Bicriterial relocation of the Viennese ambulance service

Eingereicht von:

Martin Petz

DIPLOMARBEIT

Zur Erlangung des akademischen Grades

Magister rerum socialium oeconomicarumque

Fakultät für Wirtschaftswissenschaften

der Universität Wien

Studienrichtung: Internationale Betriebswirtschaft

Betreuer: a.o. Univ.-Prof. Dr. Walter Gutjahr

Wien, im März 2008

Abstract

The organization of emergency medical services is characterised by competing objectives. Besides low response times of the ambulances other aspects have to be considered as well. To address this problem a dynamic relocation strategy for ambulance vehicles is established in this work. One objective is to maximize the resident population reachable within a given time frame. This can be done relatively easy for a set number of vehicles, but whenever a vehicle is dispatched to a call the number of available vehicles changes and relocations may be necessary to maximize the population coverage. However too many relocations would be unreasonable for the ambulance crews. Therefore the second objective is to minimize the number of relocations. The problem of dynamically relocating the vehicles is resolved by an a priori approach which solves all possible states in advance. Each time the number of vehicles changes the appropriate precalculated solution is applied. The high computational complexity of this strategy is met by Pareto Ant Colony Optimization. PACO is a specialized metaheuristic for multiobjective optimisation problems inspired by the foraging behaviour of real ants. Different algorithm versions are established and programmed in MatLab in order to explore the capabilities of the algorithm. Variations in the pheromone structures and size of the solution space, as well as approaches using shifting ant numbers are tested for their influence on the convergence behaviour of the algorithm. Multiple solutions are the outcome of the algorithm. These sets of solutions are called approximation sets, as they are an approximation of the pareto optimal front. The presence of several optimization criteria prevents an objective rating of the approximation sets, but a combination of unary measures can be used to assess certain quality aspects. In this work the found solutions ratio, the average distance and the hypervolume metric are implemented. The knowledge of the real pareto optimal front allows for a far better evaluation of the performance. Therefore the algorithms are tested on a problem instance small enough to completely enumerate all solutions, before they are used to develop a relocation strategy for the NEF-system (Notarzt Einsatz Fahrzeug) of the Viennese ambulance service.

Zusammenfassung

In der Organisation von Rettungsdiensten ergeben sich immer wieder konkurrierende Ziele. Neben einer möglichst hohen Verfügbarkeit der Rettungsfahrzeuge müssen bei der Aufstellung der Rettungsflotte auch andere Aspekte berücksichtigt werden. Zur Lösung dieses Problems wird in dieser Arbeit eine dynamische Reallokations-Strategie für Rettungs Fahrzeuge entwickelt. Eines der Ziele ist die Anzahl der innerhalb eines vorgegebenen Zeitraums erreichbaren Wohnbevölkerung zu maximieren. Für eine bestimmte Anzahl an Fahrzeugen kann eine gute Aufstellung relativ einfach gefunden werden, jedoch verändert sich die Anzahl der Fahrzeuge jedes Mal, wenn ein Fahrzeug zu Einsatz kommt. In diesem Fall müssen die Fahrzeuge möglicherweise neu verteilt werden, um eine maximale Abdeckung zu erreichen. Zu viele Fahrten allein zur Neustrukturierung sind den Fahrzeugbesatzungen allerdings nicht zuzumuten. Deshalb soll in einem zweiten Ziel die Anzahl der Neuverteilungen minimiert werden. Die dynamische Neuverteilung wird durch einen a priori Ansatz gelöst, bei dem im Vorhinein für alle möglichen Zustände eine Lösung errechnet wird. Wenn sich der Zustand ändert, kann die jeweilige Lösung angewandt werden. Der hohe Rechenaufwand dieser Methode wird mit Hilfe der Pareto Ant Colony Optimization bewältigt. PACO ist eine auf Mehrzieloptimierung spezialisierte Metaheuristik, dessen Prinzip durch das Verhalten von Ameisen bei der Futtersuche inspiriert ist. Unterschiedliche Algorithmus Varianten werden entwickelt und in MatLab programmiert um das Potential des Algorithmus zu erforschen. Variationen der Peromonstrukturen, unterschiedlich große Lösungsräume und ein Ansatz mit veränderlicher Ameisenanzahl werden auf ihren Einfluss auf den Algorithmus getestet. Der Algorithmus liefert gleich mehrere Lösungen als Ergebnis. Da mehrere Zielvorgaben bestehen ist es nicht möglich diese Lösungs-Sets objektiv zu reihen. Es können jedoch Kennzahlen errechnet werden, um bestimmte Aspekte der Lösungsgüte zu beschreiben. In dieser Arbeit kommen der Anteil der gefundenen Lösungen, die durchschnittliche Distanz und die Hypervolume Metrik zur Anwendung. Ist die pareto optimale Front bekannt, kann die Qualität der Lösungs-Sets genauer bewertet werden. Deshalb werden die Algorithmen zunächst an einer Probleminstanz getestet, die klein genug ist alle Lösungen zu errechnen, bevor sie benutzt werden, um eine Relokalisierungs-Strategie für das NEF-System (Notarzt Einsatz Fahrzeug) der Wiener Rettung zu entwickeln.

Table of contents:

Appendix:

List of Figures:

List of Tables:

1. Introduction

The organization of an ambulance service poses several interesting problems, since it is affected by various aspects that have to be taken into consideration. One mayor goal is to decrease the response times of the ambulances as the urgency of most cases demands a fast reaction. With higher numbers of ambulance vehicles the average response time can be decreased, but on the other hand budget restrictions have to be taken into consideration as well. Such competing objectives occur relatively often in practical applications. The common absence of one single optimal solution makes this type of problem rather difficult to solve. Instead of one optimal solution, whole sets of solutions with incomparable quality can be found.
An exemplarily problem arises from the location of emergency vehicles at different waiting stations in order to maximize the population reachable within a given time. For a set number of vehicles the optimal locations can be found, but if one of these vehicles has to be dispatched to an emergency, the number of available vehicles changes. In this case the vehicles may have to be relocated in order to reach the highest possible coverage. Of course it is unacceptable for the crews to relocate the ambulances every time an emergency occurs. A reasonable compromise between high population coverage and small number of relocations has to be found. This problem is addressed by this work.

In this work a bicriterial location problem for the dynamic relocation of emergency vehicles is established. The problem is based on the Maximum Expected Coverage Relocation Problem by Gendreau et al. (2006). To solve the bicriterial problem the Pareto Ant Colony Optimization Metaheuristic (PACO) is employed Dörner et al. (2002). Variations of PACO are tested to explore the capabilities of the algorithm. Subsequently the model is applied to the ambulance service system of Vienna.

In the next chapter the different models proposed for ambulance location problems are explained. Chapter 3 points out the working mechanism of the Ant Colony Optimization and the use of Pareto Ant Colony Optimization for multicriterial problems. In chapter 4 the Algorithms based on the Pareto Ant Colony Optimization Metaheuristic and their variations are explained. The full code of an exemplary algorithm is given in Appendix B. All other codes used in this work can be found on the enclosed CD. Many metrics have been proposed to assess the performance of multiobjective optimizers, because the evaluation of the solution quality is rather complex. The different metrics of performance used in this work are described in chapter 5. A comparison of the algorithm versions and the results of the Viennese relocation problem can be found in chapter 6. Finally I state my conclusions in chapter 7.

2. Ambulance Location Problems

Various models have been developed for the emergency vehicle location problem. Early approaches were based on deterministic models like the Location Set Covering Problem (LSCP) by Toregas et al. (1971) and the Maximal Covering Location Problem (MCLP) by Church and ReVelle (1974). The LSCP minimizes the number of vehicles necessary to cover all demand, while the MCLP maximizes the covered demand points with a given number of emergency vehicles. Both models ignore the change in coverage once a vehicle is dispatched to a call. To reflect this problem probabilistic models have been established. They take into account that the vehicles are unavailable for calls while they are busy. The Maximum Expected Covering Location Problem (MEXCLP) proposed by Daskin (1983) is an early representative of these probabilistic models. A different approach is used in dynamic models. These models solve a static model for the number of vehicles available every time a vehicle is dispatched to a call. This can be problematic, as the calculations have to be done in short time in order to respond fast enough to the changes. Gendreau et al. (2001) proposed the Dynamic Double Standard Model (DDSM) that is solvable in real time. The necessary computational power is attained by the implementation of a tabu-search heuristic and the use of parallel computing. Furthermore Gendreau et al. (2006) proposed an alternative model that precalculates the solutions for all possible states of the dynamic system. This model is called the Maximum Expected Coverage Relocation Problem (MECRP). A new model named Maximum Coverage Minimum Relocation Problem (MCMRP) was developed to address the ambulance relocation problem in the course of this work. It is based on the MECRP and extends this model to a bicriterial optimization problem.

2.1 The Maximum Expected Coverage Relocation Problem

In their work Gendreau et al. formulate a mathematical model to solve the problem of emergency vehicle location. The MECRP is based on a dynamic model, instead of a static or probabilistic model. Static models neglect the change in coverage when emergency vehicles are dispatched to a call. Probabilistic models are one way to take this problem into account. The idea of dynamic models is to solve a static model for the relocation of vehicles each time an emergency vehicle is dispatched to a call. So each time the state changes a new solution has to be calculated. This is called the a posteriori approach. Alternatively Gendreau et al. proposed an a priori approach where several solutions are precomputed in order to be applied if the corresponding event occurs. This approach addresses the problem not to have enough time to calculate a new solution, if two events occur in quick succession. The MECRP computes a solution for every possible state that can happen in advance. So for each possible number of available vehicles, a solution is calculated. Unfortunately this method is only feasible for a relatively small number of possible states.

The aim of the MECRP is to maximize the population reachable within a given time, while the number of relocations of emergency vehicles is limited. Instead of implementing relocation costs, the number of relocations is constrained by a fixed maximal value of relocations. There may be at most α_k waiting sites changed, if the system moves from state k to $k+1$ (if $k<=n-1$). If the system moves from state k to state $k-1$ (if $k>=1$), at most α_k+1 waiting sites may be changed. In this case at least one waiting site has to be changed, because a vehicle has to be dispatched to a call. The binary variables u_{jk} are defined as 1 if and only if $j \in W$ ceases to be a waiting site when the system moves from state k to state $k+1$ ($k=1,\dots,n-1$). In this way multiple counting of waiting site changes can be avoided. The population covered by the different numbers of available vehicles is weighted by the probability to encounter the corresponding number of available vehicles. The probability q_k of having k vehicles available of a total number of n vehicles is obtained by binomial distribution:

$$q_k = \binom{n}{k} p^k (1-p)^{n-k} \qquad (k = 0,\dots,n) \qquad (1)$$

The probability p that a vehicle is available is calculated by $p = 1 - \frac{\lambda}{n\mu}$. Where λ is the arrival rate of calls, and μ is the average service rate.

The MECRP is designed on a directed graph $G = (V \cup W, A)$. The vertex set of the demand points is given by *V*, while *W* is a vertex set of potential waiting sites for $n \leq |W|$ emergency vehicles. *A* is a set of arcs defined on $(V \cup W)^2$. Each arc $(i, j) \in A$ has a defined driving time. The demand d_i at vertex $i \in V$ is given by the population of the area assigned to vertex *i*. A vertex $i \in V$ is covered by a vertex $j \in W$, if the driving time from *j* to *i* is less than *r*. *r* represents the coverage radius of the vehicles in driving time. W_i is the subset of vertices of *W* covering $i \in V$. The binary variable x_{jk} equals 1, if and only if a vehicle is located at waiting site $j \in W$ in state *k*. The binary variable y_{ik} equals 1, if and only if demand point $i \in V$ is covered by at least one vehicle in state *k*.
The mathematical formulation is as follows:

$$\text{Maximize} \quad \sum_{k=1}^{n} \sum_{i \in V} d_i q_k y_{ik} \tag{2}$$

$$\text{subject to:} \quad \sum_{j \in W_i} x_{jk} \geq y_{ik} \qquad (i \in V, k = 0, \ldots, n) \tag{3}$$

$$\sum_{j \in W} x_{jk} = k \qquad (k = 0, \ldots, n) \tag{4}$$

$$x_{jk} - x_{j,k+1} \leq u_{jk} \qquad (j \in W, k = 1, \ldots, n-1) \tag{5}$$

$$\sum_{j \in W} u_{jk} \leq \alpha_k \qquad (k = 1, \ldots, n-1) \tag{6}$$

$$x_{jk} \in \{0,1\} \qquad (j \in W, k = 0, \ldots, n) \tag{7}$$

$$y_{ik} \in \{0,1\} \qquad (i \in V, k = 0, \ldots n) \tag{8}$$

$$u_{ik} \in \{0,1\} \qquad (j \in W, k = 1, \ldots, n-1) \tag{9}$$

The objective function (2) maximizes the weighted population coverage. Constraint (3) controls that vertex $i \in V$ is only covered if at least one vehicle is located in W_i. Constraint (4) regulates the number of vehicles for each state. Constraints (5), (6) and (9) are used to limit the number of waiting site changes to a fixed maximal value α_k when the system moves from state *k* to state *k+1*. While constraint (5) limits the number of relocations to u_{jk}, constraint (6) assures that all u_{jk} of state *k* do not exceed the maximal allowed number of waiting site changes α_k. Constraints (7), (8) and (9) simply define x_{jk}, y_{ik} and u_{jk} as binary values. The solution of the MECRP gives the locations for k emergency vehicles for every possible state k. It can be applied to a dynamic problem, by solving the MECRP once and using the solution values of x_{jk} when the system moves to the appropriate state k.

The formulation as an integer linear programming marks an interesting way of solving a dynamic relocation problem. In a clever way the covered demand is maximised and the number of relocations is limited while a complex multicriterial problem can be avoided. However the computational complexity limits the solution capability to problems with relatively small numbers of vehicles. In the next chapter a different approach to the same problematic is presented. The Maximum Coverage Minimum Relocation Problem MCMRP is formulated as a biobjective optimization problem. The increase in computational complexity will be met by the use of an advanced multiobjective optimization algorithm called Pareto Ant Colony Optimization Dörner et al. (2002).

2.2 The Maximum Coverage Minimum Relocation Problem

Similar to the MECRP, the Maximum Coverage Minimum Relocation Problem (MCMRP) uses a directed graph $G = (V \cup W, A)$. *V* represents the vertex set of demand points and *W* stands for the vertex set of potential waiting sites for $n \leq |W|$ emergency vehicles. A is a set of arcs defined on $(V \cup W)^2$. The driving time is defined in minutes for each arc $(i, j) \in A$. The demand for the emergency service is assumed to be relational to the population density. Therefore the demand d_i at vertex $i \in V$ is given by the population of the area assigned to vertex *i*. A demand point $i \in V$ is said to be covered by a waiting station $j \in W$ if it can be reached from *j* within a driving time of *r*. The variable *r* defines the coverage radius of the emergency vehicles. The subset of vertices of *W* covering $i \in V$ is defined as W_i. Like the MECRP the MCMRP uses an apriori approach for the dynamic model formulation. This means that all possible states of the model are precomputed. The number of states is defined by the number of available vehicles *k* waiting for a call (*k=0,…,n*). Whenever a vehicle is dispatched to a call the system moves from state *k* to state *k-1*. For each call only one vehicle is being dispatched. Emergencies requiring more than one vehicle are treated as a succession of calls. For each state two binary variables are defined. x_{jk}=1 if a vehicle is located at $j \in W$ in state *k*, and y_{ik}=1 if the demand point $i \in V$ is covered by at least one waiting station in state *k*.
Two objective functions measure the quality of a solution. The first objective function maximises the expected coverage. The coverage of each state *k* has to be weighted by the probability q_k of being in state *k*. The probabilities q_k of having *k* vehicles available of a total number of *n* vehicles are calculated in the same way as in the MECRP with the use of formula (1). The first objective function is therefore stated as in (12).
The second objective function minimises the expected number of relocations. If somebody calls, the system moves from state *k* to state *k-1*, and one of the *k* vehicles has to be dispatched to the call. In order to achieve fast response times the closest vehicle will be chosen. The remaining vehicles may have to be relocated, depending on the chosen waiting locations for *k-1* vehicles, as well as on which vehicle has been dispatched to the call.

Figure 2.1 illustrates a scenario where in state *k* all three waiting stations are staffed by three emergency vehicles. If an emergency occurs in the neighbourhood of waiting station 1 the emergency vehicle of waiting station 1 will be dispatched to the emergency, and the system moves to state *k-1*. As the population density in the vicinity of waiting station 1 is relatively high, it will be beneficial to relocate an emergency vehicle to this location. In this case one vehicle would be relocated. If the emergency occurs in the neighbourhood of waiting station 3, no vehicle will have to be relocated. The population density in the vicinity of waiting station 3 is comparatively low, and a relocation of the other vehicles would not lead to a higher coverage. Therefore the number of relocations is dependent on the requested emergency vehicle. In scenarios with higher numbers of waiting stations more than one relocation might be necessary. In general 0 to k-2 vehicles have to be relocated, depending on the waiting sites chosen for state *k-1* and the requested emergency vehicle.

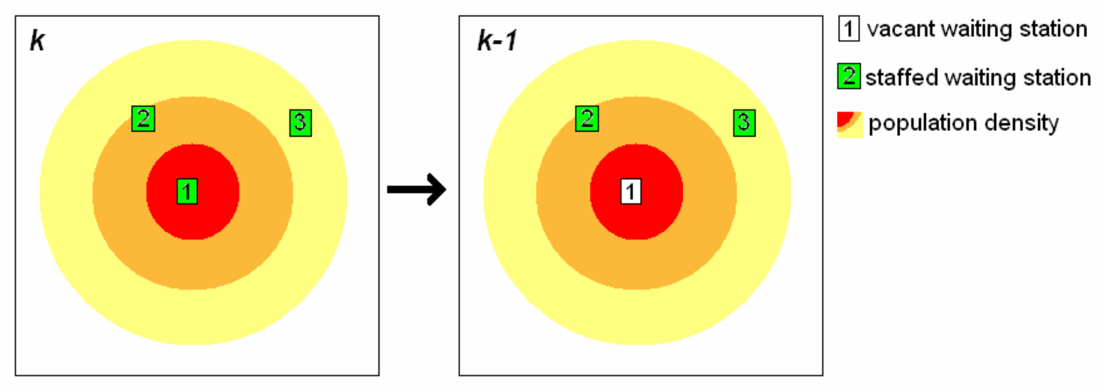

Figure 2.1: Relocation scenario.

The number of relocations that have to be made if the system moves from state *k* to state *k-1*, and the vehicle at $j \in S_k$ is dispatched to a call is given by rel_{jk}. Where S_k is the subset of vertices of *W* chosen to be used in state *k*.

$$rel_{jk} = hamming\ distance(S_k \backslash j, S_{k-1}) \tag{10}$$

Equation (10) gives the number of relocations rel_{jk} as the hamming distance between the waiting sites chosen for state *k* without the dispatched vehicle at *j* and the waiting sites chosen for state *k-1*.

The possibility of different numbers of relocations for each system state change requires the calculation of the probabilities for these different events. In state *k* a maximum of *k* different relocation numbers are possible, because k different vehicles could be dispatched. The probability that a vehicle is dispatched from waiting station $j \in S_k$ in state *k* is said to be dependent on the population of the covered demand points $i \in V_j$. V_j is the subset of vertices of V covered by waiting station $j \in W$. The probability of the different relocation numbers can be given by equation (11).

$$q_{jk}^{rel} = \frac{\sum_{i \in V_j} d_i}{\sum_{j \in S_k} \sum_{i \in V_j} d_i} \tag{11}$$

Note that demand points covered by more than one waiting station are counted repeatedly. For a realistic representation these demand points would have to be assigned to a waiting station. For example the closest waiting station could be chosen. Unfortunately these calculations are computational intensive. Therefore the relaxed model of equation (11) has been implemented. Given the number of relocations rel_{jk} (10) and the according probability q_{jk}^{rel} (11) the second target function can be formulated as the number of vehicles to be relocated between state *k=(2,…,n)* and state *k-1* according to which vehicle is being dispatched, weighted with q_k summed over all states *k* (13).

The mathematic formulation of the MCMRP is as follows:

Maximize $$\sum_{k=1}^{n} \sum_{i \in V} d_i y_{ik} q_k \tag{12}$$

Minimize $$\sum_{k=2}^{n} \sum_{j \in S_k} rel_{jk} q_{jk}^{rel} q_k \tag{13}$$

Subject to: $$\sum_{j \in W_i} x_{jk} \geq y_{ik} \qquad (i \in V, k = 0,...,n) \tag{14}$$

$$\sum_{j \in W} x_{jk} = k \qquad (k = 0,...,n) \tag{15}$$

$$x_{jk} \in \{0,1\} \qquad (j \in W, k = 0,...,n) \tag{16}$$

$$y_{ik} \in \{0,1\} \qquad (i \in V, k = 0,...,n) \tag{17}$$

Constraint (14) assures that vertex $i \in V$ is only covered if at least one vehicle is located in W_i. Constraint (15) regulates the number of vehicles for each state. Constraints (16) and (17) define x_{jk} and y_{ik} as binary values.

The MCMRP addresses the same problematic as the MECRP. The main difference is that the MCMRP uses a bicriterial approach to find a balanced solution with respect to the expected coverage and the number of relocations. Being a multiobjective optimization problem there usually is no single optimal solution. Instead many solutions can be found which can not be rated because of the multiple objectives. In order to find these solutions a multiobjective optimization algorithm can be used. In this work the MCMRP was solved by the Pareto Ant Colony Optimization Algorithm (PACO) Dörner et al. (2002). More information about multiobjective optimization and PACO can be found in chapter 3.

3. Ant Colony Optimization

The algorithm used in this work is a metaheuristic called Pareto Ant Colony Optimization. It is a specialised version of the Ant Colony Optimization metaheuristic, capable of solving multiobjective problems. The main idea of the mechanism of this metaheuristic is derived from the behaviour of real ants. In the following a brief introduction to Ant Colony Optimization and description of the Pareto Ant Colony Optimization Metaheuristic are given.

3.1 Ants

We all have already seen so-called ant streets. Ants build these streets, when they find a rich food source and try to bring it to their nest. Biologists have done research on how the ants manage to build these streets and found that some ant species are capable of finding the shortest path between the nest and the food source. This perception aroused the interest of people with a mathematical background, since the construction of shortest paths is an important problem with applications in many fields.

A good introduction to Ant Colony Optimization was given by Dorigo and Stützle (2004). The main mechanism behind the self-organization of the ants is stigmergy. That is a way of indirect communication by modifications of the environment. The ants deposit pheromones while they are on their search for food. Other ants sense the trails of this chemical and tend to follow these trails. The more pheromone is deposited on a path the more likely it is that other ants will use this path. These simple rules allow ants to find short paths from the nest to the food source. The mechanism can be easily explained by an experiment that was designed by Deneubourg et al. (1990). In the double bridge experiment an ant nest is connected to a food source by bridges with two branches of varying length. By providing different branch lengths the pheromone laying and following behaviour of the ants was studied. In the first experiment a double bridge with branches of equal lengths was used as depicted in Figure 3.1.

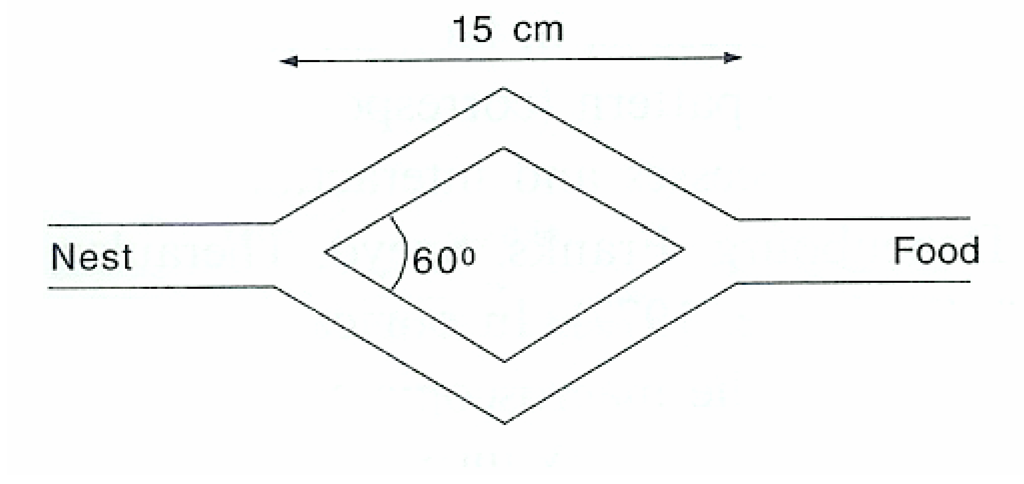

Figure 3.1: Double bridge experiment with equal branch lengths.[1]

When the ants cross the bridge they have to choose one of the branches. Initially they randomly choose each branch with equal probability, because there is no pheromone on the branches. Gradually more and more ants use one of the two branches, until eventually almost all ants use only one branch. Given that both branches are equally attractive, only by random fluctuation more pheromones are deposited on one of the branches. As following ants are attracted by higher pheromone concentrations they tend to choose this branch and strengthen the pheromone trail even more as they use this trail. Repeated experiments show that both branches are chosen with equal probability.

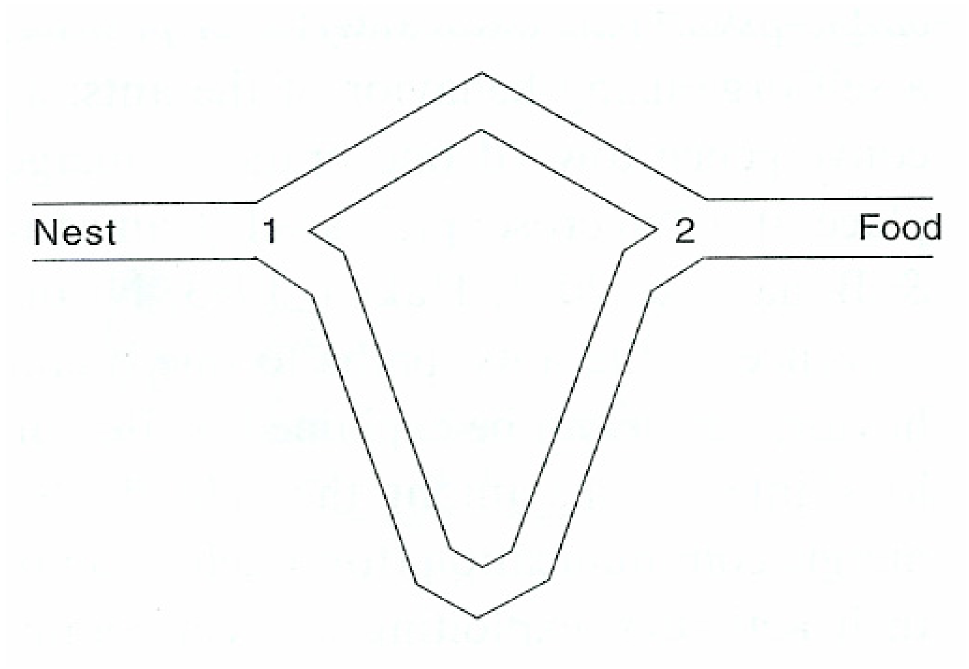

Figure 3.2: Double bridge experiment with different branch lengths.[2]

[1] Source: Dorigo, M., Stützle, T.: “Ant Colony Optimization”, The MIT Press, page 3, 2004

In the next step of the experiment the ants are confronted with different branch lengths as depicted in figure 3.2. The long branch is twice as long as the short branch. Again the ants are initially indifferent between the two branches and choose each branch with equal probability, because there is no pheromone. But after some time the majority of the ants uses the shorter branch. This happens, because the ants using the short branch are much faster and therefore are the first to reach the food source. When they return to the nest they have to decide between the two branches again. But this time there is more pheromone on the short branch, which leads the ants to use the short branch again and deposit even more pheromone on it. After some time the ants travelling on the short branch will have deposited much more pheromones and the majority of the ants will follow this pheromone trail. Interestingly even then some ants use the long branch. This can be interpreted as some kind of "path exploration", which should enable them to find new shorter paths.

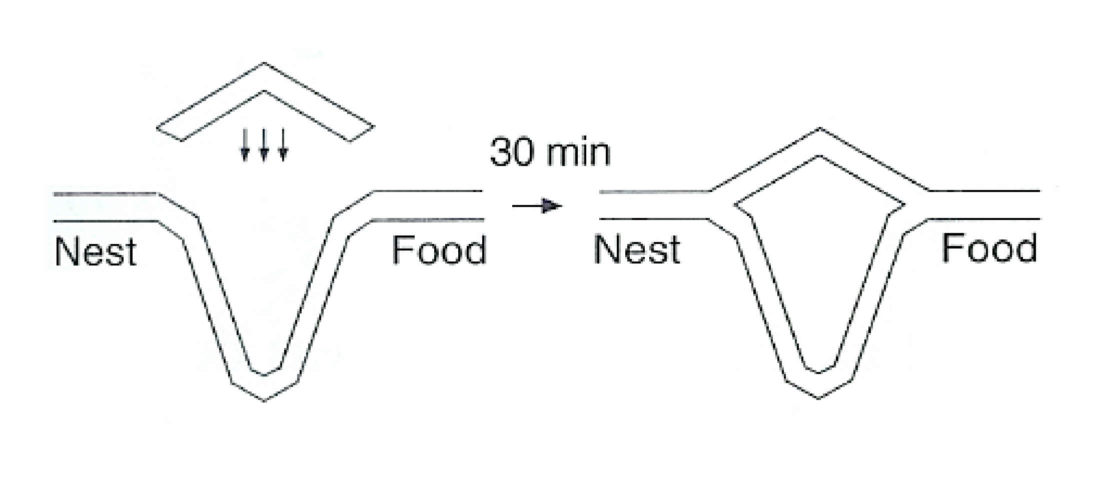

Figure 3.3: Double bridge experiment with added branch.[2]

A third step of the experiment investigated if the ants can use this "path exploration" behaviour to switch to a new shorter connection after convergence. The setting is depicted in Figure 3.3. First the ants are offered only one bridge, and after 30 minutes a shorter branch is added. It can be observed that the newly added branch is used only sporadically, and the ants do not switch to the shorter path. This is because, when the short branch is added, the majority of the ants is attracted by the strong pheromone trail and thus reinforces it. The evaporation rate of the pheromones is too slow, as to "forget" the path the ants have converged to.

[2] Source: Dorigo, M., Stützle, T.: "Ant Colony Optimization", The MIT Press, page 5, 2004

The observations of the ant behaviour led to algorithms that mimic the ant optimization capability. First ant algorithms were used to understand the mechanism that controls the ant behaviour and therefore simply imitated the ant behaviour. To successfully apply this solution strategy on more complex problems, it is necessary to add some more rules to the behaviour of the artificial ants. The third part of the double bridge experiment showed that the evaporation of the pheromones did not play a significant role for real ants. Artificial ants benefit from higher evaporation rates as they allow more exploration of the solution space, as a prerequisite to find local optima. The general higher complexity of the problems tackled with ant algorithms gives rise to more additions to the ant behaviour. On complex graphs the formation of loops raises a serious problem. The pheromones lead to a reinforcement of these loops, preventing good solutions. In order to avoid loops it is necessary to give the artificial ants some memory. If the ants memorize their path, the loops can be removed from the path before the pheromone is deposited. To further improve the convergence performance of ant algorithms it is possible to use global information. For instance the amount of pheromone that is deposited by an ant can be dependent on the solution quality. The best solutions can be found more quickly by using this information. Alternatively only the best of some simultaneously running ants can be allowed to deposit pheromones. These are only some of the extensions that could be used to build a high performing ant algorithm. The Ant Colony Optimization metaheuristic explained in the next section is one of the most successful examples of ant algorithms.

3.2 Ant Colony Optimization

The Ant Colony Optimization Metaheuristic is inspired by the foraging behaviour of real ants described in the previous chapter. It is a general pattern of a heuristic that can be applied to a wide range of problems. Being an approximate algorithm the loss of the certainty to attain optimal results is accepted in return for efficiency. This makes the algorithm suitable for *NP*-hard problems, where exact algorithms are often limited to small instances. The algorithm as described by Dorigo and Stützle (2004) mainly consists of three procedures: ConstructAntsSolutions, UpdatePheromones and DaemonActions.

3.2.1 ConstructAntSolutions

The ConstructAntSolutions process controls how “ants” (agents) independently construct solutions. As the ants move over the problems construction graph they randomly build solutions. If a node has been visited it means that the solution variable represented by this node is included in the solution. For instance in the case of a Travelling Salesman Problem, a visited node represents a visited city. The movement of the ants is stochastically decided and influenced by heuristic and pheromone information. Heuristic information represents a priori information about the problem instance, or runtime information provided by a source different from the ants. For example the heuristic information can give the cost to include the next node into the solution. The heuristic and pheromone information is stored on the arcs connecting the nodes. When the ant decides what node to visit next, it “senses” the information on the arcs and decides randomly according to the information. Arcs with higher amounts of pheromone and better heuristic values have a better chance to be included in the solution. The final (in some cases partial) solution is then evaluated and the information passed to the next process UpdatePheromones.

3.2.2 UpdatePheromones

In the UpdatePheromones process the pheromone trails are modified. The information of the solution quality is used to decide where to deposit more pheromone. In addition a rate of pheromone evaporation is implemented. This enables the algorithm to forget old solutions and explore new areas of the search space to find better solutions. Otherwise the algorithm could converge too fast to a suboptimal solution, because the first randomly found solutions would have a strong influence.

3.2.3 DaemonActions

The DaemonActions process is optional. The converging mechanism of the ants often can be improved by the addition of external operations. With the help of daemon actions global information can be used to guide the optimization process. For instance only the best ants can be allowed to deposit pheromones. Or the amount of pheromones can be dependent on the solution quality. Daemon actions can be used to decide when to activate local optimization procedures.

```
procedure ACOMetaheuristic
    ScheduleActivities
        ConstructAntSolutions
        UpdatePheromones
        DaemonActions…%optional
    end-ScheduleActivities
end-procedure
```

Figure 3.4: The ACO metaheuristic in pseudo-code.[3]

The basic framework of the Ant Colony Metaheuristic is depicted in figure 3.4. The ScheduleActivities construct organises the cooperation of the three parts of the Ant Colony Optimization algorithm. How these three processes are scheduled and synchronised is up to the designer, and should be adapted for the specific problem characteristic.

3.3 Pareto Ant Colony Optimization

The Pareto Ant Colony Optimization Metaheuristic is a variant of the Ant Colony Optimization Metaheuristic specially designed for multiobjective optimization problems. This metaheuristic approach has been introduced by Dörner et al. (2002) for a multiobjective portfolio selection problem. The challenge of multiobjective optimization problems is that competing objectives usually avoid the existence of one single optimal solution. Several solutions of undistinguishable quality may be possible. In order to overcome the difficulty of deciding between several solutions, it is possible to translate the problem under consideration into a singleobjective problem. By the assignment of weights to the different objectives one single objective function can be built. Unfortunately in most cases it is very difficult for the decision maker to give meaningful weights to represent his preferences. Therefore it is an advantage to calculate the pareto-optimal front, and then select one of these solutions.

[3] Source: Dorigo, M., Stützle, T.: "Ant Colony Optimization", The MIT Press, page 38, 2004

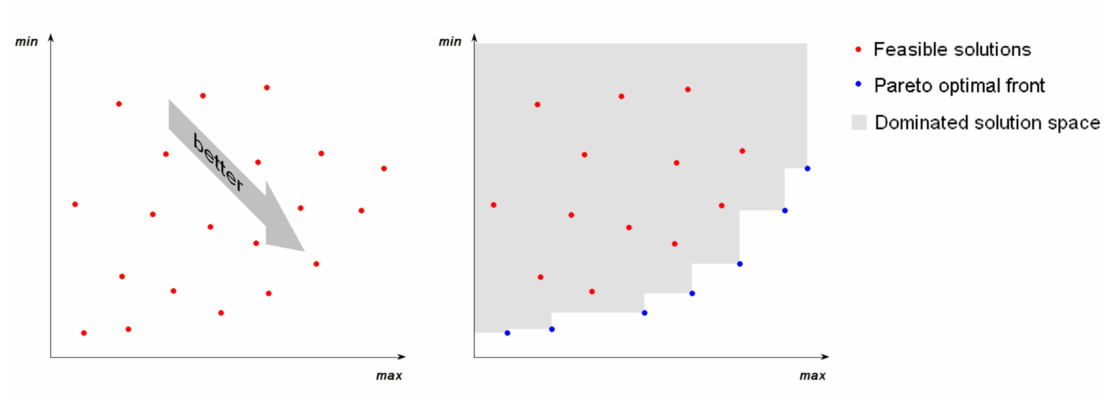

Figure 3.5: The pareto-optimal front.

The pareto-optimal front as depicted in figure 3.5 consists of all non-dominated solutions of the solution space. A solution is said to be dominated, if there exists another solution that is superior in at least one objective, while not being inferior in the other objectives. In other words dominated solutions are inferior to solutions who dominate them. On that account dominated solutions can be excluded from the set of possible best solutions. The remaining solutions which dominate all other possible solutions and do not dominate each other are called efficient or pareto-optimal. The solutions of the pareto-optimal front provide a reasonable trade-off between the objectives.

The calculation of the efficient solutions is not trivial. Larger instances of problems cannot be calculated by brute force within acceptable time. Therefore heuristic approaches like Genetic Algorithms, Simulated Annealing, Tabu Search and Pareto Ant Colony Optimization have been adapted for multiobjective combinatorial optimization problems.

The solution finding process of PACO basically relies on the same principles as ACO. Ants create solutions randomly and influenced by pheromone trails. The pheromones are applied according to the solution quality. The ability to get good solutions for multiple objectives is achieved by the use of multiple pheromone trails. For each objective a corresponding pheromone matrix is established. These matrices are weighted randomly and combined into a single pheromone matrix that is then used for the solution creating process. The weighting of the objectives is necessary to enable the procedure to converge to an optimum. The constructed solutions are evaluated with respect to all objectives and the pheromone deposits are made according to the weights. After a defined number of iterations the algorithm stops the converging process for the set weighting and repeats the procedure with different

random weightings. The converging process has to be repeated with different random weightings in order to find a good approximation set of the pareto optimal front.
In the next chapter the PACO Algorithm and some variants used in this work to solve the MCMRP are presented.

4. Algorithms

This section gives an introduction to the different algorithms which were employed for the ambulance relocation problem. Chapter 4.1 presents the basic structure common to all variants. Different pheromone structures as well as a version featuring a smaller search space were used. These algorithms are presented in chapter 4.2 and 4.3 respectively. Further enhanced versions with shifting ant numbers are described in chapter 4.4.

4.1 Basic Structure

All versions of the algorithm feature the same basic structure that follows the principles of the PACO Metaheuristic presented in Dörner et al. (2002). A high-level representation of a PACO algorithm is given in pseudo code in figure 4.1.

```
procedure PACO
    for Period = 1 to Π
        SetWeights
        InitialisePheromones
        for Iteration =1 to M
            for Ant = 1 to s
                CreateSolution
                EvaluateSolution
            end-for
            FindBest
            PheromoneUpdate
            AddSolution
        end-for
    end-for
end-procedure
```

Figure 4.1: The Pareto Ant Colony Optimization in pseudo-code.

The SetWeights construct defines the weights of the objectives randomly and different for each period. In each period the problem is solved according to a fixed weighting. Only the repeated calculation with different weightings allows the finding of a good approximation set. The pheromone structures have to be reset for each period by the InitializePheromones construct. The number of iterations M defines how long the algorithm searches with one set of weights. The CreateSolution construct simulates the run of the ant. As the path is constructed by the ant, the solution is put together. Each solution is evaluated with respect to the weightings of the objective by the EvaluateSolution construct. The FindBest construct compares the *s* solutions constructed in each iteration. The best solution is stored for later use with the PheromoneUpdate and AddSolution construct. In the PheromoneUpdate procedure the evaporation and deposit of pheromones is administered according to the found solution. The AddSolution procedure integrates the solution to the approximation set and sorts out the dominated solutions. The final outcome of the PACO procedure is a set of solutions which dominate all other found solutions but do not dominate each other. This set is called approximation set as it is an approximation of the pareto optimal front.

4.2 Pheromone Structures

The pheromone structures represent the directed graph on which the ants can move in order to build solutions. Therefore the modelling of the pheromone structures is determined by the way the solutions are to be constructed. The structure can be designed freely, but the solution construction process must be able to create all possible solutions. The performance of the algorithm and the information stored by the pheromones can be altered by the design of the pheromone structures. In the following two methods that were compared in this work will be presented.

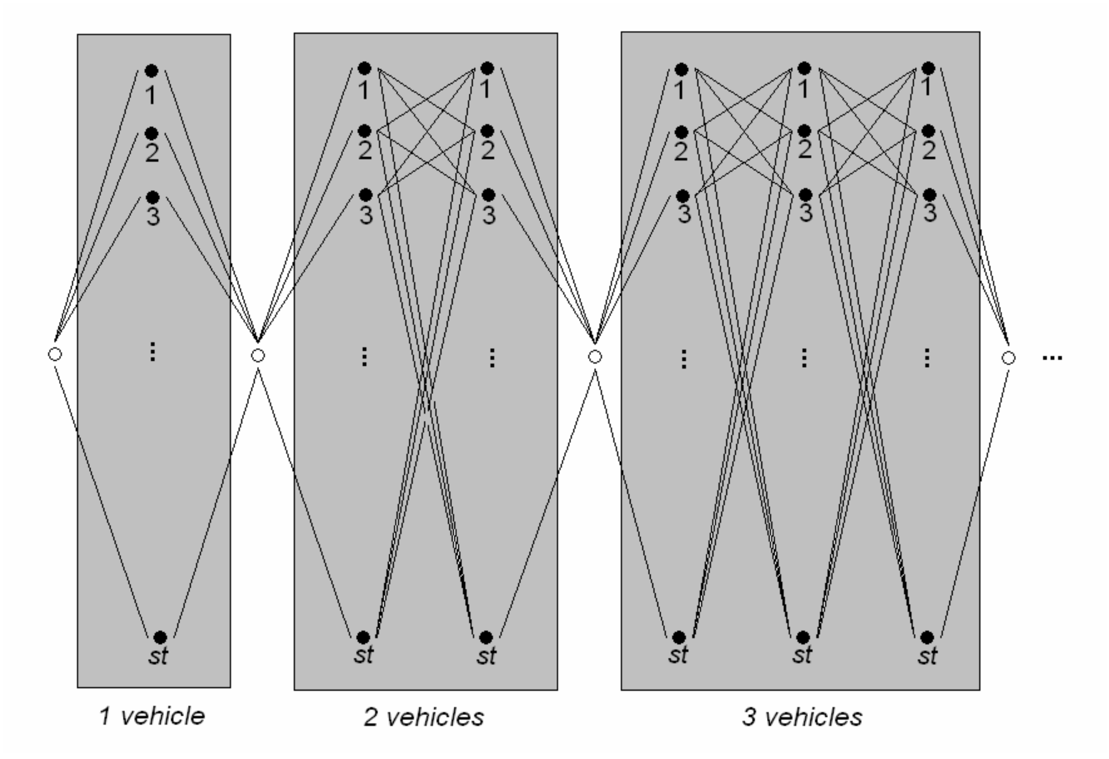

Figure 4.2: Pheromone Graph A.

Figure 4.2 depicts the pheromone structure version A. Here the ant starts on the node to the left. By moving on to another node the ant randomly selects one station for state *k=1*. In states where *k>1* the ant subsequently adds stations to the solution until *k* stations have been selected. Note that stations can only be selected once in each state. Therefore the corresponding nodes of chosen stations have to be removed from the graph in order to avoid multiple assignments. Between the states the ant moves to a blank-point before it starts the selection process for the next state. The trail of the ant is represented by the connections between the chosen stations. As the pheromones have to be deposited on this trail, the pheromones can be given in matrices with the following structure.

$$\begin{bmatrix} \tau_1^{(1)} \\ \vdots \\ \tau_{st}^{(1)} \end{bmatrix};\begin{bmatrix} \tau_{1,1}^{(2)} & \cdots & \tau_{1,st}^{(2)} \\ \vdots & & \vdots \\ \tau_{st,1}^{(2)} & \cdots & \tau_{st,st}^{(2)} \end{bmatrix};\begin{bmatrix} \tau_{1,1}^{(3,1)} & \cdots & \tau_{1,st}^{(3,1)} \\ \vdots & & \vdots \\ \tau_{st,1}^{(3,1)} & \cdots & \tau_{st,st}^{(3,1)} \end{bmatrix}\begin{bmatrix} \tau_{1,1}^{(3,2)} & \cdots & \tau_{1,st}^{(3,2)} \\ \vdots & & \vdots \\ \tau_{st,1}^{(3,2)} & \cdots & \tau_{st,st}^{(3,2)} \end{bmatrix};\ldots$$

These matrices represent the graph in figure 4.2. An interesting point of this way to design the pheromone structure is that each element of the matrix stores the information of two stations to be selected. The station the ant is coming from and the station it is moving to. This gives the algorithm the ability to mark awarding combinations of stations. For the relocation problem at hand this feature can be beneficial. It is very likely that some combinations of stations persistently contribute to high solution quality, when they are implemented in different solutions.

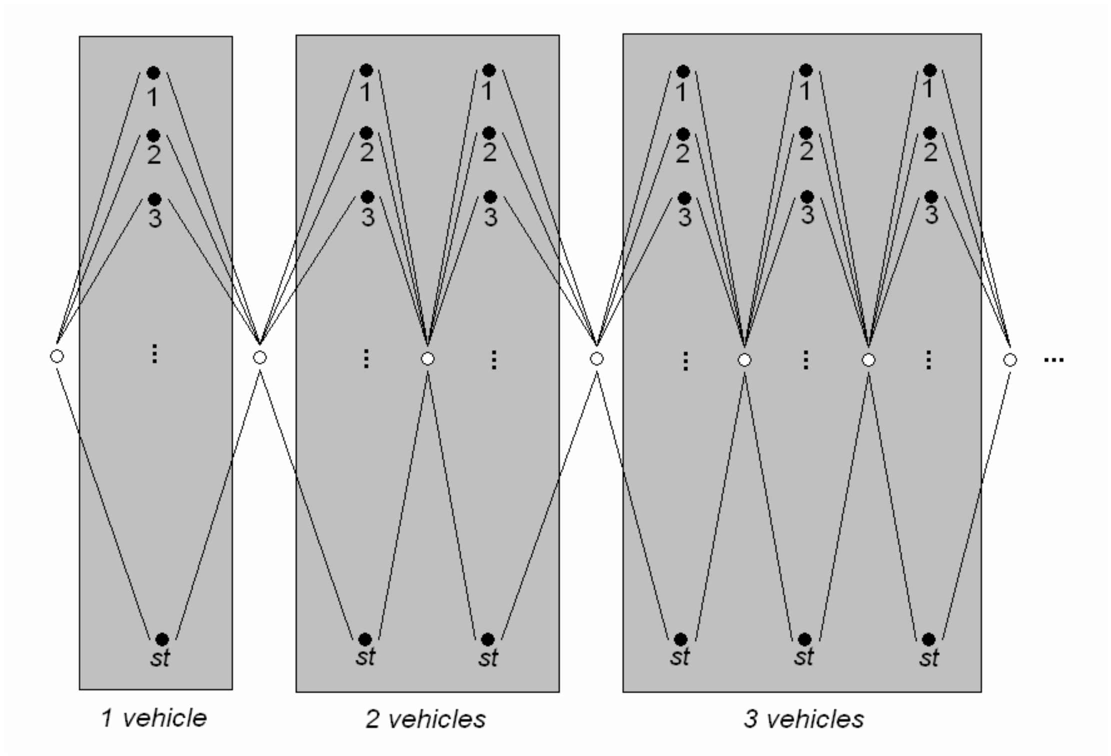

Figure 4.3: Pheromone Graph B.

The second variant of the pheromone structures is depicted in Figure 4.3. Here the ant returns to a blank-point after each selection of a station. Again the corresponding nodes of already visited stations have to be removed from the graph in order to avoid multiple assignments in each state. In difference to the first version the trail of the ant is not given by the connecting arcs. Instead the trail simply consists of the "visited" stations. The pheromone structure can be represented by a combination of vectors.

$$\begin{bmatrix} \tau_1^{(1)} \\ \vdots \\ \tau_{st}^{(1)} \end{bmatrix}; \begin{bmatrix} \tau_1^{(2,1)} \\ \vdots \\ \tau_{st}^{(2,1)} \end{bmatrix}, \begin{bmatrix} \tau_1^{(2,2)} \\ \vdots \\ \tau_{st}^{(2,2)} \end{bmatrix}; \begin{bmatrix} \tau_1^{(3,1)} \\ \vdots \\ \tau_{st}^{(3,1)} \end{bmatrix}, \begin{bmatrix} \tau_1^{(3,2)} \\ \vdots \\ \tau_{st}^{(3,2)} \end{bmatrix}, \begin{bmatrix} \tau_1^{(3,3)} \\ \vdots \\ \tau_{st}^{(3,3)} \end{bmatrix}; \dots$$

Evidently this version is simpler in the way that it only stores the information of the used stations. But the higher information capacity of version A comes at a price. The pheromone structure of version A consists of $st + \sum_{i=1}^{vh-1} i \times st^2$ elements, while version B only needs $\sum_{i=1}^{vh} i \times st$ elements. The exponential growth of the number of elements of version A leads to a significant difference even for small instances. The problem used in this work to evaluate the algorithms features twelve stations (*st=12*) and four vehicles (*vh=4*). The numbers of elements of the pheromone structure are therefore 876 and 120 respectively. Hence the question is whether the use of additional information is worth the trouble to manipulate a complex pheromone structure, or if it is favourable to keep the structure simple in order to speed up the calculations.

4.3 Search Space

This chapter describes the improvement of the solution construction process. As illustrated in the last chapter about pheromone structures, all possible solutions can be built with the solution construction process. However the freedom of movement enables the ants to create identical solutions with the use of different paths. This happens because the waiting stations can be selected in different orders. But the order of the waiting stations is irrelevant for the solution. If all permutations of the solutions are treated as varying, the algorithm has to choose between far more solutions. This effect amplifies for higher instances of the problem. The number of solutions is given by $\begin{pmatrix} st \\ vh \end{pmatrix}$, while a solution space including all permutations of the solutions differentiates between $\begin{pmatrix} st \\ vh \end{pmatrix} vh!$ solutions.

Take a small example with four waiting stations and three vehicles. Here four different solutions are possible in state *k=3* (locate three vehicles).

$$\begin{array}{lccc} Sol1: & 1 & 2 & 3 \\ Sol2: & 1 & 2 & 4 \\ Sol3: & 1 & 3 & 4 \\ Sol4: & 2 & 3 & 4 \end{array}$$

As you can see the solutions are all sorted in ascending order. The idea is to build the solutions in ascending order during the solution creation process, and therefore avoiding the permutations of the solutions. In this example three out of four solutions begin with station 1 and one solution with station 2. When the ant selects the first station it will choose randomly. The probability to select station 1 has to be risen by $\frac{3}{4}$ because three out of four solutions begin with station 1. Station 2 has to be selected with a probability of $\frac{1}{4}$. Stations 3 and 4 can't be selected in the first step. The selection of the second and third station can be done analogously. Figure 4.4 shows the probabilities that have to be applied during the selection process. The probability to select a station is dependent on the number of solutions that can be built with this station on this place of the solution. All four solutions will be chosen with equal probability if these probabilities are used during the selection process.

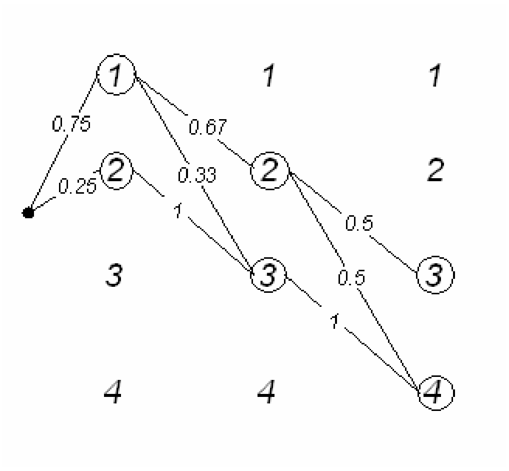

Figure 4.4: Conditional probabilities for the selection process.

These probabilities are computed in advance and stored in an array *ps*. *ps(i,j,k)* gives the number of possible solutions if station *i* is chosen, for the *j*-th vehicle to assign, if *k* vehicles are to be assigned. In the CreateSolution process these probabilities are applied to the pheromone matrices to control the solution construction.
The solution space of the problem used in this work to evaluate the algorithms (st=12, vh=4) could be reduced from 11880 to 495 solutions. This reduction should speed up the convergence behaviour, as fewer solutions have to be compared.

4.4 Shifting Ant Number

During preliminary testing of the algorithm a common characteristic could be detected. While the algorithm is able to find good approximation sets of the pareto optimal front in a short time, the actual solutions of the pareto optimal front seem to be hard to find.

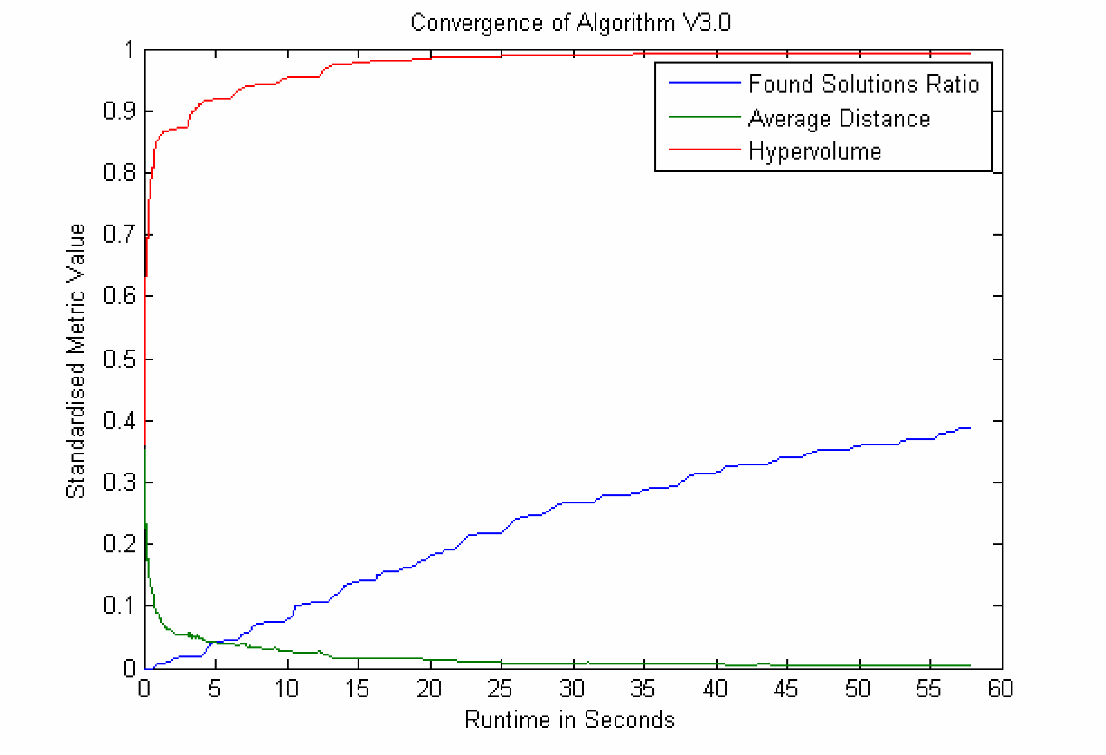

Figure 4.5: Convergence of algorithm V3.0.

Figure 4.5 shows a graph depicting the convergence behaviour of algorithm version 3.0. The coloured lines show the average metric values of 20 runs of approximately 60 seconds. Three metrics were used to measure the quality of the approximation sets. The found solutions ratio gives the percentage of found solutions of the pareto optimal front. The average distance metric measures the average distance of the solutions to the pareto optimal front. The hypervolume metric indicates the size of the region of the solution space that is weakly dominated by an approximation set. More information about the metrics used in this work to evaluate the solution quality can be found in chapter 5. Higher values of hypervolume and found solutions ratio and lower values of average distance indicate better approximation sets. The graph shows that after a runtime of 20 seconds good approximation sets have been found. The hypervolume and average distance metric values can only increase slightly with extended runtime. The found solutions ratio increases far slower, finding less than 40 percent of the solutions of the pareto optimal front after 60 seconds of runtime. It seems as the approximation set converges very fast close to the pareto optimal front, but the solutions of the pareto optimal front are hard to find. This does not seem to be a problem of local optimality, as tests with high values for the pheromone evaporation parameter did not lead to better results. (Confer Appendix C: Determination of the parameters.) A raise in the number of ants used by the algorithm helps to find more solutions of the pareto optimal front, but this increases the runtime directly proportional.

The idea to overcome this problem was to increase the number of ants in the later stages of the algorithm, when the found approximation set is close to the pareto optimal front. This should enable the algorithm to intensify the search when better solutions are hard to find. The idea is transposed by linearly increasing the number of ants during the run of the algorithm. A minimum of three ants is used in order to maintain the proper functionality of the algorithm. The number of ants is determined for each iteration by equation 18. *s* is a definable parameter giving the maximum number of ants. *M* gives the number of iterations.

$$AntNr = \max(3, \left\lceil \frac{iteration}{\frac{M}{s}} \right\rceil) \qquad (18)$$

Interestingly this modification led to clearly inferior results, as will be shown in chapter 6.1.3. A reason for the bad performance may be the small number of ants in the early stages of the algorithm. With only few ants exploring the search space, many pheromones are deposited on relatively poor solutions misdirecting the algorithm. This showed that it is important to find good solutions in the early stages of the algorithm. The principle was therefore reversed and the algorithm modified in the way that the number of ants should be decreased linearly. Equation 19 shows how the number of ants used for algorithm version 3.2 is calculated.

$$AntNr = \max\left(3, s + 1 - \left\lceil \frac{iteration}{\frac{M}{s}} \right\rceil\right) \qquad (19)$$

The good results of algorithm version 3.2 confirm the assumption that intensified search in the early stages of the algorithm helps to direct the algorithm towards good solutions. A detailed evaluation of the results is given in chapter 6.

5. Metrics of Performance

For multiobjective optimization problems it is substantially more complex to define the quality of approximated solutions than for single-objective optimization problems. The reason for this is the usual absence of a single optimal solution. Instead the results of multiobjective generally consist of a whole set of incomparable solutions, where each solution is superior to the others in some objectives and inferior in other objectives. These solutions which dominate all other possible solutions but do not dominate each other are called the pareto-optimal set. The aim of multiobjective optimization algorithms is to find this set or a good approximation of it. In order to assess multiobjective optimization algorithms some tool to rate the approximation sets is needed. As can be seen in figure 5.1 this task can be difficult.

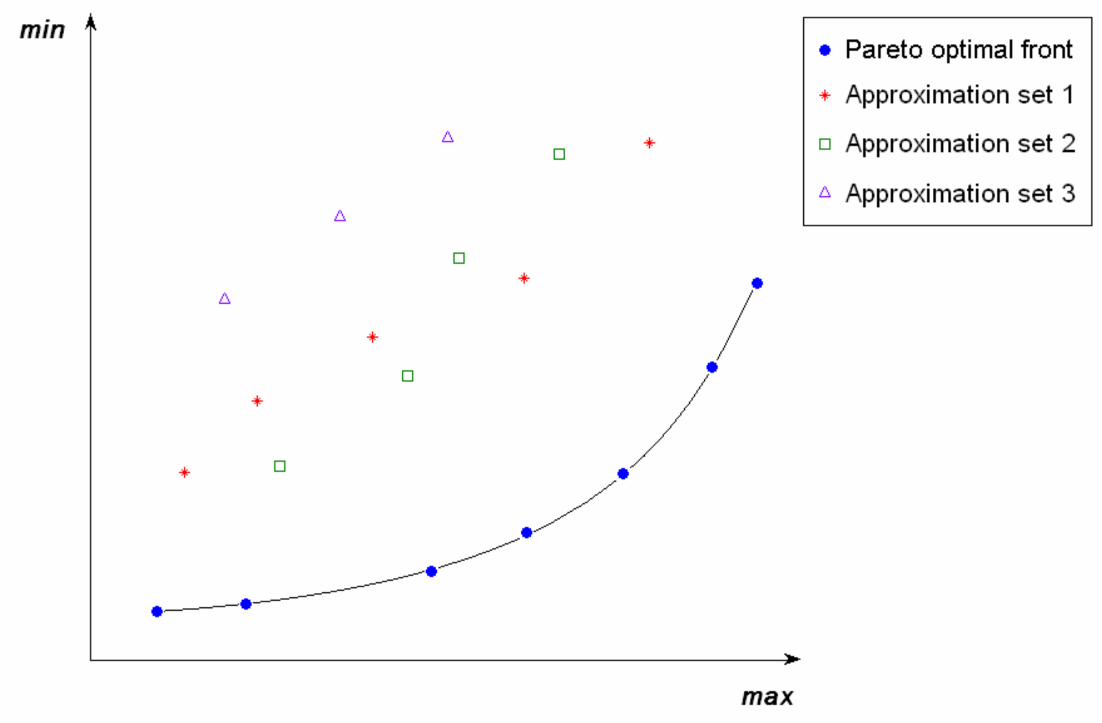

Figure 5.1: Approximation sets and the pareto optimal front.

Figure 5.1 depicts a two-dimensional problem with maximization on the x-axis and minimization on the y-axis. So solutions to the lower right dominate those above and left of them. It is difficult to say whether approximation set 1 or 2 is better, because they both partly dominate each other. It seems to be easier to compare approximation

set 3 to the others, because it is dominated by them. So approximation set 3 can be stated as worse, jet it is not clear how much better the other sets are and in what aspects.
Unfortunately there is no common agreement on standards for the quality evaluation. A popular solution to this problem is to calculate unary quality measures. Such measures assign a number that reflects a certain quality aspect of an approximation set. Usually some of these measures are combined in order to cover the different aspects. There are also binary quality measures, which assign numbers to pairs of approximation sets. Another solution is the attainment function approach trying to estimate the probability to obtain arbitrary goals in objective space from multiple approximation sets. Zitzler et al. (2002) showed that existing unary quality measures can fail in indicating whether an approximation set is better than another, even if several of them are used. They proposed a binary quality measure that is able to display which approximation set is superior according to dominance relations. But the better indication ability comes at the cost of additional complexity. The number of indicator values to be considered is quadric in the number of approximation sets as opposed to linear for unary indicators.

According to Zitzler, Thiele and Deb (2000) the optimization goal consists of multiple objectives:

- "The distance of the resulting nondominated set to the pareto-optimal front should be minimized.
- A good (in most cases uniform) distribution of the solutions found is desirable. The assessment of this criterion might be based on a certain distance metric.
- The extent of the obtained nondominated front should be maximized, i.e., for each objective, a wide range of values should be covered by the nondominated solutions."

Therefore it was decided to use a set of unary quality measures to evaluate the solutions of the algorithm versions. The three metrics of performance that were implemented in this work are described in the following.

5.1. Found pareto-optimal Solutions Ratio

The best possible approximation of the pareto-optimal front is the pareto-optimal front itself. So an intuitive measure is to look at how many solutions of the pareto–optimal front were found. This metric calculates the quotient of the amount of solutions of the approximation set A, which are also part of the pareto-optimal front P, to the total amount of solutions in P. (cf. Jaszkiewicz, 2004)

$$M_1(A) = \frac{|A \cap P|}{|P|} \tag{20}$$

The value of this metric can range from 0 to 1, With 0 indicating that no solution of the pareto-optimal front was found, and 1 indicating that all solutions of the pareto-optimal front were found. It can easily be seen that higher values are desirable. Nevertheless it is more important to pattern the distribution of the pareto-optimal front, instead of knowing all the solutions of the pareto-optimal front. In the end the decision maker has to select one of the solutions according to his weightings of the objective functions.

This metric is easily interpreted and shows a small computational complexity. The main disadvantage is that the pareto-optimal front has to be known. So the use of this metric is restricted to relatively small problems. It also has a weakness in the early stages of an Algorithm, when typically relatively bad approximation sets are found. The metric does not differentiate between approximation sets without any solutions of the pareto-optimal front, thus giving no indication of solution quality until the late stages of the algorithm, when the pareto-optimal front is reached.

5.2. Average Distance

This metric calculates the average distance from each solution **a** of the approximation set **A** to the closest solution **p** of the pareto-optimal front **P**. The metric is formulated by Zitzler et al. (2000) as following:

$$M_2(A) = \frac{1}{|A|} \sum_{a \in A} \min\{\|a - p\|; p \in P\} \tag{21}$$

Where || · || denotes a distance metric. Different methods can be used to calculate the distance between two solutions. For the bidimensional case in this work the Euclidean distance was used. It is necessary to normalise the objective values, if the objectives have different ranges. Otherwise some objectives would have a bigger impact on solution quality than others.

In general lower values of this metric indicate smaller distances, with a value of 0 meaning that the approximation set equals the pareto-optimal front. But lower values do not necessarily mean better solutions, as even dominated solutions can have lower distances. Very differently distributed approximation sets may have the same distance value. Jaszkiewicz (2004) pointed out that regions containing many solutions have a bigger impact on the average distance than regions containing fewer solutions. Interestingly the addition of another pareto-optimal solution to an approximation set can result in worse distance values, although this can only lead to an improved approximation set. The convergence graphs in appendix D show this characteristic. One of the strengths of this metric is that unlike the first described metric it gives insight in the convergence process of an algorithm even in the early stages. The shortcomings of the average distance metric give rise to the need to combine this metric with other metrics in order to get a good estimation of solution quality.

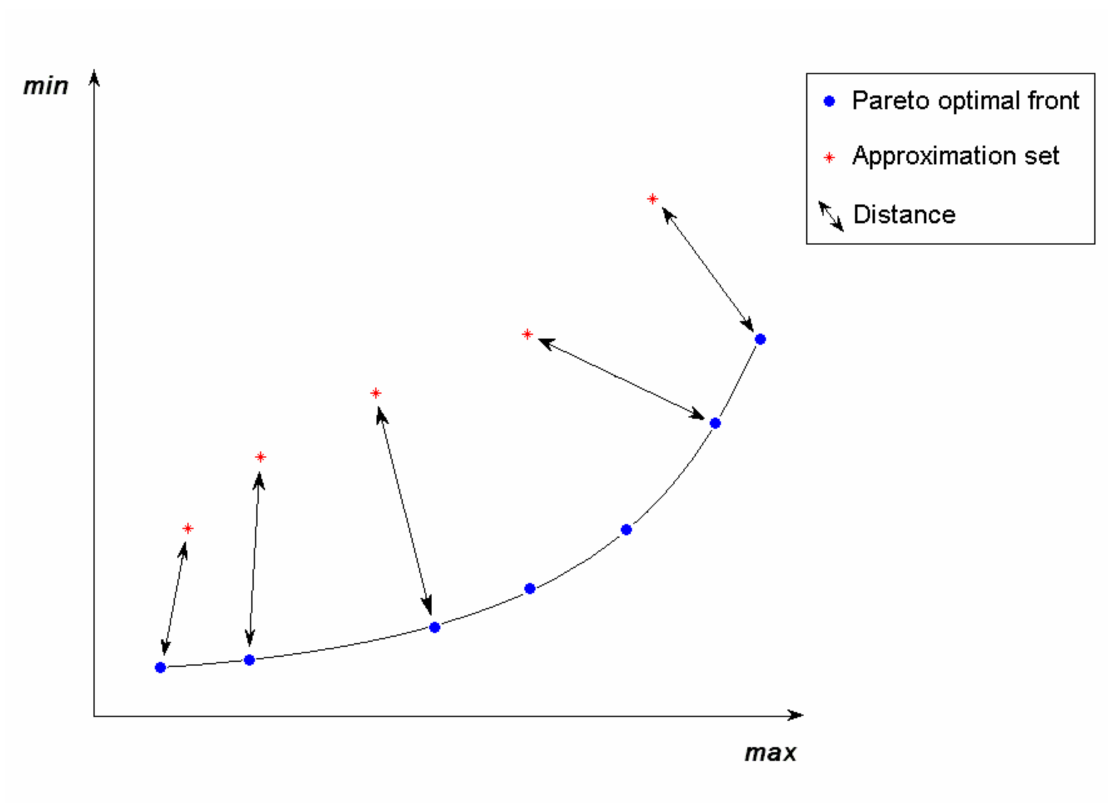

Figure 5.2: The distance metric.

Figure 5.2 depicts a two-dimensional problem with maximization on the x-axis and minimization on the y-axis. Solutions to the lower right dominate those above and left of them. The arrows illustrate the distances between the solutions of the approximation set and the closest solutions of the pareto-optimal front.

5.3. Hypervolume

This metric was first introduced by Zitzler and Thiele (1998). The hypervolume indicates the size of the region of the solution space that is weakly dominated by an approximation set. Each solution of the approximation set dominates a region of the solution space. The volume of the weakly dominated region is calculated from a hypercube that spans between the solution and a reference point. The volume of the merged hypercubes of all solutions gives the hypervolume. Nebro et al. (2006) defined the hypervolume in the following way:

$$M_3 = volume\left(\bigcup_{a=1}^{|A|} v_a\right) \tag{22}$$

Where $\mathbf{v_a}$ gives the volume of the region of the solution space that is dominated by solution **a** of the approximation set **A**.

The two-dimensional case is displayed in figure 5.3. Here the objective on the x-axis is to be maximized and the objective on the y-axis is to be minimized. Solutions weakly dominate the area above and left of them. The grey area between the solutions of the approximation set and the reference point illustrates the hypervolume. The graph points out that higher hypervolume values (bigger grey area) indicate better solution quality.

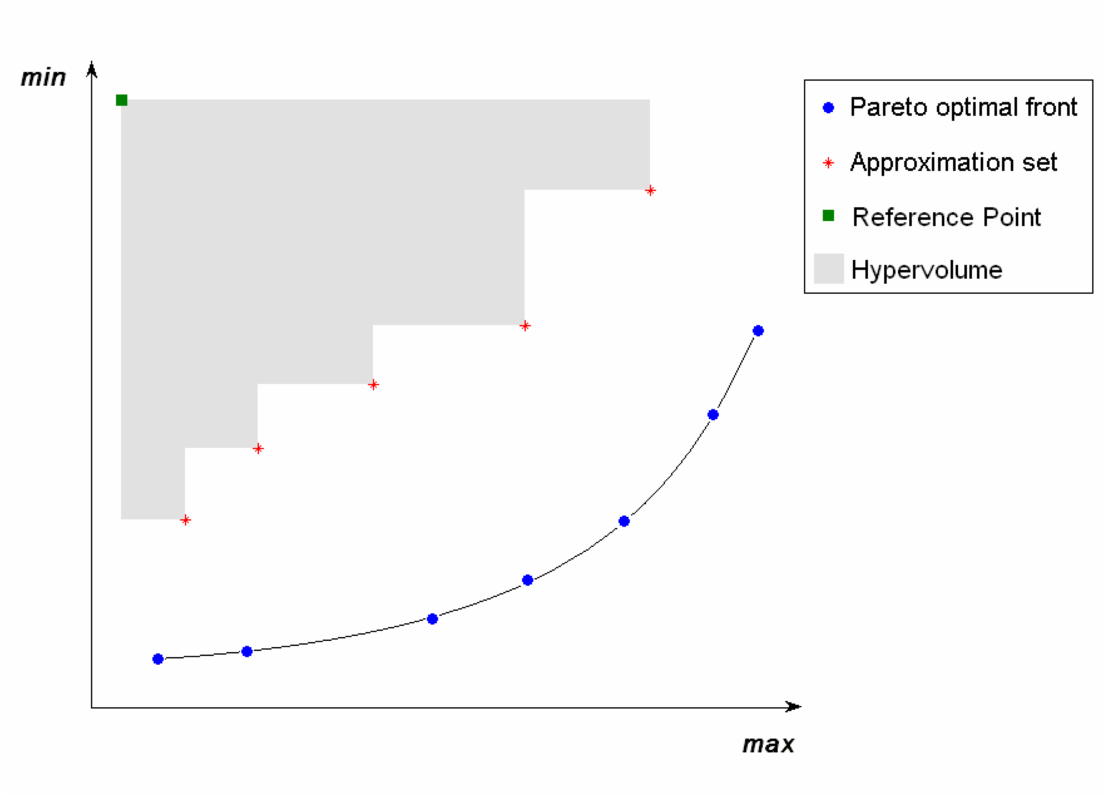

Figure 5.3: The hypervolume metric.

The selection of the reference point is an important step, as different reference points can change the ordering of pairs of incomparable approximation sets. Fonseca et al. (2005) suggest to use the bounds of the objective space if they are known. Otherwise it is possible to concatenate all approximation sets and compute the nadir point. Then the nadir point has to be shifted in order to be strictly dominated by all points in the approximation set. This point can then be used as the reference point. In this work the worst case values of the objectives were used to build the reference point.

Zitzler et al. (2002) stated that hypervolume was the only unary indicator (they were aware of) that would hold for $\ntriangleright$-compatibility and $\triangleright$-completeness. $\ntriangleright$-compatibility means that, if the metric indicates that solution A is better than solution B we know that B is not better than A according to dominance relations. $\triangleright$-completeness means that all cases where A is better than B are detected by the indicator. This is a valuable quality, because many unary indicators have the problem that they may evaluate approximation set A as better than B, even if B actually is superior according to dominance relations. Another strength of the hypervolume metric is the ability to comprise the diversity of an approximation set. The hypervolume can be calculated fast in the two-objective case, but large numbers of objectives implicate high computational cost.

6. Comparison and Results

In this chapter the different algorithms that were applied to the ambulance relocation problem are compared, and the results of the Viennese relocation problem are presented. In addition the important parts of the procedure are explained to ease the understanding of the results.

6.1 Comparison of the Algorithms

The different versions of the algorithms were already tested during the development. The findings of these first tests led to new ideas and even more versions. However for a meaningful comparison the versions have to be tested under the same circumstances. In this case this means that, all versions use the same runtime, and the optimal parameters for this runtime have to be found by a standardised procedure. This procedure is described in detail in appendix C. It is important to adjust the parameters for every version, because changed convergence characteristics can require different parameter settings in order to fully benefit from the changes. The calculations for the comparison were conducted on a model of smaller scale due to time constraints. In this model up to four vehicles (vh=4) have to be located on twelve stations (st=12) to cover the demand of 400 demand points.
The results are illustrated in three boxplots, one for each metric. The underlying data was gathered in twenty runs of sixty seconds for each algorithm version. The meaning of the metrics is described in chapter 5. High values of the found solutions ratio and the hypervolume as well as low values of the average distance indicate good solution quality. The biggest differences can be seen in the found solutions ratio. Unfortunately this metric offers the lowest solution quality indication performance. But together with the other two metrics, which offer good solution quality indication, the different versions of the algorithm can be evaluated. In addition the convergence behaviour of the algorithms can be analysed to assess the algorithm. The graphs depicting the convergence behaviour of the different algorithm versions are given in appendix D.

6.1.1 Pheromone Structures

Algorithm versions 1 and 2 differ in the way the pheromones are structured. Version 1 uses a pheromone structure able to store more information, but this ability comes at the price of higher complexity. In Version 2 the number of elements of the pheromone structure is minimized reducing the complexity of the structure. This way only necessary information is stored. The results show that the additional information stored in version 1 is actually being used by the algorithm. If both versions are executed with the same number of iterations, algorithm version 1 can converge faster and leads to better results. But the higher complexity slows the process so that within 60 seconds only ~102.900 solutions can be calculated, while algorithm version 2 is able to calculate 40% more solutions in the same time (~144.000 solutions). The boxplots in figures 6.1, 6.2 and 6.3 show that if given the same amount of cpu-time, algorithm version 2 can outperform algorithm version 1. So in this case it is beneficial to use the simple but fast approach.

6.1.2 Search Space

Inspired by these results the next evolution of the algorithm was altered with the aim to reduce the complexity even further. Algorithm versions 1 and 2 use the same solution construction procedure, that allows the ants to move freely in order to build the solutions. Unfortunately the freedom of movement allows them to create identical solutions over different paths. In other words this means that the algorithm has to choose not only between all possible solutions, but between all permutations of all possible solutions. In algorithm version 3 the solution space is reduced to a minimum. For every possible solution there exists only one path leading to that solution. Still all solutions can be constructed with equal probability. The necessary additional calculations can be done in advance, and therefore are not slowing down the algorithm. As assumed this modification improves the solution quality and enables the algorithm to converge faster to good solutions. Note that the difference in performance of algorithm versions 2 and 3 should increase with problem size, as the number of permutations rises exponentially.

6.1.3 Shifting Ant Number

Algorithm versions 3.1 and 3.2 feature modifications of the code which caused unexpected results. In both versions the number of ants is not constant as it is in the previous versions, but is allowed to change over the runtime. In algorithm version 3.1 the number of ants increases linearly. The intention was to enable the algorithm to intensify the search behaviour in the late stages of the run. Unfortunately the results show clearly that this measure decreases the performance. This can be explained, if the change in the number of ants is seen as a change in the amount of pheromones that are deposited. With small numbers of ants fewer calculations are done before the pheromones are applied. Therefore lots of pheromones are deposited on relatively poor solutions in the early stages of the algorithm misdirecting the converging process. In algorithm version 3.2 the number of ants linearly decreases. As the first approximation sets seem to be very important for the convergence behaviour, the intensified search in the early stages could improve the performance. The results confirm this assumption. Algorithm version 3.2 provides the best results for the runtime of 60 seconds. A maximum of 40 ants is used in this version. Of course this large number of ants would also help to find better solutions in the late stages of the algorithm, but the constant use of large numbers of ants would take a lot of runtime. Therefore the varying ant number seem to enable the algorithm to use the resources efficiently for the important stages of the process, while it is still possible to calculate enough iterations.

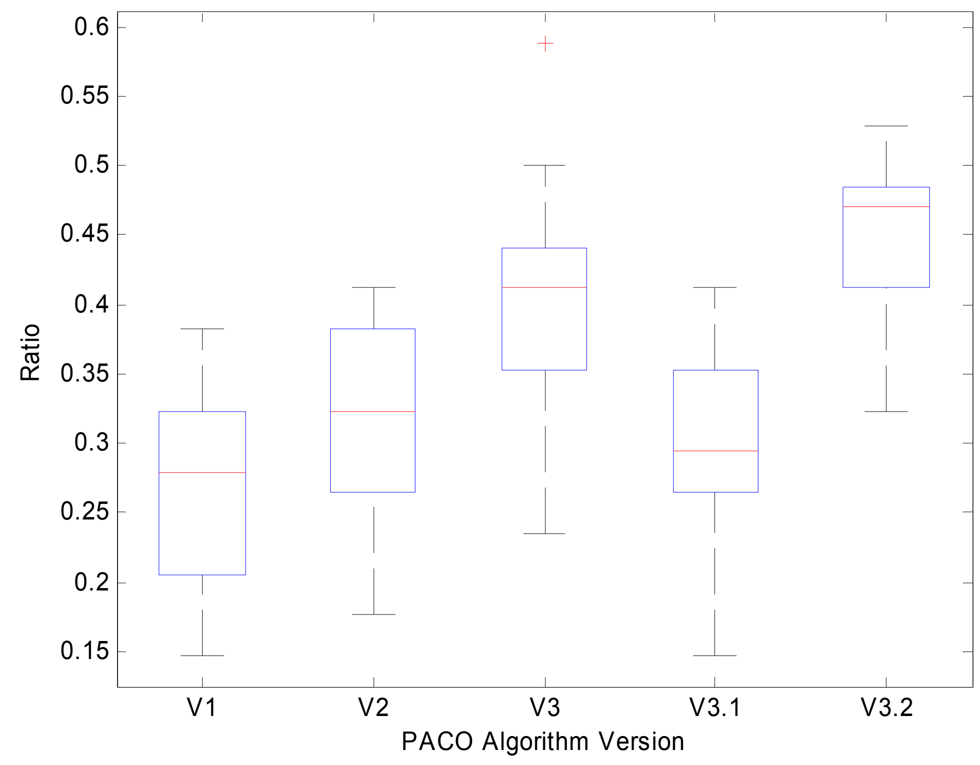

Figure 6.1: Comparison of the algorithms: Found solutions ratio.

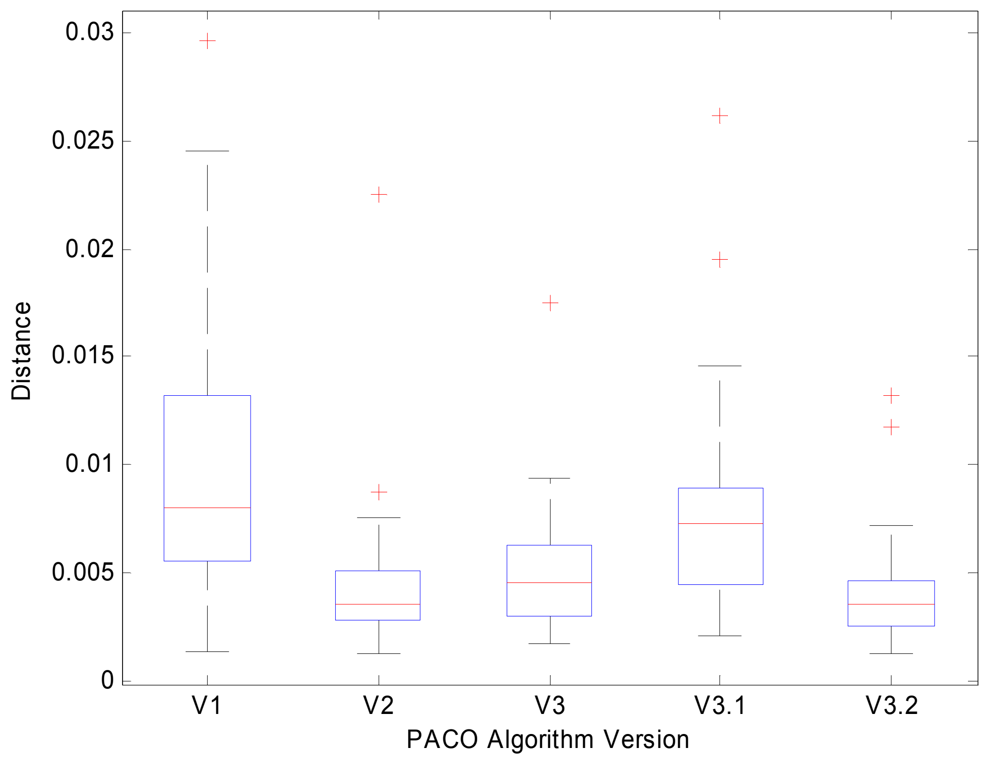

Figure 6.2: Comparison of the algorithms: Average distance.

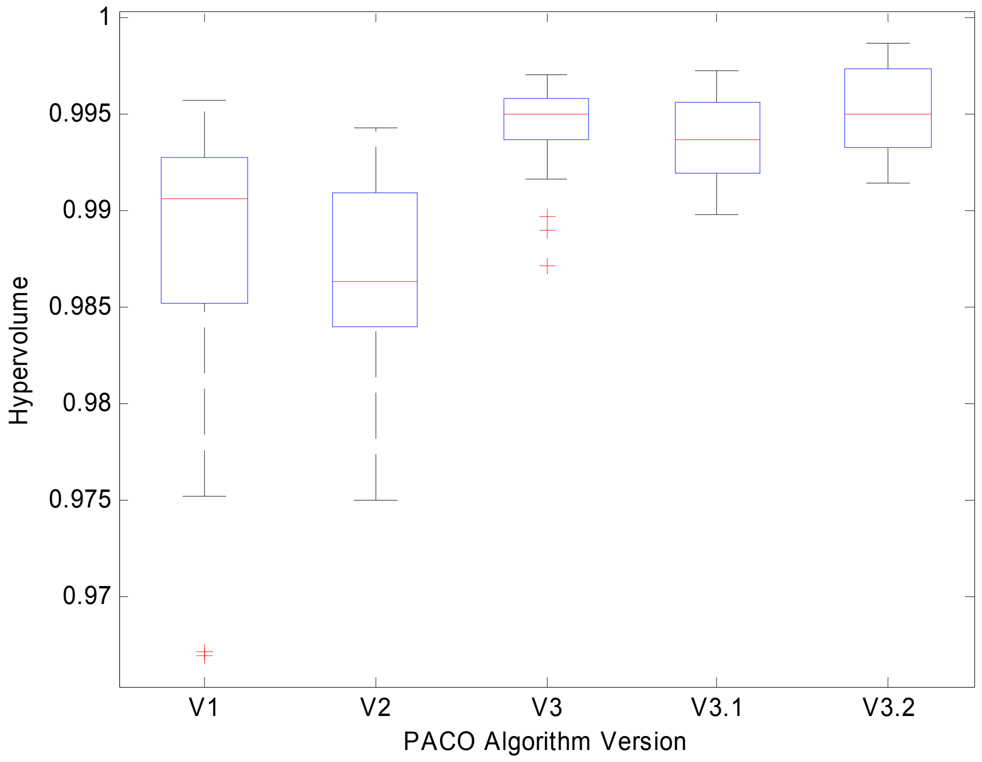

Figure 6.3: Comparison of the algorithms: Hypervolume.

6.2 Results of the Viennese Relocation Problem

The Viennese ambulance service (Wiener Rettung) is a public organization responsible for the provision of emergency services in Vienna. To fulfil the various tasks a heterogeneous fleet with differently assembled staff is maintained. There are 40 ambulances staffed with paramedics, who are able to respond to most emergencies. These ambulances are also used to take over non-emergency tasks like transportation services to hospitals. For difficult cases, where an emergency physician is needed, there exist 17 NEF vehicles (Notarzt Einsatz Fahrzeug), which are staffed with a paramedic and an emergency physician. Incoming emergency calls are answered centrally, so that the appropriate vehicle can be sent according to the criteria of the NEF-system. This system enables a higher availability of emergency physicians as well as an efficient use of the resources. The waiting sites of the NEF vehicles are distributed over Vienna in a way that each possible emergency site can be reached within 12 minutes after the call.

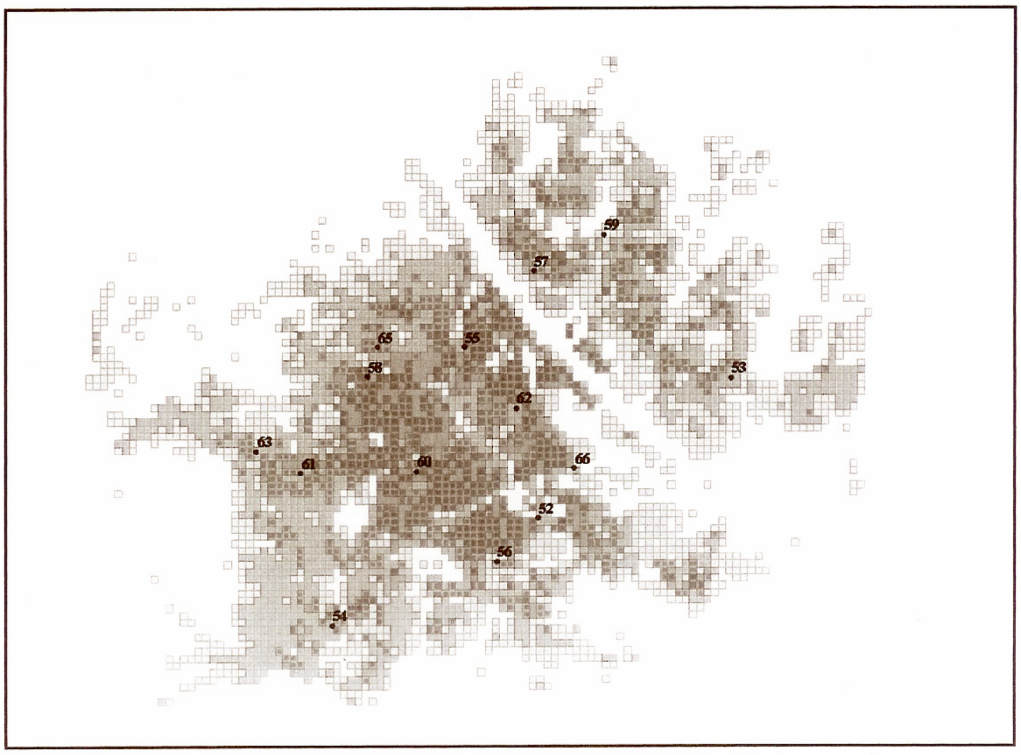

Figure 6.4: Demand Points and Waiting Sites in Vienna[4]

[4] Source: Dorner, T.: “Comparing Location Models for Emergency Vehicles in Vienna”, page 20, Vienna, 2006

Thanks to the work of Theresa Dorner (2006) very accurate data of the situation in Vienna could be implemented in this study. Her diploma thesis contains precise information on the waiting locations, demand points and the distances between them. In the following the establishment of the data applied in this work is described. The necessary processing of the geographic data was done with the geographic information system (GIS) software ArcView and its extension WIGeoNetwork. With the use of this software the exact waiting site locations available to the NEF system could be geocoded. For the establishment of the demand points the territory of Vienna was partitioned into 250x250m squares. The centroid of these squares was used to locate the demand point, while the resident population within these squares represents the demand. The information of the population distribution was provided by Statistik Austria. The total population of 1.721.987 inhabitants is allocated to 3.920 demand points with a population > 0. Figure 6.4 depicts the distribution of the demand points and waiting sites in Vienna. Squares without any resident population are left out, while darker squares indicate higher population density. The distances between the waiting stations and the demand points were calculated with data basing on real street data of TeleAtlas. This way the effect of different street categories and traffic rules could be considered.

In order to implement the PACO Metaheuristic on the Viennese problem, the algorithms were adapted to the scale of the problem. The algorithm needs upper and lower bounds of the target functions. They are used to evaluate the solutions found during the run of the algorithm and consequently affect on which solution paths pheromones are added. Fortunately the actual best and worst values of the population coverage value could be found in reasonable computation time. All waiting site allocations were fully enumerated for each state in order to find the maximum and minimum population coverage of each state. Then the best and worst values were combined to build the upper and lower bound respectively. The values of target function 1 range from 1,643,477.34 to 1,721,791.81. Note that these values cannot be directly interpreted as the population coverage, because they give the sum of the population coverage of each state weighted by the probability of the state. The upper bound of the relocation value is given by the sum of the maximum number of relocations of each state multiplied by the corresponding probability:

$$worst\_tf2 = \sum_{k=1}^{vh} (k-1) * q_k \tag{23}$$

The actual lowest possible value of target function 2 cannot be found within reasonable computation time. A very low relocation value was found by the evaluation of a solution with a hamming distance between states of zero. With this method a value of 0.88752 was obtained. It can be estimated that the lowest possible value lies slightly below 0.88752. In order not to exceed the lowest possible value the lower bound was set to 0. The lower and upper bounds of target function 2 are therefore 0 and 8.94167 respectively. The calculation of the probabilities of the states demands the knowledge of two characteristics: the average service rate μ and the arrival rate of calls λ. According to Dorner (2006) these values are μ=1.2 and λ =8.47 and base on real data provided by the Viennese ambulance service. With the use of binomial distribution (Formula 1) the probabilities q_k were calculated. The optimal parameters for the different algorithm versions were determined in a standardised procedure described in appendix C. Applied to the larger problem instance they lead to a runtime of approximately 50 minutes.

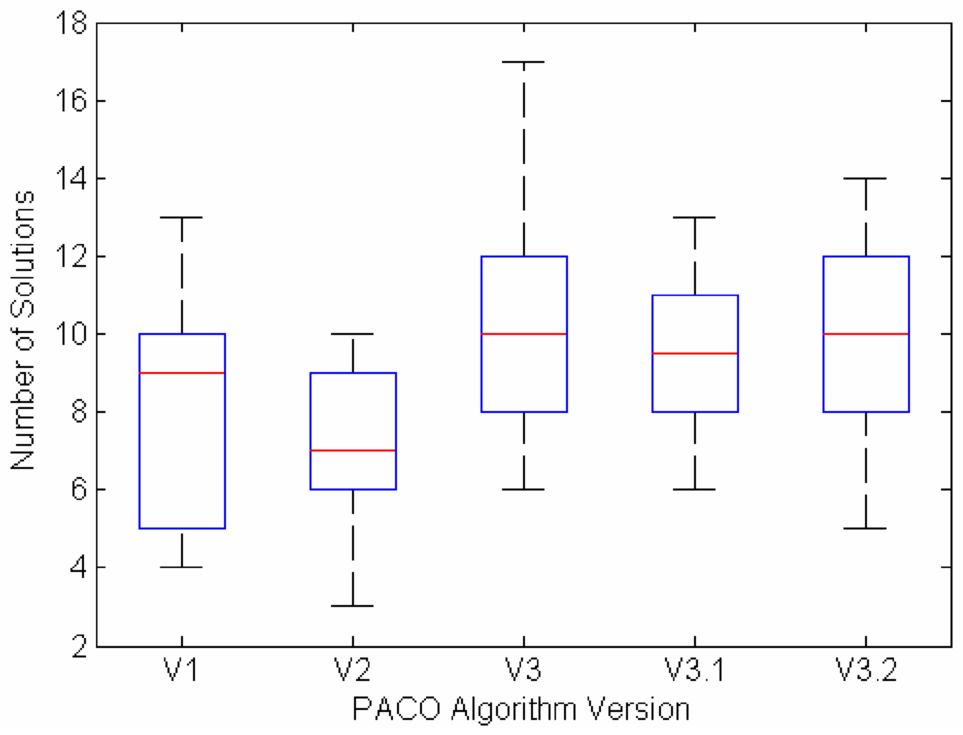

Figure 6.5: Number of Solutions of the Algorithm Versions.

For each algorithm version 10 approximation sets were calculated for the Viennese relocation problem. Because of the size of the problem a full enumeration would take too much computational time, so the pareto optimal front is unknown. Therefore the solutions ratio and distance metric could not be used to evaluate the quality of the approximation sets. The number of solutions of the approximation sets is given in a boxplot in figure 6.5. The different versions show a uniform behaviour in this aspect. Relatively few pareto optimal solutions are included in the approximation sets compared to the total number of up to 154.000 calculated solutions in every single run.

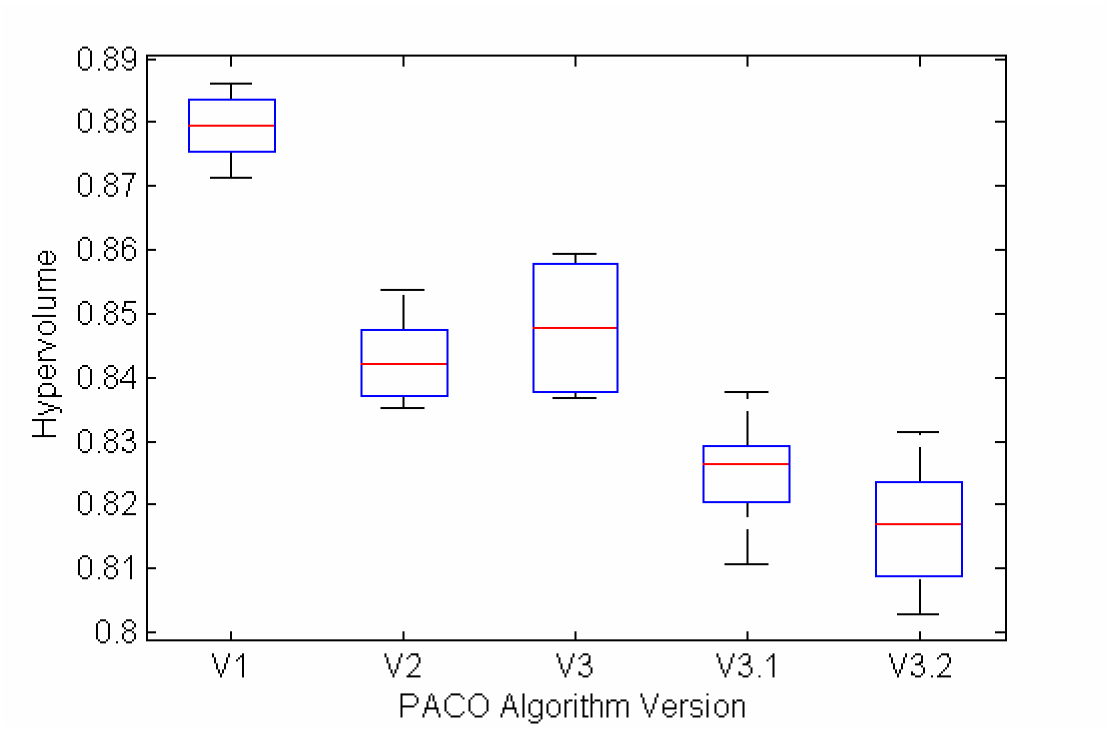

Figure 6.6: Hypervolume Values of the Algorithm Versions.

The hypervolume metric is better qualified to compare the algorithm versions, as it can evaluate the quality of the approximation sets. Figure 6.6 gives a boxplot of the hypervolumes attained by the different algorithm versions. The results are quite surprising, because in two aspects they don't reflect the previous findings. The performance of algorithm version 3.2 falls behind all other versions although it was the best performing algorithm at the test problem. It is jet unclear why the larger problem instance leads to this change. Further investigation would be necessary to explain the dissimilar impact of decreasing ant numbers on the performance of the algorithm. Algorithm version 1 provides the best hypervolume values. In this case the change can be explained by the conceptual difference. The complex pheromone structure of version 1 allows more information to be stored, but takes a bit more of cpu-time. For the small test problem it was beneficial to solve more solutions in the same time. The larger problem size allows algorithm version 1 to profit from the conceptual advantage and makes it the best performing version. It is interesting to see that the problem size can have such an impact on the algorithm performance. Figure 6.7 shows all solutions of the approximation sets of the best and worst performing algorithm versions 1 and 3.2. Better solutions are closer to the bottom right corner. With the help of this graph the superiority of algorithm version 1 is visualized, and it can be seen that the solutions of algorithm version 3.2 do not include any solutions of the pareto optimal front, as they are dominated by the solutions of algorithm version 1. However both versions operate on a high level as they are relatively close to the boundaries. An arbitrary selected solution from the lower right corner achieves in 97.6% of the time a population coverage higher than 99.9% while it meets a relocation value of 1.08. If the selected relocation strategy would be applied, a high level of population coverage could be achieved with a relatively small number of relocations.

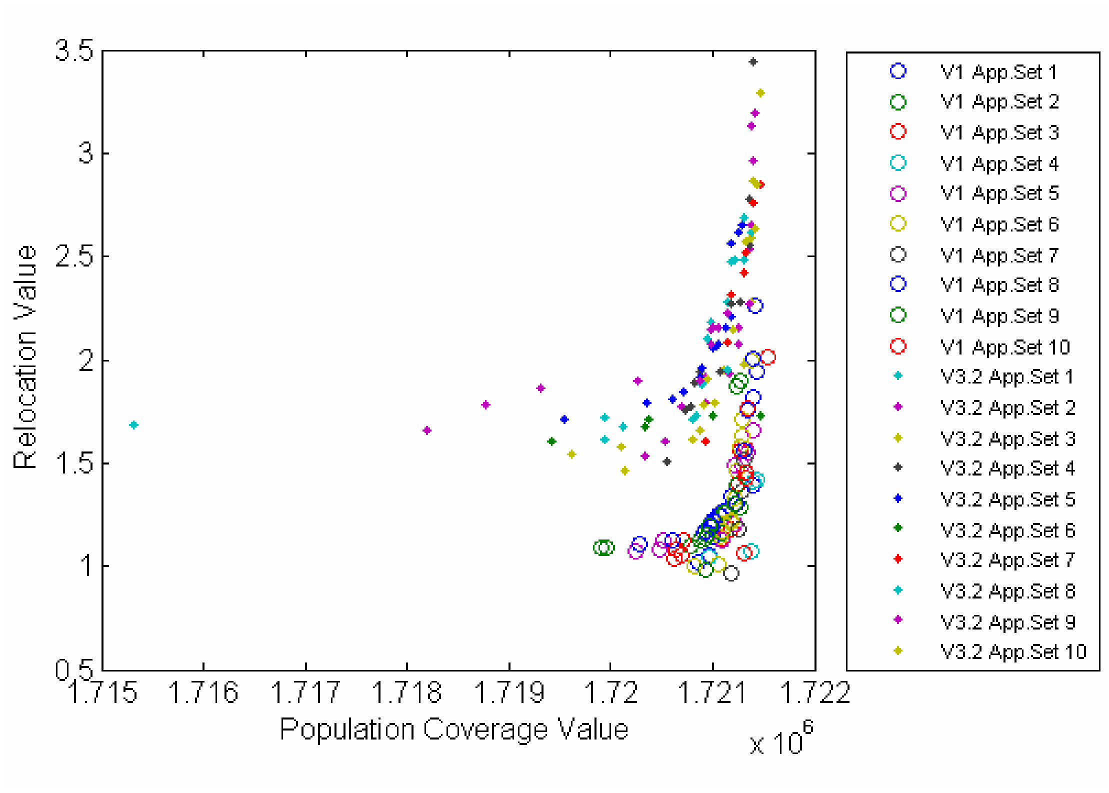

Figure 6.7: Approximation Sets of Algorithm Versions V1 and V3.2.

7. Conclusion

Based on the Maximal Expected Coverage Relocation Problem by Gendreau et al. (2006) a bicriterial ambulance relocation problem has been developed. In addition to the extension to a bicriterial problem, the way to assess the number of relocations was refined. It was shown how the Pareto Ant Colony Algorithm can be used to face the increased computational complexity caused by the extension.

The results of this work help to understand the mechanisms behind the Pareto Ant Colony Algorithm, but they also bring up new questions. One major question was whether to use a simple approach or to rely on a sophisticated concept even if the complexity slows down the algorithm. The results of the test problem clearly favoured the fast and simple approach, but the complex approach could employ its advantages when it was applied to the large problem instance. The reduction of the solution space improved the performance in both cases as it was expected, whereas the experiments with varying ant numbers generated ambiguous results. Only when applied to the small model the decreasing ant number could improve the performance. Further research is needed to understand under what circumstances a varying ant number can increase the performance.

The final pareto optimal solutions proposed for the Viennese ambulance relocation problem show that the implementation of a relocation strategy can effect high rates of population coverage despite relatively few relocations. But the high values of the population coverage also result from the large radii of the ambulance vehicles. Even with only one vehicle 90% of the population can be reached within 12 minutes, if it is located on waiting station 14. Therefore an implementation of a relocation strategy could be used to maintain stricter time constraints and minimize the response times with the given number of NEF-vehicles.

References:

Church, R.L., ReVelle, C.S.: "The maximal covering location problem", Papers of the Regional Science Association 32, 101-118, 1974

Daskin, M.S.: "The maximal expected covering location problem: formulation, properties and heuristic solution", Transportation Science 17, 48-70, 1983

Deneubourg, J.L., Aron, S., Goss, S., Pasteels, J.-M.: "The self-organizing exploratory pattern of the Argentine ant", Journal of Insect Behaviour, 3, 159-168, 1990

Dorigo, M., Stützle, T.: "Ant Colony Optimization", The MIT Press, 2004

Dörner, K., Gutjahr, W., Hartl, R., Strauss, C., Stummer, C.: "Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection", Kluwer Academic Publishers, 2002

Dorner, T.: "Comparing Location Models for Emergency Vehicles in Vienna", Wien, 2006

Gendreau, M., Laporte, G., Semet, F.: "A dynamic model and parallel tabu search heuristic for real-time ambulance relocation", Parallel Computing 27, 1641-1653, 2001

Gendreau, M., Laporte, G., Semet, F.: "The Maximal Expected Coverage Relocation Problem for Emergency Vehicles", Journal of the Operational Research Society 57, 22-28, 2006

Jaszkiewicz, A.: "Evaluation of Multiple Objective Metaheuristics", in: Sevaux, M., Sörensen, K., T'kindt, V.: Metaheuristics for Multiobjective Optimization, Springer, Berlin, 2004

Nebro, A., Luna, F., Alba, E., Beham, A., Dorronsoro, B.: “AbYSS: Adapting Scatter Search for Multiobjective Optimization”, Tech Rep. ITI-2006-2, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, 2006

Toregas, C., Swain, R., ReVelle, C.S., Bergmann, L.: “The location of emergency service facilities”, Operations Research 19, 1363-1373, 1971

Zitzler, E., Thiele, L.: “Multiobjective optimization using evolutionary algorithms - a comparative case study.” In Eiben, A., Bäck, T., Schoenauer, M., Schwefel, H.: Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V), pages 292–301, Springer, Berlin, 1998

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Grunert da Fonseca, V.: “Performance Assessment of Multiobjective Optimizers: An Analysis and Review”, TIK-Report No. 139, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, 2002

Zitzler, E., Thiele, L., Deb, K.: “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”, Evolutionary Computation 8(2): 173-195, Massachusetts Institute of Technology, 2000

Appendix:

A. List of Variables

best_zf1	…	best possible value or upper bound of target function 1
best_zf2	…	best possible value or upper bound of target function 2
c	…	parameter for pheromone-update (deposit)
coverage(i)	…	poputation reachable within r minutes from station i
D(x,y,station)	…	distance array of distances between demand points and waiting stations
di(i)	…	demand of demand point i given by population
M	…	number of iterations
PI	…	number of periods
ps(i,j,k)	…	number of possible solutions if station i is chosen, for the j-th vehicle to assign, for k vehicles
qk(k)	…	probability of being in state k
r	…	action radius of the emergency vehicles, given in minutes
rho	…	parameter for pheromone-update (evaporation)
s	…	number of ants
SOL	…	solution matrix representing the approximation set
SOL_A	…	list of solutions found by the ants within one iteration
SOL_c	…	solutions of the complete enumeration
SOL_c_s	…	standardised solutions of the complete enumeration
SOLi	…	best solution of solution set SOL_A
st	…	number of potential waiting stations
tau_com	…	pheromone matrix of weighted combined target functions
tau_f1	…	pheromone matrix of target function 1
tau_f2	…	pheromone matrix of target function 2
vh	…	number of vehicles
weight_f1	…	weight of target function 1
weight_f2	…	weight of target function 2
worst_zf1	…	worst possible value or lower bound of target function 1
worst_zf2	…	worst possible value or lower bound of target function 2

B. Code of PACO V3.2

In this section version 3.2 of the Pareto Ant Colony Algorithm is given in Matlab code. Matlab is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation.[5] The other versions of the algorithm as well as all other codes used in this work are given on the enclosed CD. All calculations were performed on an Athlon64 3500+ PC system. A list of all variables used in this code can be found in appendix A.

```
[vh,st,r,D,di,qk,coverage]=create_TESTData;

PI=25
M=308
s=40
rho=0.01
c=50

function SOL = PACO(PI,M,s,rho,c,D,di,vh,st,r,qk,coverage)
%------------------------------
%Pareto Ant Colony Optimization
%PI    ... number of periods (each period runs with new random weights)
%M     ... number iterations
%s     ... number of ants
%rho   ... parameter for pheromone-update (evaporation)
%c     ... parameter for pheromone-update (deposit)
%D     ... distancearray (x,y,station)
%di    ... demand
%vh    ... number of vehicles
%st    ... number of stations
%r     ... range of vehicles
%qk    ... probability that k vehicles are available
%------------------------------
high=100000;
SOL=zeros(1,12); SOL(1,12)=high;%create first solution
best_zf1=[8873.546107017692;];
worst_zf1=[4422.241490884563;];
bw_zf1=best_zf1-worst_zf1;
best_zf2=[0.546514972263641;];%!!! value calculated by complete enumeration !!!
worst_zf2=[2.2954881516229;];
bw_zf2=worst_zf2-best_zf2;
%upper and lower bounds can be used for best- and worst-values

ps = create_ps(st,vh);

for Periode=1:PI
    weight_f1=rand;%assign weights for F1 and F2 randomly
    weight_f2=1-weight_f1;
    [tau_f1,tau_f2,tau_com]=create_tau(st,vh,weight_f1,weight_f2);
    %create pheromone-matrices
    for Iteration=1:M
        AntNr=max([3 s+1-ceil(Iteration/(M/s))]);
        %lineary decreases the number of ants, beginning with s (minimum value=3)
        for Ant=1:AntNr
            X=createX(tau_com,vh,st,ps);%run an ant
            SOL_A(Ant,:)=createSOLi(X,D,di,vh,st,r,qk,coverage);
            %calculates the target values
        end
        SOLi.=.findbest(SOL_A,weight_f1,weight_f2,AntNr,
                        worst_zf1,best_zf2,bw_zf1,bw_zf2);
        %searches for the best solution
[tau_f1,tau_f2,tau_com]=Phero_update(rho,c,SOLi,st,vh,weight_f1,weight_f2,tau_f1,
```

[5] http://www.mathworks.com (4.10.2007)

```
                                         tau_f2);%pheromone-update
        SOL = sortfun(SOLi,SOL);% add solution
    end
end

function ps = create_ps(st,vh)
%-----------------------------
%ps(i,j,k) gives the number of possile solutions if station i is chosen,
%for the j-th vehicle to assign, for k vehicles.
%-----------------------------
ps=zeros(st,vh,vh);
for i=1:vh
    ps(i:st,i,i)=1;
end
for i=2:vh
    for j=i-1:-1:1
        for k=1:st-1
            if ps(k+1,j+1,i)>0
                ps(k,j,i)=sum(ps(k+1:st,j+1,i));
            end
        end
    end
end

function [tau_f1,tau_f2,tau_com]=create_tau(st,vh,weight_f1,weight_f2)
%---------------------------------------------------------------------
%creates the pheromone-matrices for TF1, TF2 and weighted combined
%---------------------------------------------------------------------
tau_f1=zeros(st,vh,vh);
for i=1:vh
    tau_f1(:,1:i,i)=1;
end
tau_f2=tau_f1;
tau_com=weight_f1*tau_f1+weight_f2*tau_f2;

function X = createX(tau_com,vh,st,ps)
%---------------------------------------------------
%creates an solution X (run an ant)
%---------------------------------------------------

X=zeros(st,vh);%creates blank solution-matrix X
tau_all=tau_com.*ps;
for i=1:vh
    currentloc=0;
    for j=1:i
        randnumber = rand * sum(tau_all(currentloc+1:st,j,i));
        bound=0;
        count=0;
        while randnumber > bound
            count=count+1;
            bound = bound + tau_all(currentloc+count,j,i);
        end
        X(currentloc+count,i)=1;
        currentloc=currentloc+count;
    end
end

function SOLi = createSOLi(X,D,di,vh,st,r,qk,coverage)
%---------------------------------------
%calculates the target values of X
%---------------------------------------

SOLi(1,1)=find(X(:,1));
SOLi(1,2:3)=find(X(:,2));
SOLi(1,4:6)=find(X(:,3));
SOLi(1,7:10)=find(X(:,4));
SOLi(1,11)= EvaluationF1(X,D,di,vh,st,r,qk);
SOLi(1,12)= EvaluationF2_V2(SOLi,qk,coverage);

function targetvalue1 = EvaluationF1(X,D,di,vh,st,r,qk)
%------------------------------------
%calculates targetvalue of F1
%targetvalue1 = percentage of covered population, weighted with qk summed over all k.
%------------------------------------

coverage_di=zeros(20,20,vh);
%coverage_di(x,y,numberVehicles)...saves the demand(di) of covered demand points
for i=1:vh
   for j=1:st
```

```
      if X(j,i)==1,
         for x=1:20
            for y=1:20
               if D(x,y,j)<=r,
                  coverage_di(x,y,i)=di(x,y);
               end
            end
         end
      end
   end
end
parttargetvalue=zeros(vh,1);%coverage of 1,2,...,vh available vehicles
for i=1:vh
    parttargetvalue(i,1)=sum(sum(coverage_di(:,:,i)));
end
targetvalue1=qk*parttargetvalue;

function targetvalue2 = EvaluationF2_V2(SOLi,qk,coverage)
%-----------------------------------
%calculates the targetvalue of F2
%targetvalue2 = The number of vehicles to be relocated between two
%consecutive states k according to which vehicle is called (the probability
%that a vehicle is called from station st is dependent on the covered
%population (=coverage(st))), weighted with qk summed over all k.
%-----------------------------------
rel=zeros(3,4);%number of vehicles to be relocated
q_rel=zeros(3,4);%probability that vehicle is called

v2=zeros(1,2);
v2(1)=any(SOLi(1,1)==SOLi(1,2));
v2(2)=any(SOLi(1,1)==SOLi(1,3));
for i=1:2
    sel=ones(1,2);
    sel(1,i)=0;
    rel(1,i)=1-v2*sel';
    q_rel(1,i)=coverage(SOLi(1,i+1))/sum(coverage(SOLi(1,2:3)));
end

v3=zeros(1,3);
v3(1)=any(SOLi(1,2:3)==SOLi(1,4));
v3(2)=any(SOLi(1,2:3)==SOLi(1,5));
v3(3)=any(SOLi(1,2:3)==SOLi(1,6));
for i=1:3
    sel=ones(1,3);
    sel(1,i)=0;
    rel(2,i)=2-v3*sel';
    q_rel(2,i)=coverage(SOLi(1,i+3))/sum(coverage(SOLi(1,4:6)));
end

v4=zeros(1,4);
v4(1)=any(SOLi(1,4:6)==SOLi(1,7));
v4(2)=any(SOLi(1,4:6)==SOLi(1,8));
v4(3)=any(SOLi(1,4:6)==SOLi(1,9));
v4(4)=any(SOLi(1,4:6)==SOLi(1,10));
for i=1:4
    sel=ones(1,4);
    sel(1,i)=0;
    rel(3,i)=3-v4*sel';
    q_rel(3,i)=coverage(SOLi(1,i+6))/sum(coverage(SOLi(1,7:10)));
end

targetvalue2=0;
for i=1:3
    targetvalue2=targetvalue2+qk(1,i+1)*(rel(i,:)*q_rel(i,:)');
end

function SOLi =
findbest(SOL_A,weight_f1,weight_f2,AntNr,worst_zf1,best_zf2,bw_zf1,bw_zf2)
%------------------------------------------------------------------
%searches the best solution of SOL_A according to the current weighting.
%------------------------------------------------------------------

for i=1:AntNr
    targetvalue(i) = weight_f1 * ((SOL_A(i,11)-worst_zf1)/bw_zf1-1)*-1 + weight_f2 *
((SOL_A(i,12)-best_zf2)/bw_zf2);
end
[Min_value,Min_index] = min(targetvalue);
SOLi=SOL_A(Min_index,:);
```

```
function
[tau_f1,tau_f2,tau_com]=Phero_update(rho,c,SOLi,st,vh,weight_f1,weight_f2,tau_f1,tau_f
2)
%-------------------------------------------------------
%evaporation of pheromones and deposit of pheromones on the path of he best ant
%-------------------------------------------------------

incl=zeros(st,vh,vh); %incl=1 if part of best solution SOLi
incl(SOLi(1,1),1,1)=1;
incl(SOLi(1,2),1,2)=1;
incl(SOLi(1,3),2,2)=1;
incl(SOLi(1,4),1,3)=1;
incl(SOLi(1,5),2,3)=1;
incl(SOLi(1,6),3,3)=1;
incl(SOLi(1,7),1,4)=1;
incl(SOLi(1,8),2,4)=1;
incl(SOLi(1,9),3,4)=1;
incl(SOLi(1,10),4,4)=1;
tau_f1  = (1-rho) * tau_f1 + rho * weight_f1 * c * incl;
%pheromone-update for pheromonetrail of F1
tau_f2  = (1-rho) * tau_f2 + rho * weight_f2 * c * incl;
%pheromone-update for pheromonetrail of F2
tau_com = weight_f1 * tau_f1 + weight_f2 * tau_f2;
%combined weighted pheromone-matritces

function SOL = sortfun(SOLi,SOL)
%---------------------------------------------------
%integrates solution SOLi into approximation set SOL,
%and deletes dominated solutions in SOL
%---------------------------------------------------
dominated=0;
for i=1:size(SOL,1)
    if SOLi(1,11)<=SOL(i,11) & SOLi(1,12)>=SOL(i,12)%new solution is dominated
        return %abort, because new solution is dominated or already integrated
    elseif SOLi(1,11)>=SOL(i,11) & SOLi(1,12)<=SOL(i,12)
    %new solution dominates at least one old solution
        dominated(end+1)=i; %list dominated solutions
    end
end
SOL(dominated(1,2:end),:)=[]; %delete dominated solutions
SOL(end+1,:)=SOLi; %integrate new solution

function [vh,st,r,D,di,qk,coverage]=create_TESTData
%------------------------------------
%creates data needed for calculation
%------------------------------------

vh=4; %number of vehicles
st=12; %number of stations
r=4; %reach of vehicles
S=[4 6;4 14;6 3;6 10;9 6;10 10;10 15;12 6;13 9;14 4;15 14;16 8]; %coordinates of
stations

DP=zeros(20,20);
count=1;
for j=1:2:40
    for i=1:20
        DP(i,j)=i;
        DP(i,j+1)=count;
    end
    count=count+1;
end

D=zeros(20,20,st); %creates distance-array (x,y,station)
for j=1:st
   D(:,:,j)=abs(DP(1:20,1:2:40)-S(j,1))+abs(DP(1:20,2:2:40)-S(j,2));
end

di=[0,0,0,10,10,10,10,10,10,10,10,10,10,10,10,10,10,0,0,0;
    0,0,10,10,38,10,10,10,10,10,10,10,10,20,10,10,10,10,10,0;
    0,10,10,20,38,10,10,10,10,10,10,60,60,60,20,10,10,10,10,0;
    20,20,20,20,55,76,76,76,10,10,10,60,60,60,60,20,20,10,10,10;
    20,20,44,55,55,76,76,76,76,88,88,60,60,60,60,20,20,10,10,10;
    20,20,44,55,55,76,76,76,88,88,88,88,40,40,40,40,55,10,10,10;
    20,20,44,44,55,76,76,76,88,88,88,88,70,70,70,55,55,10,10,10;
    20,20,44,44,55,76,90,90,88,88,70,70,70,70,70,55,55,20,20,20;
    66,66,66,80,80,90,90,90,90,100,70,70,70,70,70,55,55,20,20,20;
    66,80,80,80,80,90,90,90,100,100,100,70,70,70,70,55,20,20,20,20;
    10,80,80,80,70,90,90,100,100,100,88,70,70,70,70,55,20,20,20,20;
    10,60,70,70,95,95,95,99,100,88,88,88,70,70,70,20,20,20,20,20;
    10,60,60,70,74,95,95,99,88,88,88,50,50,70,70,44,20,20,20,20;
    10,60,60,60,74,74,95,99,88,88,70,70,50,50,44,38,20,20,20,20;
    10,50,50,60,60,74,74,99,60,60,70,70,50,44,44,38,38,38,38,20;
    10,50,50,60,60,60,74,60,60,60,50,50,44,44,44,38,38,38,38,38;
    40,50,50,50,40,40,40,60,60,60,50,50,44,44,44,38,38,33,33,33;
    40,50,50,50,40,40,40,40,50,50,50,30,44,44,44,33,33,33,33,33;
    40,50,50,40,40,40,40,40,40,50,50,30,33,33,33,33,33,33,0,0;
    40,40,40,40,40,40,40,40,40,50,50,30,33,33,33,33,33,33,0,0;];

p=1-1.03/(vh*1.46); % p=probability that vehicle is available
qk=zeros(1,vh); % qk(1,k)=probability of having k availabe vehicles
for k=1:vh
    qk(1,k)=(factorial(vh)/(factorial(vh-k)*factorial(k)))*p^k*(1-p)^(vh-k);
end

E=zeros(20,20,st);
for i=1:st
    for j=1:20
        for k=1:20
            if D(j,k,i)<=r
                E(j,k,i)=di(j,k);
            end
        end
    end
end
for i=1:st
    coverage(i)=sum(sum(E(:,:,i)));
end
```

C. Determination of the Parameters

The version of the PACO algorithm used in this work utilises 5 parameters (*PI, M, s, rho* and *c*).

- *PI* adjusts how many times the algorithm will start over with new weightings.
- *M* defines the number of iterations of the algorithm with one weighting.
- *s* defines the number of ants within one iteration. In the case of algorithm version 3.1 and 3.2 *s* defines the maximal number of ants.
- *rho* is the factor of pheromone evaporation.
- *c* defines the pheromone deposit.

The parameters *PI, M* and *s* influence the length of calculation time directly proportional. For each of these tree parameters a doubling of the parameter approximately leads to a doubling in calculation time. As *PI*M*s* gives the total number of solutions calculated during the run of the algorithm, it is easy to estimate the total runtime of the algorithm. After some initial testing of the algorithms it was decided to optimise the parameters for a total runtime of 60 seconds of cpu-time. First a set of different promising values for *PI* (*PI*=[15, 20, 25, 30]) and *s* (*s*=[10, 20, 30, 40]) were chosen. Then the value of *M* was set in order to lead to a runtime of approximately 60 seconds.

PI	M	s	PI*M*s (calculated solutions)	estimated runtime (sec)
15	1021	10	153150	60,00851097
20	766	10	153200	60,02810239
25	613	10	153250	60,04769381
30	510	10	153000	59,94973672
15	510	20	153000	59,94973672
20	383	20	153200	60,02810239
25	306	20	153000	59,94973672
30	255	20	153000	59,94973672
15	340	30	153000	59,94973672
20	255	30	153000	59,94973672
25	204	30	153000	59,94973672
30	170	30	153000	59,94973672
15	255	40	153000	59,94973672
20	191	40	152800	59,87137105
25	153	40	153000	59,94973672
30	128	40	153600	60,18483373

Table C.1: Values of PI, M and s.

The combinations of PI, M and s given in table C.1 were combined with different values of rho (rho=[0.01, 0.02, 0.03]) and c (c=[20, 50, 80]) leading to a total of 144 different combinations. Then the algorithm was run with all these combinations of parameters and the end results were evaluated by the three metrics of performance described in chapter 5. This step was repeated 4 times and the mean values of the metrics were built. For a better evaluation quality the 10 best combinations of parameters were recalculated 6 times in order to get statistically firm results. The 10 best combinations and the results are given in table C.2.

PI	*M*	*s*	*rho*	*c*	M_1 ratio	M_2 distance	M_3 hypervolume	weighted metric values
30	255	20	0,02	80	0,39117647	0,00425771	0,99468548	0,96482707
30	128	40	0,02	50	0,42352941	0,0047455	0,99643719	0,967437
30	170	30	0,02	20	0,43823529	0,00417195	0,99584964	0,96796244
30	170	30	0,03	20	0,44117647	0,00342286	0,99566027	0,96821114
25	153	40	0,01	80	0,42941176	0,00863221	0,99628488	0,96646609
25	153	40	0,02	80	0,35882353	0,00481677	0,99539982	0,96350603
25	306	20	0,01	80	0,42941176	0,00398189	0,9956579	0,96745366
25	153	40	0,02	50	0,42941176	0,00612594	0,99601658	0,96704358
25	204	30	0,02	50	0,43529412	0,00333889	0,9966363	0,96857663
20	383	20	0,01	50	0,45882353	0,00416691	0,99625383	0,96925609(max)

Table C.2: The 10 best combinations of parameters (10 repetitions).

The quality of the results was assessed by a weighted combination of the metrics. The weights were set according to the indication quality of the metrics (cf. chapter 5) and the range of attained values. M_1, M_2 and M_3 were weighted with 5%, 30% and 65%. The high weighting for M_3 was chosen because hypervolume has the best indication quality. In addition the range of the attained values was small in comparison to the range of values of M_1. While the values of M_3 were ranging between 0.973 and 0.996, the values of M_1 were ranging between ~0.1 and ~0.5. So the weighting of M_1 had to be chosen comparatively small in order to limit the impact of metric M_1.

In the next step the best combination of parameters was used for further optimizations of the parameters *rho* and *c*. A total of 25 combinations of parameters was built with PI=20, M=383, s=20, rho=[0.005, 0.01, 0.015, 0.02, 0.025] and c=[30, 40, 50, 60, 70]. Again the first run was repeated 4 times, and then the 10 best combinations were recalculated 6 times. According to these calculations the best combination of parameters for algorithm version 3 is:

PI=20, *M*=383, *s*=20, *rho*=0.01, *c*=60

This procedure was repeated for all versions of the algorithm leading to the parameters listed in table C.3.

Algorithm Version	*PI*	*M*	*s*	*rho*	*c*
V1	15	343	20	0.02	40
V2	20	240	30	0.025	50
V3	20	383	20	0.01	60
V3.1	30	128	80	0.02	40
V3.2	25	308	40	0.01	50

Table C.3: List of best parameters for 60 seconds of runtime.

D. Convergence Graphs

These graphs depict the convergence behaviour of the different algorithm versions. The metric values are averaged values of twenty runs of 60 seconds. Information on the three metrics of performance can be found in chapter 5.

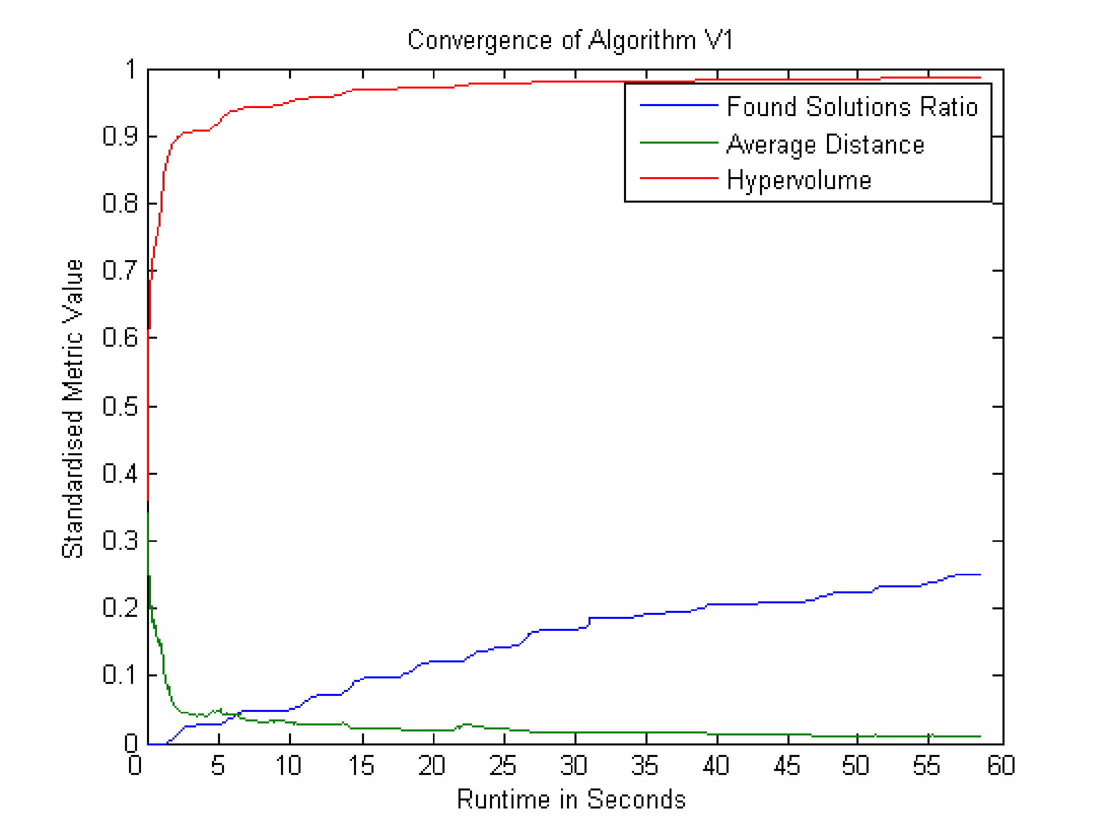

Figure D.1: Convergence Graph of Algorithm Version 1

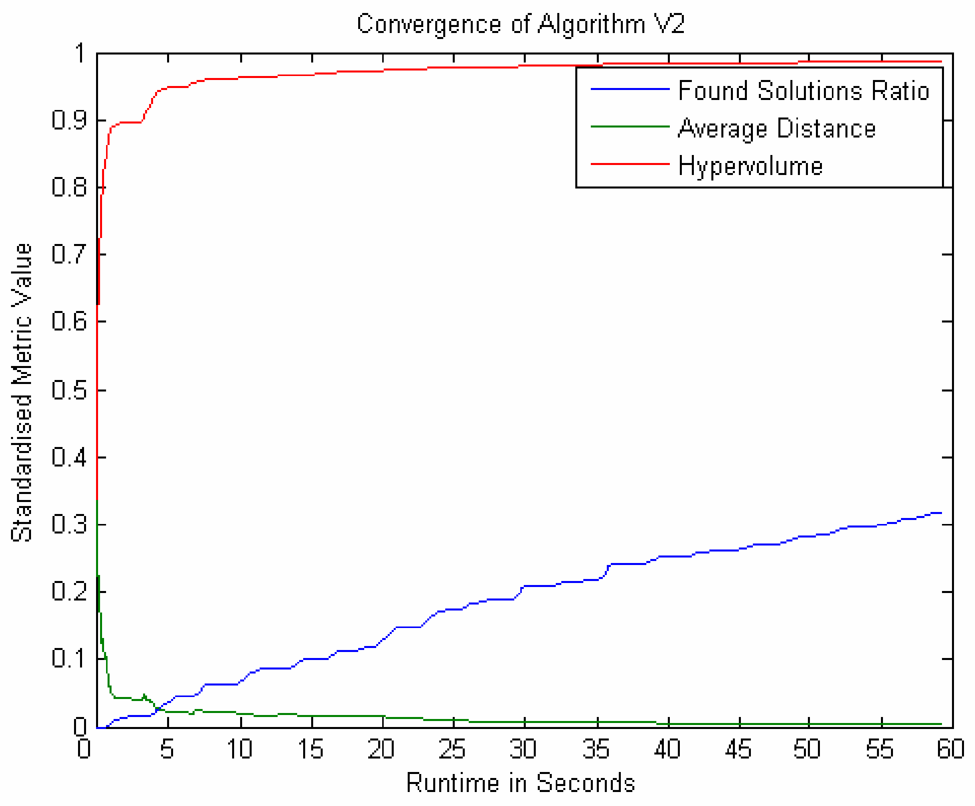

Figure D.2: Convergence Graph of Algorithm Version 2

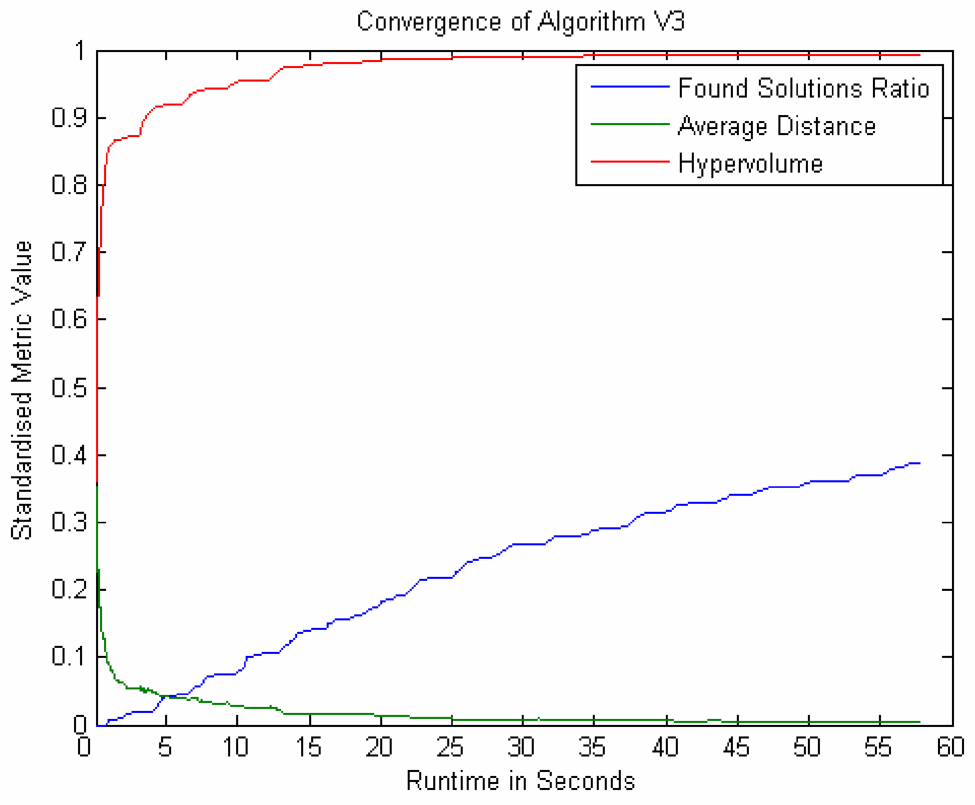

Figure D.3: Convergence Graph of Algorithm Version 3

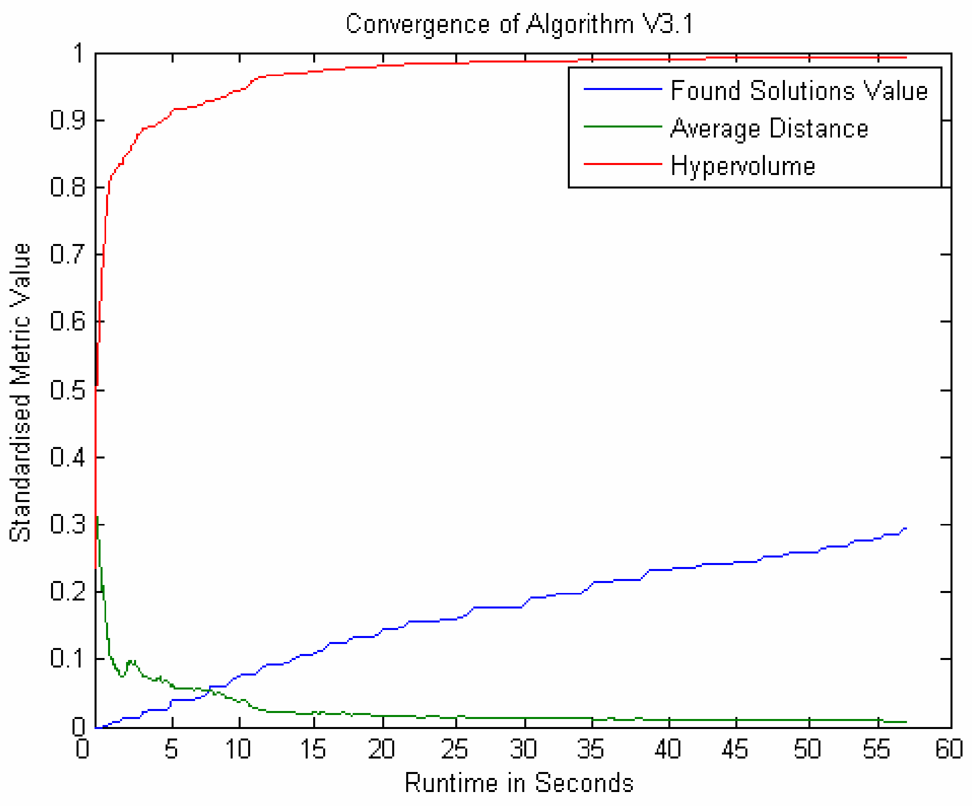

Figure D.4: Convergence Graph of Algorithm Version 3.1

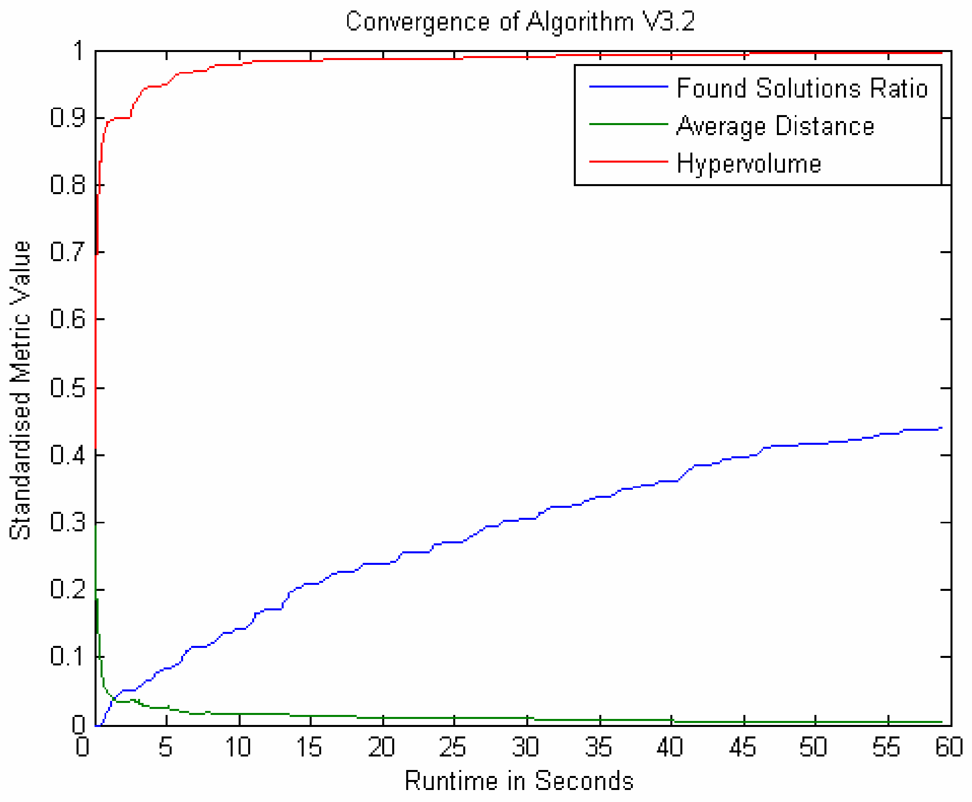

Figure D.5: Convergence Graph of Algorithm Version 3.2

E. Curriculum Vitae

Martin Petz
Anton Baumgarnerstr. 44/B2/211
1230 Wien
0681 10 217 200
martin_petz@chello.at

Personal Information

Date of Birth	27.07.1981
Place of Birth	Vienna
Citicenship	Austria

Education

1999-date	Studies of International Business Administration Specialized in Operations Research and Logistics Management University of Vienna Faculty of Business, Economics and Statistics Brünnerstraße 72, 1210 Vienna
1995-1999	Bundesoberstufenrealgymnasium Anton-Kriegergasse 25, 1230 Vienna
1991-1995	Bundesrealgymnasium Anton-Kriegergasse 25, 1230 Vienna

Employment History

2001-2002	Civilian Service Magistratsabteilung 47 Sozialer Stützpunkt f. d. 18. u. 19. Bezirk Schulgasse 19, 1180 Vienna
1998	Siemens AG Austria Abteilung Energie- und Sondertechnik Siemensstraße 88-92, 1210 Vienna

Skills and Qualifications

Foreign Languages	English, French
Software	MS Office, SPSS, SAP, Adonis Programming abilities in MatLAB
Driving Licence	2001