MASTERARBEIT

Titel der Masterarbeit

“Genetic Algorithms to solve Multi-Objective Optimization Problems - Adopting NSGA-II to a Periodic Vehicle Routing Problem”

Verfasser

Hans-Christian Zohmann, Bakk.

angestrebter akademischer Grad

Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2011

Studienkennzahl lt. Studienplan:	A 066 926
Studienrichtung lt. Studienplan:	Wirtschaftsinformatik
Betreuer:	ao. Univ.-Prof. Mag. Dr. Walter Gutjahr

Abstract

The aim of the paper is to evaluate the applicability of a multi-objective genetic algorithm to a Periodic Vehicle Routing Problems (PVRP). In the beginning of the paper the theoretical background of the areas involved will be outlined. The PVRP will be explained from a business administrative point of view, followed by a positioning of the PVRP in the area of optimization problems. In the following section, firstly, genetic algorithms in general and then multi-objective genetic algorithms will be discussed. A further discussion of the NSGA-II in detail will be followed by the presentation of approach how the genetic algorithm can be adopted to a PVRP. The main characteristic of this approach is the division of the problem into a master problem and a sub-problem. The NSGA-II will be responsible for solving the master problem which refers to providing a periodic delivery schema. The sub-problem refers to solving a Travelling Salesman Problem (TSP). The sub-problem will be solved as an Integer Linear Program (ILP) with the help of CPLEX in one case and with a Next-Neighbor heuristic enhanced by a two-opt move in the later case. After introducing the NSGA-II there will be given an overview about the structure of the actual PVRP implementation and the interdependencies to CPLEX. Then the setting up of the test environment and how the test runs are designed will be presented. In the final sections the results from the various test runs will be discussed showing the performance of the two different PVRP implementations.

Contents

1 Introduction

The aim of the paper is to adopt the NSGA-II algorithm to a Periodic Vehicle Routing Problem (PVRP) which is a special form of a Vehicle Routing Problem (VRP). The specialty of the NSGA-II is that the algorithm can optimize more than just one objective at the same time. Moreover, the NSGA-II offers a mechanism for constraint handling. Thus, the NSGA-II will be adopted to a PVRP where two objectives should be optimized simultaneously and where certain constraints need to be considered. In total, two different implementations will be provided to see how well the NSGA-II can handle the PVRP and to experience how the results differ between these two implementations.

The main question needed to be answered in order to reach the goals specified above is how the NSGA-II can be adopted to a PVRP. Therefore, in the second chapter the business background of the PVRP will be pointed out trying to give a general understanding of the problem to be solved. Moreover, the various variants and aspects of a VRP are discussed.

The third section outlines an approach to solve a PVRP. Therefore, a general background and working mode of genetic algorithms will be presented. The capability of optimizing more objectives at the same time is related to a certain group of genetic algorithms, the so-called multi-objective evolutionary algorithms (MOEAs). With the introduction of MOEAs also concepts like pareto optimality and pareto front are explained. Furthermore, a measure for evaluating the performance of a MOEA, the so-called hypervolume indicator is introduced. This measure is essential as it allows comparing the performance of different MOEA implementations. After that, the NSGA-II will be discussed in detail showing important characteristics and how the algorithm will be adopted to the PVRP.

The PVRP will be divided into a master problem and a sub-problem. The NSGA-II will be responsible for solving the master problem which refers to providing a periodic delivery schema. The sub-problem refers to solving a Travelling Salesman Problem (TSP) and will be called from within the master problem. The master problem needs the information from the sub-problem in order to provide a good periodic delivery schema. For the sub-problem two implementations will be provided. One implementation will make use of IBM ILOG CPLEX Optimizer to solve the sub-problem; the other implementation will be based on a heuristic.

In section 4 the program written to solve the PVRP will be described. This section contains a description of the main components, expected input parameters and output produced by the program.

The following section informs about the setting up of the test environment and the test procedure. The setting up of the test environment involves the design of the test instances, tuning of the PVRP implementation and planning of the test runs.

In section 6 the results of the test runs will be presented. In the final section a summary of the paper will be given.

2 Business Background of the Vehicle Routing Problem (VRP)

The idea of this chapter is to provide the reader with the necessary business background of the problem to be researched. The following sections should make clear the importance of the topic chosen and its position in the business world.

2.1 Informal introduction of the research problem

The research problem focuses on a company which is supplying its customers with goods. It is of no importance whether these goods are purchased or produced by the company itself. The paper is only looking at the aspects of distributing the goods to the customers.

The problem under discussion involves finding a periodic delivery schema which determines when to deliver to which customer with a certain amount of goods. A period could be e.g. a week running from Monday to Friday or from Monday to Sunday. A customer could be delivered once within a period, twice within a period or even every day of that period. The delivery schema to be found must be periodic which means that the schema can be adopted for the current week but also for the following weeks and is still meeting all requirements. The frequency a customer is served and the number of goods delivered to the customer must ensure that the customer does not run out of stocks.

The goal then is to find a periodic delivery schema for a network of customers which satisfies the daily demands of all the customers and at the same time minimizes transportation and inventory costs. If a 5 days period running from Monday to Friday is considered, then the transportation costs are given by 5 tours starting and ending at a depot including all the customers to be delivered. The inventory costs are a sum of all the inventory costs of all the customers for the 5 days period.

2.2 Positioning of the VRP in the Business Environment

The problem introduced above belongs to the group of Vehicle Routing Problems (VPR) and is a typical problem in the area of supply chain management (SCM) and logistics [1]. SCM deals with managing the so-called supply chain. The supply chain comprises the various phases a product goes through, from its first occurrence till the product is eventually in consumer hands. Figure 1

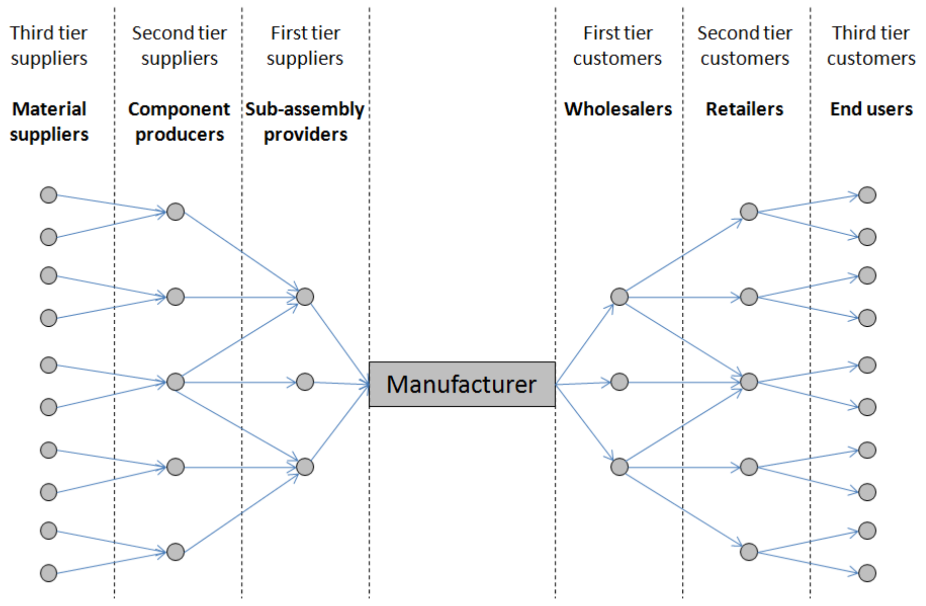

Figure 1: Typical Supply Chain [2]

shows a typical supply chain around a manufacturer [2]. To the left of the manufacturer the purchasing and assembly process for a certain product is shown, to the right of the manufacturer the distribution process until the product reaches the end users is depicted.

Considering the problem to be researched the company trying to find an optimal delivery schema for its customers could be a wholesaler (first tier customers) supplying a number of retailers (second tier customers) having certain daily demands at their premises made up by the demands of their customers (third tier customer) as shown in figure 1.

Organizing a supply chain in an efficient way is a rather complex process and, therefore, requires to be managed in a way that optimizes the benefits for suppliers and consumers. Already within an organization exist conflicting goals which hinder reaching the overall goal of an optimized supply chain. According to [2] the purchasing might look for the most reliable supplier, the inventory management might strive for low unit costs, the main goal of warehousing might be having fast stock turnover and transport management might look for full vehicle loads.

Furthermore, it is important that the organizations within a certain supply chain recognize that they share an overriding objective, namely satisfying their final customers [2]. If only one member of a supply chain fails to provide the expected service, then eventually all members of that supply chain will suffer a loss. This calls for cooperation among the members of a supply chain in order to be able to satisfy the final customers and to be able to compete with other supply chains.

2.3 The Perspective of the Customer

In this section the perspective of a retailer (see figure 1) within a supply chain is discussed. The retailer is purchasing a certain product from a supplier, keeps the product in stock and sells these products to the end consumers. The retailer wants to satisfy the end consumers by not running out of stocks, i.e. meeting the customers demands. This calls for a proper inventory management which is responsible for the control of stock levels within an organization [2].

The questions arising here are when should the retailer place an order and how much should the retailer order. These questions obviously result in a trade-off between large, infrequent orders and small, frequent orders. The first case gives high average stock levels but low costs for transport. On the other hand, the second case gives low average stocks but high costs for transport.

Keeping stocks is an inevitable necessity in order to overcome shortages or unexpected rises in demand. Stock serves as a buffer between supply and demand and gives the retailer the necessary flexibility to keep the business running smoothly. In order to minimize the overall cost for holding stock, the various types of costs involved have to be considered. According to [2] the costs of carrying stock can be divided into the following four types:

- Unit Cost: The price of a product charged by the supplier
- Reorder Cost: The cost of a repeat order for a product
- Holding Cost: The cost of keeping one unit of a product in stock
- Shortage Cost: The loss of profit if demanded product is not available

The unit cost is directly linked to the price offered by the supplier and can be lowered through price negotiations or by simply finding alternative suppliers. The reorder cost denotes administrative costs occurring when repeatingly placing orders. The holding cost not only involves the costs for keeping a product in stock but also involves the opportunity costs caused by the money tied up. The shortage cost not only involves a loss of profit for now but also a loss of reputation and potential future profits.

Having found a promising supplier, knowing the costs of carrying stock and knowing the end customers demands it is possible to find the economic

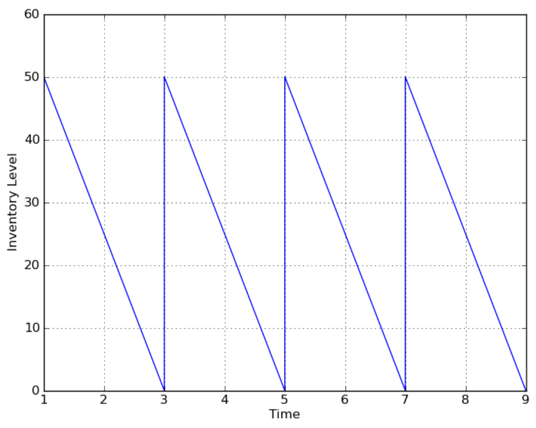

Figure 2: Inventory Model with Constant Demand Rate

order quantity (EOQ) which is the optimal size for an order in a simple inventory system [2]. The EOQ model is a deterministic inventory model where it is reasonable to assume that the rate of demand is constant [3]. A constant demand rate simply means that the same amount of items is taken from inventory every day. Considering the stock level over a longer period of time with a constant demand rate gives a repeated pattern of stock cycles as shown in figure 2. Here, the initial stock level is 50 and the daily demand amounts up to 25. This means that within two days the goods on stock are sold out which calls for a reliable supplier providing new goods on the third, fifth, seventh day and so on. One way to overcome some uncertainty regarding supply or demand is keeping safety stock.

An alternative for the retailer is to outsource the activities around managing the inventory. The outsourcing of managing inventory activities can be limited to only some activities or can comprise all activities relating to inventory management.

2.4 The Perspective of the Supplier

In this section the perspective of the supplier delivering goods to the customer from the section above will be discussed. The supplier could be either the manufacturer or the wholesaler (first tier customers) depicted in figure 1. As mentioned before in section 2.1 it is of no importance whether these goods are purchased or produced by the supplier himself. Figure 3 shows an example of the supplier's depot (blue square) and his customers (green dots).

Basically, each customer has his own demand and chooses to order when it

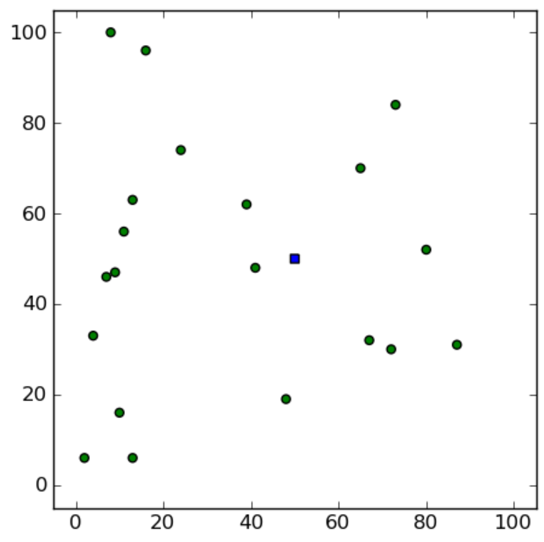

Figure 3: Supplier and Customers

suits him best according to the implemented inventory management policy. The customers manage their inventories themselves and can optimize the purchasing process individually. The supplier accepts the orders from his customers and delivers the ordered goods within the promised lead time. If the orders of the customers vary a lot and are not on a regular basis, then the supplier finds it hard to achieve optimization through proper scheduling and planning of the delivery tours. If the supplier can rely on regular orders which stay the same over periods, then improved scheduling and planning is possible.

2.5 Vendor Managed Inventory (VMI)

However, if the supplier and his customers come to an arrangement which makes the supplier responsible for finding a schedule when to deliver and how much to deliver to each customer, then an improved solution for all parties involved might be found. An example for such an arrangement between supplier and customers is the so-called Vendor Managed Inventory (VMI). According to [4] in a VMI partnership, the supplier makes the main inventory replenishment decisions for his consumers. The vendor monitors the inventory of the buyer and makes periodic resupply decisions regarding order quantities, shipping and timing.

As in the VRP announced in section 2.1 the supplier knowing the average demand and the current stock of all customers can start to optimize the delivery process. By deciding when to deliver and how many goods to deliver, the supplier can ensure full truck loads, optimized tours as well as ensuring that the customers do not run out of stock. Moreover, the customer neither needs to monitor his stock, nor to order goods anymore.

The idea is to reduce costs (transportation costs, inventory costs), but at the same time guaranteeing a high service level. Moreover, the supplier has more security that he can deliver on time as he is responsible for scheduling, thus, increasing security for the customers that they will obtain the goods on time as well.

2.6 Defining the Vehicle Routing Problem and its Variants

The VRP is a generic name for a whole class of problems in which a set of routes for one or more vehicles based at one or several depots must be retrieved for a certain number of customers. There exist numerous variants of the classical VRP depending on certain features and characteristics. Combinations of the following characteristics/features form the different variants of the VRP:

- one vehicle or multiple vehicles
- vehicles with or without capacity
- homogenous or heterogeneous vehicle fleet
- one depot or multiple depots
- planning period finite or infinite/periodic
- each customer is visited exactly once or multiple times within a delivery tour
- customers must be visited within certain time windows
- vehicle can pickup goods at depot between deliveries
- vehicle picks up at customer and delivers to customer
- supplier manages customer's inventory

The list above is not complete but shows the most important characteristics which make up the variants listed and described below. Extending the classic VRP by one or more of the features from above leads to a VRP of higher complexity.

Multi Depot Vehicle Routing Problem (MDVRP) In the MDVRP the focus lies on the existence of multiple depots responsible for storing the products and serving as starting and end point for vehicle tours. In problems with only one depot all the vehicles start and terminate their tours at the same depot, whereas, in a MDVRP vehicle tours start and end in different depots. The MDVRP is NP-hard which means that it is not possible to find an optimal solution within reasonable time [5].

Split Delivery Vehicle Routing Problem (SDVRP) The main characteristic of this VRP in contrast to the other variants is that each customer can be visited more than once. As a consequence the demand of a customer may be greater than the vehicle capacity and a customer may be served by more than just one vehicle from a fleet. The objective of the CVRP is to find a set of vehicle routes that minimizes the total distance travelled by not exceeding the vehicle capacity for each tour [6].

Periodic Vehicle Routing Problem (PVRP) The idea behind the PVRP is to construct vehicle routes over multiple days within a planning period. Each day of the planning period a fleet of capacitated vehicles delivers the customers. The objective of the PVRP is to find a set of tours which minimizes the total travel cost taking into consideration the vehicle fleet capacity and visit requirements [7].

VRP with Time Windows (VRPTW) The VRPTW is a classic VRP extended by time windows, i.e. that the customers must be delivered within a certain time. This results in the same objective as in the classic VRP which is finding a route for the vehicles to serve all customers minimizing the travel costs. However, in addition to the tour capacity constraint, also the time window constraint defined by the customers must be considered [8].

VRP with Backhauls (VRPB) Extending the classical VRP with backhauls allows a vehicle to pickup goods on the return trip after delivers have been made. Thus, the objective is to minimize the total distance travelled over a set of tours [9].

VRP with Pick-Ups and Deliveries (VRPPD) In the classical VRP the vehicles pick up the goods at a depot and then deliver the goods to the customers which means that all goods to be delivered originate from a depot. In case of a VRPPD, however, goods need to be delivered to customers but also need to be picked up at customer locations. The objective and constraints stay the same; the total distance travelled should be minimized considering maximum distance and maximum capacity constraints. NP-hard [10].

Inventory Routing Problem (IRP) The IRP shares the routing problem with all the VRPs above which means shipping products from a supplier to customers minimizing the transportation costs. In case of the IRP the supplier does not only need to consider the demand of the customers but also inventory, limited storage capacity and stock holding costs. Thus, the supplier is responsible for the managing the inventory at the customers avoiding stock-outs. Furthermore, the objective function has to take into account the stock holding costs [11].

Considering the problem to be researched as described in 2.6 the following characteristics/features can be determined:

- one vehicle with capacity
- one depot
- each customer is visited exactly once within a delivery tour
- planning period is infinite/periodic
- supplier manages customer's inventory

The the PVRP and the IRP are the two variants which match the problem to be researched most precisely. The problem to be researched and the PVRP have in common the characteristic of being periodic. Periodic means that a feasible solution follows a certain cyclic pattern and can be adopted continously for multiple periods guaranteeing the same results provided that the problem parameters do not change.

Except for the IRP all the VRP variants introduced above have exactly one objective function which is to minimize the distance travelled. In case of the IRP not only the distance travelled should be minimized but also the stock holding costs. This results in either one aggregate objective function consisting of travelling costs and stock holding costs or in two objective functions, one objective function for the travelling costs and one objective function for the stock holding costs.

As being the supplier managing the inventories of the customers in addition to delivering the products, it is vital to consider the transportation costs independently from the stock holding costs. The supplying party will bear the transportation costs and each customer will bear the stock holding costs occurring at his warehouse. Considering these two objectives independently allows the parties involved to evaluate whether a found solution is in favor of the supplier or the customers. Therefore, a solution approach is needed which can handle minimizing two objective functions at the same time.

In this paper the problem to be discussed will always be called a PVRP as the planning horizon is more than just one period. However, it needs to be kept in mind that the actual problem is a special variant of a PVRP. The PVRP is extended by the characteristic that the supplier is responsible for managing the customers' inventory (IRP). Considering this, it can be realized that this constellation represents a VMI as defined in 2.5.

3 Approach to Solving Vehicle Routing Problems

This chapter describes an approach based on an algorithm called Non-dominated Sorting Genetic Algorithm II (NSGA-II) to solve the Periodic Vehicle Routing Problem defined in chapter 2. The NSGA-II belongs to the group of genetic algorithms, more specifically, the algorithm belongs to the group of multi-objective evolutionary algorithms. Therefore, first genetic algorithms and multi-objective evolutionary algorithms will be explained. After a general understanding of the working mode of genetic algorithms the approach based on the NSGA-II to solve a PVRP will be put forward.

3.1 Genetic Algorithms

From the second half of the 20th century on the natural evolution has become a source of new strategies for solving optimization problems [12]. In the scientific world [8] the development of genetic algorithms is attributed to Holland [13]. Concepts from the nature have been adapted to match the requirements of solving optimization problems in areas such as logistics, production, medicine, etc. Genetic algorithms are simply simulating the process of the natural evolution.

According to [14] the main concept behind the natural evolution can be explained as the ability to survive through adaption to a changing environment. The natural evolution incorporates an innovative problem solution mechanism that allows to efficiently find solutions for complex problems.

The following list shows a selection of terms which appear in the natural evolution as well as in the area of evolutionary algorithms (see Table 1):

Name	Natural Evolution	Evolutionary Algorithms
Population:	set of humans/anmials	set of chromosomes/individuals
Chromosome:	blueprint of human/animal	string of characters or numbers
Gene:	part of a chromosome	character, feature, variable
Allele:	specification of a gene	value of a character
Fitnes:	survivability of human/animal	goodness of a chromosome
Reproduction:	production of offsprings	creation of new chromsomes
Generation:	population at a certain point of time	same as for natural evol.

Table 1: Evolutionary Vocabulary in Nature and Computer Science [14]

Genetic algorithms work with a population of chromosomes or also so-called individuals. A chromosome/individual represents a possible solution to a certain problem and is expressed by a string of characters or numbers.

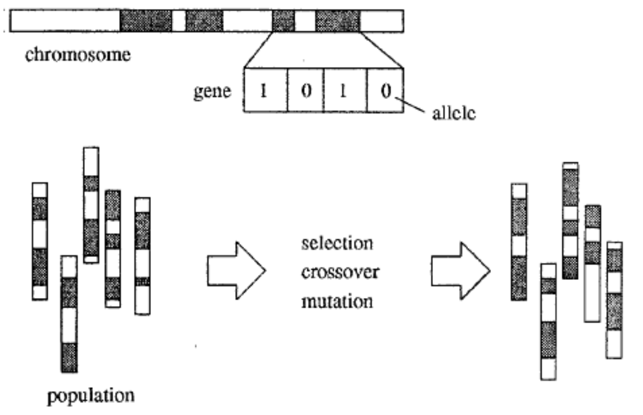

Figure 4: Basic components of a genetic algorithm [8]

Figure 4 shows an example of a simple chromosome structure consisting of binary genes having values '0' or '1'. Depending on the environment or a specific problem some chromosomes of a population will perform better, some others will perform worse. Therefore, each chromosome is assigned a fitness score which evaluates a chromosome regarding a specific problem.

Furthermore, figure 4 shows the reproduction process of a genetic algorithm by undertaking selection, crossover and mutation operators. In this reproduction process highly fit chromosomes/individuals of a population are selected (selection), recombined (crossover) and partly manipulated (mutation) in order to build an evolved population of new possible solutions.

The process of deriving a new population from an original population is called generation and is repeated again and again. With an increasing number of generations the evolving population will contain a higher proportion of characteristics possessed by good performing individuals from previous generations. In the course of the evolution of the population the individuals with a bad evaluation will die out. According to [15] the most promising areas of the search space are explored by favoring the mating of the more fit individuals. The procedure of a simple genetic algorithm can be summarized as follows:

1. Randomly initiated population
2. Calculation of fitness
3. Selection
4. Execution of genetic operators (recombination/mutation)

3.2 Multi-objective Evolutionary Algorithms (MOEAs)

As discussed in section 2.6 a solution approach is required which allows the optimization of two objectives simultaneously. In the literature [16] a problem with multiple objective functions to be optimized is called multi-objective optimization problem (MOOP). Fonseca and Fleming [16] point out that evolutionary algorithms (EAs) can handle multi-objective optimization problems (MOOPs) very well. EAs inherently find multiple pareto-optimal solutions in one single simulation run since they work with a set of solutions in each generation, thus, an EA can be extended in a way to maintain multiple solutions [17].

3.2.1 Pareto optimality

As in case of MOOP where more objectives at a time are considered, there does not exist one single optimal solution. In contrast to single objective optimization problems, there exists rather a set of alternative solutions. These alternative solutions within a search space are also called Pareto-optimal solutions or Pareto-optimal front. The model for a MOOP can be represented by a minimization/maximization function described as a vector function f that maps m decision variables to n objectives [16, 18, 19, 20]:

$$\begin{aligned}
&min/max\ y = f\left(x\right) = \left(f_1\left(x\right), f_2\left(x\right), \ldots, f_n\left(x\right)\right)\\
&subject\ to\ x = \left(x_1, x_2, \ldots, x_m\right) \in X\\
&\qquad\qquad y = \left(y_1, y_2, \ldots, y_m\right) \in Y
\end{aligned}$$

where x is called the decision vector and X is the parameter space; y is the objective vector and Y is the objective space. In [18] Pareto optimality is defined as a set of solutions of a MOOP consisting of all decision vectors for which the relevant objective vectors cannot be improved in any dimension without decreasing any other. Considering a minimization problem a decision vector $a \in X$ is said to dominate a decision vector $b \in X$ (written as $a \succ b$) only if:

$$\begin{aligned}
\forall i \in \{1, \ldots, n\}: &\quad f_i\left(a\right) \leq f_i\left(b\right) \quad \wedge\\
\exists j \in \{1, \ldots, n\}: &\quad f_j\left(a\right) < f_j\left(b\right)
\end{aligned}$$

Following this, if a decision vector $a \in X$ dominates a decision vector $b \in X$, any member of decision vector a is said to be non-dominated or non-inferior [20].

3.2.2 Measuring Pareto Optimality

As MOOPs do not deliver one single solution consisting of a single scalar value but a non-dominated collection of vectors the question arises whether there exist quantitative measures revealing information about the quality of the outcomes. Due to the fact of optimizing more than just one objective at the same time and obtaining more than one solution within a test run, the need for special measures arises.

In case of a standard TSP a test run delivers only one objective value. Consequently, various problem instances of the TSP can be easily compared with each other with the help of the objective value. When having a minimization problem simply the problem instance with the lowest objective value dominates the other problem instances.

In case of the NSGA-II solving a PVRP, there are two objective functions (Transportation and Storage costs) involved and usually there is more than just one possible solution making up the pareto front. This calls for a metrics indicating the performance of an optimization technique allowing the comparison of different multi-objective optimization implementations. However, Zitzler et al [21] emphasize the difficulty of defining a quality measure for multi-objective optimization problems as the optimization goal itself consists of multiple objectives. Therefore, in [21] the following aspects are suggested which define a good solution of a MOOP:

- Minimization of the distance of the resulting non-dominated set to the Pareto-optimal front
- A good (in most cases uniform) distribution of the solutions is found desirable.
- Maximization of the extent of the obtained non-dominated front, i.e., for each objective a wide range of values should be present

In this paper the focus will be laid on the hypervolume indicator (also known as Lebesgue measure or S metric). According to [22] the hypervolume measure is one of the most frequently applied measures. The hypervolume measure takes into consideration the size of the objective value space which is covered by a set of non-dominated solutions (e.g. A, B and C as shown in figure 5). In order to be able to calculate the multi-dimensional regions made up by the non-dominated solutions an arbitrary reference point needs to defined. By adding together the areas covered by the Pareto-optimal solutions a measure is obtained which allows comparing the performance of different multi-objective optimization implementations.

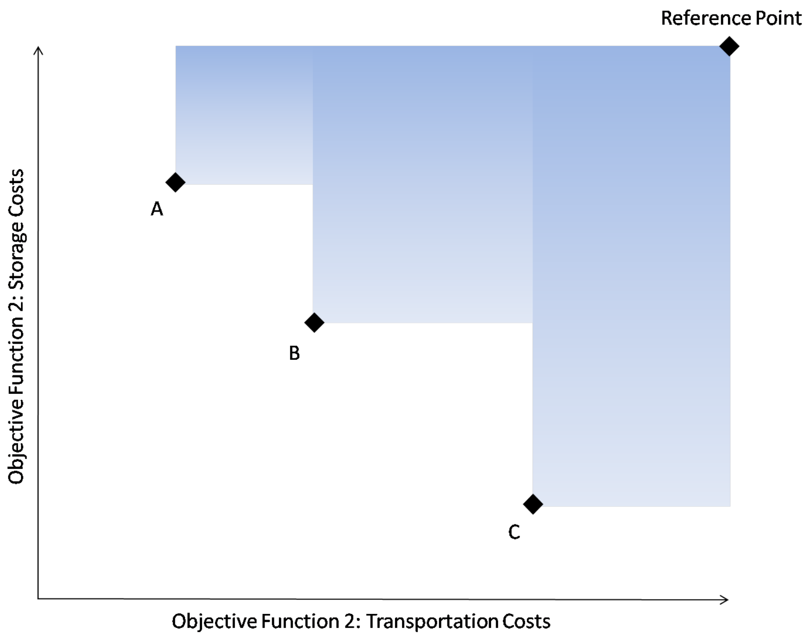
Reference Point
A
B
C
Objective Function 2: Storage Costs
Objective Function 2: Transportation Costs

Figure 5: Hypervolume Indicator

However, in [23] the weakness of the hypervolume measure is pointed out as the choice of the reference point can have an effect on the evaluation of the performance of various non-dominated sets. In case of the PVRP introduced in this paper, a reference point will be chosen which refers to the maximal transportation and storage costs occurring in a test instance.

3.2.3 Introducing the NSGA-II

The NSGA-II [17] is an improved version of the NSGA presented in [20] and was chosen to be adopted for the PVRP as the NSGA-II is reported to deliver promising results when tackling multi-objective optimization problems. The main improvements of the NSGA-II approach can be summarized as follows [17]:

Better non-dominated sorting algorithm The original implementation of the non-dominated sorting used in the NSGA was criticized for its computational complexity of $O(MN^3)$ where M refers to the number of objectives and N is the population size. The execution of the highly complex non-dominated sorting in every generation caused the NSGA to be computationally expensive for large population sizes. The new

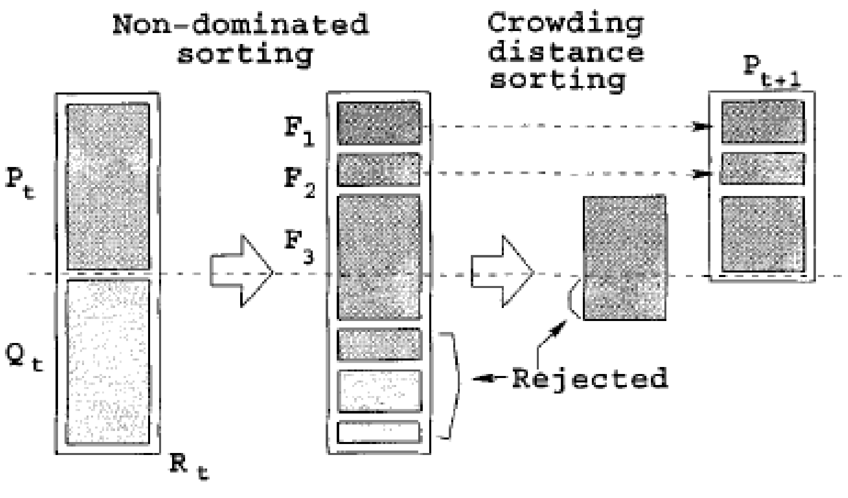

Figure 6: NSGA-II procedure [17]

implementation of the non-dominated sorting in the NSGA-II reduced the time complexity to $O\left(MN^2\right)$.

Introduction of elitism In the NSGA-II elitism was introduced which results in speeding up the performance of the NSGA-II and helps to keep good solutions once found.

Elimination of the need for specifying a sharing partner In order to obtain a good spread of solutions a sharing function method involving a sharing parameter is used in the NSGA. One problem with this approach is that the performance of the sharing function method largely depends on the chosen value for the sharing parameter. Moreover, the sharing function method is highly complex ($O\left(N^2\right)$) as each solution must be compared with all other solutions in the population.

In the NSGA-II the sharing function approach is replaced with a crowded-comparison approach. Now, the NSGA-II does not require the specification of a parameter for maintaining diversity and the computational complexity is reduced as well.

Basically, the NSGA-II varies from a simple genetic algorithm only in the way the selection operator works and the enhancements just described above. The crossover and mutation operators work the same way.

Figure 6 shows the overall procedure of the NSGA-II. After the initialization of a population as known as from genetic algorithms, the population is sorted based on non-domination into several fronts ($F_1, F_2, F_3, etc.$). The so-called first front is a completely non-dominated set, i.e. none of the individuals from the first front are dominated by any other individuals. The individuals of the second front are dominated by the individuals of the first front, but dominate all the individuals from the subsequent fronts. The same is valid for the next fronts. The individuals from the third front are dominated by the individuals from the previous fronts and so on. Every individual in each front is assigned a fitness value which is called rank by Deb et al. [17]. The value of the rank assigned depends on the front of an individual, i.e. individuals from the first front have rank 1, and individuals from the next front have a rank incremented by one and so on.

Furthermore, the NSGA-II considers a measure called crowding distance which indicates the distance of an individual to its neighbors. With the help of the crowding distance the NSGA-II tries to ensure better diversity in the population. The crowding distance is assigned front-wise and basically, represents the Euclidean distance between the individuals based on their m objectives in a m dimensional hyper space.

As soon as the individuals are sorted based on non-domination and the crowding distance is computed for all the individuals, then a binary tournament selection with a crowded-comparison-operator is carried out. Parents are selected from the population by using binary tournament selection based on the rank and the crowding distance. An individual is selected if the value of the rank is smaller or if the crowding distance is greater than the other. The selected population generates offsprings from crossover and mutation operators. The population with the current population and current offsprings is sorted again based on non-domination and only the best individuals are selected according their rank and crowding distance.

3.3 Implementing the NSGA-II as a PVRP

As described before in 2.6 the goal of the PVRP is to find a set of tours which minimizes the total costs travelled and the stock holding costs simultaneously. This kind of problem is a typical MOOP as defined in 3.2 and therefore, perfectly suits the requirements of a genetic algorithm, such as the NSGA-II. However, the NSGA-II needs assistance in solving the defined problem.

Therefore, the solution approach will be decomposed into two phases. In the first phase each customer will be assigned for each day within the planning period whether the customer will be serviced or not. The NSGA-II will be responsible for this assignment. In the second phase each tour according to the assignment of phase 1 will be optimized, thus, leading to an improved overall solution. Finding an optimal tour is known as a Travelling Salesman

Problem (TSP). For solving the TSP two different approaches haven been implemented: (1) Integer Program and (2) 2-opt move based heuristic.

This 2 phase approach can also be seen as a master problem (NSGA-II) and a sub-problem (TSP).

3.3.1 Problem Formulation

The specialty of the specified problem can be seen in the fact that not only one but two objectives are to be optimized at the same time. Minimizing the transportation costs automatically leads to higher storage costs. At the same time minimizing the storage costs leads to higher transportation costs.

Objective: Minimizing transportation and storage costs

Feasibility: The solution is feasible if all constraints of the specified VRP are satisfied, i.e. deliveries must be scheduled in a way so that the daily stock levels do not fall below a certain limit. Therefore, there exists one constraint ensuring that each customer is service at least once a week. Additionally, there exist 5 more constraints (Mon - Fri) ensuring that the daily delivery amount does not exceed the truck capacity.

Formulation: Minimizing the total costs of all tours while minimizing the costs for holding stocks. The tour with minimum costs for each day is found with the help of the sub-problem.

3.3.2 Chromosome Representation

The chromosomes (also called individuals) can be represented by characters or numbers. In case of the actual PVRP a binary representation was found to be the best way to describe a solution. A binary variable determines whether a certain customer is delivered on a certain day of a period. Table 2 shows the binary representation for a PVRP with two customers and a delivery period from Monday to Friday. The number of customers and the period length define the number of binary variables (number of customers x period length). According to table 2 customer 1 is delivered on Monday and Wednesday; customer 2 is delivered on Tuesday and Thursday.

Customer 1					Customer 2				
Mon	Tue	Wed	Thu	Fri	Mon	Tue	Wed	Thu	Fri
1	0	1	0	0	0	1	0	1	0

Table 2: Chromosome Representation

3.3.3 Generation of Initial Population

The population size defines how many individuals are computed for each generation and is expected as an input parameter. Following this a population size of 4 requires the generation of 4 individuals for the first generation and all the following generations. This initial population is generated randomly and could look like as follows for a population size of 4:

```
Individual 1: 00001 00101 11110 00111 00000
Individual 2: 10000 10011 11100 10111 01111
Individual 3: 01010 00101 00010 00010 11011
Individual 4: 10001 11010 01101 01110 00101
```

From the above chromosome representation it can be learnt that a pattern should be found how to supply 5 customers for a period of 5 days. Each block represents a delivery schema for a certain customer. The spaces are not necessary; they only should help the reader to realize which binary variables belong to a customer.

3.3.4 Selection Process

With the help of a mechanism called binary tournament selection [24] individuals from a population will be randomly chosen as parents being responsible for mating and reproduction. In case of the binary tournament selection tournaments are held between pairs of individuals, i.e. always 2 individuals will be randomly chosen and compared with each other. Always the dominating individual will be selected as parent. If two individuals are evaluated equally then there is a 50 percent chance for both of them to be selected for further processing.

For a population size of 4, 2 pairs of individuals will be selected and the winner of each tournament becomes a parent. Below it can be seen that individual 2 was chosen to compete with individual 1 and individual 1 proved to be a better solution and therefore was select as a parent. Among the two remaining individuals, individual 4 proved to be the better solution. This process happens a second time for the original population, thus in the end there are four parents. As the pairs are randomly selected the same constellation as shown below can be the consequence. In both cases individual 1 and individual 4 are selected as parents. Individual 1 dominates individual 2 and individual 4 dominates individual 3.

```
Individual 2 vs. Individual 1: Individual 1
Individual 3 vs. Individual 4: Individual 4
Individual 2 vs. Individual 1: Individual 1
Individual 3 vs. Individual 4: Individual 4
```

Due to the selection process chosen the population size for the implementation of the NSGA-II must always be a factor of 4. This way it is ensured that enough parents are selected from the population, thus, allowing the generation of a sufficient number of children keeping the population size constant.

3.3.5 Reproduction

The reproduction process is responsible for combining useful traits from parent chromosomes and passing them on to the children [8]. The reproduction process involves crossover and mutation. The idea behind a crossover is to move toward promising regions of the search space by matching good parents to construct better offsprings. On the other hand, with the help of mutation it is thought to avoid over homogenous populations by randomly swapping binary variables of an individual.

In order to perform crossover the NSGA-II requires a crossover probability as an input parameter. The crossover is applied to two parents. For each position there exists the given probability that a crossover will be executed which simply means exchanging the values of the binary values of parent 1 and parent 2 at the relevant position. For example, a crossover rate of 0.2 means that there is a 20 percent chance that the binary variables of two parents at a certain position become eligible for a crossover. Once the binary variables of two parents are eligible for a crossover, then there is a 50 percent chance of exchanging the binary variables of the two parents. If the binary variables of the two parents have the same value, then the crossover does not have any effect.

As shown below the crossover applied to two parents results in 2 offsprings. The mark (C) indicates at which position a crossover has been carried out. The possible swaps are: (0,1) to (1,0), (1,0) to (0,1), (0,0) to (0,0) and (1,1) to (1,1).

```
Parent 1: 00001 00101 11110 00111 00000
Parent 2: 10001 11010 01101 01110 00101
          C C   C      C    CC  C   C C
Child  1: 10001 10101 11110 01110 00101
Child  2: 00001 01010 01101 00111 00000
```

For accomplishing mutation the NSGA-II also needs a mutation probability as an input parameter. This parameter defines the probability of changing the value of an individual's binary variable from either 0 to 1 or from 1 to 0. After having executed the crossover operands, then the mutation operand will be applied to all individuals of a population. Below an example is shown with an assumed mutation probability of 0.2. The binary variables marked with 'M' indicate a successfully mutated gene.

```
Individual: 10001 10101 11110 01110 00101
               M      M  MM   M  M  M
Individual: 10011 10100 10010 11100 10101
```

As shown, with the selection and reproduction operators the NSGA-II can find more and more promising solutions. In case of the PVRP implementation based on the NSGA-II, the master problem implementation just presented relies on the implementation of a sub-problem which will be introduced in the next section.

3.4 Solving the Sub-Problem: TSP

The sub-problem which needs to be solved within the execution of the PVRP is a TSP and is called by the master problem. The master problem provides all parameters which are needed in order to solve the problem. The parameters provided involve the distances between the relevant customers and the depot. As the aim of this paper is to compare the results of the PVRP with two different TSP implementations, two different implementations will be presented here. One approach is based on CPLEX and delivers the optimal solution. The second approach is a heuristic which does not necessarily give an optimal solution but promises to deliver good results.

3.4.1 Problem Formulation

Here, in contrast to the master problem only one objective needs to be minimized.

Objective: Minimizing transportation costs for a given number of customers

Feasibility: The solution is feasible if all constraints of the specified TSP are satisfied, i.e. the depot and each customer in the given tour is visited exactly once

Formulation: Finding the sequence of customers to be visited which minimizes the transport costs

3.4.2 Sub-Problem: Integer Program

The first implementation of the TSP was done with the help of IBM ILOG CPLEX Optimizer[1]. IBM ILOG CPLEX Optimizer is a software for solving linear optimization problems and will be simply referred to as CPLEX in this paper. The TSP under discussion consists of a set of customers (n) and the transportation costs (vector c) between these customers. The TSP was modeled as an Integer Linear Program (ILP) as follows in CPLEX considering a binary decision variable x:

Objective:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Subject To:

$$\sum_{j=1}^{n} x_i = 1(i = 1, \dots, n)$$

$$\sum_{i=1}^{n} x_j = 1(j = 1, \dots, n)$$

$$x_{ij} \in \{0, 1\}$$

The idea is to find a set of edges which guarantees the minimal travelling costs (objective function) and connects all nodes involved (constraints). However, the solution returned by CPLEX is not necessarily a valid TSP solution. Below there are shown two possible solutions for a TSP problem with 6 nodes involved. Both solutions are valid regarding the ILP model from above, but only one of them is a valid TSP solution.

```
Valid TSP tour:     (1 2 3 4 5 6)
Not valid TSP tour: (1 2 3)(4 5 6)
```

In both cases the solution adheres to the constraints so far and the truck would visit each customer exactly once. In the second case the solution does not consist of one but of two tours. Therefore, it must be checked whether the solution returned by CPLEX is a valid TSP solution or not. If the solution

[1]http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (last access: 23.04.2011)

consists of more than one sub-tour, then constraints need to be added to the model from above in order to avoid the generation of found sub-tours. After the modification of the model, the ILP will be executed once again. It will be checked if there still exist sub-tours in the returned solution. If not, then the optimal and valid TSP solution was found, otherwise the model needs to be adapted again and again until a solution without sub-tours is provided by CPLEX.

3.4.3 Sub-Problem: Nearest-Neighbor Heuristic

The alternative implementation of the TSP is based on the nearest-neighbor which is improved by the 2-opt move. The nearest-neighbor algorithm belongs to the group of greedy heuristics and is straightforward. The next customer to be serviced is always the one who incurs the minimal travelling costs for the supplier. Once a customer was serviced within a tour, then the customer will not be considered anymore. The truck starts at the depot and transports the products always to the nearest customer. As soon as all the relevant customers were serviced, the truck returns to the depot.

In order to increase the efficiency of the nearest-neighbor algorithm the heuristic was enhanced by the so-called 2-opt move. After an initial generation of a tour with the nearest-neighbor heuristic, the tour is scanned for improvement possibilities. The idea behind the 2-opt move is that a given tour could be easily improved if two nodes are exchanged with each other. Following this, for each node of a given tour a simulated exchange with every other node is carried out. If the simulated exchange promises a better overall solution of the TSP, then the exchange of two nodes will be processed for real. The heuristic scans for better solutions until the simulated exchanges do not deliver improvements anymore.

This implementation is not expected to deliver the optimal solution of a TSP, but still should deliver good results in relatively short time. In later sections this implementation of the sub-problem will be simply referred to as 2-opt move heuristic.

4 Program Description

The PVRP was implemented as a C++ program and is composed of the following components:

Main component	C++
NSGA-II [25]	C++
TSP CPLEX	C++ and CPLEX
TSP 2-opt move	C++

Table 3: PVRP Components

The main component takes care of reading the input parameters (see table 4) and the test instance (see table 5). Furthermore, the test instance data is prepared in a way to satisfy the needs of the NSGA-II and the TSP implementations. This involves converting the x and y coordinates of the customers to a vector consisting of the distances between the customers. In case of a test instance with 20 customers and 1 depot, then a vector of size 210 is created to hold all the distances. Assuming $n = 21$ the size of the vector is determined as follows: $(n + (n - 1))/2$.

The NSGA-II specific implementation is based on the NSGA-II presented by Deb et al. [25]. The C implementation[2] provided by Deb was converted into pure C++ code and modified in a way to satisfy the needs of the PVRP implementation, especially the minimiation function of the problem (see 3.2.1 and 3.3.1).

The PVRP implementation controls the program flow and the interaction between the NSGA-II specific implementation and the TSP implementation. Depending on the TSP solver input parameter (see table 4) the PVRP implementation will make use of CPLEX or the 2-opt move heuristic to solve the TSP.

The program will be started with the parameters defined in table 4 and generates as many generations as defined in the input parameter 'Generation'.

4.1 Input

The program expects the following input parameters as specified in table 4. According to the instance parameter a certain instance file is read into the program and prepared for the PVRP. The instance file to be read needs to follow the structure as defined in table 5. The data defined in the instance file

[2]http://www.iitk.ac.in/kangal/ (last access: 23.04.2011)

Generation	Defines how many generations to be considered (stop criterion)
Population size	Defines the population size; must be a factor of 4
Mutation rate	Specifies the probability of the mutation rate (must be between 0 and 1)
Crossover rate	Specifies the probability of the crossover rate (must be between 0 and 1)
TSP solver	Determines the TSP implementation to be used (0=CPLEX; 1=2-opt Move heuristic)
Random seed	Specifies the random seed for the random number (must be between 0 and 1)
Instance	Specifies the instance to be optimized

Table 4: PVRP Implementation - Expected input parameters

describe the problem to be solved. Additionally, the reference point for the stock holding costs needs to specified in the instance file. The reference point for the transportation costs does not need to be specified as this reference point is calculated in the initialization phase of the PVRP program.

No of customers	Specifies the size of the problem
Period length	Specifies the length of the planning period
Truck capacity	Specifies the maximum a truck can deliver a day
Reference point (storage costs)	Specifies the reference point for the storage costs
X-Y coordinates	For each customer the x and y coordinates are specified
Demand	For each customer the daily demand is specified
Storage costs	For each customer the storage costs per item are specified

Table 5: Structure of instance file

4.2 Output

The output of the program is a file showing the final feasible population. For each individual of the population the parameters from table 6 are shown. For each individual two objective values (transportation and storage costs) are displayed. Moreover, the service constraint and the capacity constraint is shown. Constraints with a negative value indicate that the individual found does not represent a feasible solution. The subsequent binary string refers to the periodic delivery schema. The last two values are the rank and the crowding distance of the individual.

Objective values	Shows the transport and storage costs
Service constraint	Shows the value of the constraint ensuring that each customer is supplied at least once
Capacity constraints	Shows the values of the 5 constraints ensuring that the truck capacity is not exceeded on one day of the period
Binary string	Represents which day each customer is serviced
Rank	Shows the rank of the individual
Crowding distance	Shows the value of the individual's crowding distance

Table 6: PVRP Implementation - Output file

A second output file logs information about every test run as shown as in table 7. Besides the basic parameter settings, the logged information contains the runtime a test run needs to compute all generations and the hypervolume of the solution found. Moreover, the reference point for the transportation costs is included in the file.

Runtime (seconds)	Shows time needed to compute all generations
Hypervolume	Shows the performance of the test run (see 3.2.2)
Generations	Number of computed generations
Population size	Shows the population size of test instance
Mutation rate	Shows mutation rate of the test run
Crossover rate	Shows crossover rate of the test run
Random seed	Shows the random seed of the test run
Instance	Refers to the relevant instance file
TSP solver	Shows the TSP implementation used
Reference Point (transportation costs)	Shows the reference point for the transportation costs

Table 7: PVRP Implementation - Log file

5 Setting up of a Test Environment

5.1 Test Procedure

As defined in the introductory chapter the aim of the paper is to evaluate to which degree the NSGA-II heuristic is applicable for a PVRP. Therefore, a number of tests need to be executed in order to find out how well the NSGA-II performs on the PVRP. The setting up of the test environment contains the steps as follow and is discussed in detail in the subsequent sections:

1. Design of Problem Classes
2. Parameter Tuning
3. Execution of Test Runs

5.2 Design of Problem Classes

Table 8 shows the parameters defining the various problem classes of test instances. In total there exist 16 problem classes of test instances. The PVRP as defined in the introductory chapter restricts the number of depots to one. As a consequence there exists only one depot in each test instance.

The number of customers is set to either 20 or 30 in the test instances to be generated. The reason for having two size ranges is to see how the PVRP implementations perform on larger problems compared to smaller ones. The numbers of customers chosen should neither be too small nor too large. The number should not be too small as it is tried to create test instances close to the real world problem. However, the number of customers should not be too large in order to avoid too long and complex test runs. In one half of the test instances the customers will be distributed evenly, whereas in the other half of the test instances the customers will be clustered around certain points.

The spread of the demand will be weak in one half of the test instances and strong in the second half. The truck capacity will be 50% above the daily average demand for the test instance used for parameter tuning; a truck capacity of around 33% and 66% above the daily average demand will be used for the later test runs. The daily average demand is the amount of goods needed to be delivered every day to satisfy all the customers within a period. An increasing truck capacity above the daily average demand results in a larger tolerance and, thus, relaxes the tour capacity constraint.

The depot and the customers will be located in a map of size 100 $\times$ 100. For all test instances the depot will be located at the point ($x = 50; y = 50$). In case of the evenly distributed customer landscapes the x and y variables

Number of Depots	1
Number of Customers	20 or 30
Distribution of Customers	Evenly distributed or clustered
Spread of Demand	Weak [40. . . 80] or strong [20. . . 100]
Truck Capacity	33%, 50% or 66% above daily average demand

Table 8: Design of Problem Classes

will be randomly generated. In case of the clustered distribution of customers, first, in each quadrant of the 100 × 100 map one point will be randomly generated and then, the customers will be randomly located around these four points.

The range of the weak spread of the demand is within 40. . . 60 and the range of the strong spread is within 20. . . 80. The truck capacity directly depends on the randomly generated demand as the daily average demand serves as the basis for the calculation of the truck capacity.

5.3 Parameter Tuning

The objective of parameter tuning is to find a parameter setting which allows the algorithm to find better solutions. Test runs with various parameter settings are necessary as parameters such as population size, mutation and crossover rates differ strongly depending on the underlying problem [14].

For the parameter tuning one random test instance with 20 customers and one depot has been generated. Table 9 shows the parameters for the generation of the tuning instance. The parameters are chosen in a way to generate an average test instance. Figure 7 shows the tuning instance with the location of the depot (blue square) and all the 20 customers (green dots) to be served.

Number of Depots	1
Number of Customers	20
Distribution of Customers	Evenly distributed
Spread of Demand	Strong [20. . . 80]
Truck capacity	50% above daily average demand

Table 9: Design of Tuning Instance

The tuning instance is then used to find the optimal parameter setting considering the population size, mutation and crossover rate. With the help of the hypervolume indicator (see 3.2.2) the most promising parameter setting

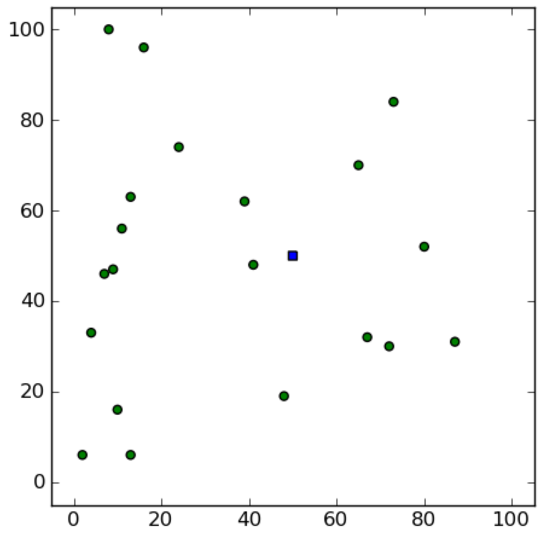

Figure 7: Tuning instance

will be identified. The values of the reference parameters for the tuning instance are:

1. Reference value - Transportation Costs: 189305
2. Reference value - Storage Costs: 478400

The parameter tuning was accomplished in two phases. The first phase was a broad search for the area of best parameter settings with a smaller number of generations and population size. The second phase was responsible for confirming the results from the first phase and searching more thoroughly the confirmed areas with a larger number of generations and population size.

In the first phase, in total 3 test runs with the following seeds were executed: 0.5, 0.6 and 0.7. The number of generations was limited to 60 and the population size chosen was 100. For each test run all the combinations shown as in table 10 were processed resulting in a total of 108 executions of the NSGA-II.

Table 11 shows the best 5 parameter combinations from a multiple test runs with changing parameter settings and random number seeds. The four columns denote the parameter settings (mutation and crossover rate), the hypervolume and standard deviation. The hypervolume column refers to the average hypervolume over the 3 test runs and the following column shows its

Mutation	Crossover
0.0	0.0
0.2	0.2
0.4	0.4
0.6	0.6
0.8	0.8
1.0	1.0

Table 10: Tested Parameter Combinations

standard deviation. These results clearly show that the current implementation performs best with a mutation rate of 0 and a high crossover rate. The best performing parameter setting with a hypervolume of 0.4474 as shown in the table 11 has a mutation rate of 0 and a crossover rate of 0.8.

Mutation	Crossover	Hypervolume	Std. Deviation
0	0.8	0.4474	0.0140
0	0.6	0.4402	0.0172
0	0.4	0.4355	0.0184
0	1.0	0.4310	0.0146
0	0.2	0.4129	0.0058

Table 11: Best 5 Parameter Combinations - Tuning Phase 1

These results indicate that the NSGA-II implementation delivers the best solutions when no mutation operators at all are performed. In the second phase of parameter tuning it will be analyzed whether the results of the first phase can be confirmed or not. Moreover, the search will be extended to parameter values with two decimal places.

In table 12 the results of the test runs with random seeds 0.1, 0.2, 0.3, 0.4 and 0.5 are shown. For this test series the number of generations was set to 250 and the population size was limited to 20. These results show that the best parameter settings from tuning phase 2 are much better than the best parameter settings from tuning phase 1. Considering the input parameters of this test series, the best parameter setting from tuning phase 1 delivers a hypervolume of 0.2991 which is about 25% lower than the hypervolumes of the parameter settings as shown in table 12.

The parameter setting with a mutation rate of 0.02 and a crossover rate of 0.18 has the highest hypervolume (0.4009) in the tuning phase 2 and, therefore, is regarded as the overall best performing parameter setting. This parameter setting will be applied for all NSGA-II test runs involving comparisons between the implementations of the different sub-problems.

Mutation	Crossover	Hypervolume	Std. Deviation
0.02	0.18	0.4009	0.0230
0.02	0.29	0.4001	0.0235
0.01	0.18	0.3991	0.0105
0.02	0.11	0.3973	0.0129
0.02	0.22	0.3968	0.0199

Table 12: Best 5 Parameter Combinations - Tuning Phase 2

5.4 Generation of Test Instances

Table 13 and table 14 show an overview of all the test instances which are generated according to section 5.2. For each problem class there have been generated 3 test instances which results in a total of 48 test instances; table 13 shows 24 instances each with 20 customers and table 14 shows 24 instances each with 30 customers.

The first column denotes the problem class and the second column refers to the consecutive number of the relevant test instance. The following column (Dist) shows whether the test instance has an evenly distributed or clustered customer landscape. The fourth column (D range) refers to the demand range specified for the relevant test instance. For all test instances the storage costs range between 5 and 100. Apart from describing the characteristics of the test instances, table 13 and 14 also provide information about the daily average demand the supplier has to satisfy (Avg D), the truck capacity of the supplier (Truck C) and the reference points for the storage costs (RefPnt(S)) and transport costs (RefPnt (T)).

Class	Instance	Dist	D range	Avg D	Truck C	RefPnt (S)	RefPnt (T)
1	1	even	20...100	1104	1500	575180	202880
1	2	even	20...100	1233	1500	937130	161040
1	3	even	20...100	1012	1500	611170	202340
2	4	even	20...100	1226	1800	768420	190055
2	5	even	20...100	1401	1800	674310	170565
2	6	even	20...100	1287	1800	878440	203780
3	7	clust	20...100	1188	1500	829520	149435
3	8	clust	20...100	1174	1500	678840	145050
3	9	clust	20...100	1257	1500	806760	164305
4	10	clust	20...100	1182	1800	639470	155585
4	11	clust	20...100	1168	1800	762830	154210
4	12	clust	20...100	1152	1800	743400	157255
5	13	even	40...80	1188	1500	708690	197885
5	14	even	40...80	1182	1500	696670	203800
5	15	even	40...80	1175	1500	738730	201035
6	16	even	40...80	1170	1800	757300	207280
6	17	even	40...80	1176	1800	693250	160490
6	18	even	40...80	1055	1800	600210	193735
7	19	clust	40...80	1255	1500	755690	168595
7	20	clust	40...80	1282	1500	756170	172985
7	21	clust	40...80	1200	1500	694290	189975
8	22	clust	40...80	1260	1800	765320	129065
8	23	clust	40...80	1196	1800	719300	138870
8	24	clust	40...80	1162	1800	693110	159315

Table 13: Test Instances: 20 Customers

Class	Instance	Dist	D range	Avg D	Truck C	RefPnt (S)	RefPnt (T)
9	25	even	20...100	1475	2200	883660	241375
9	26	even	20...100	1961	2200	1166910	208815
9	27	even	20...100	1599	2200	954560	247970
10	28	even	20...100	1513	2800	839000	262320
10	29	even	20...100	1836	2800	1262010	233330
10	30	even	20...100	1859	2800	1044230	207590
11	31	clust	20...100	1512	2200	893620	247410
11	32	clust	20...100	1765	2200	1086650	241440
11	33	clust	20...100	1674	2200	1038830	227385
12	34	clust	20...100	1702	2800	901260	188685
12	35	clust	20...100	1631	2800	1183450	186035
12	36	clust	20...100	1720	2800	973430	222005
13	37	even	40...80	1864	2200	1129950	263150
13	38	even	40...80	1733	2200	1041980	236370
13	39	even	40...80	1863	2200	1106900	235775
14	40	even	40...80	1821	2800	1080970	231555
14	41	even	40...80	1801	2800	1072000	236395
14	42	even	40...80	1706	2800	1037200	228665
15	43	clust	40...80	1752	2200	1092200	216155
15	44	clust	40...80	1780	2200	1066420	204855
15	45	clust	40...80	1829	2200	1038900	234265
16	46	clust	40...80	1703	2800	1067480	226605
16	47	clust	40...80	1836	2800	1079090	212250
16	48	clust	40...80	1784	2800	1115940	215370

Table 14: Test Instances: 30 Customers

6 Results

In this section the results of the test runs with the implementation of the PVRP described in the previous section are discussed. The main focus of the test runs lies on the comparison between the PVRP implementation based on CPLEX and the PVRP implementation based on the 2-opt move heuristic.

6.1 Displaying the solution of a PVRP

In the following subsection a solution of test instance 1 (see table 13 in section 5.4) will be presented. The number of generations and the population size were set to 100. The mutation rate was set to 0.002 and the crossover rate was set to 0.18. The random seed chosen was 0.1 and the TSPs were solved with the help of the 2-opt move heuristic-based PVRP implementation. The optimization process took 1710 seconds and delivered a hypervolume of 0.3946.

6.1.1 Pareto-Front

Figure 8 shows the pareto front for problem instance 1 after 100 generations. In total, 100 points are depicted in figure 8; each point represents a possible solution. The slope of the pareto-front shows the trade-off between the two objectives to be optimized. Solutions to the left tend to have lower transportation costs but higher storage costs; solutions to the right tend to have higher transportation costs but lower storage costs.

As a consequence a decision maker can choose among the pareto-optimal solutions. The solutions range from solution A to solution B and have the objective values as shown in table 15. Considering the perspectives of the supplier and the customers, a solution near point A is preferable for the supplier and a solution near point B is preferable for the customers. However, this overview does not reveal the individual storage costs of all the customers involved.

Solution	Transportation Costs	Storage Costs
A	104376	338892
B	176016	30987

Table 15: Solution A vs. B

Figure 9 shows the pareto-front as in figure 8 for test instance 1 after 100 generations. Additionally, it shows the pareto-fronts for the same test instance after 10 generations and after 1000 generations. From figure 9 it

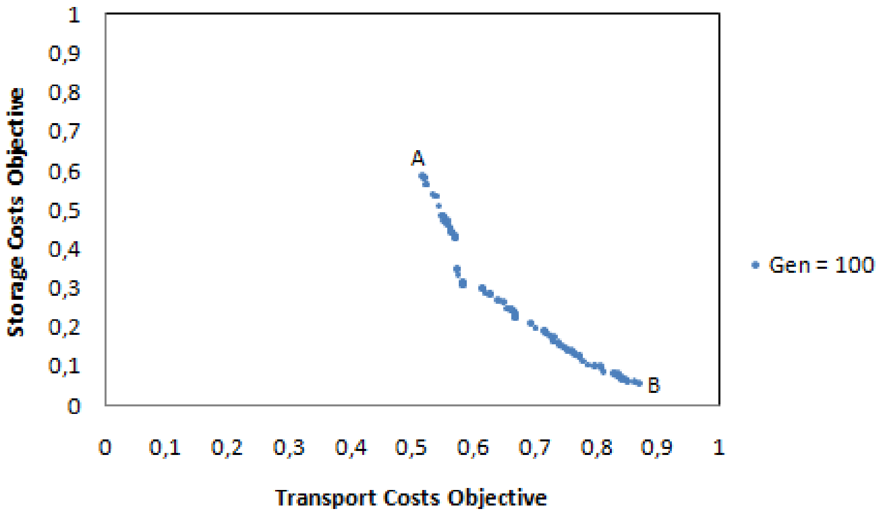

Figure 8: NSGA-II: Pareto-front

can be clearly recognized that the pareto-front moves to the left with an increasing number of generations. This movement towards left indicates a better set of alternative solutions for the decision maker and can be confirmed by an increasing hypervolume:

- Hv 1: 0.2562 (10 Generations)
- Hv 2: 0.3946 (100 Generations)
- Hv 3: 0.4573 (1000 Generations)

Moreover, it can be recognized that the pareto-fronts after 100 or 1000 generations are made up of more points (alternative solutions) than after only 10 generations. This simply implies that the NSGA-II has not found that many alternative solutions after 10 generations as after 100 or even more generations. Besides that, it can be recognized that the NSGA-II finds a better spread of solutions with an increasing number of generations. The applied NSGA-II after 10 or 100 generations still has several discontinuous regions, whereas the applied NSGA-II after 1000 generations is already converging to the true front keeping the discontinuous regions to a minimum.

6.1.2 Presenting the Periodic Delivery Schema of solution A

Table 16 shows the periodic delivery schema of solution A introduced before in section 6.1.1. The service level varies from customer to customer. For ex-

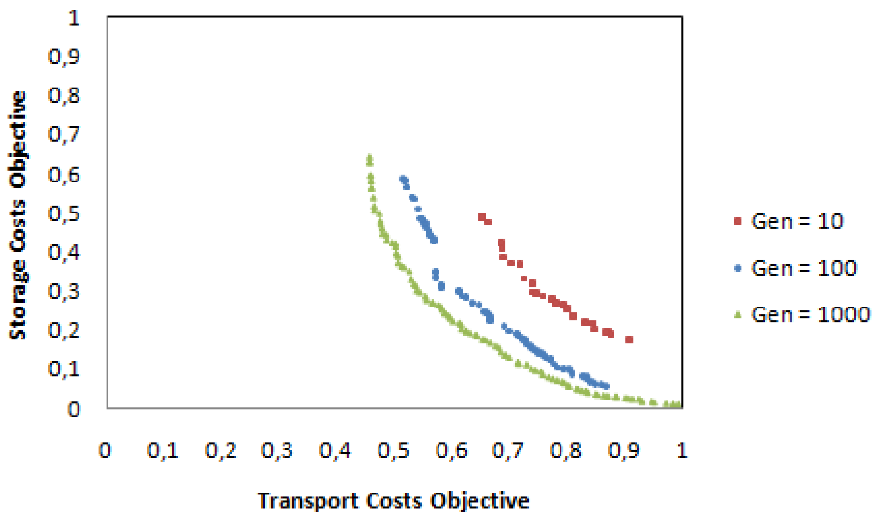

Figure 9: NSGA-II: Pareto-fronts

ample, customer 1 is serviced only once a week (Tuesday), whereas customer 11 is serviced 4 times a week. Consequently, customer 1 needs a storage capacity having the capability of keeping products demanded within one week. Customer 11 only needs a storage capacity for the relevant product from Wednesday to Thursday. These different service levels result in much higher periodic storages costs for customer 1 (€ 19380.00) compared with customer 11 (€ 2016.00). In case of solution B from section 6.1.1, both customers would be serviced every day and, thus, no storage costs would occur at all.

6.1.3 Presenting transport routes of Solution A

From the values in table 16 the supplier knows which customer should be supplied on which day. However, this representation does not reveal anything about the actual tour of each day in the planning period. Therefore, with the help of the TSP sub-problem implementation it is possible to find the optimal tours regarding the travel costs. Figure 10 shows a comprised form of the periodic delivery schema, not only indicating which customer should be serviced which day but also showing the exact route the supplier's truck has to follow in order to keep travelling costs to a minimum level. Moreover, figure 10 displays the customer landscape and the delivery routes from Monday to Friday.

Customer	Mon	Tue	Wed	Thu	Fri
01	0	1	0	0	0
02	1	1	0	0	0
03	1	1	0	0	1
04	0	0	0	1	0
05	1	0	0	0	0
06	0	0	1	1	0
07	0	1	1	0	1
08	0	0	0	1	1
09	1	0	0	0	0
10	0	1	1	0	1
11	1	1	1	0	1
12	0	0	1	0	0
13	0	0	0	1	0
14	0	0	0	1	0
15	0	1	0	0	1
16	1	0	1	0	0
17	0	0	1	1	0
18	1	0	0	0	0
19	1	0	0	0	0
20	0	0	1	0	0

Table 16: Periodic delivery schema of Solution A

6.2 Comparison of PVRP Implementations

In this section the PVRP with two different TSP solver implementations will be compared. One implementation of the TSP sub-problem makes use of CPLEX, the other implementation relies on the nearest neighbor algorithm improved by a 2-opt move modification. The input for the TSP problem is basically always one column of the assignment table 16 and the travelling costs associated from one point to another. The main difference between the TSP solver implementations is that the CPLEX-based implementation always finds the optimal solution, whereas the 2-opt move implementation only finds good solutions but not necessarily the best solution. However, the CPLEX implementation which always delivers the optimal solution to a TSP problem needs much more time to solve a problem than the heuristic-based implementation.

The question arising here is whether the PVRP implementation is delivering better results when relying on a TSP solver which is coming up with optimal solutions but takes relatively long time for the problem solving pro-

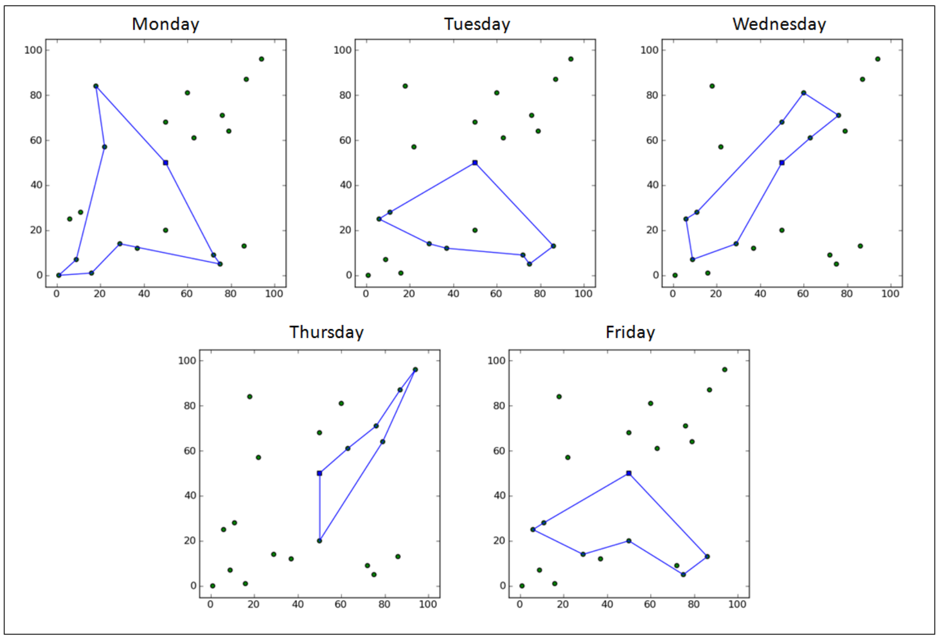

Figure 10: Solution A: Tours: Mon - Fri

cess or when relying on a fast TSP solver only coming up with relatively good results. For each individual of a generation there needs to be solved 5 TSP problems (Mon - Fri), in case of a population size of 100 and a number of generations of 100 this would result in a maximum of $5 \times 100 \times 100$ TSP problems needed to be solved. This implicitly raises the question whether the problem solving time or the solution quality of the TSP solvers affects the overall result to a higher degree.

6.2.1 Comparison of PVRP implementations with equal number of generations

In order to compare the PVRP implementation with the two different TSP solvers a series of test runs haven been executed. Therefore, the PVRP has been applied to the test instances introduced in section 5.4.

In a first test run the parameters were set to the following values: The number of generations was set to 250 and the population size was set to 20. The mutation rate was set to 0.02 and the crossover rate was set to 0.18. Figure 11 shows the average hypervolumes of the problem classes 1 - 8; figure 12 shows the average hypervolumes of the problem classes 9 - 16. Each problem class consists of 3 test instances and each of these test instances was solved three times (random seed: 0.1, 0.2, 0.3).

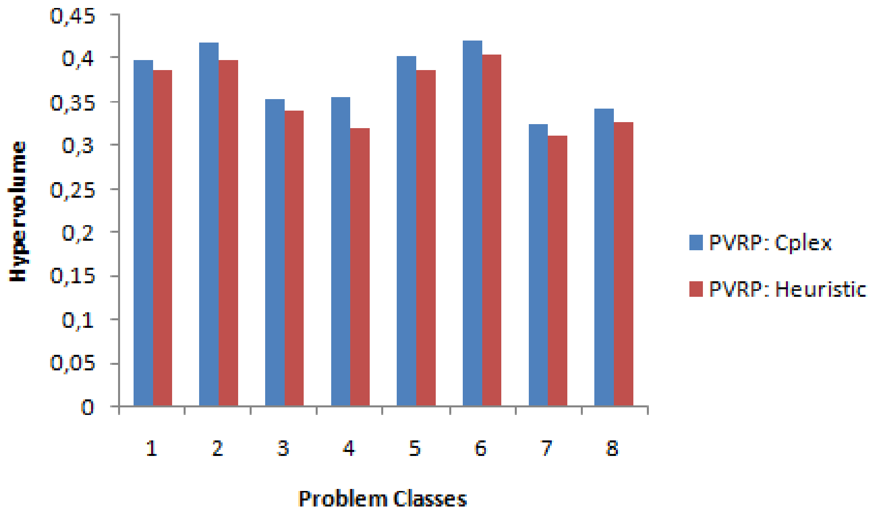

Figure 11: Comparison of Hv: PVRP: CPLEX vs. Heuristic (1 - 8)

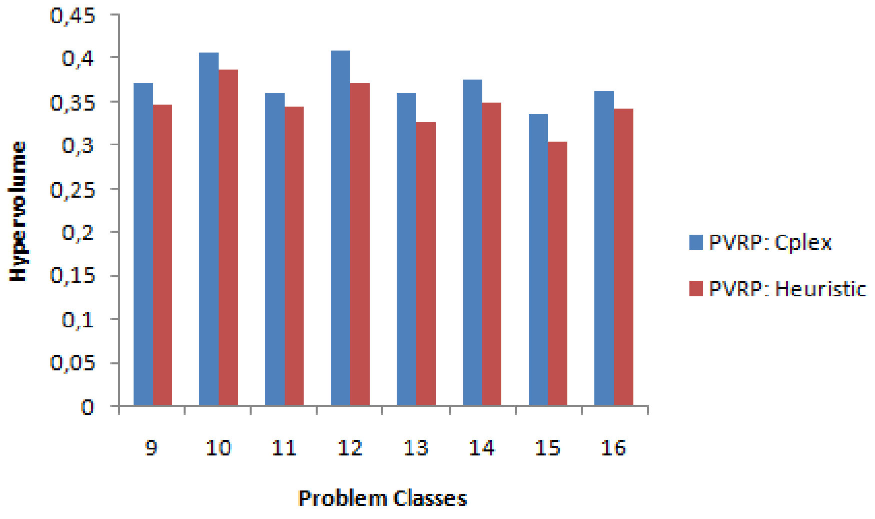

Figure 12: Comparison of Hv: PVRP: CPLEX vs. Heuristic (9 - 16)

The problem classes are numbered from 1 to 8 (Figure 11), respectively from 9 - 16 (Figure 11). For each problem class the average hypervolume is shown for the CPLEX-based PVRP (left bar) and heuristic-based PVRP (right bar). It can be seen that for all problem classes the PVRP implementation with CPLEX as TSP solver has better results. When looking at the individual test results, the CPLEX-based PVRP delivered better results in 118 cases and only in 26 cases the PVRP with the heuristic TSP solver delivered better results. In other words, the PVRP implementation with CPLEX delivered better results in 81.94% of the test runs when having the same parameter setting. Moreover, the average hypervolume of the CPLEX-based test runs was about 5.87% higher than the heuristic-based PVRP test runs. However, the PVRP implementation relying on CPLEX needs about 10086 seconds on average to solve a test instance, whereas the other PVRP variant does not even need 1 second.

6.2.2 Comparison of PVRP implementations with similar runtimes

As a consequence, the number of generations was increased for the test runs with the 2-opt move-based PVRP implementation to produce test results with similar runtimes. Therefore, the number of generations was set to 1000000 resulting in an average runtime of 1041 seconds needed to solve a test instance but which is still only one tenth of the average runtime of the CPLEX variant.

The average time the PVRP implementations need for each problem class is shown in figure 13. The implementation relying on the heuristic TSP solver seems to be very constant regarding the runtime. The problem classes with 20 customers (1 - 8) are solved within an average runtime of around 700 seconds, whereas the problem classes with 30 customers (9 - 16) are solved within an average runtime of around 1400 seconds. The problem solving times of all problem classes with 30 customers exceed the problem solving time of the problem classes with 20 customers.

Problem Classes	CPLEX-based PVRP	Heuristic-based PVRP
01 - 16	15602	375
01 - 08	18735	91
09 - 16	11596	230

Table 17: Comparison of standard deviations (runtime)

Table 17 shows the standard deviation of the runtimes of the heuristic-based PVRP implementation which is 375. The CPLEX-based PVRP implementation gives a much higher standard deviation which amounts up to

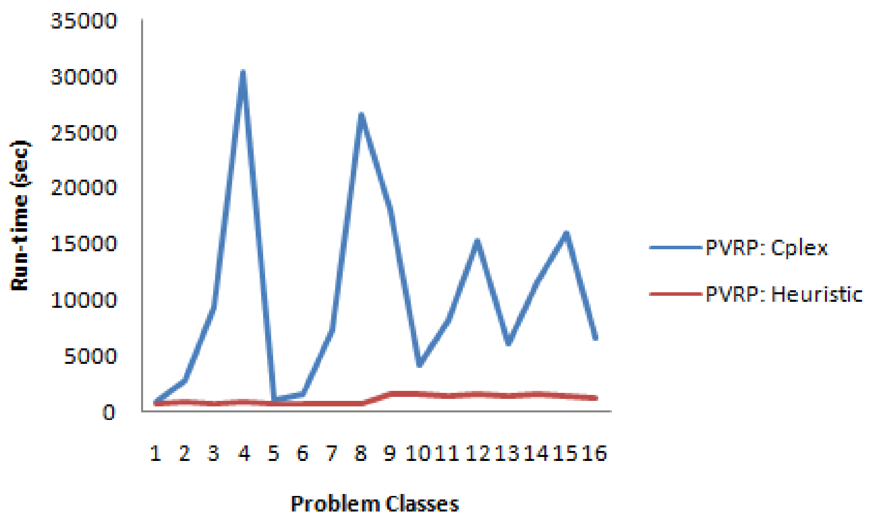

Figure 13: Comparison of runtime: PVRP: CPLEX vs. Heuristic (1 - 16)

15602. The remarkable fact when looking at the CPLEX-based implementation is that problem classes with 30 customers do not necessarily need longer time in order to be solved compared to problem classes with only 20 customers. This fact is underlined when comparing the average runtime of the problem classes with 20 customers (34336 sec) with the average runtime of the problem classes with 30 customers (10913 sec). When considering the CPLEX-based implementation, then the problem classes 4 and 8 are the ones with the highest average runtimes although only 20 customers need to be serviced. In contrast, the problem classes 1, 5 and 6 are the ones which need less time to be solved than any other problem classes and to a certain degree can even compete with the average runtimes of the equivalent problem classes of the heuristic-based implementation (e.g. Problem Class 1: 917s vs. 706s).

As indicated by the standard deviations presented in table 17 the runtimes of the CPLEX-based implementation fluctuate to a much higher degree than the runtimes of the heuristic-based implementation.

Figure 14 and figure 15 provide the comparison of the hypervolumes of the PVRP implementations with the runtimes just discussed. With the adaption of the number of generations to be computed the heuristic-based PVRP implementation now can surpass the results of the CPLEX-based implementation. Especially, for the problem classes with 20 customers to be serviced the heuristic-based implementation has better results in all the problem classes (1 - 8). When looking at the problem classes with 30 customers, the results look much more balanced. In 4 (10, 12, 14 and 16) out of 8 problem classes the CPLEX-based implementations delivers slightly better results regarding the hypervolume, whereas in the other 4 problem classes the heuristic-based PVRP implementation comes up with better results.

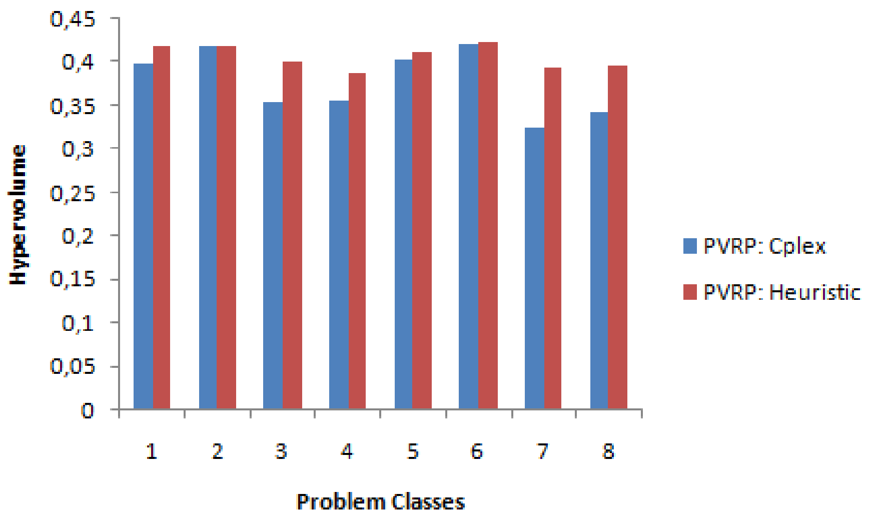

Figure 14: Comparison of Hv: PVRP: CPEX vs. Heuristic (1 - 8)

Overall, the heuristic-based implementation performs better than the CPLEX-based implementation. This can also be confirmed by comparing the individual results; in 93 out of 144 cases the heuristic-based PVRP delivered better hypervolumes than the CPLEX-based PVRP which results in a percentage of 64.50%. As discussed before the heuristic-based PVRP obtains better average hypervolumes by needing much less time to solve the test instances.

6.2.3 Development of Hypervolume over Time

Figure 16 shows the development of the hypervolume for the CPLEX-based PVRP (blue line) and the heuristic-based PVRP (red line) implementation over 500 generations (Test instance 1). The test run was performed with a population size of 100 and a random seed of 0.1. The hypervolume was measured every 10 generations; the hypervolume of the CPLEX-based PVRP after one generation is 0.1928 and increases up to 0.4795 after 500 generations, whereas the hypervolume of the heuristic-based PVRP initially starts with 0.1763 and has a final value of 0.4744.

After one generation the CPLEX-based implementation delivers a hypervolume which is about 8.56% higher than the hypervolume of the heuristic-

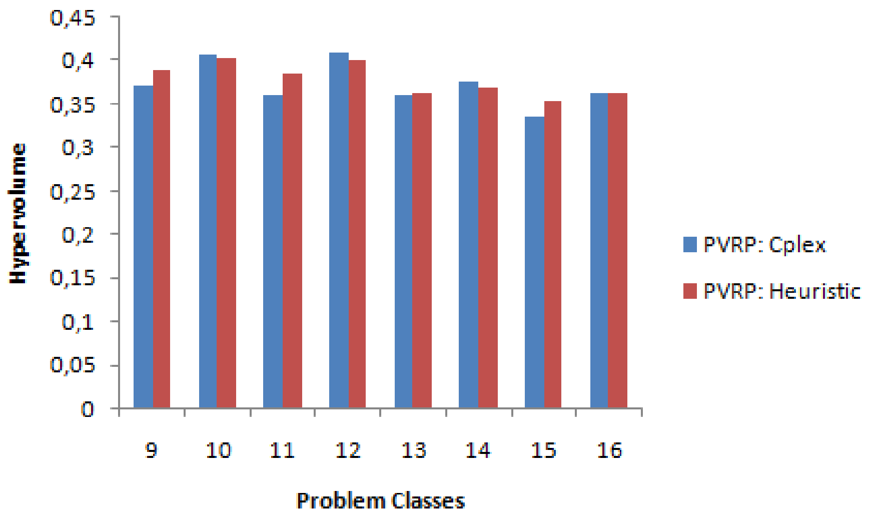

Figure 15: Comparison of Hv: PVRP: CPEX vs. Heuristic (9 - 16)

based implementation. At the end of the test run this difference in hypervolumes is reduced to only 1.06%. In total, the hypervolume of the CPLEX-based implementation increases by 148.61%. The strongest improvement can be recognized within the first ten generations (48.20%), respectively within the first 100 generations (89.59%). The following 100 generations only provide an increase of 6.76% and this increase is further reduced to 3.52% for the next 100 generations. The last 100 generations merely come up with an increase of less than 1% (0.88%). The behavior of the hypervolume development of the heuristic-based PVRP implementation is similar. The improvement of the hypervolume starts off slightly slower, but proves to maintain higher increase rates within the execution of the 500 generations.

6.3 Discussing the Influence of the Problem Class Aspects

6.3.1 Truck Capacity

All the uneven-numbered problem classes have a relatively small truck capacity; all the even-numbered problem classes have a relatively larger truck capacity. Problem classes with 30 customers have a larger truck capacity than the problem classes with 20 customers in order to be able to satisfy the higher demand caused by the additional customers.

When directly comparing an uneven-problem class with its neighboring

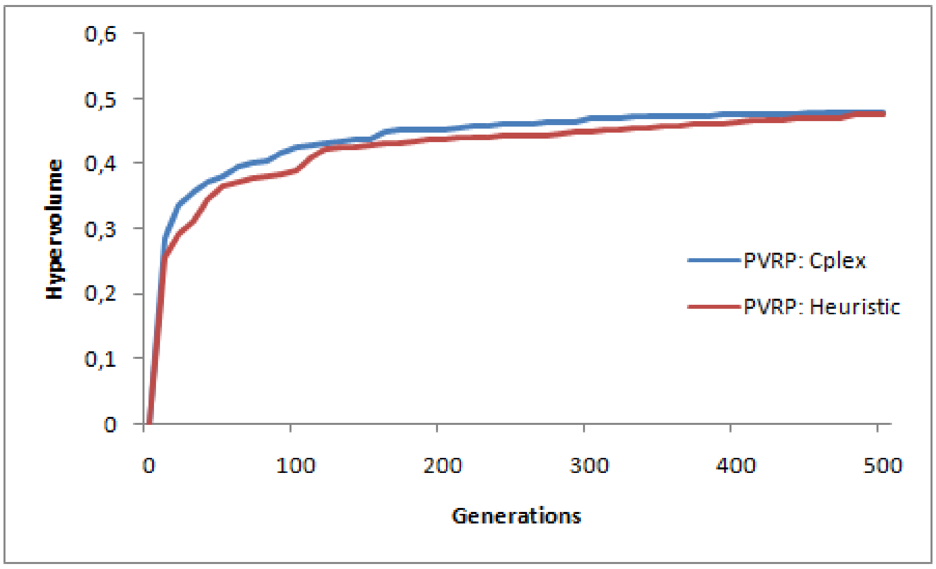

Figure 16: Comparison of Hv over time (500 gen)

problem class (e.g. 1 vs. 2, 3 vs. 4, etc.), then all parameters specifying the problem class except the truck capacity are equal. Figures 11, 12, 14 and 15 show that the problem classes with a larger truck capacity (even-numbered problem classes) tend to have higher hypervolumes. In figure 11 all the problem classes with higher truck capacity have better results regarding the hypervolume compared to the corresponding neighboring problem class. According to the figures 12 and 14 in 7 out of 8 cases problem classes with a higher truck capacity deliver better hypervolumes.

This tendency can be explained due to the mitigation of the truck capacity constraint. Consequently, the PVRP implementation can find more and better solutions allowing to reaching a better hypervolume. However, this additional space for optimization also leads to increased runtimes for the even-numbered problem classes. For both, the CPLEX-based and the heuristic-based PVRP implementations, in 6 out of 8 cases the even-numbered problem classes need more time to finish the optimization problem than the uneven-numbered problem class.

6.3.2 Demand Range

The demand range of a problem class is a similar constraint like the truck capacity. Depending on the specification of this constraint the PVRP implementation has more or less space for optimization. The problem classes from 1 - 4 and 9 - 12 have a broader demand range, whereas the remaining problem classes have a smaller range. The comparison of the problem classes with their corresponding problem classes regarding the demand range (1 vs. 5, 2 vs. 6, etc.) confirms that a more severe constraint leads to lower hypervolumes, but in turn also requires less time to finish the execution of the optimization process. For both PVRP implementations, in 6 out of 8 cases the problem classes with a broader demand range deliver higher hypervolumes. The problem classes with a broader demand range need more time in 5 out of 8 comparisons in case of the CPLEX-based implementation and need more time in 7 out of 8 comparisons in case of the heuristic-based implementation.

6.3.3 Distribution of Customers

The distribution of the customers to be serviced is either even or clustered. Both PVRP implementations find it harder to obtain good hypervolumes when the customer distribution is clustered compared to an even distribution. In case of the CPLEX-based implementation, in 7 out of 8 comparisons the problem classes with an even distribution have a relatively higher hypervolume. In case of the heuristic-based implementation all problem classes with an even distribution perform better than their clustered counterparts. The higher complexity induced by the clustered distribution of the customers also leads to longer runtimes, but this is only true for the CPLEX-based implementation. Here, in 6 out of 8 cases the problem classes with a clustered distribution need more time to get solved. Surprisingly, the clustered problem classes do not necessarily take more time to finish the optimization process when considering the heuristic-based PVRP implementation. Only in 3 out of 8 cases the clustered problem classes take more time than the evenly distributed problem classes.

6.3.4 Problem Size

The problem sizes tested are related to the number of customers to be delivered. In the problem classes 1 - 8, there need to be serviced 20 customers and in the problem classes 9 - 16 need to be serviced 30 customers. Problem class 1 can be directly compared with problem class 9 as all other problem characteristics are kept the same. Problem class 2 can be directly compared with problem class 10 and so on.

When comparing the hypervolumes of the corresponding problem classes (1 vs. 9, 2 vs. 10, etc.) for the CPLEX-based PVRP, it cannot be concluded that the problems with smaller problem classes deliver better hypervolumes. In 4 out of 8 direct comparisons the problem classes with 20 customers (1, 2, 6 and 7) have higher hypervolumes, but in the other 4 cases they have lower hypervolumes. However, it is remarkable that the 4 problem classes (with 20 customers) delivering higher hypervolumes all have an even distribution of customers. This indicates that the CPLEX-based implementation is less affected by a clustered distribution with an increasing problem size. It neither can be recognized that the runtimes are increasing due to a larger problem size (see figure 13).

In case of an evenly distributed customer landscape with an increasing problem size the CPLEX-based PVRP finds it harder to obtain good results and needs more and more time. But in case of a clustered customer landscape with an increasing problem size, the implementation can find better solutions in even less time. The increasing problem size seems to reduce the complexity of the clustered customer landscape.

When directly comparing the hypervolumes of the corresponding problem classes for the heuristic-based PVRP, similar results can only be confirmed for the test runs with 250 generations. The test runs with 1000000 generations give totally different results. The way the customers are located in the customer landscape seems to hardly impact the hypervolume and runtime. In 7 out of 8 comparisons the hypervolume of the problem classes with 20 customers surpasses the hypervolume of the problem classes with 30 customers. Moreover, all the problem classes with 30 customers have a runtime which is about 100% longer than the problem classes with 20 customers.

The difference in how the PVRP implementations behave can be explained by the working mode of the underlying TSP solvers. First of all, the TSP solver needs more time than the heuristic-based TSP solver to solve an individual TSP problem. Additionally, the CPLEX-based TSP solver is prone to certain complex TSP problems; when trying to solve a TSP problem of higher complexity then the runtime will increase further. On the other hand, the heuristic-based TSP solver will solve an individual TSP problem very fast and will not need extra time to find a solution for TSPs of higher complexity. Tricky TSPs only could cause the heuristic-based TSP solvers to find a solution which is relatively far from the optimal solution provided by the CPLEX-based TSP solver.

7 Summary

This paper discusses an approach to solve a Periodic Vehicle Routing Problem (PVRP) with the help of the NSGA-II. The PVRP is a special variant of the generic Vehicle Routing Problem (VRP) which refers to a whole class of problems in which a set of routes for one or more vehicles based at several depots must be retrieved for a certain number of customers. The main characteristics which make the PVRP so special are an infinite planning period and that the inventories of the customers are managed by the supplier. As a consequence the supplier needs to consider storage costs and transportation costs when defining a periodic delivery schema for the customers to be serviced.

The NSGA-II is a genetic algorithm which belongs to the group of multi-objective evolutionary algorithms (MOEAs) and is capable of optimizing more objectives simultaneously. Consequently, the NSGA-II suits the needs of the PVRP with two objectives to be minimized perfectly. Moreover, the NSGA-II does not only deliver one solution but delivers a whole set of solutions. The solutions of this set are deemed to be pareto-optimal which means that the increase of one objective of a solution from the set leads to a decrease in the other objective of that solution. Thus, the NSGA-II provides the supplier with a number of alternative solutions representing the trade-off between the storage and transportation costs.

The pareto-optimal solutions enable a supplier to select a periodic delivery schema with relatively high transportation costs and low storage costs favoring the customers or with relatively low transportation costs and high storage costs favoring the supplier. This scope of action provided by the set of solutions can help the supplier and the customers to find a solution which satisfies both parties to a high degree.

For solving the PVRP the NSGA-II needs assistance in solving a sub-problem. The NSGA-II is only responsible for solving the master problem which refers to assigning deliveries to customers within the planning period. The sub-problem is a so-called Travelling Salesman Problem (TSP) and optimizes the delivery tours provided by the master problem. Two implementations are provided for the sub-problem. In the first implementation the TSP is modeled as an Integer Linear Program and is solved with the help of IBM ILOG CPLEX Optimizer. The second implementation is based on a heuristic called Nearest Neighbor enhanced by 2-opt move. The CPLEX-based TSP solver is expected to deliver optimal solutions whereas the heuristic-based TSP solver is deemed to deliver only good solutions. However, the heuristic-based TSP solver is much faster in finding a relatively good solution than the CPLEX-based TSP solver in finding an optimal solution.

Test runs were executed for the CPLEX-based and the heuristic-based PVRP implementation. When comparing the performance of the implementations with the same number of generations and population size, then the CPLEX-based PVRP finds slightly better solutions than the heuristic-based PVRP but needs much more time to compute all the generations. When increasing the number of generations for the heuristic-based PVRP to obtain similar runtimes, then the CPLEX-based implementations delivers worse results.

Both PVRP implementations react similar to the different problem classes. For example, an increasing truck capacity or a broader demand range leads to a better performance while the runtime is extended for both implementations. They also find it harder to obtain good hypervolumes when the customer distribution is clustered compared to an even distribution. The higher complexity induced by the clustered distribution of the customers also leads to longer runtimes, but this is only true for the CPLEX-based implementations. Surprisingly, the clustered problem classes do not necessarily take more time to finish the optimization process when considering the heuristic-based PVRP implementation.

Interestingly, it cannot be concluded that the problems with smaller problem classes deliver better hypervolumes. In case of an evenly distributed customer landscape with an increasing problem size the CPLEX-based PVRP finds it harder to obtain good results and needs more and more time. But in case of a clustered customer landscape with an increasing problem size, the implementation can find better solutions in even less time. The increasing problem size seems to reduce the complexity of the clustered customer landscape. In contrast, the way the customers are located in the customer landscape seems to hardly impact the hypervolume and runtime of the heuristic-based PVRP.

The difference in how the PVRP implementations behave can be explained by the working mode of the underlying TSP solvers. First of all, the CPLEX-based TSP solver needs more time than the heuristic-based TSP solver to solve an individual TSP problem. Additionally, the CPLEX-based TSP solver is prone to certain complex TSP problems; when trying to solve a TSP problem of higher complexity then the runtime will increase further. On the other hand, the heuristic-based TSP solver will solve an individual TSP problem very fast and will not need extra time to find a solution for TSPs of higher complexity. Tricky TSPs only could cause the heuristic-based TSP solvers to find a solution which is relatively far from the optimal solution provided by the CPLEX-based TSP solver.

Summarizing the work accomplished it can be said that the paper and the relating implementation of the NSGA-II were highly challenging, but also very interesting. The challenging aspects were the inherent complexity of the problem to be researched and the technologies involved. For example, when considering a problem with 20 customers and a planning horizon of 5 days, the problem to be solved is represented by 100 binary variables. In case of a population size of 100, the PVRP implementation computes 100 individuals per generation. For each individual always 5 TSPs need to be computed. Consequently, the computing time of the TSP sub-problem is vital as it is required to solve a large number of TSPs within the execution of the PVRP implementation. The resulting computational expense called for well-planned test series. Depending on the parameter settings chosen test series could vary from minutes to several days. As a consequence, repetition of test runs may lead to suffering a huge delay in progress.

The technologies involved were not of extreme complexity, but, nevertheless it was my first encounter with IBM ILOG CPLEX Optimizer which required me to learn how to implement an optimization model for the optimizing software and how to link it with the PVRP implementation. Moreover, due to the necessary interaction between the C++ program and the CPLEX optimizer program development and testing were hindered. However, altogether it was a great and sound project which did not have to overcome any real obstacles.

List of Figures

List of Tables

References

[1] R. H. Balou, *Business Logistics/Supply Chain Management.* Prentice Hall, 5 ed., 2004.

[2] D. Waters, *Supply Chain Management An Introduction to Logistics.* Palgrave MacMillan, second ed., 2009.

[3] D. R. Anderson, D. J. Sweeney, and T. A. Williams, *Introduction to Managment Science: Quantitative Approaches to Decision Making.* West Publishing Company, 7 ed., 1994.

[4] M. Waller, M. E. Johnson, and T. Davis, "Vendor-managed inventory in the retail supply chain," *Journal of Business Logistics*, vol. 20, pp. 183 – 203, 1999.

[5] W. Ho, G. T. Ho, P. Ji, and H. C. Lau, "A hybrid genetic algorithm for the multi-depot vehicle routing problem," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 4, pp. 548 – 557, 2008.

[6] C. Archetti and M. G. Speranza, "The split delivery vehicle routing problem: A survey," in *The Vehicle Routing Problem: Latest Advances and New Challenges* (B. Golden, S. Raghavan, and E. Wasil, eds.), vol. 43 of *Operations Research/Computer Science Interfaces Series*, pp. 103–122, Springer US, 2008.

[7] P. M. Francis, K. R. Smilowitz, and M. Tzur, "The period vehicle routing problem and its extensions," in *The Vehicle Routing Problem: Latest Advances and New Challenges* (B. Golden, S. Raghavan, and E. Wasil, eds.), vol. 43 of *Operations Research/Computer Science Interfaces Series*, pp. 73–102, Springer US, 2008.

[8] K. C. Tan, L. H. Lee, Q. L. Zhu, and K. Ou, "Heuristic methods for vehicle routing problem with time windows," *Artificial Intelligence in Engineering*, vol. 15, no. 3, pp. 281 – 295, 2001.

[9] E. Nishimura, A. Imai, and S. Papadimitriou, "Yard trailer routing at a maritime container terminal," *Transportation Research Part E: Logistics and Transportation Review*, vol. 41, no. 1, pp. 53 – 76, 2005.

[10] G. Nagy and S. Salhi, "Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries," *European Journal of Operational Research*, vol. 162, no. 1, pp. 126 – 141, 2005.

[11] L. Bertazzi, M. Savelsbergh, and M. G. Speranza, “Inventory routing,” in *The Vehicle Routing Problem: Latest Advances and New Challenges* (B. Golden, S. Raghavan, and E. Wasil, eds.), vol. 43 of *Operations Research/Computer Science Interfaces Series*, pp. 49–72, Springer US, 2008.

[12] K. Weicker, *Evolutionaere Algorithmen.* Teubner Verlag, 2007.

[13] J. H. Holland, *Adaption in Natural and Artificial Systems.* MIT Press, 1975.

[14] I. Gerdes, F. Klawonn, and R. Kruse, *Evolutionaere Algorithmen.* Vieweg Verlag, 2004.

[15] D. Beasley, D. R. Bull, and R. R. Martin, “An overview of genetic algorithms: Part 1 - fundamentals,” *University Computing*, vol. 15, no. 2, pp. 58 – 69, 1993.

[16] C. M. Fonseca and P. J. Fleming, “An overview of evolutionary algorithms in multiobjective optimization,” *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182 – 197, 2002.

[18] E. Zitzler and L. Thiele, “Multiobjective optimization using evolutionary algorithms a comparative case study,” in *Parallel Problem Solving from Nature PPSN V* (A. Eiben, T. Bck, M. Schoenauer, and H.-P. Schwefel, eds.), vol. 1498 of *Lecture Notes in Computer Science*, pp. 292 – 301, Springer Berlin / Heidelberg, 1998. 10.1007/BFb0056872.

[19] D. A. Veldhuizen and G. B. Lamont, “Multiobjective evolutionary algorithms: Analyzing the state-of-the-art,” *Evolutionary Computation*, vol. 8, no. 2, pp. 125 – 147, 2000.

[20] N. Srinivas and K. Deb, “Muiltiobjective optimization using nondominated sorting in genetic algorithms,” *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[21] E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results,” *Evolutionary Computation*, vol. 8, no. 2, pp. 173 – 195, 2000.

[22] M. Emmerich, N. Beume, and B. Naujoks, "An emo algorithm using the hypervolume measure as selection criterion," in *Evolutionary Multi-Criterion Optimization* (C. Coello Coello, A. Hernndez Aguirre, and E. Zitzler, eds.), vol. 3410 of *Lecture Notes in Computer Science*, pp. 62–76, Springer Berlin / Heidelberg, 2005.

[23] J. Knowles and D. Corne, "On metrics for comparing non-dominated sets," in *Congress on Evolutionary Computation (CEC 2002)*, pp. 711 – 716, IEEE Press, 2002.

[24] D. E. Goldberg and K. Deb, *A Comparison of Selection Schemes used in Genetic Algorithms*, pp. 69 – 93. Morgan Kaufmann Publishers, 1991.

[25] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms.* Wiley-Interscience series in systems and optimization, John Wiley & Sons, Ltd., first ed., 2001.

A Appendices

A.1 Zusammenfassung

Das Ziel dieser Masterarbeit ist die Evaluierung der Anwendbarkeit eines Genetischen Algorithmus auf ein periodisches Vehicle Routing Problem (PVRP). Bei dem zu evaluierenden Algorithmus handelt es sich um den sogenannten "Non-Dominated Sorting Algorithm Version II" (NSGA-II). Der angesprochene Algorithmus gehört zur Gruppe der Genetischen Algorithmen und zeichnet sich durch die gleichzeitige Optimierung mehrerer Zielfunktionen aus.

Im Zuge der vorliegenden Arbeit wird der NSGA-II auf ein periodisches Vehicle Routing Problem angewandt. Bei dem Problem handelt es sich um eine Fragestellung aus dem Bereich des Transport- bzw. Lagerwesens. Es ist ein Warendepot vorhanden und mit Hilfe eines Fahrzeuges sollen Kunden periodisch an gewissen Tagen der Woche beliefert werden. Für ein solches Problem werden u.a. Transportkosten, Lagerkosten, täglicher Bedarf und Anfangslagerbestand angenommen.

Der anzuwendende Algorithmus liefert eine Menge von Lösungen für ein konkretes PVRP. Eine Lösung bezeichnet ein periodisches Belieferungsschema und drückt aus, an welchen Tagen welche Kunden beliefert werden sollen. Bei der Anwendung des NSGA-II auf das vorgestellte PVRP sind zwei Zielfunktionen involviert, denn es werden sowohl die Transportkosten als auch die Lagerkosten minimiert. Außerdem müssen Nebenbedingungen eingehalten werden, die sicherstellen, dass den Kunden die Waren nicht ausgehen und der tägliche Bedarf stets gedeckt werden kann.

Die Lösungen, die ein NSGA-II liefert sind pareto-optimal. Das heißt, dass der Anstieg einer Zielfunktion die gleichzeitige Reduktion der anderen Zielfunktion bedeutet. Bei der Wahl eines Belieferungsschemas kann somit aus einer Menge an alternativen Belieferungsschemata ausgewählt werden. Diese Auswahl drückt demzufolge den Trade-off zwischen Transport- und Lagerkosten aus. Folglich existieren Belieferungsschemata, die den Anbieter favorisieren; gleichzeitig gibt es aber auch Lösungen, die die zu beliefernden Kunden bevorzugen. Aus dieser Menge an Belieferungsschemata kann nun eines gewählt werden, welches die unterschiedlichen Anforderungen der Kunden und des Anbieters am besten abdecken.

Das zu lösende Problem wird unterteilt in ein Hauptproblem und ein Subproblem. Das Hauptproblem wird in C++ umgesetzt und bezeichnet das oben beschriebene PVRP. Im Vordergrund steht hierbei die Zuordnung an welchen Tagen einer Planungsperiode die Kunden beliefert werden sollen. Um die Minimierung der Transportkosten für das Beschaffungsschema gewährleisten zu können, muss für jeden Tag die Fahrtroute mit den geringsten Transportkosten berechnet werden. Dieses Subproblem ist auch bekannt als Travelling Salesman Problem (TSP). Für das Subproblem wurden zwei Im-

plementierungsvarianten erstellt.

Eine Implementierungsvariante des Subproblems wurde mit Hilfe von IBM ILOG CPLEX (ilog.com/products/cplex/) gelöst. Die zweite Implementierungsvariante wurde mit einer modifizierten Variante des "Nearest Neighbour Algorithmus" umgesetzt. Durch zusätzliche Anwendung des "2-opt move"wird versucht in einem weiteren Schritt eine bessere Lösung zu finden. Die erste Implementierungsvariante liefert stets eine optimale Lösung für ein TSP, braucht für die Lösungsfindung jedoch relativ länger als die zweite Variante, welche im Normalfall nur gute Lösungen findet.

Testläufe wurden durchgeführt mit der CPLEX-basierten Subproblem-Variante sowie mit dem modifizierten "Nearest Neighbour Algorithmus". Vergleicht man das PVRP mit den zwei verschiedenen Subproblem Implementierungen bei gleichen Parametereinstellungen, so findet die CPLEX-basierte Variante bessere Lösungen bei höherer Laufzeit. Passt man die Parametereinstellungen an, sodass beide Implementierungsvarianten ähnliche Laufzeiten vorweisen, dann liefert die Variante mit der zweiten Subproblemimplementierung bessere Ergebnisse.

Interessanterweise reagieren beide PVRP Implementierungen ähnlich auf unterschiedliche Problemklassen. Beispielsweise bedeutet ein größeres Transportvolumen des Fahrzeugs, dass die NSGA-II Implementierung bessere Lösungen findet, dabei aber auch mehr Zeit zur Lösungsfindung in Anspruch genommen wird. Das heißt, dass bei Problemklassen mit "gelockerten"Nebenbedingungen ein größerer Suchraum vorhanden ist. Dementsprechend dauert die Lösungsfindung länger, führt aber gleichzeitig zu besseren Ergebnissen.

Es ist ebenso erkennbar, dass die PVRP Implementierung bei einer gleichmäßigen Verteilung der Kunden bessere Lösungen im Vergleich zu Kundenverteilungen in Form von "Clustern"findet. Für die CPLEX-basierte Implementierung bedeuten geballte Anhäufungen von Kunden längere Laufzeiten, die Heuristik-basierte Implementierung reagiert nicht notwendigerweise mit längeren Laufzeiten. Weiters kann beobachtet werden, dass Problemklassen mit weniger zu beliefernden Kunden nicht automatisch bessere Lösungen liefern.

Das unterschiedliche Verhalten der beiden PVRP Implementierungen kann durch die Eigenschaften der darunterliegenden TSP Implementierungen erklärt werden. Der CPLEX-basierte TSP Solver benötigt mehr Zeit zur Lösungsfindung als der Heuristik-basierte TSP Solver. Außerdem weist die CPLEX-basierte Implementierung besonders schlechtes Verhalten in Bezug auf komplexe TSPs auf. Die Heuristik-basierte Implementierung reagiert stabiler auf komplexe Problemklassen; kein wesentlich größerer Zeitaufwand ist zu bemerken.

A.2 Curriculum Vitae

CURRICULUM VITAE

Persönliche Daten

Hans-Christian Zohmann
Am Schöpfwerk 29/9/56, 1120 Wien
Tel: 0677/377 95 18
E-Mail: a0102290@unet.univie.ac.at

Geburtsdatum	17.02.1982
Geburtsort	Wien
Familienstand	ledig
Nationalität	Österreich

Ausbildung

2007 – 2011	Masterstudium Wirtschaftsinformatik (WINF), Uni Wien
2001 – 2007	Bakkalaureatsstudium Wirtschaftsinformatik, Uni Wien/TU Wien Abgeschlossen am: 31.01.2007 Bakkalaureatsstudium Sinologie, Uni Wien Nicht abgeschlossen
1996 – 2001	International Business College Hetzendorf, 1120 Wien

Berufserfahrung

2011
(15.03 – jetzt)

SAP HCM ADVICE Consultant
HCM ADVICE, Wien, Österreich (http://hcm-advice.eu)
- Programmierung, Customizing u. Beratung im Bereich SAP Human Capital Management

2010
(01.02 – 31.07)

ITP Region China Department (Praktikum)
Volkswagen Group China, Beijing, China (http://www.vw.com.cn)
- Analyse, Dokumentation u. Implementierung von konzernweiten Prozessen für den lokalen IT User Help Desk in Beijing
- Bewertung u. Prototyping von iPad Anwendungen für die Vertriebsabteilung
- Analyse, Konsolidierung u. Visualisierung von Produktdaten
- Verifikation, Dokumentation u. Testen von Interfaces

2008 – 2010
(10.03 – 31.01)

Junior Software Developer (Teilzeit: 25h)
Qualysoft, Wien, Österreich (www.qualysoft.at)
- Implementierung von Webanwendungen (Java, Spring, JSF)
- Model Driven Development (AndroMDA, MagicDraw)
- Implementierung von Geschäftsprozessen u. Workflows (C#, Java Script, XML, XSLT, Oracle)

2007
(07.07 – 08.07)

.NET Development mit C# (Praktikum)
s IT Solutions, Wien, Österreich (www.s-itsolutions.at)
- Entwicklung von Tools in C#
- Programmierung von SQL Stored Procedures auf MS SQL Server

Kenntnisse

EDV	MS Office (Word, PowerPoint, Excel, Access), MS Project MS Visio, Adonis, MagicDraw, Eclipse C++, Java, Python, SQL (mySQL, Oracle, MS SQL Server)	
Sprachen	Deutsch	Muttersprache
	Englisch	fließend (mündlich/schriftlich)
	Chinesisch	fließend (mündlich), HSK Level 6
	Italienisch/Spanisch	Basiskenntnisse

Auslandserfahrung

2010 (01.02 – 31.07)	Praktikum bei Volkswagen Group China (siehe Berufserfahrung) Beijing, China
2007 (01.02 – 31.05)	Auslandssemester im Rahmen des Masterstudiums WINF Dublin City University, Dublin, Irland
2004 / 2005	Auslandsjahr im Rahmen des Sinologie-Studiums, Universität Tongji, Shanghai, China