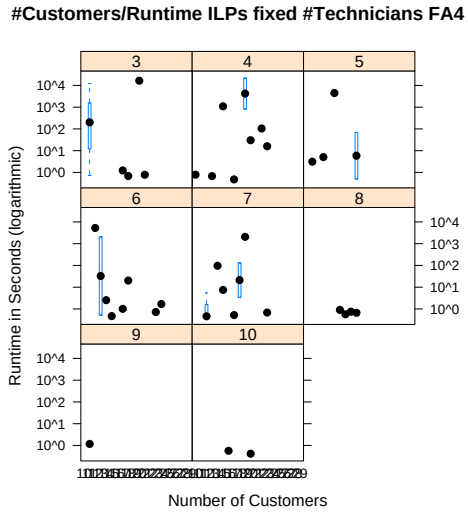
(a) *BM5, #Customer from 10 to 29*



(b) *BM6, #Customer from 10 to 29*



(c) *BA10, #Customer from 10 to 29*



(d) *FA4, #Customer from 10 to 29*

Figure 4.3: Relation between Number of Customers and Runtime for ILPs with fixed Number of Technicians for individual test instances *BM5*, *BM6*, *BA10* and *FM4*

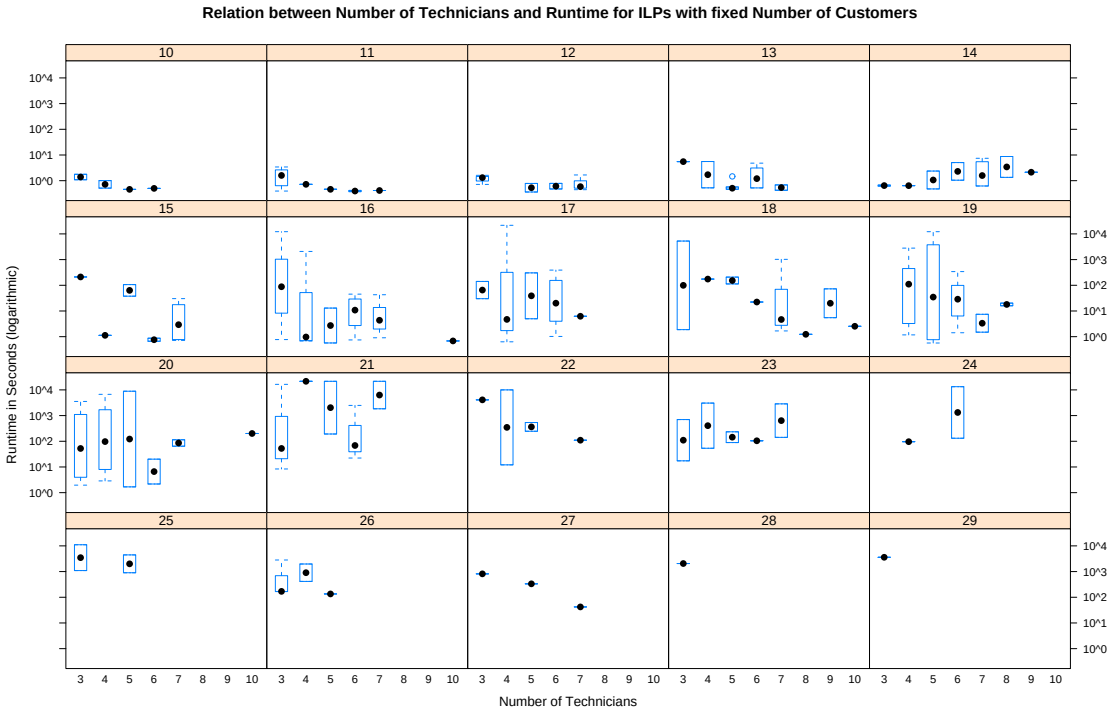
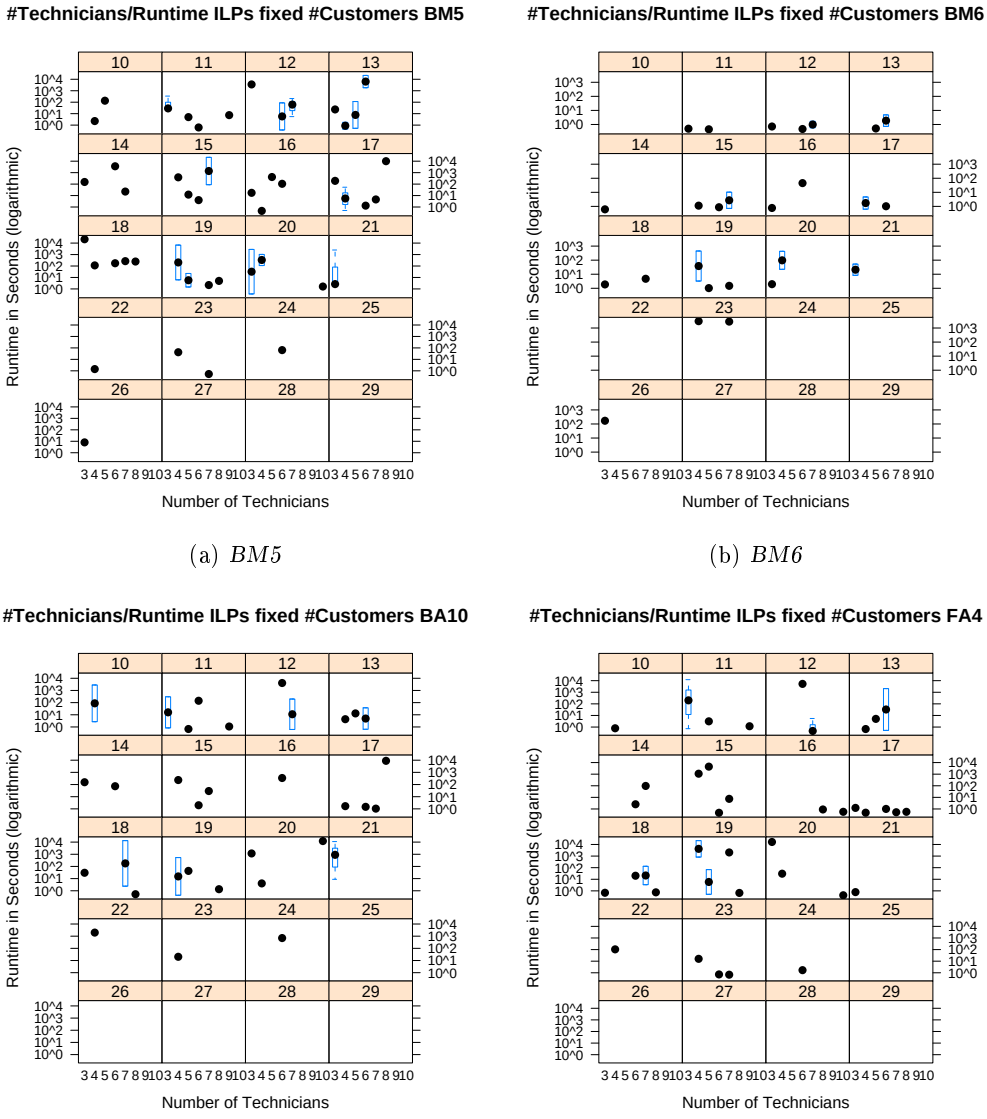


Figure 4.4: Relation between Number of Technicians and Runtime for ILPs with fixed Number of Customers



(a) *BM5*

(b) *BM6*

(c) *BA10*

(d) *FA4*

Figure 4.5: Relation between Number of Technicians and Runtime for ILPs with fixed Number of Customers for individual test instances *BM5*, *BM6*, *BA10* and *FM4*

4.5.1.2 Heuristic Scheduling Performance

As we can see in Figures 4.8 thru 4.11 on page 51, the performances of our heuristic scheduling approaches vary depending on the instance they are currently trying to solve.

For our heuristic approaches we see a similarly strong correlation between runtime and number of customers as we had seen for our *ILPs*. The relation between the runtime and the number of customers is displayed in Figure 4.6 for every algorithm combination. Particularly the *Greedy Assignment* approach combined with *ILPs* in the *Tour Construction Phase* show exceedingly high run times for higher numbers of customers. It is noticeable that there seems to be a good correlation between the fraction $\frac{\text{number of customers}}{\text{number of technicians}}$ in a given test instance and the runtime of all combinations that contain *ILPs*, again in particular the combinations with *Greedy Assignment* in the *Assignment Phase*. The bigger the fraction, the higher the runtime. This means the *ILPs* actually perform worse when they have to assign many customers to few technicians, which jives with our observations on the effect of the number of technicians on the *ILP* runtime in Section 4.5.1.1 on page 42.

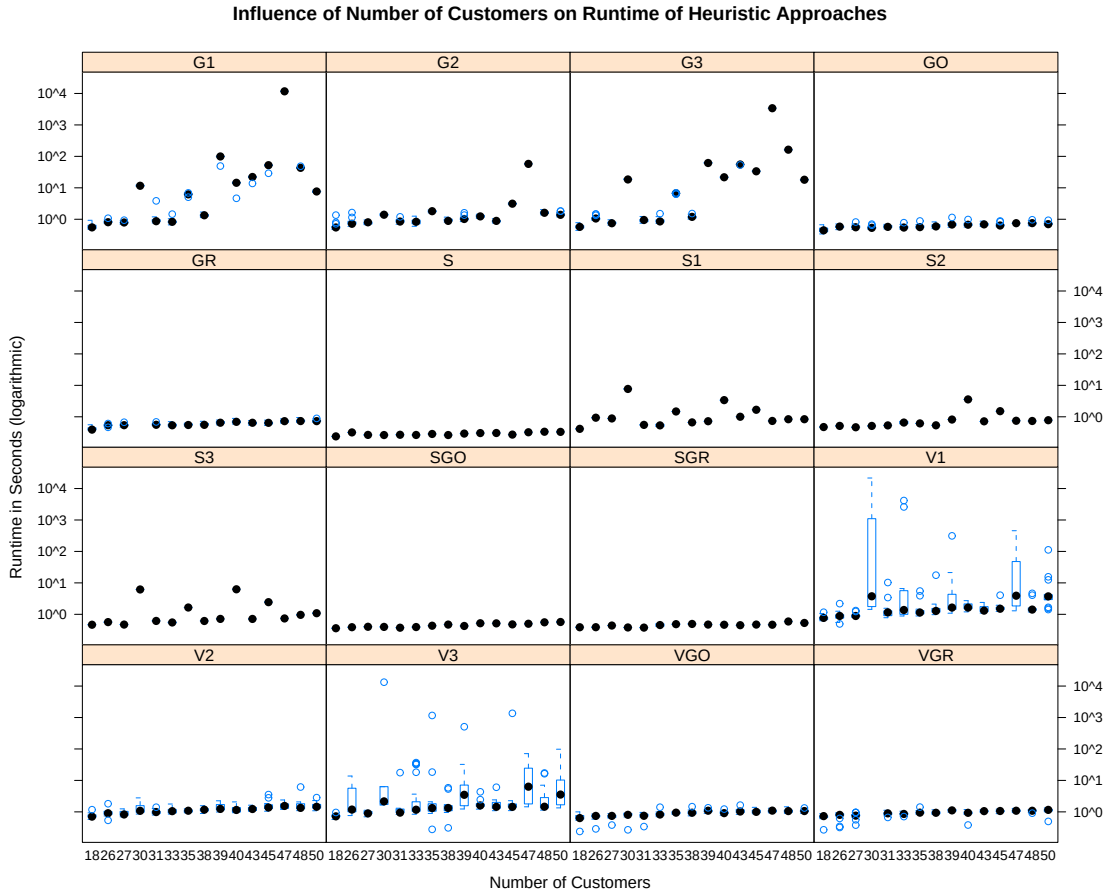


Figure 4.6: Influence of Number of Customers on Runtime of Heuristic Approaches

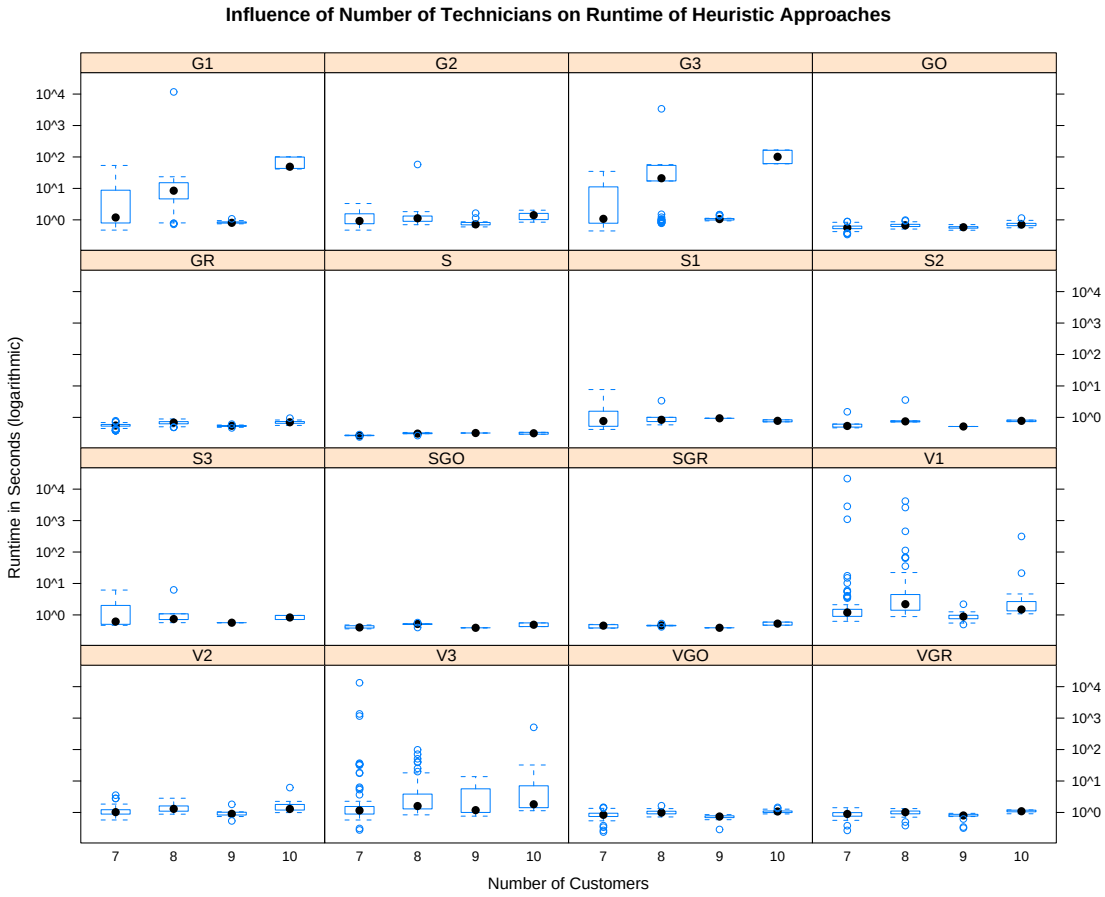


Figure 4.7: Influence of Number of Technicians on Runtime of Heuristic Approaches

4.5.1.3 Conclusions on Performance

The performance analysis shows that our *ILPs* are capable of solving instances with up to 7 technicians and and 20 customers (mean runtime for $I1 \ldots I3$ is 50 seconds). At 7 technicians and 21 customers we have the first peak with over 5,800 seconds, followed by 7 technicians / 22 customers at 150 seconds and 7 technicians / 23 customers at 1,440 seconds mean runtime.

The runtime seems to be influenced by various factors. The test instances themselves have a very significant impact on the runtime. Besides that we have found indications that the runtime correlates to the number of technicians and customers, as well as the fraction $\frac{\text{number of customers}}{\text{number of technicians}}$. The bigger this fraction, the higher the runtime.

Our heuristic approaches are significantly faster that the *ILPs* for lager problem sizes. While our *WSRP* formulation calculated on an instance with 45 customers and 7 technicians for 3 weeks until it was finally canceled, our heuristic approaches typically find solutions within seconds for the examined problem sizes.

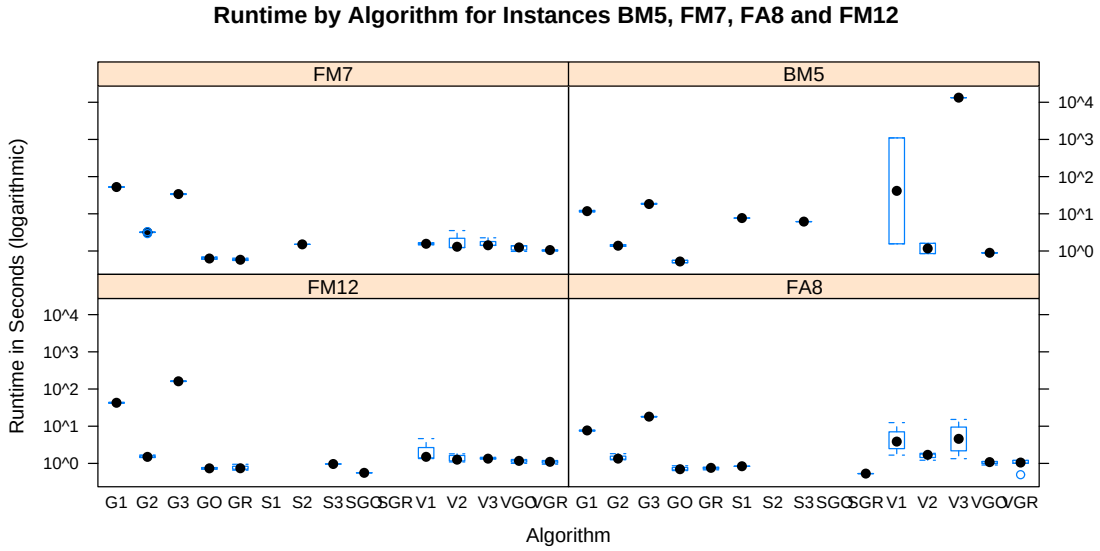


Figure 4.8: Runtime by Algorithm for Instances BM5, FM7, FA8 and FM12

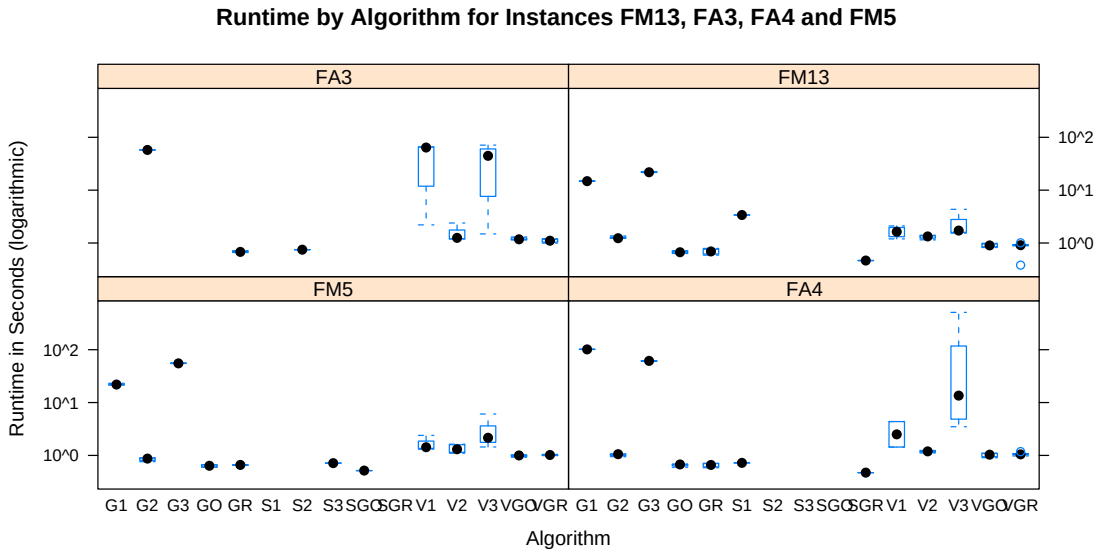


Figure 4.9: Runtime by Algorithm for Instances FM13, FA3, FA4 and FM5

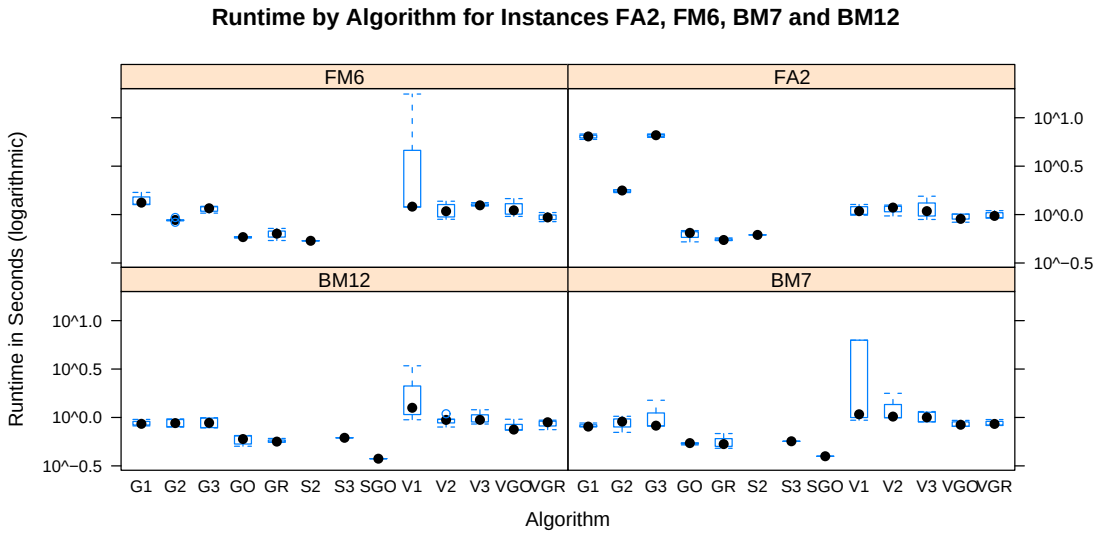


Figure 4.10: Runtime by Algorithm for Instances FA2, FM6, BM7 and BM12

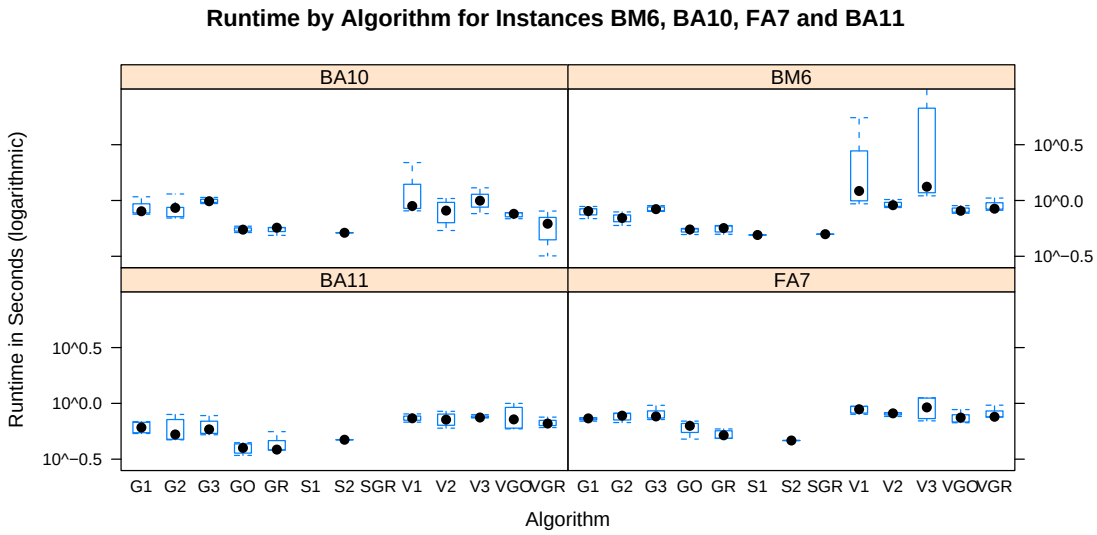


Figure 4.11: Runtime by Algorithm for Instances BM6, BA10, FA7 and BA11

Our *ILP* formulations perform reasonably well on most instances for solving the *TSP* problem in the *Tour Construction Phase* of our *Decomposition Strategies*. However, for some test instances we see excessively high run times that exceed our limit of 6 hours computation time.

4.5.2 Analysis of Scheduling Quality

In this section we take a closer look at the schedules we obtained using our different approaches and compare them to the manual schedules that were used in real-life.

In the end, we will find answers to some of our questions from Section 4.4.2.1 on page 40.

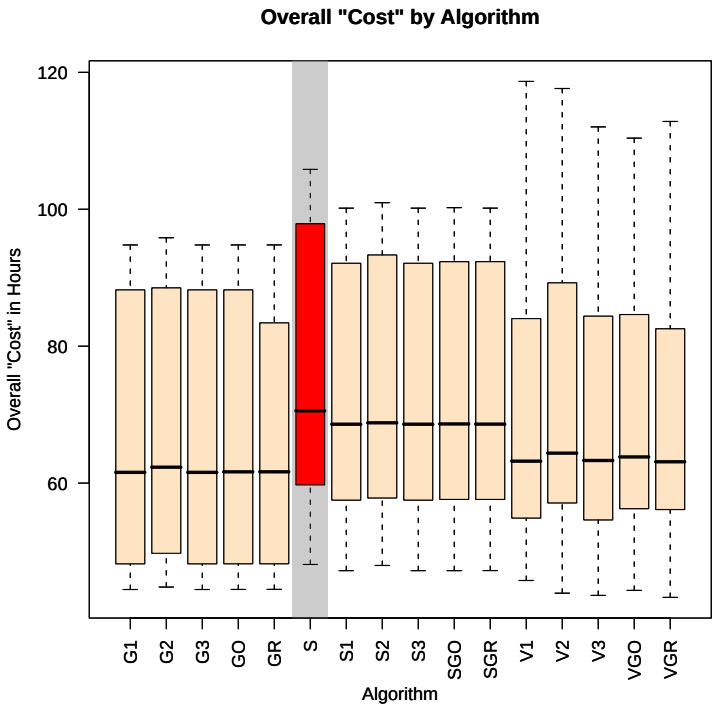
4.5.2.1 Overall Cost of Schedules

In Figure 4.12a on the next page we see that the overall cost obtained by our scheduling algorithms is typically lower than of the schedules that were actually followed in the real world (denoted by Algorithm "S"). Because the test instances themselves are strongly differing in overall cost as we can see in Figure 4.12b on the following page, the variance of our obtained overall costs in Figure 4.12a is also pretty high. We will get a better impression of the scheduling performance by isolating the test instances and looking at each of them separately.

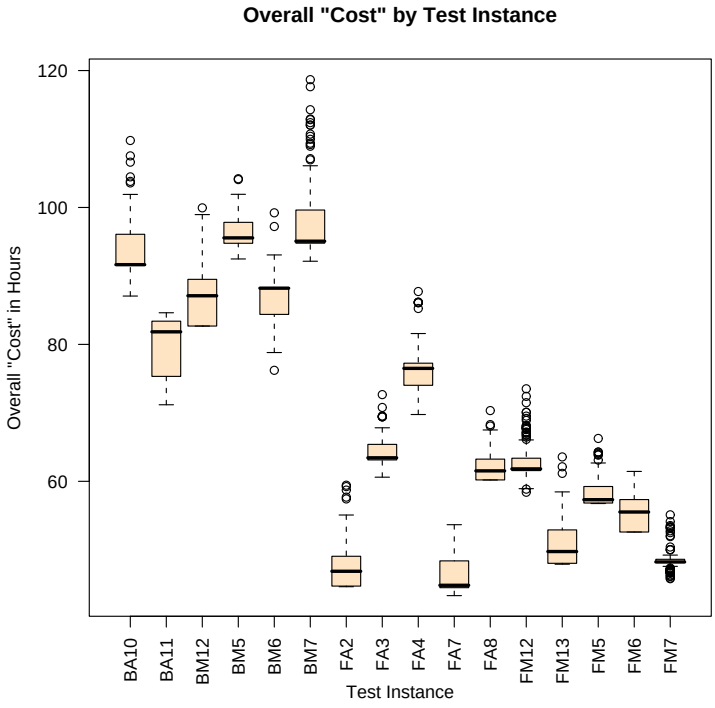
When looking at the test instances isolated, we can clearly see how the obtained schedules are improved over the actual real-world schedule. Figure 4.13 on page 55 depicts this for test instance *BM6* and also shows how the overall cost obtained by our indeterministic heuristic algorithms using the *k-means* based approach varies during multiple runs while those using our deterministic *Greedy* algorithms always come to the same solution.

Instance: **Algorithm ↓**	*BM5*	*BM6*	*BM7*	*BM12*	*BA10*	*BA11*
G1	94.77	88.21	94.78	82.68	91.53	83.39
G2	95.84	88.51	95.07	87.79	91.64	83.55
G3	94.77	88.21	94.78	82.68	91.53	83.39
GO	94.77	88.21	94.79	82.68	91.54	83.39
GR		88.21	94.79	82.69	91.54	83.39
S	104.07	99.22	105.83	96.54	99.25	82.62
S1	99.83	91.23	100.16	92.97	93.88	80.07
S2	99.89	92.74	100.96	96.76	93.88	81.27
S3	99.83	91.23	100.16	92.97	93.88	80.07
SGO	99.83	91.70	100.24	92.97	95.53	80.25
SGR	99.83	91.70	100.16	92.98	95.53	80.25
V1	97.74	83.41	101.31	88.09	96.69	75.23
V2	99.26	88.14	101.94	94.46	97.40	76.55
V3	59.66	84.00	99.68	88.67	96.61	75.38
VGO	96.92	83.23	99.69	89.20	95.92	76.81
VGR		84.81	103.27	88.98	96.56	75.37

Table 4.5: Mean Overall "Cost" in Hours by Algorithm for instances *BMx* and *BAx*



(a) Overall "Cost" by Algorithm



(b) Overall "Cost" by Test Instance

Figure 4.12: Overall Cost by Algorithm respectively Test Instance

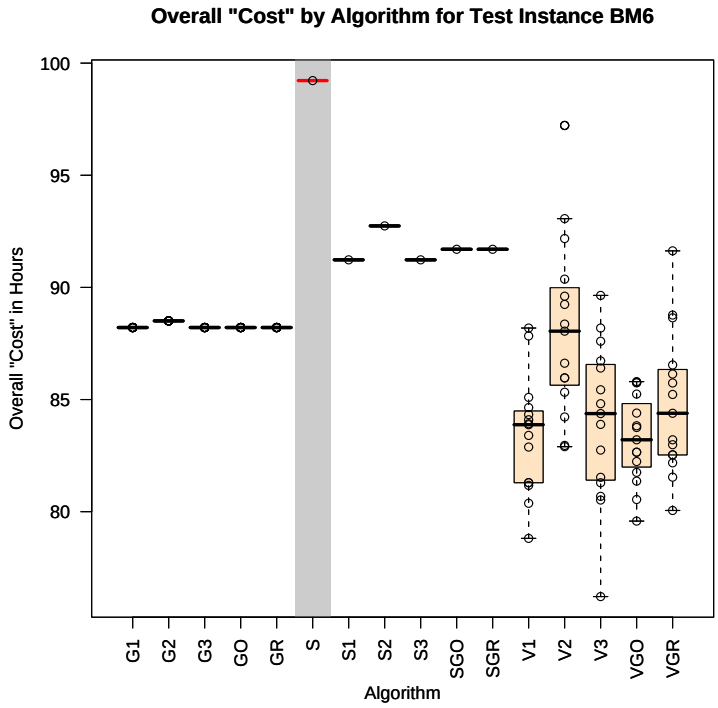
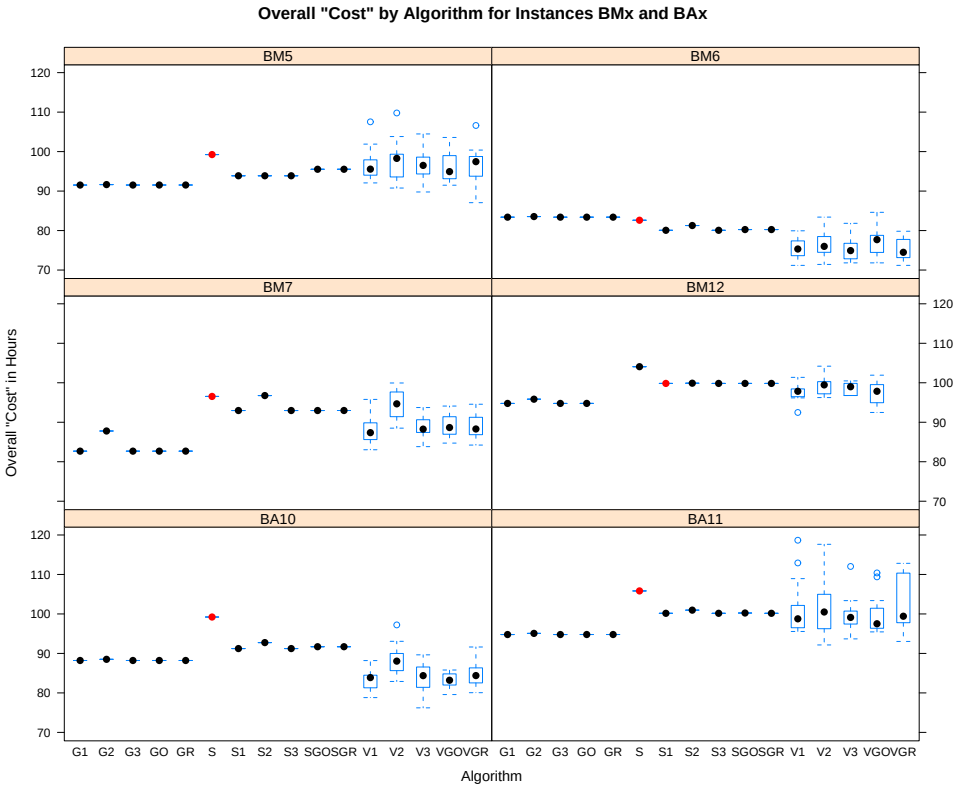


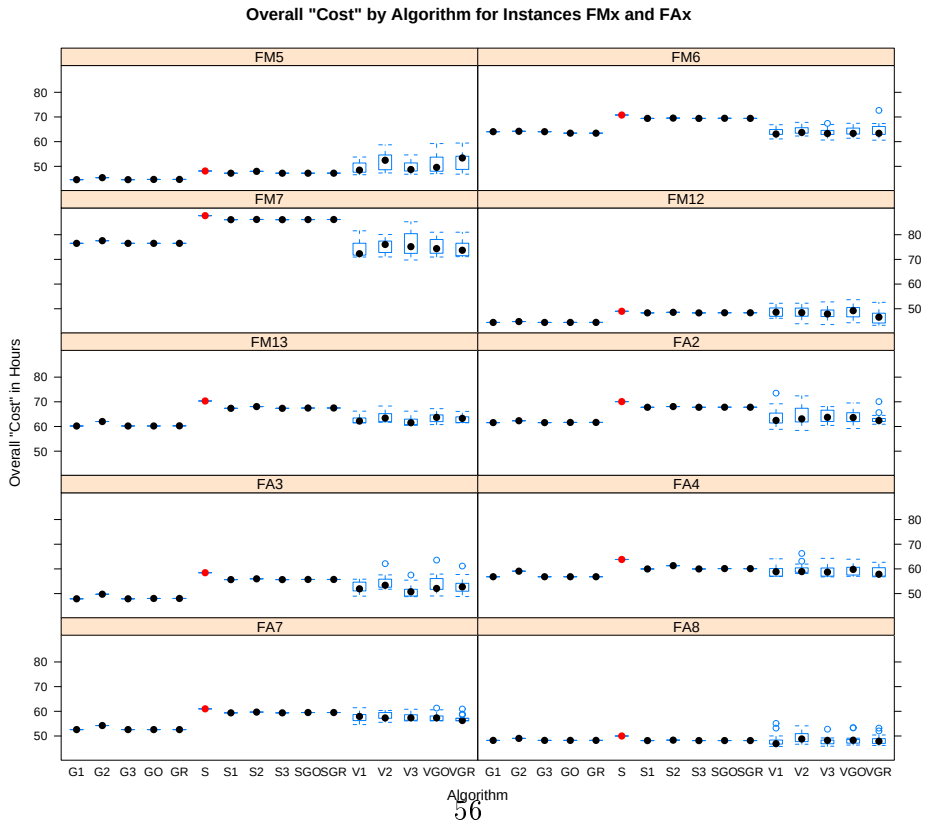
Figure 4.13: Overall "Cost" by Algorithm for Test Instance *BM6*

Instance: **Algorithm ↓**	*FM5*	*FM6*	*FM7*	*FM12*	*FM13*
G1	56.83	52.59	48.22	61.57	47.89
G2	59.07	54.23	49.03	62.32	49.74
G3	56.83	52.59	48.22	61.57	47.89
GO	56.83	52.59	48.22	61.65	48.02
GR	56.83	52.59	48.22	61.65	48.02
S	63.82	61.01	50.00	70.07	58.46
S1	59.98	59.38	48.14	67.78	55.66
S2	61.32	59.69	48.31	68.08	55.98
S3	59.98	59.38	48.14	67.78	55.66
SGO	60.13	59.51	48.15	67.80	55.72
SGR	60.11	59.51	48.15	67.80	55.72
V1	59.03	57.44	48.25	63.81	52.41
V2	59.93	58.08	49.49	64.49	54.60
V3	59.42	57.58	48.05	64.17	51.26
VGO	59.41	57.55	48.43	63.70	53.72
VGR	58.93	57.03	48.46	63.17	53.19

Table 4.6: Mean Overall "Cost" in Hours by Algorithm for Instances *FMx*



(a) Instances BMx and BAx



(b) Instances FMx and FAx

Figure 4.14: Overall "Cost" by Algorithm for Test Instances

Instance: **Algorithm**	*FA2*	*FA3*	*FA4*	*FA7*	*FA8*
G1	44.61	64.04	76.50	44.46	60.20
G2	45.41	64.20	77.58	44.81	62.01
G3	44.61	64.04	76.50	44.46	60.20
GO	44.70	63.44	76.50	44.46	60.20
GR	44.70	63.44	76.50	44.47	60.26
S	48.11	70.79	87.72	48.98	70.33
S1	47.21	69.42	86.09	48.33	67.36
S2	47.97	69.54	86.18	48.55	68.07
S3	47.21	69.42	86.09	48.33	67.36
SGO	47.21	69.47	86.13	48.38	67.48
SGR	47.23	69.43	86.15	48.38	67.50
V1	49.49	63.64	73.99	48.86	62.72
V2	51.89	64.54	75.32	48.57	63.74
V3	49.76	63.67	75.92	48.22	62.01
VGO	51.18	64.09	75.45	48.97	63.68
VGR	52.21	64.57	74.24	46.92	63.00

Table 4.7: Mean Overall "Cost" in Hours by Algorithm for Instances *FAx*

As we can see in Figure 4.13 on page 55, our heuristic schedules, in the example of test instance *BM6*, yield significant improvements in the routing of the technicians regarding the overall cost compared to the real-world schedules. Over all algorithm combinations A and test instances I the median improvement rate of the overall cost Ψ_O determined in our test runs compared to the overall cost of the original *SAR* Ψ_S is given as

$$\Phi_{A,I} := \frac{1}{\mid A \mid} \sum_{A} \frac{\Psi_O}{\Psi_S} \quad \forall \quad I$$

For our experiments, $\Phi_{A,I}$ lies between 0.5% and 13.2% with a $\text{mean}_I(\Phi_{A,I})$ of 8% and a $\text{median}_I(\Phi_{A,I})$ of 9.3%. The obtained mean overall costs by algorithm and test instance are given in Tables 4.5 on page 53 thru 4.7 on the preceding page for further reference.

4.5.2.2 Overall Technician Cost

The overall cost that a schedule possesses is a good indicator for the scheduling quality. However, we also need to check the feasibility of the resulting schedules. Since we have the requirement to schedule all customers in the input set and are not allowed to leave any customer without service, we have to weaken our maximum working cost constraint for the heuristic approaches such that the constraint is obeyed as long as there is any technician available who can perform the service without exceeding his maximum working cost. Whenever all technicians are filled up, we discard this constraint for the current assignment and assign the customer to the closest technician that possesses all skills required to service the customer. The resulting schedules are therefore potentially not feasible according to the strict feasibility rules that require both technician skills and maximum working cost to be obeyed. The motivation for this approach is the typical proceeding in the practice. While over hours are an exceptional cost that should be prevented wherever possible, customer satisfaction is typically a more important goal. Therefore, a service dispatcher would rather have some of his technicians do over hours than leaving a customer unserviced, although his *CFO* may rebuke him for doing so.

Figure 4.16 on page 60 displays, for each combination of algorithm and test instance, the overall cost that a technician route has. The *S*,...,*SGR* solutions, which conserve the actual technician-customer assignments from the real world, have a very similar overall technician cost of approximately 20 to 35 hours in in all cases, as expected (see Section 4.1 on page 30 or information on how to interpret hours in our test instances). Both the *Greedy Assignment* (*Gx* := *G1*... *GR*) and the *k-means* approaches (*Vx* := *V1*... *VGR*) vary in overall technician cost depending on the test instance. In the example of instance *BM5*, both *Gx* and *Vx* have longer tours than the SAR, while for instance *BM12* both *Gx* and *Vx* have shorter tours. For both test instances we yielded moderate improvements regarding the overall cost of the schedule (compare Figure 4.14a on page 56), but only for instance *BM12* we also stay below the allowed range of maximum working cost. When we compare the results regarding overall cost and overall technician cost for instance *BA10*, we find that the overall cost improves for the *Gx* as well as the *Vx* approaches, while the overall technician cost is lower than in the SAR for the *Gx* whereas it is higher for the *Vx*.

In general, we cannot conclude that an improvement of the overall cost results in an increase of the overall technician cost, which would lead to the concern that the improvements in scheduling cost are typically gained in part by shifting customer assignments to fewer technicians, thus resulting in a higher overall technician cost. In contrast, there seems to be a potential for savings without exceedingly raising the overall technician cost. This theory is supported by Figure 4.15 which puts the overall cost in relation to the maximum overall technician cost obtained throughout all test runs.

The variation of the overall technician cost in a single schedule is a good indicator for the feasibility of the schedule. In Figure 4.16 on the next page we see that the *Gx* typically have a greater variation of overall technician costs than the *Vx*. The SAR based solutions (*Sx*) have the smallest variation of overall technician cost out of all algorithm combinations, as expected. We therefore conclude that the *k-means* approach is superior to the *Greedy Assignment* approach with regard to the variation of the overall technician cost within a schedule, which can be interpreted as a balancing of the work load among the technicians.

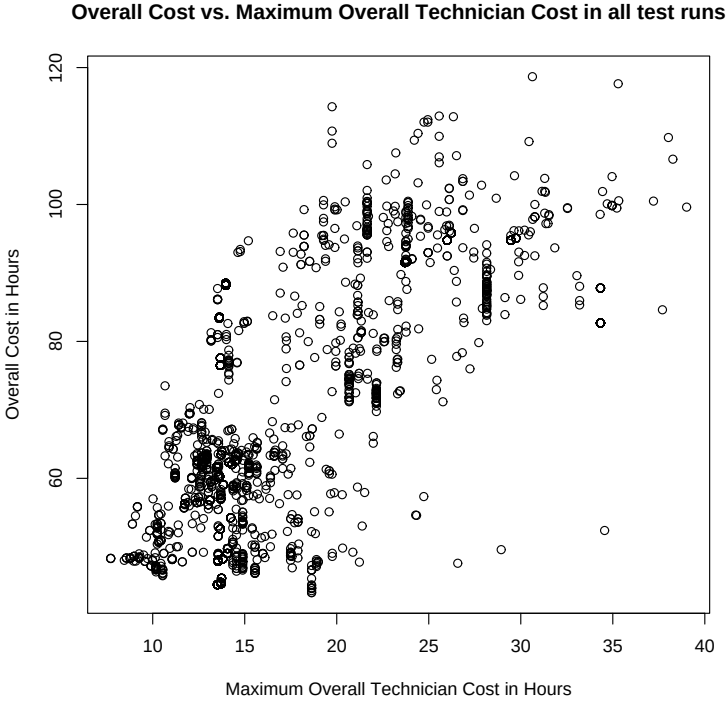


Figure 4.15: Overall Cost vs. Maximum Overall Technician Cost in all test runs

4.5.2.3 Conclusions on Scheduling Quality

Our *Greedy Assignment* approach has the main drawback that it does not necessarily generate technician-customer assignments that are properly balanced over the techni-

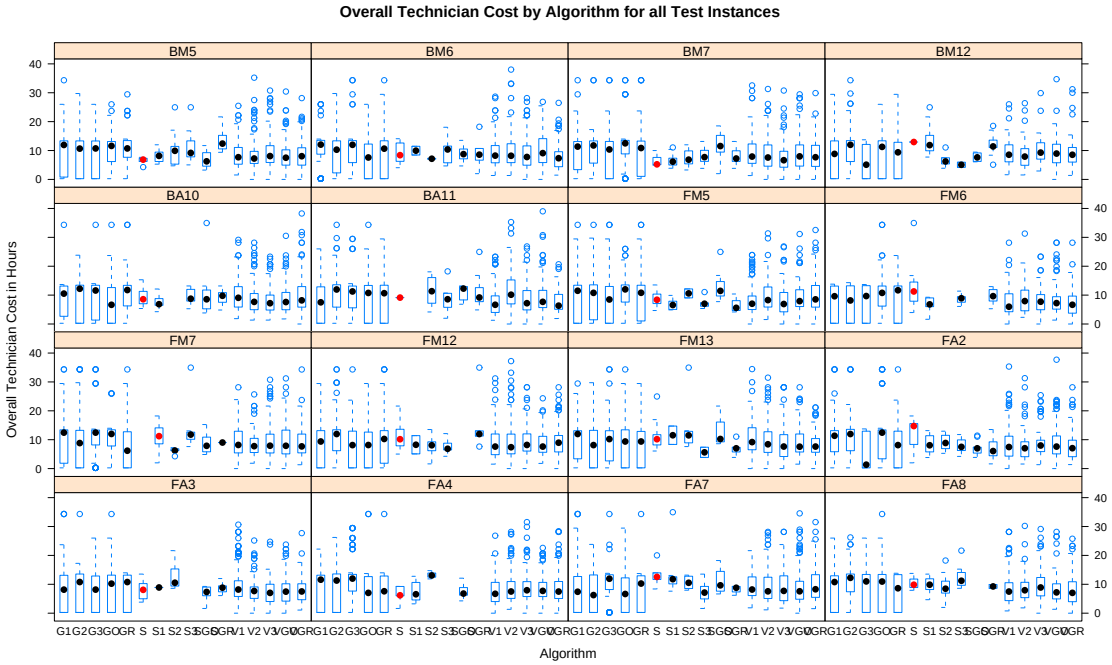


Figure 4.16: Overall Technician Cost by Algorithm for all Test Instances

cians. The resulting schedules therefore tend to be rather unrealistic and typically can not compete against the schedules obtained using the *k-means* approach in the *Assignment Phase* of our *Decomposition Strategy*.

Our *k-means* approach behaves reasonably well on most of our test instances, however depending on the utilized method for the *Tour Construction Phase* it may still show outliers regarding runtime.

Our *Greedy Roundtour Optimization* approach has a constantly low runtime. In combination with one of the *2-Opt* alternatives presented the *Greedy Roundtour Optimization* approach can be considered a viable approach for creating technician routes based on given technician-customer assignments.

5 Final Remarks

We close this thesis with a summary of our conclusions and and outline of further optimization prospects.

5.1 Conclusions

In this work we compared several different approaches for creating technician-customer assignments and round tours. We have also presented a set of real-world test instances which we used for our experiments. We have seen how the quality of the resulting schedules is affected by the chosen combination of scheduling approaches and what effects the different combinations have on the runtime of our approaches.

While we gained improvements of the overall cost in every case compared to the actual schedules performed in real-life, the combination of the *Greedy Construction* algorithm and *ILPs* (*G1*...*G3*) typically yielded the biggest savings. We also found that the resulting maximum work shift within a schedule varies significantly depending on the test instance and that the *Greedy Construction* based solutions show a bigger variance in the resulting overall technician costs, while the *k-means* based solutions distribute the workload more evenly among the technicians, i.e. they yield better technician-customer assignments than *G1*...*G3*. For our real-world application, an even distribution of the workload among the technicians with few outliers for overall technician cost is desirable, we therefore favor the *k-means* approach for the creation of technician-customer assignments.

The runtime behavior is less predictable for all approaches that use *ILPs* in the Tour Construction Phase than for the purely heuristic ones. The combinations based on *Greedy Assignment* and *ILPs* vary most out of all observed combinations and they have the most significant outliers. The *GO* and *GR* combinations on the other hand, which are purely heuristic, have a very constant runtime behavior. The combinations based on *k-means* show a similar trend. While those combined with *ILPs* tend to have exceedingly high run times for difficult test instances, the *VGO* and *VGR* combinations perform comparably quick on almost all instances.

Because of the better technician-customer assignments generated by the *k-means* based approaches, a more predictable runtime behavior and despite the by trend slightly higher overall costs compared to the *Greedy Assignment* based approaches, we consider *k-means*

the superior approach for creating technician-customer assignments and thus for creating improved schedules.

Our SAR based approaches behave very similar to each other regarding overall cost, regardless of whether *ILPs* or *Greedy* are used in the Tour Construction Phase. Also the variation of overall technician costs is fairly low, which indicates that the work days are pretty well balanced among the technicians. The comparison with our *Gx* and *Vx* instances reveals a potential for improvement of the overall schedule, however. Especially when the space of customers to choose from is broadened by allowing to pick from multiple days worth of work and also varying which customer is served on which day, we expect to see even bigger potentials for savings.

5.2 Further optimization prospects

The variations in the solution quality between our different Tour Construction approaches make further experiments with alternative algorithms worthwhile. In this section we will outline two examples for alternatives and provide a brief summary of their main aspects.

Christofides Algorithm

An alternative to our examined approaches in the Tour Construction Phase is the *Christofides Algorithm*, which is a heuristic algorithm to find a near-optimal solution to the instances of the *TSP* that satisfy the triangle inequality. The basis of *Christofides' algorithm* is a *Minimum Spanning Tree* (*MST*) in the graph. There are two well-known algorithms to calculate a *MST*, *Kruskal's algorithm* and *Prim's algorithm*. Both are greedy algorithms that run in polynomial time. While *Prim's algorithm* is more performant it also requires complex data structures (so-called *Fibonacci Heaps* or *AF-Heaps* [FW90]). For typical problem sizes also *Kruskal's algorithm* is reported to be sufficiently performant and the effort for implementation is significantly smaller than for *Prim's algorithm*.

Kruskal's algorithm first creates a forest F (a set of trees), where each vertex in the graph is an individual tree. It then creates a set S containing all the edges in the graph. While S is not empty, it removes an edge with minimum weight from S. If the removed edge connects two different trees, it adds it to the forest, combining the two trees into a single tree, otherwise it discards the edge. In the end, the forest has only one component and forms a minimum spanning tree of the graph [Kru56].

Christofides algorithm then finds a perfect matching with minimal weight in the complete graph over vertices with odd degree in the *MST* and combines the edges of the matching and the *MST* to form a multigraph. In this multigraph it then forms an *Eulerian path* and finally eliminates already visited nodes. The result is a *Hamiltonian path* which represents an improved version of the initial round tour.

The Savings Algorithm

The Savings Algorithm is a heuristic algorithm for forming sequenced round tours out of a given customers. The number of technicians is not known in the beginning of the algorithm and is itself a decision variable. This may be a worthwhile approach in situations where the daily workload over all technicians strongly varies and our constraint of providing service to every customer shall be obeyed [Odo04].

In the first step all customers are connected to the depot c_0 through a roundtrip, which mens it starts with $N-1$ round tours. It then merges the round tours node by node, by maximizing the following savings function:

$$savings(c_i, c_j) := \text{dist}(c_0, c_i) + \text{dist}(c_0, c_j) - \text{dist}(c_i, c_j)$$

The algorithm allows to check for various side constraints in each iteration of tour merges. It terminates when no more feasible merges of routes can be found.

List of Algorithms

1 Greedy Construction Algorithm 20
2 Improve Voronoi Partitioning 24
3 Greedy Roundtour Optimization Algorithm 26
4 Greedy 2-Opt Algorithm 29

Index

Christofides flow-based model, 15
2-Opt, 20, 27, 28, 41, 60

AF-Heaps, 62
Assignment Phase, 41, 48, 60

Christofides Algorithm, 62
Christofides algorithm, 62
Christofides' algorithm, 62
cost
    on-site working cost, 12–16, 20, 30, 32–34
    travel cost, 12–16, 20, 21, 23, 24, 26, 29, 30, 32–34, 63

Decomposition Strategies, 52
Decomposition Strategy, 7, 19–21, 41, 60
depot, 12, 13, 63

Eulerian path, 62

feasible routing solution, 13–15, 19
Fibonacci Heaps, 62

Greedy, 21, 24, 53, 62
Greedy 2-Opt, 20, 27, 28
Greedy Assignment, 20, 23, 31, 41, 48, 58, 59, 61
Greedy Construction, 7, 19, 20, 23, 26, 31, 41, 61
Greedy Roundtour Opt., 31
Greedy Roundtour Optimization, 20, 26–28, 40, 41, 60

Hamiltonian path, 62

k-means, 20, 21, 27, 28, 31, 41, 53, 58–61
Kruskal's algorithm, 62

Maximum Working "Cost", 7, 12, 13, 15, 21, 22, 26, 32, 33, 36, 37, 40, 58

Post Optimization, 41
Prim's algorithm, 62

set
    of customers, 12, 20, 29
    of skills, 12
    of technicians, 12, 20, 26
skills
    set of skills, 12
    skill requirements, 12, 13, 15, 22, 36
    skill set, 12, 20, 22
    technician skills, 12, 13, 15

Tour Construction, 41
Tour Construction Phase, 41, 48, 52, 60

Voronoi Cell, 21, 23, 24
Voronoi Cells, 21
Voronoi Center, 21–25
Voronoi Centers, 21
Voronoi Partitioning, 21, 23, 25

6 Acronyms

CFM	Christofides flow-based model
CFO	Chief Financial Officer
ERP	Enterprise Resource Planning
ILP	Integer Linear Program
JSP	Job Shop Scheduling Problem
LP	Linear Program
MST	Minimum Spanning Tree
mTSP	Multiple Traveling Salesman Problem
SAR	Service Activity Report
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem
WSMRP	Water Scheduling Makespan Routing Problem
WSCRP	Water Scheduling Combined Routing Problem
WSRP	Water Scheduling Routing Problem

Bibliography

[AL97] Emile Aarts and Jan K. Lenstra, editors. *Local Search in Combinatorial Optimization*, chapter 8, pages 215–310. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[Bec03] J. Christopher Beck. Vehicle routing and job shop scheduling: What's the difference? In *Proc. of the 13th International Conference on Automated Planning and Scheduling*, pages 267–276, 2003.

[Bek06] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, June 2006.

[BG07] M. Bakhouya and J. Gaber. An immune inspired-based optimization algorithm: Application to the traveling salesman problem. *Advanced Modelling and Optimization*, 9(1):105–116, 2007.

[BSl95] Julien Bramel and David Simchi-levi. A location based heuristic for general routing problems. *Operations Research*, 43:649–660, 1995.

[Cro57] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, pages 791–812, November-December 1957.

[DBL90] *31st Annual Symposium on Foundations of Computer Science, 22-24 October 1990, St. Louis, Missouri, USA*, volume II. IEEE, 1990.

[FW90] Michael L. Fredman and Dan E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. In *FOCS* [DBL90], pages 719–725.

[GH06] Zong Woo Geem and Han Hwangbo. Application of harmony search to multi-objective optimization for satellite heat pipe design. Technical Report UKC 2006 AST-1.1, US-Korea Conference, Aerospace Science & Technology, 2006.

[GLP05] Zong Woo Geem, Kang Seok Lee, and Yongjin Park. Application of harmony search to vehicle routing. *American Journal of Applied Sciences*, 2(12):1552–1557, 2005.

[Kru56] Jr. Kruskal, Joseph B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[KSVW08] Sven O. Krumke, Sleman Saliba, Tjark Vredeveld, and Stephan Westphal.

Approximation algorithms for a vehicle routing problem. *Mathematical Methods of Operations Research*, February 2008.

[LW05] Andrew Lim and Fan Wang. Multi-depot vehicle routing problem: A one-stage approach. *IEEE Transactions on Automation Science and Engineering*, 2(4):397–402, October 2005.

[MRD07] Waqar Malika, Sivakumar Rathinamb, and Swaroop Darbhaa. An approximation algorithm for a symmetric generalized multiple depot, multiple travelling salesman problem. *Operations Research Letters*, 35(6):747–753, November 2007.

[NC81] P. Toth N. Christofides, A. Mingozzi. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, December 1981.

[Odo04] A. R. Odoni. Networks: Lecture 2. *ORSA-Journal of the Computing*, November 2004.

[Rei91] G. Reinelt. Tsplib - a travelling salesman problem library. *ORSA-Journal of the Computing*, 3(4):376–384, Fall 1991.

[RSLI77] Daniel J. Rosenkrantz, Richard E. Stearns, Philip M. Lewis, and II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

[SS07] Rathinam Sivakumar and Raja Sengupta. 5/3-approximation algorithm for a multiple depot, terminal hamiltonian path problem. Research Report UCB-ITS-RR-2007-1, Institute of Transportation Studies, University of California, Berkeley, May 2007.