Lars Wagner

Introduction of an EA and Fuzzy Approach to solve a Task-Allocation Problem

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF

FAKULTÄT FÜR INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

Introduction of an EA and Fuzzy Approach to solve a Task-Allocation Problem

Bachelor Thesis

Lars Wagner

June 30, 2020

Supervisor:	Prof. Dr.-Ing. habil. Sanaz Mostaghim
Advisor:	Dr.-Ing. Heiner Zille
Advisor:	Dr.-Ing. Christoph Steup

Lars Wagner: *Introduction of an EA and Fuzzy Approach to solve a Task-Allocation Problem*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2020.

Abstract

Evolutionary Algorithms (EA) and Fuzzy Systems are introduced to solve a Task-Allocation Problem. One use case is the planning of tours in a company that provides maintenance services for different industrial machines. The used problem which is based on the vehicle routing problem (VRP) differs from the state-of-the-art where capacity, distance, and time window constraints are optimized. In the proposed problem the assignment of employees to different routes has to be optimized that the requirements at the different stations are met.

Both approaches are used to solve the given problem through the creation of a prototype framework. The approaches were evaluated by different benchmark instances of varying sizes.

The current state-of-the-art is examined. The implementation of the prototype framework is explained extensively.

The results of the EA are promising. On the contrary, the used fuzzy approach fell short on expectations.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Companies have a high planning effort for scheduling field works. The requirements of the tasks are mostly informal values described by words. This description by words is not exact because each person interprets the requirements differently. Therefore these conditions are fuzzy. Also, it is hard to determine crisp values for the different requirements of the task. Currently, most of the research done optimized the route length of the different worker routes or optimized the delivery of goods in a specific time frame per customer on these routes. Based on, the search space of these problems is huge and exact solving algorithms find the solutions. But with increasing complexity the time needed increases exponentially [14, 3]. Metaheuristics are used to find solutions. For optimization, we will use an evolutionary algorithm that uses fuzzy logic to solve the assignment problem of the required conditions. In mind that this approach is more intuitive and suitable to the problem a comparison to an approach with crisp modeling is done.

1.2 Goals

The work aims to solve an assignment problem. The assignment problem is to send employees on suitable routes so that all stations can be serviced. Each station has certain requirements in various skills, these skills must be available to the employee so that the station can be maintained. A good solution is characterized by a minor violation of the requirements or even would not have any violation if the circumstances of the problem allow it.

The work continues to make the following contributions:

On the one hand, a literature overview is given, which consists of similar problems and points out the solution approaches.

On the other hand, the vehicle routing problem is extended by fuzzy systems, so that the needs of the customers/stations can be represented imprecisely.

An evolutionary algorithm will be developed to solve the problem. Fuzzy systems will be used to optimally cover the needs of the customers. The algorithm is not limited to a single problem instance. It can solve the problem formulation both for crisp values and with imprecise values.

Finally, an evaluation is carried out using benchmarks. Here, different problem instances are created and then checked with the help of benchmarks. First, the assignment problem is solved as a real value problem. In the second test, the benchmarks are applied to the fuzzified real values.

1.3 Structure of Thesis

First, the thesis clarifies fundamentals used later on. Furthermore, a literature overview is given. In Chapter 4 the proposed approach is discussed. Here the whole concept of the algorithm is described and modeling approaches are explained. After that, an evaluation of the approach is done. Different benchmarks are used to analyze the performance of different problem sizes. This work is ended with a summary of its results and an overview of opportunities for future work in Chapter 6.

2 Fundamentals

2.1 Evolutionary Algorithms

Evolutionary algorithms are a metaheuristic which is used for numerical and optimization problem where no exact solution algorithm exists or the exact solution can simply not be achieved efficiently. A definition of optimization problem is given by Kruse in[13].

Definiton 2.1 *(Optimization problem) An optimization problem consists of a pair* (Ω, f) *where* Ω *is the search space of all potential solutions for the problem and* $f : \Omega \rightarrow \mathbb{R}$ *the evaluation function which assigns a quality assessment* $f(\omega)$ *to each solution* $\omega \in \Omega$.
An element $\omega \in \Omega$ *is an* ***exact solution*** *if it is a global maximum of* (Ω, f).
An element is a ***global maximum*** *of* f *if* $f(\omega') \leq f(\omega)$[1], $\quad \forall \omega' \in \Omega$

The functionality of an evolutionary algorithm can be seen as a guided random search over the search space. Furthermore, the algorithm approximates a good solution by the fitness value. Hence it is assumed that better solutions result in a higher value. The algorithm applies the concept of biological evolution and is based on the insights of the Darwinian Evolution[13]. In the following section, the procedure of an evolutionary algorithm is explained in detail.

[1] $f(z) \leq f(x)$ where x is a better solution than z

2.1.1 Procedure of an EA

A basic understanding of how an evolutionary algorithm works will be given by this section. Therefore the general procedure will be shortly explained. Later in this chapter, the different components will be examined in further detail.

An evolutionary algorithm consists of different steps which are accumulated to an update loop. An episode of that update loop is referred to as iteration. Hereby, the whole process imitates evolution.

Before we can enter into the update loop the algorithm needs to be initialized. Here all necessary components of the algorithm are setup.

After the initialization, the population has to be evaluated. As a result, each individual gets it is own fitness value. The fitness value expresses the quality of an individual corresponding to a fitness function which is the quality measure of a given problem. A change in the fitness function also changes the different fitness values of the individuals in the population.

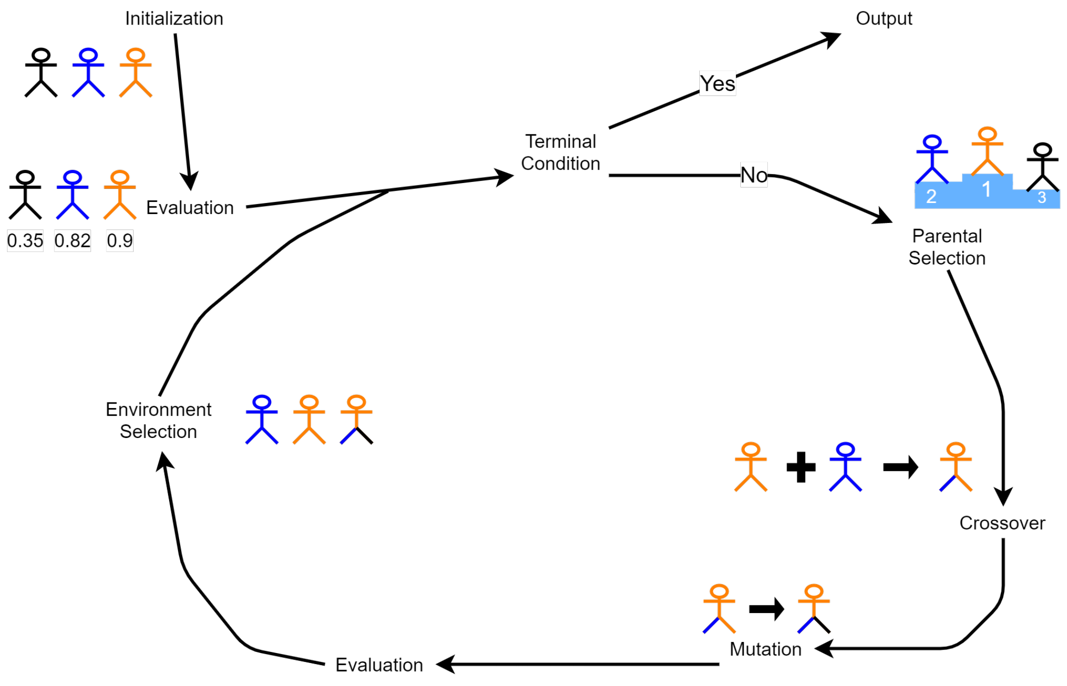


Figure 2.1: The course of an evolutionary algorithm

The following steps are part of an update loop that runs continuously. Next, the terminal condition is checked. This condition decides if the algorithm terminates. Most of the time this condition is a consolidation of two aspects. On the one hand, we define a fitness threshold that, if it is reached by at least one candidate of the current population, terminates the algorithm. To make sure that the algorithm terminates, on the other hand, a maximum number of iterations is given which ends the algorithm if the limit is reached. If the first terminal condition is not reached because of the problem size the second makes sure that the algorithm finishes.

If at least one of the termination criteria is satisfied the output is created and printed by the algorithm. Otherwise, the update loop is executed for another iteration.

The parental selection is the first step of the loop. In this step, we decide which individuals should be used to reproduce our population. The probability of each individual to engage in the reproduction depends on it is own fitness value. The higher the fitness value the higher is the probability to engage in the reproduction assuming the evolutionary algorithm is used to solve a maximization problem.

When the individuals are selected we go on to the crossover. The genotypes of two individuals get mixed by different operators. Individuals with mixed genes are referred to as children. The genotypes of the children are based on their parents.

Preceding new individuals were created based on the part which was chosen by the parental selection. Out of these individuals, a given percentage or number of the individuals will be altered after the reproduction by mutation. This procedure simulates the influence of random genetic mutations or other influences on the genetic material for example X-radiation.

Another evaluation is needed to assess the newly created individuals as preparation for the environment selection.

Currently, we exceeded the original population size, therefore, the size is adjusted in the environment selection. Based on the fitness values a new generation is taken out of the population by a selection operator. These individuals are the basis for the next iteration.

Concluding the terminal condition is checked again and dependent on its outcome the next iteration starts.

2.1.2 Encoding

The representation of a problem also called encoding is an important part of an optimization problem. Hereby, The encoding has two separate representations that correlate with each other. The two representations are genotype and phenotype. The genotype describes the internal structure for the optimization problem. The phenotype is the representation of the candidate in the search space. There are a few desirable properties that the encoding should have. One property would be that similar phenotypes are represented by similar genotypes that means if we have a slight difference between the two phenotypes the corresponding genotypes should only differ by a small amount, too. Similar candidates should have further similar fitness values because this property builds the basis of the algorithm. Another aspect is that we have to make sure that all possible solutions can be represented by the encoding. Furthermore, we have to ensure that no invalid candidates can occur. That is especially important for the different operators in the algorithm's routine. If a valid candidate is turned invalid by an operator the encoding should provide repair functions to correct that issue or should punish the invalid candidates. An invalid candidate is a solution outside of the solution space. The interrelationship between the two types can be seen in Figure 2.2[16].

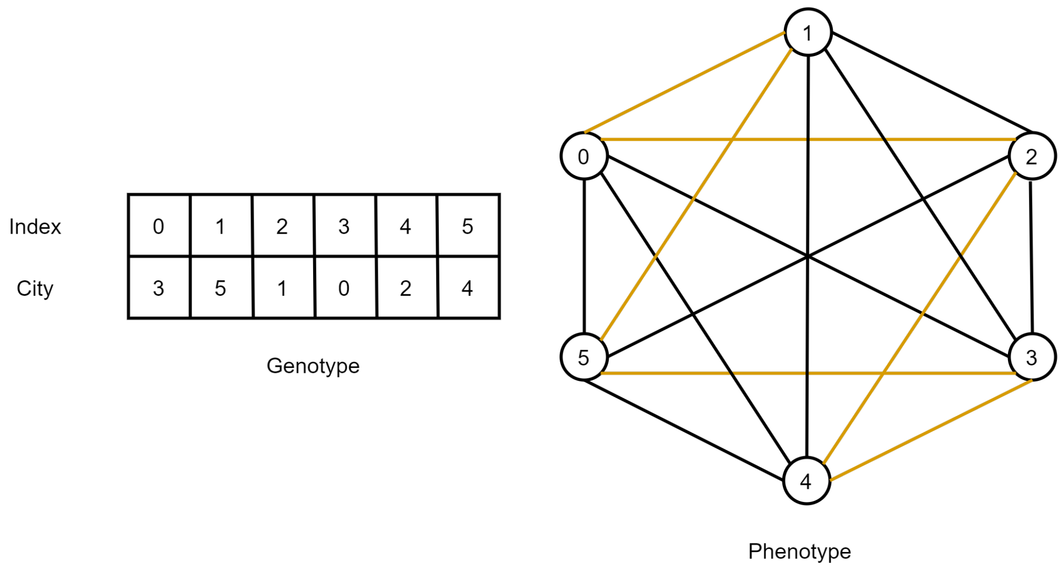


Figure 2.2: Correlation between genotype and phenotype exemplary for the TSP

The genotype of the problem is an integer array of the size six. A round tour between the cities is encoded by a permutation. The orange highlighted connections correspond to the exemplary given permutation.

2.1.3 Fitness

The core of the algorithm is represented by the fitness function. It is the pivot of the algorithm because the function assesses the different individuals. Here it has a huge impact on which individuals are chosen in the selection process. A change here causes also a change in the result. The fitness function assigns each individual a value which corresponds to the quality of the solution to the given problem [13]. A "good" fitness function fulfills a few requirements. The function should be continuous. This property excludes poles and gaps in the graph of the function which would have a negative influence on the optimization. Additionally in Section 2.1.2 we mentioned that similar candidates should have similar values. This property means that a "good" fitness function avoids hamming cliffs. Assumed that the optimization problem in Figure 2.2 is a minimization problem the fitness function could be the sum of all the distances between the cities on the route.

2.1.4 Selection

Therewith an optimization can be performed the algorithm needs to select individuals from the population. First, it has to select a part of the population for reproduction, recombination, and mutation also known as parental selection. Furthermore, with a selection operator, the next generation is chosen from the population which is the environment selection. Two typical selection operators are the roulette wheel selection and the tournament selection. For the selection itself, the selection pressure is the decisive factor which is influenced by the fitness values. The selection pressure describes how likely it is that the best individual is selected for the next generation. If the selection pressure is high it is very likely that the best individual results in the next generation. A low selection pressure results in an exploration of the solution space [13]. Here a tradeoff has to be done between the improvement of the solution and the exploration. Another way to ensure that the quality of a generation keeps a threshold is by elitism. Here a few of the best individuals are saved and updated after each iteration. This process ensures that the algorithms keep a base quality where it can improve from [13].

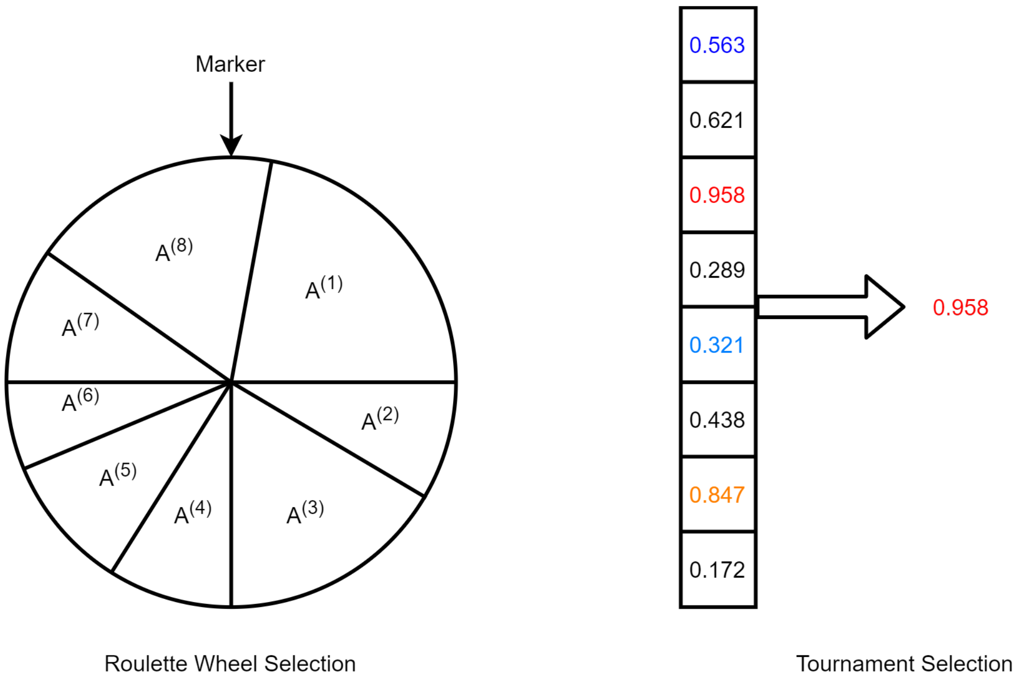


Figure 2.3: Roulette wheel selection and tournament selection of size 4

Independently which of the operators is chosen the fitness value created by the fitness function decides which individuals are picked from the population. The fitness function is modeled in a way that it creates a symbiosis with the selection operator. It is difficult to use a fitness value which is maximized in a selection operator which prefers the value of a minimization problem. The graphic in Figure 2.3 shows two commonly operators the roulette wheel selection and the tournament selection. In the roulette wheel selection, the fields $A^{(i)}$ on the wheel are distributed by the relative fitness values of the different individuals i. Let us assume that the optimization problem is a maximization problem. Whereby greater fitness values results in a higher relative fitness value and accordingly to higher relative fitness in a greater field on the roulette wheel [13]. The relative fitness is calculated by (2.1).

$$fitness_{rel}(x) = \frac{fitness(x)}{\sum_{s \in pop(t)} fitness(s)} \tag{2.1}$$

The other common operator is the tournament selection. Here individuals are randomly picked corresponding to the tournament size. These individuals compete against each other in a tournament.

2.1.5 Genetic operators

The evolutionary algorithm needs genetic operators for the exploration of the search space. Simultaneously, the fitness value is improved on the way of the exploration. Furthermore, these operators make sure that the solutions are diverse. They prevent the algorithm to get stuck in a local optimum. Therefore different operators are used which have different benefits and tasks for the algorithm. The three mainly used operators are reproduction, crossover, and mutation[13].

Reproduction

The reproduction operator copies individuals of the current generation into the next one hereby it ensures that the healthiest individuals have a higher chance to get into it. Hence the quality of the population will improve step by step through the iterations or in is stagnating in the worst case[13].

Crossover

The crossover operator is also known as recombination is one of the two big factors in an EA which should diversify the solutions in the population. This operator takes two individuals from the population and creates new individuals from them. The generated individuals are also called children. There is a huge amount of specific crossover operators. They can range from simple single-point crossovers over uniform crossovers to individually setup operators that are operating on a provided scheme. There is no "best" operator given because the effects and results are changing from problem to problem [11, 13].

The execution of three different crossovers can be seen in Figure 2.4. The first child is obtained by the use of single-point crossover. The cut point was set between the fifth and sixth index so that the parents consists now of an anterior half and rear half. The first half of the chromosome of the child contains the data of the first parent. The other half (blue digits) is taken from the second parent[11, 13]. With this simple crossover, the size of the population is doubled. The population would now contain the two parents

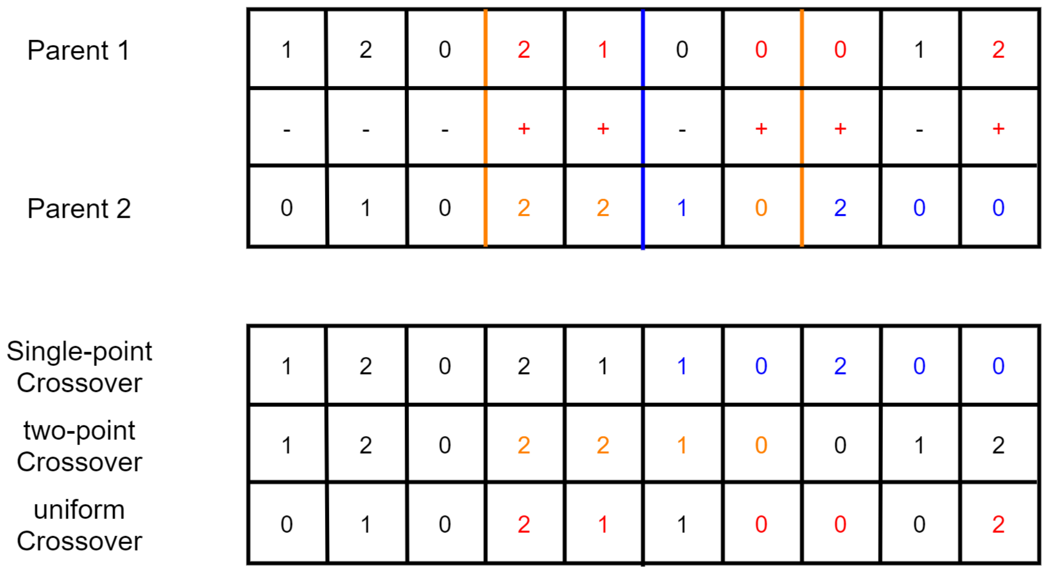


Figure 2.4: Different crossover operators

plus the two children created by the crossover. The two-point crossover uses two cutting points which were set between the third and fourth index as well as between the seventh end eighth. For the recombination, the inner part (orange digits) between the cutting points were swapped between the two parents. A scheme is needed to perform the uniform crossover. The scheme defines which parts of the chromosome should be exchanged. Here the plus indicates the genes (red digits) which are swapped between the parents. The crossover can be created with the encoding in a way that it considered all conditions of the problem[11, 13].

Mutation

The other factor which ensures that a degree of diversion is added to the population is the mutation operator. Like in nature, which the EA is imitating, sudden mutations can occur. Therefore after the recombination, a part of the population will be mutated[13].

For the mutation, the child of the uniform crossover from Figure 2.4 where used as a basis. In Figure 2.5 the results of two common mutation operators are shown. The first mutation changed a single gene of the input individual (red digit). Another commonly used mutation operator is to swap two genes with each other. This operator was used for the second mutation (orange digits)[11, 13, 19]. New genetic material is added by this process to the evolutionary

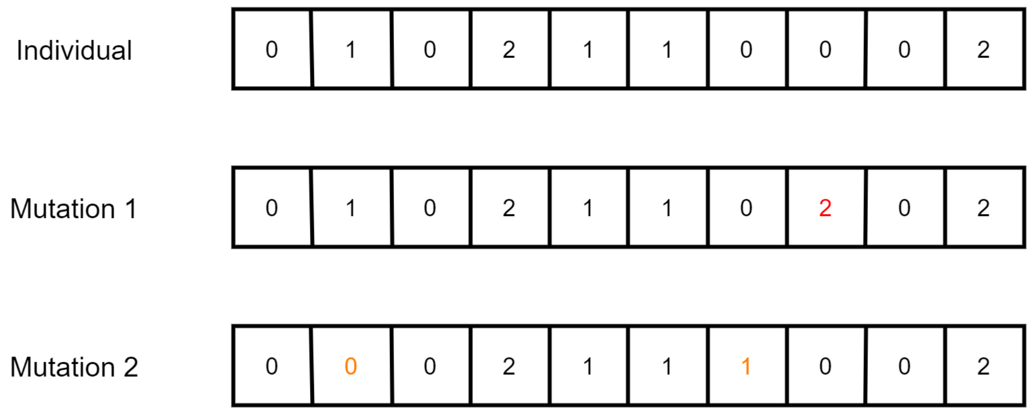


Figure 2.5: Different individuals resulting from mutation

algorithm which can be part of better solutions that were not possible to this point because the necessary components were not present until now.

2.2 Vehicle Routing Problem

The vehicle routing problem (VRP) is an extension of the traveling salesman problem (TSP), which are both graphical problems. These problems are often used as a basis for the modeling of optimization problems where the traveled distance has to be optimized. In the traveling salesman problem, each city is represented by a node, and all cities are connected by edges. The goal is to find an optimal route between cities which should be minimal. The only constraint which is given is that each city can be visited only once. A detailed definition can be found in [1]. A possible representation of the reality would be a trip around the world where you start and end in the same country.

In the vehicle routing problem different TSP's have to be solved, so that total distance is minimized. Different tours have to be planned which all start and end at a central depot. The number of tours is given. For example in reality it is represented by the size of the carpool or staff size. This problem can grow in complexity by adding different constraints. The vehicle routing problem was originally introduced by Dantzig[8]. Two commonly researched fields are the capacitated vehicle routing problem (CVRP) and the vehicle routing problem with time windows (VRPTW). In the CVRP the vehicle got a capacity that has to be regarded in the planning of the tours. A contempt of the constraint has negative consequences. The VRPTW includes a time constraint. Here the nodes have to be served in a given time window. Also, a combination of both versions is possible. The result would be the capacitated vehicle routing problem with time windows (CVRPTW)[20, 6]. The VRP is the problem that all delivery services like UPS, DHL, and DPD have to solve daily.

2.3 Task-Allocation Problem

The Task-Allocation Problem that is used in this thesis is based on the VRP. A few points were modified so that the model represents the problem which can be seen in Figure 4.1.

The nodes of the VRP have three values. Each value belongs to a skill so that each of these values represents a skill level. The nodes can be combined into different routes. These routes are served by employees as tours. Each route is only served by one worker. The employees have like every node their own sets of the three different skill levels. The skill levels on the side of the nodes can be seen as a requirement which the employee of the tour has to fulfill. This fulfillment is the optimization criteria so that the quality of every solution can be expressed by the violation/fulfillment of the requirements on the node side by the employee. Out of this reason, an optimization of the distance as well as the encoding of the order in which the different stations are served is not executed. It is assumed that the distance and the sequence of the tour are both included in the allocation of the different tasks to the employees. Section 4.1 goes into detail and explains all correlations. The goal is to find an optimal allocation between stations and workers so that the requirements of the stations are fulfilled. Hereby each worker is assigned to a route with different stations.

2.4 Fuzzy Systems

2.4.1 Fuzzy Sets

In the set theory, a value can either be a member of the set or not be a member. The idea behind fuzzy sets is that we do not separate between these two values of membership. Instead, fuzzy sets have partial membership, so we look at which degree or extent the value is a member of a fuzzy set [21].

Kruse[13] defines a fuzzy set as the following:

Definiton 2.2 *(Fuzzy Set) A fuzzy subset or simply a fuzzy set μ of a set X (the universe of discourse) is a mapping $\mu : X \to [0,1]$, which assigns to each element $x \in X$ a degree of membership $\mu(x)$ to the fuzzy (sub)set μ. The set of all fuzzy (sub)sets of X is denoted $F(X)$. A conventional set $M \subseteq X$ can be viewed as a special fuzzy set by identifying it with its characteristic function or indicator function.*

$$I_M : X \to \{0,1\}, \quad x \mapsto \begin{cases} 1 & \textit{if } x \in M \\ 0 & \textit{otherwise} \end{cases}$$

Representation

A fuzzy set can be associated with linguistic expression. This expression can describe an imprecise value or an imprecise interval. "around 3", "of middle height", "very tall" are examples of these expressions. The sets which model these expressions should be convex. That means that the set should monotonically increase to a specific point and from there monotonically decrease [13].

Fuzzy sets can be represented by different functions that model different basic shapes. In praxis, only a few simple shapes are used. With this representation, each set can be uniquely specified by a few parameters. Commonly used parametric fuzzy sets are triangular functions. [13]

The triangular functions can be specified like this:

$$\Lambda_{A,B,C} : \mathbb{R} \to [0,1], x \mapsto \begin{cases} \frac{x-a}{b-a} & \text{if } a \le x \le b \\ \frac{c-x}{c-b} & \text{if } b \le x \le c \\ 0 & \text{otherwise} \end{cases}$$

Membership vs Probability

The membership of a value towards a set should not be mistaken as a probability. These are two disparate concepts. To illustrate the difference between the gradual membership and probability we take a look at different statements. "This tree is old." This statement is fuzzy, because of the imprecise value "old" a numerical evaluation is not possible because we cannot measure them. "The banana comes probably from Ecuador." Here we have an uncertainty about the statement *from Ecuador*. The last statement will be: "Red cars are very big." It is a combination of uncertainty with a fuzzy value. *big* cannot be defined in more detail. With the modificator *very* we can assume the grade of it is bigness. Further a statistical assertion over the set of all cars is made. The explanation is based on the example given by Bezdek[4, 17].

2.4.2 Fuzzification

A fuzzy set can consist of different fuzzy sets that describe the linguistic variable. The different sets represent the different linguistic terms of which the variable consists. For example, the linguistic variable would be the "pressure" of the tire. This variable is segmented into different terms. These terms are "low", "normal" and "high". This coherence is shown in Figure 2.6. Through the segmentation, the vague expression is more precise as a single fuzzy set for the linguistic variable with one term because now we differentiate between the various degree of memberships for the terms within the linguistic variable.

Until now we have single values. We need to find operators like in the binary or boolean logic which enable us to connect different variables. Therefore these operators should fulfill different properties.

Let us assume the candidate of a truth function of the conjunction for fuzzy propositions is the function $t : [0,1]^2 \to [0,1]$. The function needs to be commutative and associative because the truth value of the conjunction does not depend on the order of the propositions. That means

(T1) $t(\alpha, \beta) = t(\beta, \alpha)$
(T2) $t(t(\alpha, \beta), \gamma) = t(\alpha, t(\beta, \gamma))$

should hold for α, β and γ.

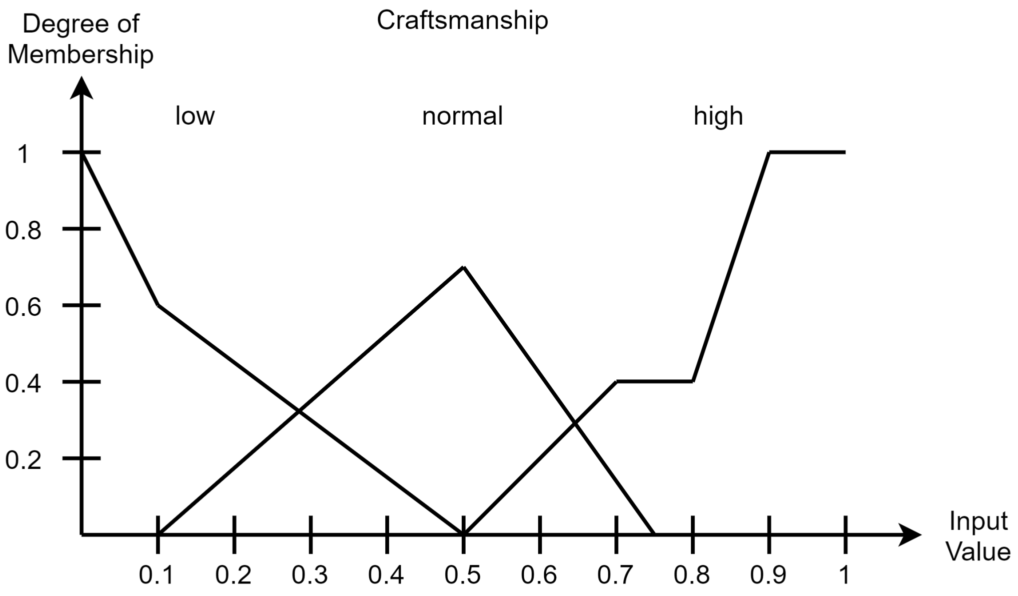


Figure 2.6: Relation between the fuzzy variable craftsmanship and its linguistic terms

In additon if x has a lower thruth value then ψ the value of the conjunction $\varphi \wedge \psi$ should be less than the thruth value of the conjunction $\varphi \wedge x$. Therefore a monotonicity criteria of t is needed:

(T3) From $\beta \leq \gamma$ follows $t(\alpha, \beta) \leq t(\alpha, \gamma)$,

for all α, β and γ. Based on the commutativity of the property (T1) and (T3) concludes that t has to be nondecreasing for both arguments.

Furthermore, we need to make sure that the truth value of φ does not change if we connect it with a true proposition of ψ. That means that the truth values of the proposition φ is the same as $\varphi \wedge \psi$. For the truth function t this leads to

(T4) $t(\alpha, 1) = \alpha,$ for all α

Definiton 2.3 *A function $t : [0,1]^2 \rightarrow [0,1]$ is called t-norm (triangular norm), if the axioms (T1)-(T4) are satisfied.*

The truth values of the conjunction coincide with the values of the t-norm which satisfies the axioms. Each t-norm can be used for the conjunction. The axiom (T4) concludes that t(1,1) = 1 and t(0,1) = 0 for every t-norm t. Furthermore using the commutativity of axiom (T1) we obtain $t(1,0) = 0$.

In addition the monotonic property of (T3) and t(0,1) = 0 result in t(0,0) = 0. Concluding that the truth function $t(\alpha, \beta) = min(\alpha, \beta)$, so that the conjunction can be expressed by the minimum of α and β.

We can use the same approach to define criteria which the candidates for the truth function for the disjunction have to satisfy. The properties (T1)-(T3) apply to the disjunction as well. The fourth axiom (T4) has to be changed that the truth function describes the disjunction and leads to

$(T4') \quad s(\alpha, 0) = \alpha,$ for all α

which means that the truth value of the disjunction between the proposition φ and the false proposition ψ is the same as the truth value of the proposition φ.

Definiton 2.4 *A function $s : [0,1]^2 \rightarrow [0,1]$ is called t-conorm (triangular conorm) if the axioms (T1)-(T3) and (T4') are satisfied.*

Using the dual concept between t-norm and t-conorm, a t-norm can be transformed into a t-conorm and vise versa. Therefore we use the De Morgan's Law. The resulting formulas corresponding to the De Morgan's Laws can be seen in (2.2) and (2.3).

$$t(\alpha, \beta) = 1 - s(1 - \alpha, 1 - \beta), \tag{2.2}$$

$$s(\alpha, \beta) = 1 - t(1 - \alpha, 1 - \beta), \tag{2.3}$$

In the end, the corresponding t-conorm of the t-norm which is described by the minimum is the maximum.

2.4.3 Operation on Fuzzy Sets

Currently, we can derive fuzzy sets form crisp numbers. These fuzzy sets are represented by their characteristic function. Operators have to be defined which allows the interaction between the sets. These operations are inspired by the classical set theory. Methods for the intersection, union, and complement will be introduced. Therefore the classical set theory will be extended to fuzzy sets.

Intersection

An element x belongs to the intersection of the sets M_1 and M_2, if and only if it belongs to both sets. This affiliation of x towards the intersection does not depend on the membership of any other value $y \neq x$ but solely on the membership of x to M_1 and M_2. This means,

$$x \in M_1 \cap M_2 \iff x \in M_1 \wedge x \in M_2 \tag{2.4}$$

We assume that the degree of membership of an element x to the intersection solely depends on the membership values of x to the fuzzy sets $\mu_1$andμ_2. The degree of membership $\mu(x)$ states to what degree x is an element of μ. If we want to calculate the membership degree of an element x to the intersection between the fuzzy sets μ_1 and μ_2 we can use equation (2.4) to do so. The degree of membership is equal to the truth value of the conjunction "x is an element μ_1 AND x is an element of μ_2". With the knowledge of Section 2.4.2 we know that the value of the conjunction of two fuzzy propositions can be modeled by a t-norm. The intersection between two fuzzy sets μ_1 and μ_2 is defined as the fuzzy set $\mu_1 \cap_t \mu_2$ with $(\mu_1 \cap_t \mu_2)(x) = t(\mu_1(x), \mu_2(x))$[13].

Union

Analogously the union of two fuzzy set can be defined with slight changes of the equation (2.4) as $x \in M_1 \cup M_2 \iff x \in M_1 \vee x \in M_2$. Which leads to $(\mu_1 \cup_s \mu_2)(x) = s(\mu_1(x), \mu_2(x))$, as the union of $\mu_1$andμ_2 with respect to the t-conorm s[13].

Complement

The formula $x \in \overline{M} \iff \neg(x \in M)$ the complement of a fuzzy set can be derived. The fuzzy set $\overline{\mu_1}(x) = 1 - \mu(x)$ results of the assignment of the truth function $w_\neg(\alpha) = 1 - \alpha$ to the negation[13].

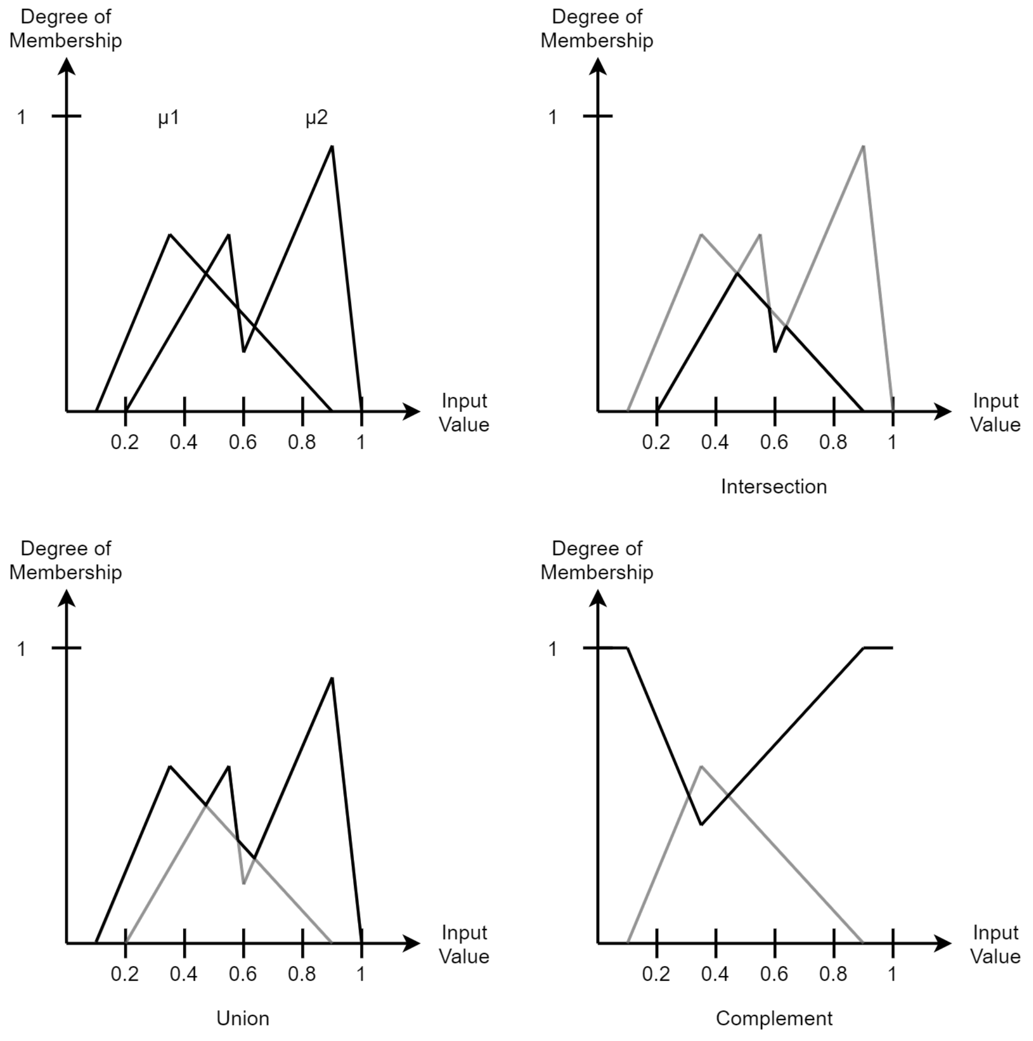


Figure 2.7: The different results of the fuzzy set operations. $\mu_1 \cap_t \mu_2$ was performed with the minimum and for the union $\mu_1 \cup_s \mu_2$ the maximum was used

2.4.4 Inference

We know how we can transform a crisp value into a fuzzy set. But what do we do if we have multiple input values which have to be combined to an output value? Here comes a step called inference that comes into play. The concept of the inference works with a rule base. This rule base includes different rules which match the input values to output values. The rule base $\mathscr{R}$ consists of if-then-rules $R \in \mathscr{R}$ which have the form

$$R: \text{ If } x_1 \text{ is } \mu_R^{(1)} \text{ and } \dots \text{ and } x_n \text{ is } \mu_R^{(n)} \text{ then } y \text{ is } \mu_R \tag{2.5}$$

where $x_1, \dots, x_n$ are input values and y is the output value. The fuzzy sets $\mu_R^{(i)}$ or μ_R are linguistic variables which could be "hot", "long distance" or "later" [15, 13].

A single rule should not be seen as a logical implication. It is more a piecewise definition of a function. If the rule base $\mathscr{R}$ consists of the rules $R_1, \dots, R_r$, we should understand it as a piecewise definition of a fuzzy function, that is

$$f(x_1, \dots, x_n) \approx \begin{cases} \mu_{R_1} \text{ If } x_1 \approx \mu_{R_1}^{(1)} \text{ and } \dots \text{ and } x_n \approx \mu_{R_1}^{(n)} \\ \vdots \\ \mu_{R_r} \text{ If } x_1 \approx \mu_{R_r}^{(1)} \text{ and } \dots \text{ and } x_n \approx \mu_{R_r}^{(n)} \end{cases} \tag{2.6}$$

The graph of the function is obtained by

$$graph(f) = \bigcup_{i=1}^{r} (\hat{\pi}_1(\{x_i^{(1)}\}) \cap \dots \cap \hat{\pi}_n(\{x_i^{(n)}\}) \cap \hat{\pi}_Y(\{y_i\})) \tag{2.7}$$

Using the minimum for the intersection and the maximum for the union results in a fuzzy set which represents the fuzzy graph of the function which is described by the ruleset $\mathscr{R}$

$\mu_{\mathscr{R}} : X_1 \times \dots \times X_n \times Y \to [0, 1]$,
$(x_1, \dots, x_n, y) \mapsto max_{i \in \{1, \dots, r\}}\{min\{\mu_{R_i}^{(1)}(x_1), \dots, \mu_{R_i}^{(n)}(x_n), \mu_{R_i}(y)\}\}$

in the case of a finite rule base $\mathscr{R} = \{R_1, \dots, R_r\}$. [13]

2.4.5 Defuzzyfication

The transformation of a value into a fuzzy set was shown in the section before. Now we need methods for the way back which transforms a fuzzy set back to a value. There are different possibilities to solve this problem. Common methods are the center of gravity, the center of area, or the mean of maxima.

Center of Gravity

The center of gravity is the normed mean of the fuzzy set. The result represents the point were the fuzzy set is balanced. The center of gravity is calculated by an integral seen in (2.8). The formula can be split into two separate ones. The first one calculates the area of the fuzzy set (2.9) and the second formula uses this result from (2.9) as a normalization factor which is applied to the integral of the base points and the membership values (2.10).**??**

$$x_{COG} = \frac{\int_{x^-}^{x^+} x\mu(x)dx}{\int_{x^-}^{x^+} \mu(x)dx} \tag{2.8}$$

$$A = \int_{x^-}^{x^+} \mu(x)dx \tag{2.9}$$

$$x_{COG} = \frac{1}{A}\int_{x^-}^{x^+} x\mu(x)dx \tag{2.10}$$

where x_{COG} is the center of gravity, A the resulting area, x the corresponding crisp value, and $\mu(x)$ the grade of membership of x[17].

Center of Area

This method is similar to the preceding one. The result represents the point, where the two subareas, which are created by that point, are equal. Therefore the area of the fuzzy set is calculated as in (2.9) and the border which creates equal subareas has to be found (2.11).

$$\int_{x^-}^{x^m} \mu(x)dx = \int_{x^m}^{x^+} \mu(x)dx \tag{2.11}$$

where x^m represents the center of area.

Mean of Maxima

The mean of maxima is also similar to the center of gravity. The calculation stays the same only that we concentrate on the intervals of the maxima. In most of the cases, the maxima of a fuzzy set are single points. If that is the case the mean of maxima is the mean of all those points of maxima[13].

3 Related Work

3.1 Fuzzy Optimization in VRP

Fuzzy-based methods were used by Brito et al. to solve the VRP where the input variables are influenced by uncertainty. The underlying problem was modeled as VRP and then transformed to a linear programming problem. The uncertainty in the information was represented by the fuzzification of the constraints. Different methods are discussed to model different information. In the end a short simple example is given [6, 5].

Nasser reviewed the different approaches by the literature. Further, he gives a summarization of the modeling approaches regarding the constraints [9].

A fuzzy chance-constrained program was developed by Cao et al. to solve the open vehicle routing problem with fuzzy demands. The modeling is based on fuzzy credibility theory which is then integrated into a differential evolution algorithm and stochastic simulations. The hybrid algorithm was able to find a good parameter setting to solve the problem. In the future, the authors want to adopt the approach to other metaheuristic techniques [7].

3.2 EA Optimization in VRP

An evolutionary algorithm solution of the VRP is introduced by Baker et al. the approach and setup of the genetic algorithm are explained. The results of the hybrid genetic algorithm with neighborhood search methods were competitive in terms of solution time and quality to the best-known results for benchmark VRPs which were obtained by tabu search and simulated annealing[2].

4 Proposed Approach

In this chapter, the modeling of the algorithm will be described. First, a formulation of the problem will be given and explained. Then we want to go into the encoding which was used. Following we will picture how the whole evolutionary algorithm works before the different components of the algorithm are described in more detail afterward.

4.1 Formulation of the Problem

In Section 2.2 and Section 2.3 an introduction of the problem which has to be solved was given. In the following section, we want to introduce important terms which are later used in the formulas and also want to give a further understanding of the problem.

The problem consists of different components that have different dependencies with each other that are visualized in Figure 4.1. The first component is the station which has to be served. Each station has its own unique skill set. There are three different skills: craftsmanship (CS), technical understanding (TU), and soft skills (SoSk). When the skills are mentioned corresponding to a station they are referred to later on as requirements. Each skill has a value in the range of [0:100].

The given circumstances can be expressed by the following mathematical annotations:

$$\begin{aligned}
s &\in S \\
s_i^t(x) &\mapsto [0:100] \\
|S| &= n \\
n &\in N
\end{aligned}$$

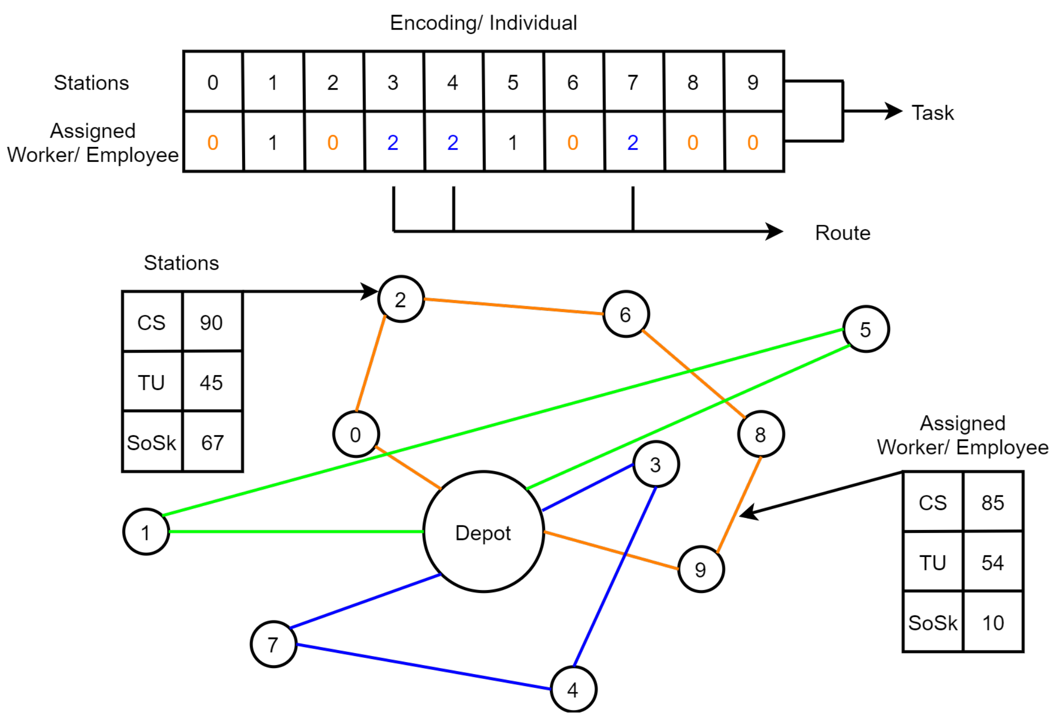


Figure 4.1: Overview of the whole procedure of the EA

where S is the set of all stations, s is a station of the set, $|S|$ is the size of the set, n is the number of stations which is provided by the user, x is an individual, i is the index of a station within an individual and $s_i^t(x)$ is the value of the requirements to the given skill t.

Another component is the worker also referred to as an employee. The workers have the same properties as the stations. If skills are mentioned corresponding to a worker they are referred to as capabilities later on. These conditions can be expressed by mathematical expressions similar to the stations.

$$\begin{aligned} w &\in W \\ w_i^t(x) &\mapsto [0:100] \\ |W| &= n \\ n &\in N \end{aligned}$$

where W is the set of all stations, w is a station of the set, $|W|$ is the size of the set, n is the number of stations which is provided by the user, x is an individual, i is the index of a station within an individual and $w_i^t(x)$ is the value of the capability to the given skill t.

The third part of the problem is a task. A task is a part of the individual shown in Figure 4.1. Furthermore, it is represented by a single index of the encoding and shows the interaction between employees and stations. The interaction between these components becomes important when the fitness values of the individuals are calculated. This allocation of a worker to a station is referred to as tasks.

An aggregation of the tasks of a single worker is a route and resembles a TSP of the VRP. In Figure 4.1 the routes of the different workers corresponding to the individual can be seen. The different routes are highlighted by the colors blue, orange, and green.

The last component condenses all the other components in itself and is the basis of the evolutionary algorithm, the encoding. The individual encodes the representation of the VRP in Figure 4.1. Hereby the correlation between the different routes and the representation in the encoding can be seen. The individual is a single instance of a population of solutions so that the values and layout of the VRP in Figure 4.1 are only explanatory. Section 4.2 goes into detail on this topic.

4.2 Encoding

As encoding for the problem, an array of integers was chosen where the length of the array is given by the number of the stations which have to be served. At each index, in the array, the worker is encoded which is serving the station.

Station	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Worker	9	1	2	1	6	2	1	1	9	2	2	1	1	6	2	2

Table 4.1: Encoding of an individual

An example of the encoding of an individual could look like Section 4.2. The worker with the id one is serving six stations, the worker with id two serves also six stations, the worker with the ids six and nine are both serving two stations.

4.3 Procedure of the Algorithm

The algorithm which is used follows the procedure in Section 2.1.1. The cycle which is used had been adjusted in a few aspects. The adjustments are mentioned when they take place. Subsequently, the routine of the algorithm is explained. An overview of the algorithm can be seen in Figure 4.2. Firstly the population of the proposed algorithm is initialized randomly. The used procedure for the initialization can be seen in Algorithm 1. Herefore each individual of the population is assigned with an array of different allocations. These allocations are assignments of workers to stations. This array is initialized randomly. Therefore a worker is selected out of the worker pool and assigned to the current station. The routine is repeated until each station has a worker assigned to it.

```
Result: Initialization of the population
Population ← 0;
while $i < numberOfIndividuals$ do
    forall $Tasks$ do
        Assign worker of the staff to the task;
    end
end
return Population
```

Algorithm 1: Initialization of the starting population

Then a first evaluation of the population is done. A detailed view is given in Section 4.4. Here the composition of the different fitness values will be explained. Following the evaluation, the terminal condition is checked. If the condition is fulfilled the output will be generated otherwise we go deeper into the evolutionary algorithm cycle.

To this point, each individual is initialized and got a fitness value assigned. Next, parental selection chooses the individuals for reproduction, crossover, and mutation form the population. The selection operator is a tournament selection. From there on the algorithm differs from the general cycle shown in Section 2.1.1. The used approach divides the chosen individuals into two groups. The first group will only be mutated. This adaptation should provide continuous input of genetic material. This is advantageous when the population is almost converged to an optimum. Then the crossover cannot provide

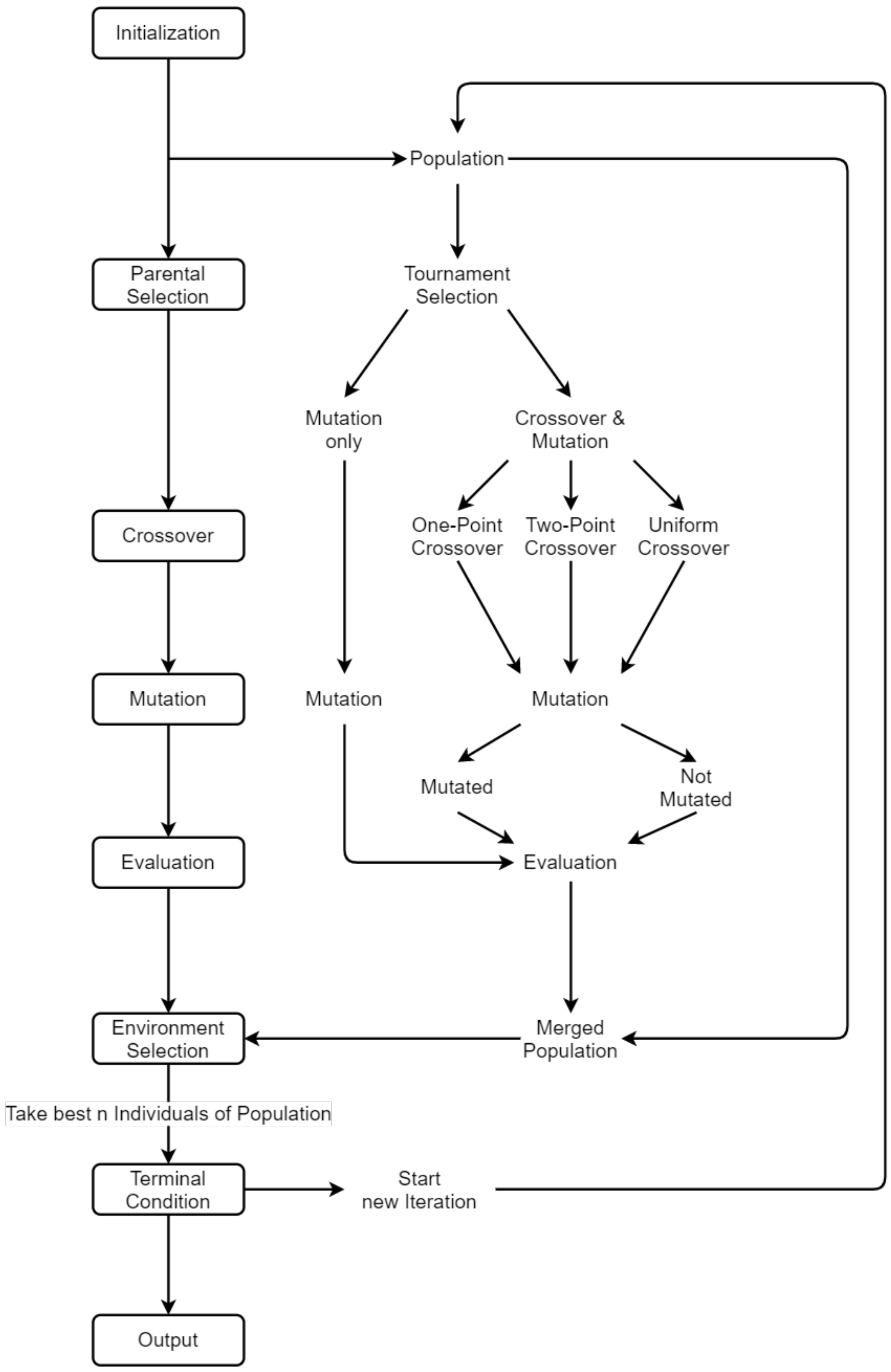


Figure 4.2: Overview of the whole procedure of the EA

any diversion because most of the solutions are very similar. Also, only small changes have to be made to find the optima. The second group pass through the described way of crossover and mutation which are shown in Figure 4.2. The used crossover and mutation operators are explained in more detail in Section 4.6 and Section 4.7. On the second route, only a part of the population is mutated. Then all newly created individuals are evaluated before the next generation can be chosen by the environment selection. After the new generation is selected the routine goes back to the start of the cycle where the terminal condition was checked. The deviations from the general cycle were made to increase diversity. Concluding that more genetic material enters the population. These mechanisms should prevent that the algorithm gets locked in a local optimum.

4.4 Fitness Function

For this step, different fitness functions are used. The fitness function is designed in a modular way that consists of different parts which are aggregated to a final value. The different partial results are saved in five variables. These variables are the head of Table 4.2. The "crisp satisfaction" is the result of the evaluation where crisp numbers were used. Contrary to the "crisp satisfaction", the "fuzzy satisfaction" includes the value of the fuzzy solution. "ConstraintWH" and "ConstraintCP" are punishment factors that can contribute to the final fitness value. The constraints should represent different conditions which should be fulfilled by the solutions. Section 4.4 goes into more detail on both constraints and their calculation. The last variable, the "completed tasks", aggregate the tasks which could be served and fulfilled by the individual. The later is only used for testing purposes in the evaluation at the end of the thesis and does not contribute to the different scores. There are eight possible problem instances. For example, the problem should be solved with crisp values regarding both constraints. The composition of this variant is shown in Table 4.2 by row four. The variant "CrispBothConstraints" is influenced by the crisp by the "crisp satisfaction" and both constraints. The values are summed and generate the fitness value for this case. The parental and environment selection operates with these final values.

	Crisp Satisfaction	ConstraintWH	ConstraintCP	Fuzzy Satisfaction	Completed Tasks
CrispNoConstraints	**X**				**TP**
CrispConstraintWH	**X**	**X**			**TP**
CrispConstraintCP	**X**		**X**		**TP**
CrsipBothConstraints	**X**	**X**	**X**		**TP**
FuzzyNoConstraints				**X**	**TP**
FuzzyConstraintWH		**X**		**X**	**TP**
FuzzyConstraintCP			**X**	**X**	**TP**
FuzzyBothConstraint		**X**	**X**	**X**	**TP**

Table 4.2: Composition of the different fitness values. Variable contributes to the fitness value if the box is checked by an X. TP stands for testing purpose in the evaluation.

Crisp Satisfaction

"Crisp Satisfaction" is the first partial result that contributes to the final fitness value. This result is calculated with crisp values. These values are in the range between zero and one. The different steps of the calculation can be seen in Algorithm 2. The function iterates over the individual which describes which worker is assigned to which station. At each index, the function looks up if the requirements were met. Therefore the difference between the requirements and the skill level of the assigned worker is calculated for each trait. If the difference is positive the value will be cut down to zero to prevent a tradeoff between "worse" and "good" allocations by "overfitting" the task which results in higher satisfaction.

```
Result: Crisp Satisfaction of the individual
CrispSatisfaction ← 0;
forall Tasks do
    forall Traits do
        CrispSatisfaction += Min(0, Worker Skill - Requirements);
    end
end
CrispSatisfaction ← CrispSatisfaction / 3;
return CrispSatisfaction
```

Algorithm 2: Calculation of the CrispSatisfaction

For example, the required value is 45. That means that each worker with a value of 45 in this trait and higher fulfills the task to it's fullest. Without the constraint, the algorithm would optimize the given in the following way. The value of 45 resembles a breakpoint. From here on all values greater than 45 result in a satisfaction greater than zero, lower values in a negative one. Now the highest possible satisfaction value is 55. If the algorithm assigns a worker with a value of 100 in this trait to the task it would result in a satisfaction level of 55. Nonetheless, this worker fulfills the task as good as a worker with a value of 45 so a huge amount of potential is wasted. Also, the "over-fitting" made it possible to compensate for bad task allocations where the satisfaction level is negative. The maximum amount which can be compensated is negative 55. There were two factors which contradicted that way of modeling. First the possibility of the compensation of bad task allocations within the individual. Secondly, the modeling does not represent reality because it does not matter how much the requirements are exceeded. If the threshold is reached the task can be fulfilled to it's fullest and can be completed. Therefore all exceeding values are restricted to zero.

$$
\begin{aligned}
Violation &= \sum_{\substack{i \in \mathbb{N} \\ 0 \leq i \leq |S|}} \sum_{t} min(0, w_i^t(x) - s_i^t(x)) \\
CrispSatisfaction &= \frac{Violation}{3} + |S|
\end{aligned}
$$

The value for each trait is added to the variable which saves the "crisp satisfaction". After all the differences over the stations were acquired the fitness value is divided by 3. The result is now in the range of zero to minus one. For an intuitive understanding of the result, the size of the set S is added. Consequently, the value of the "crisp satisfaction" is in the range of zero and the size of the set S. The value of zero describes that none of the requirements could be fulfilled. On the other hand, a value of the size of the set S describes that all stations could be maintained as a result of the fulfillment of the requirements.

Fuzzy Satisfaction

The second part of which the fitness can consist of is the fuzzy satisfaction. Here the function also iterates over the individual and selects the station and worker at the given index.

As in Algorithm 3 shown the algorithm fuzzifies all the traits in each iteration first. The fuzzified values will be then combined to a result fuzzy set. The result fuzzy sets per trait will then be unified to a result fuzzy set which will

```
Result: Fuzzy Satisfaction of the individual
FuzzySatisfaction ← 0;
forall Tasks do
    forall Traits do
        Trait Result = Inference Fuzzify(Requirement) & Fuzzify(Worker
          Skill);
    end
    UnifiedSets ← Union off all Trait Results;
    Final Result ← defuzzify UnifiedSets;
    FuzzySatisfaction ← FuzzySatisfaction + Final Result;
end
return FuzzySatisfaction
```

Algorithm 3: Calculation of the FuzzySatisfaction

represent the solution for this index. This fuzzy set which includes all partial solutions will then be defuzzified. The resulted crisp value is then added to the fuzzy satisfaction and the next iteration begins.

Following the different steps are described more specifically.

First, each trait will be fuzzified. Therefore a fuzzy set is needed. The set which is used for the fuzzification is split up in three linguistic terms which are small (S), middle (M), and great (G). The terms are defined by the following characteristic functions of the linguistic terms:

$$S(x) = \begin{cases} -2x + 1, [0; 0.5] \\ 0, otherwise \end{cases}$$

$$M(x) = \begin{cases} 4x - 1, [0.25; 0.5] \\ -4x + 3, [0.5; 0.75] \\ 0, otherwise \end{cases}$$

$$G(x) = \begin{cases} 2x - 1, [0.5; 1] \\ 0, otherwise \end{cases}$$

The characteristic functions are based on the following deliberations. The linguistic terms small, middle, and great are gradations of the skill level of the worker given to the skill. An equal distribution of the terms as well as almost the same gradient of the graph was chosen to allow smooth transitions between the different terms. Furthermore, the same characteristic functions are used for the modeling of the suitability. The goal is to find an allocation of workers which fulfills the given requirements. The case that the requirements are met is modeled by the linguistic term possible which has the same characteristic function as middle. The other terms of not possible and surpassed have their highest degree of membership at the extreme points. Their degrees of membership decrease gradually when suitability converges to the middle point. The "over-fitting" and "under-fitting" of the problem should have the same influence on the result. Therefore, the fuzzy sets were modeled symmetrically.

For example the following values are given:

Target	CS	TU	SoSk
Station	0.7	0.35	0.9
Worker	0.4	0.7	0.8

Table 4.3: Example Values

Through the fuzzification, we obtain the degrees of membership for the different terms of the fuzzy variable of the corresponding trait related to the crisp value. The crisp value of 0.7 from the station in craftsmanship (CS) will result in the degree of membership of 0.2 for middle (M) and 0.4 for great (G). The fuzzy set represented in Figure 4.3 is the set with which the crisp values are fuzzified. Like described earlier in Table 4.3 the crisp value of 0.7 in craftsmanship is applied to the different characteristic functions. This results in the two degrees of membership $M(0.7) = 0.2$ and $G(0.7) = 0.4$. These are the degrees of membership recording to the different linguistic terms. Therefore the crisp value is more affiliated towards great (0.4) than to middle (0.2). After this procedure is done for all traits we obtain two three by three matrices one for the station and one for the worker.

The results are summarized by the following table:

Now all the original trait values are fuzzified. Currently, it is not possible to decide if the worker is suited to get the job done or not. Therefore a new fuzzy variable is introduced. The suitability is constructed by the previously obtained fuzzified values. In addition to the new variable comes a rule base as well.

With this rule base, it is possible to transform the two fuzzy variables used in the fuzzification to the new fuzzy variable which describes the suitability (Suit).

Station				Worker		
Terms/ Variables	CS	TU	SoSK	CS	TU	SoSK
S	0	0.3	0	0.2	0	0
M	0.2	0.4	0	0.6	0.2	0
G	0.4	0	0.8	0	0.4	0.6

Table 4.4: Fuzzified Values for the station and worker from Table 4.3

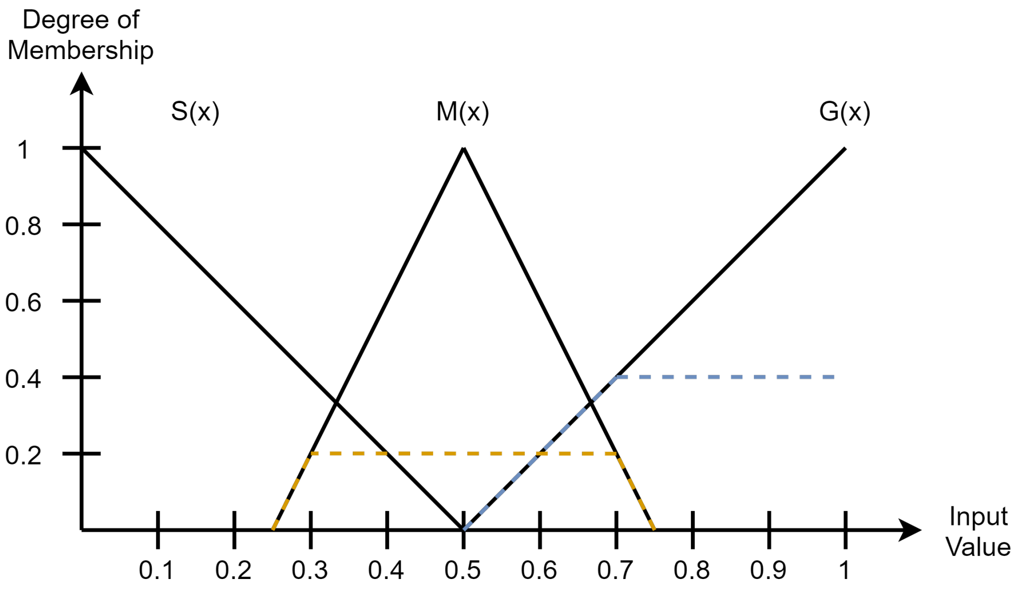


Figure 4.3: Degrees of membership of the crisp value 0.7 to the different linguistic terms of the fuzzy variable craftsmanship

Station CS	Worker CS	Suitability CS
S	S	P
S	M	SUR
S	G	SUR
M	S	NP
M	M	P
M	G	SUR
G	S	NP
G	M	NP
G	G	P

Table 4.5: Rulebase for the inference, small (S), middle(M), great (G), not possible (NP), possible (P), surpassed (SUR)

This rule base stays the same for all the fuzzy variables. Furthermore, we also need a norm which calculates the value of the suitability corresponding to the two input variables. The algorithm computes with the min-norm. This norm takes the minimum of the degrees of membership of the inputs and assigns it to the result variable. I choose the min-norm at this point to represent the point that nobody can work over his limitations, therefore, the unit which is the smallest will describe the suitability the worker has in this given combination of the inputs.

A rule can be interpreted as follows:

If $StationCS(S) \wedge WorkerCS(S) \rightarrow SuitCS(P)$

by applying the Min-Max T-(co)-norm:

$SuitCS(P) = min(StationCS(S), WorkerCS(S))$

At this point, the crisp value is fuzzified shown by Table 4.4. Exemplary I will explain the inference step based on the trait craftsmanship (CS) corresponding to the linguistic term of not possible from the suitability. On the station side the values are (0, 0.2, 0.4) and on the worker side (0.2, 0.6, 0).

The values for the suitability will be acquired in the following way:

$$\begin{aligned}
SuitCS_0(NP) &= min(StationCS(M), WorkerCS(S)) \\
SuitCS_0(NP) &= min(0.2, 0.2) \\
SuitCS_0(NP) &= 0.2
\end{aligned}$$

The other two values are calculated in the same way.

$$\begin{aligned}
SuitCS_0(NP) &= min(0.4, 0.2) \\
SuitCS_0(NP) &= min(0.4, 0.6)
\end{aligned}$$

The linguistic term not possible of the suitability fuzzy variable regarding the craftsmanship has, therefore, the values of (0.2,0.2,0.4).

After that operation, each linguistic term has attained three values through the inference. The complete result can be seen in table Table 4.6.

	Suitability								
Variables/ Terms	NP			P			SUR		
CS	0.2	0.2	0.4	0	0.2	0	0	0	0
TU	0	0	0	0	0.2	0	0.2	0.3	0.4
SoSk	0	0	0	0	0	0.6	0	0	0

Table 4.6: Result of the inference using the fuzzified values from table Table 4.4, small (S), middle(M), great (G), not possible (NP), possible (P), surpassed (SUR)

Now each linguistic term got per trait three values. These values now need to be unified. The inference calculated us the minimum the worker is capable of corresponding to the rules in the rule base. As co-norm, I choose the max-norm because the values are for sure fulfilled by the worker. As a result, the maximum value can be taken. This norm picks the maximum of the three stored values. For the suitability in the craftsmanship of the linguistic term not possible the calculation would be as follows:

$$SuitCS(NP) = max \begin{cases} SuitCS_0(NP) \\ SuitCS_1(NP) \\ SuitCS_2(NP) \end{cases}$$

The result with the corresponding fuzzy set and delineated linguistic terms can be seen in Figure 4.4.

Before the defuzzification can start there is one point left. The three different sets have to be combined so that we obtain a result fuzzy set which ca be defuzzified. To achieve the goal a union of the three different sets has to be built.

The sets of Figures 4.4 to 4.6 are taken to build the set of Figure 4.7.

Here the intersections between the different linguistic terms have to be considered. In the interval of 0.25 to 0.5, the higher degree of membership is applied. Either the value of the term not possible is applied or the value of the term possible. The same rule has to be assigned to the other interval of 0.5 to 0.75 as well. Furthermore, it has to be regarded that the values stay in the boundaries of the set. It is not possible to assign a degree of membership of 0.8 to the term of not possible at the point of 0.2 because the boundary is at 0.6.

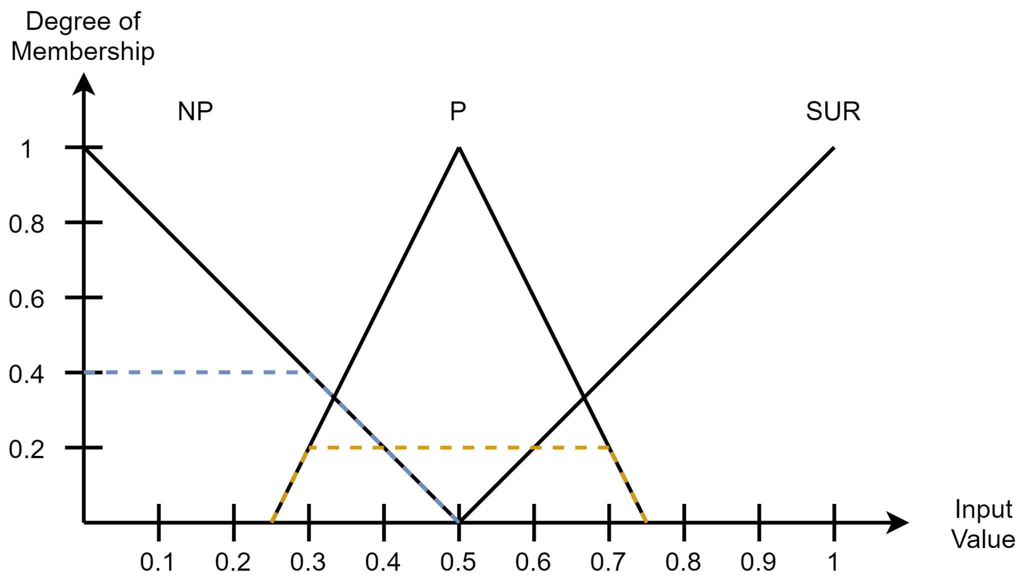


Figure 4.4: Suitability of the worker regarding the craftsmanship

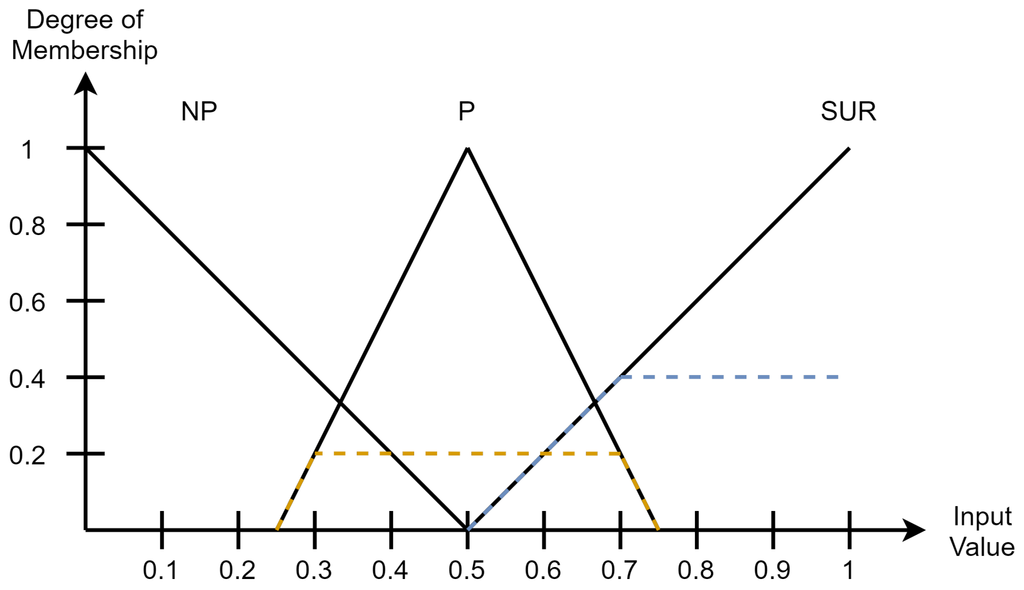


Figure 4.5: Suitability of the worker regarding the technical understanding

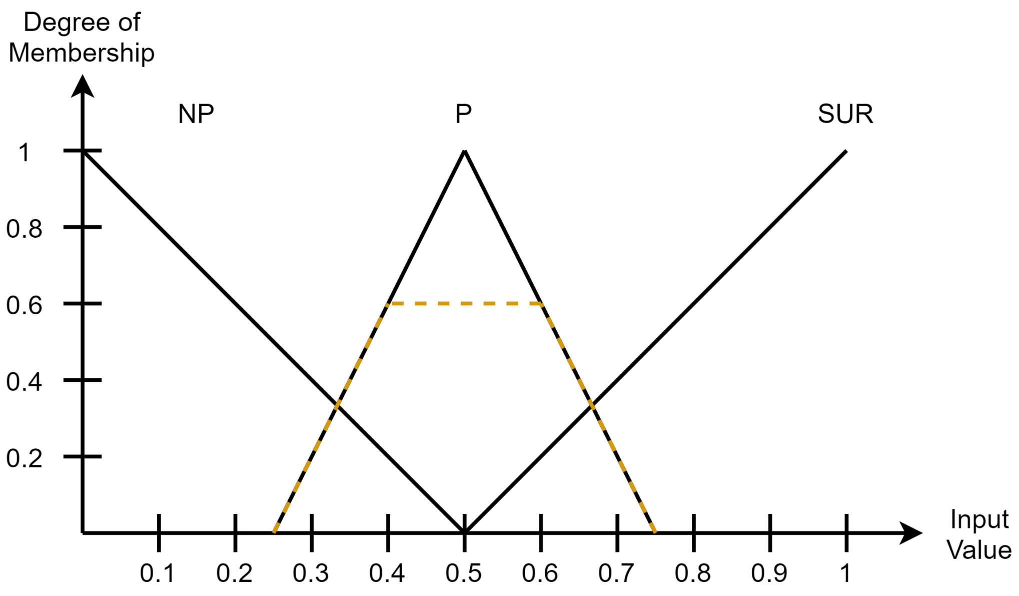


Figure 4.6: Suitability of the worker regarding the soft skills

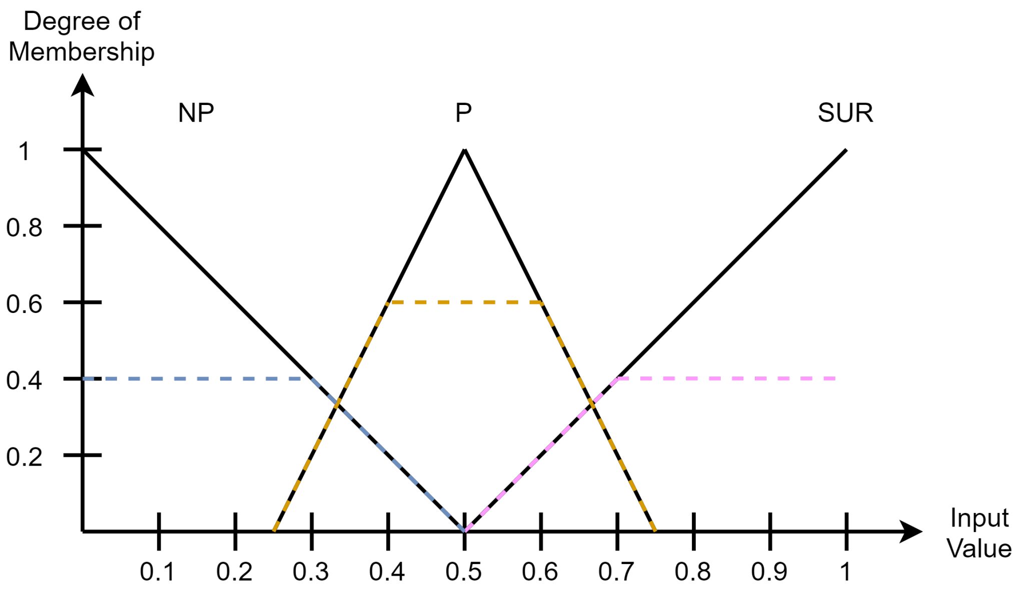


Figure 4.7: Resulting fuzzy set after building the union of all intermediate solutions

The resulting fuzzy set is shown in Figure 4.7 can now be used for defuzzification. The exact solution would be determined by an integration over the fuzzy set. Therefore the curve is split into spatial intervals which are then integrated: $A = \int\limits_{-x}^{+x} \mu(x)\mathrm{d}x$.

This comes with a high computation cost. A trade-off between computation time and the exact result is the result we choose. My solution approximates the solution. Therefore we choose base points in the solution where singletons are created. The more base points we take the better the result is. We took 10 base points for my solution. Every 0.1 step we set a base point. Additionally, the mean is created to minimize the error which comes with the approximation. Corresponding to the in Figure 4.7 the calculation look as follows:

$$
\begin{aligned}
A &= \frac{\int\limits_{0.1}^{1} x \cdot \mu(x)}{\int\limits_{0.1}^{1} \mu(x)} \\
A &= \frac{0.1 \cdot 0.4 + ... + 0,4 \cdot 0,6 + ... + 0.6 \cdot 0.6 + 0.7 \cdot 0.4 + ... + 1 \cdot 0.4}{0.4 \cdot 7 + 0.6 \cdot 3} \\
A &= \frac{0.04 + ... + 0.24 + ... + 0.36 + 0.42 + ... + 0.4}{2.8 + 1.8} \\
A &= \frac{2.64}{4.6} \\
A &= 0.574
\end{aligned}
$$

The term $\mu(x)$ represents the degree of membership of the corresponding linguistic term.

If the number would be increased to infinity we would result in the integral which calculates the exact result.

Constraints

Additionally to the different satisfaction values, two constraints have some influence on the fitness value.

The first constraint is related to the working time of the worker and will be referred to as the "working hour (WH)" constraint. The value given for this constraint represents the maximum allowed working hours. If the threshold of this constraint is exceeded a punishment factor will be added to the fitness value. The value of the factor correlates to the difference between the threshold and the exceeding value. The greater the difference, the greater will be the adjustment by the constraint.

The punishment value for this constraint is the sum of the difference between all exceeding values and the allowed value over all workers described by the following formula:

$$ConstraintWH = \sum_{i=0}^{n} max(w_i - \varphi, 0)$$

where w_i are the working hours of every single worker and φ is the allowed threshold for an employee. The result is then multiplied by the constraint weight.

The second constraint displays the carpool or staff of the company also referred to as the "carpool (CP)" constraint. The size of the carpool is shown by the value of this constraint. A violation of this constraint means that there are not enough cars in the carpool of the company to accomplish all tasks.

The adjustment value for the CP constraint is calculated by the difference of the cars which are used and needed by the worker to accomplish their routes and the cars which are provided by the carpool. In other words, the second constraint is given by the difference of the employees who were assigned by the individual and the estimated number of worker to complete the routes which are given by the user.

The CP constraint is calculated as follows:

$$ConstraintCP = max(I_w - \varphi, 0)$$

where I_w is the number of different employees assigned to the routes and φ the estimated number of workers for the problem.

We assume that both constraint thresholds are set to five. The individual is given in Section 4.2 would lead to the following constraint values. The individual has two workers which exceed the threshold of five regarding the WH constraint. Therefore the differences are calculated and summed. The punishment for this constraint will be two. Since both workers exceeding the threshold by one. This factor is then multiplied by the constraint weight which is given by the user.

The CP constraint will not be triggered because only four workers were assigned to serve the stations. In the end, only the WH constraint would influence the fitness value for the given example.

4.5 Selection Operators

Like in Section 4.3 already mentioned a tournament selection is used for the parental selection. The tournament is of size two. That means that two individuals are selected randomly from the population and compete against each other. There are always two different individuals chosen. The time the first individual is picked it is removed from the pool of possible candidates. This design decision was taken to increase the selection pressure. A result of this decision is that the individual with the worst fitness value of the generation has no chance to get into the reproduction, crossover, and mutation process as the conclusion of the tournament because the individual with the best fitness wins it. Resulting in the inevitable loss of the worst individual in all cases. The environment selection is a μ+Lambda-Selection. Here the population consists of 4 different groups of individuals which can be seen in Figure 4.2. Three groups are coming from the foregoing evaluation. In the first group are the individuals who were only mutated. The second branch consists of the individuals which were created by crossover but were not mutated. Crossover and mutation were applied to the next group. Last but not least the original population, which was used for parental selection, is added to the process. The selection operator takes the best individuals out of this population until it reached the original problem size. The resulting population is the basis of the next generation. The μ+Lambda-Selection provides a form of elitism. The algorithm thereby can only improve the candidates of solutions. If a step develops only worse solutions, which can be created by mutations and bad crossovers, they are contained and eliminated by the individuals of the start of the generation. The idea was to maintain the quality of the fitness values over the different solutions because it is more likely that a better individual is created by two individuals who have already "good" fitness values.

4.6 Crossover Operators

After the individuals for the recombination were chosen by the parental selection. Genetic diversion is created by three different operators. A one-point crossover is the first crossover operator. Here the cutting is chosen randomly and the right-hand side is then changed between both individuals. Another proportion of the new population is created by a two-point crossover. The operator does not have fixed cutting points. The cutting points are changing after each application. The two-point crossover exchanges the inner part between the individuals. The points for the cuts are both chosen randomly that the exchanged part can reach from one single allele to the whole chromosome except the first and last allele. The last group of new individuals is determined by the third operator which is a uniform crossover. The pattern of each crossover is created randomly after each application so that at each application a different scheme is used. The randomness in the setup of the different operator should make it possible that the collection of alleles which contribute to "good" solutions can be transferred between the individuals and therefore improve the overall fitness of the individuals and thereby of the population. These collections can vary in their lengths which makes it worthwhile that the operators are highly flexible. If the operators would be fixed the algorithm would converge as well but would split matching alleles which in turn results in a long time of convergence. With the chosen approach a reduction of these unintended splits is intended.

4.7 Mutation Operators

The algorithm uses two different mutation operators. In this step, small changes are applied to the genes of the chosen individuals. The algorithm uses two different mutation operators. The individuals who have to be mutated are therefore split into two groups. For the first group, the mutation operator will change a single index of the chromosome. This is done by adding an arbitrary offset to the current index which results in a new allocation of a worker to this station. The other mutation operator changes two workers of the individual with each other. Resulting in the change of the assignment of the workers to the two stations concluding a different satisfaction value.

5 Evaluation

This chapter will present the results of this thesis. First, the parameter setting of the evaluation benchmarks will be explained. Then the structure of the benchmark instances will be described. An outline of the results of a single run will be given as well. From there on the results will be presented and interpreted. At the end of the chapter, the results will be summarized to a conclusion.

5.1 Parameter Setting

The parameter setting will be examined in this section. We differentiate between global and local parameters. The global parameters will stay the same for all benchmark problems. The parameter which change between the different benchmark instances are the local parameters.

Parameter	Value
Tournament Size k	2
Size of Population, Mutation only	0.1
One-Point Crossover Size	0.33
Two-Point Crossover Size	0.33
Uniform Crossover Size	0.34
Mutation Rate	0.1
Population Size	200
Iterations	500

Table 5.1: Global variables of the algorithm

The structure of the algorithm was shown in Section 4.3. The used parameters for the different components are shown in Table 5.1. The tournament size is set to a fixed size of two to invoke a small selection pressure by the preservation of

Parameter	Value			
	Problem1	Problem2	Problem3	Problem4
ConstraintWH	5	12	20	6
ConstraintPC	5	10	30	12
Number of Stations	16	100	500	50
Number of Workers	10	25	75	20
Punishment Weights				
WeightWH	0.5	1	5	10
WeightPC	0.5	5	1	10

Table 5.2: Local variables of the algorithm

the diversity of the population. That allows the algorithm in the crossover part to exchange good sub-structures even the individual is worse than the other. This design decision should prevent that the algorithm gets stuck in a local optimum. Further diversity is added by the three crossover operators which are applied equally to the population. The different operators allow different patterns to be exchanged. The encodings of "good" individuals show similarities when they are compared with each other. Those structures within the encoding of the "good" individuals have a huge impact on the fitness value. If a crossover can exchange these coherent genes the individual will receive a boost which is represented in the fitness value. The established structures in the encoding represent partial solutions to the problem. These patterns can be very diverse. Therefore it is beneficial to have different crossover operators[12, 10]. The mutation rate and the size of the subset of the population which is only mutated are set to 0.1 [18]. The population size is set to 200 and the evaluation runs over 500 iterations. These values were chosen corresponding to the size of the problems shown in Table 5.2. Each problem has it is own threshold values for the constraints. The local variables change between different problems. Four problems were evaluated. Each problem instance was then evaluated with the different punishment weights. In the end, we have 16 different parameter setups. These setups were run for both approaches, the crisp and fuzzy approach.

5.2 Explaination of Benchmark

A benchmark instance evaluates the different problems with 31 runs each. The punishment weights which are set for the instance does not change for the different problems. Table 5.2 shows that four different problems have to be evaluated. This evaluation is done for a crisp and fuzzy approach. In total each benchmark instance has eight problems. These problems are evaluated 31 times each.

Station	HG	TV	SoSk
0	19	90	15
1	11	60	42
2	5	78	74
3	74	36	55
4	0	30	54
5	7	79	95
6	73	0	84
7	4	48	37
8	79	0	60
9	98	49	55
10	2	15	61
11	95	14	74
12	45	3	33
13	0	16	32
14	75	1	75
15	7	24	80

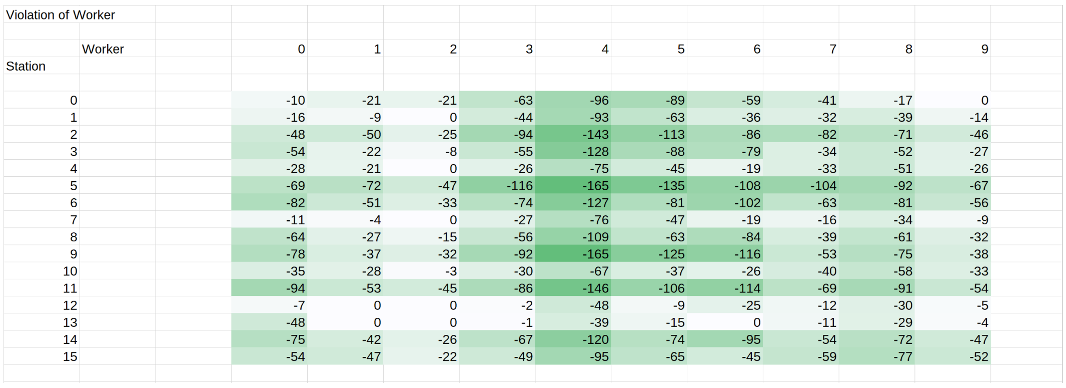
Worker	HG	TV	SoSk
0	49	80	26
1	83	69	33
2	66	69	58
3	52	27	31
4	28	7	2
5	38	1	38
6	20	31	35
7	79	49	21
8	75	85	3
9	87	95	28

Table 5.3: Requirements and Capabilities of the instance C16W10

Each run the population will be initialized randomly. The data for the problems are provided by CSV-tables. These tables characterize the requirements/ skill levels of the different stations and workers. Each line represents the values for craftsmanship, technical understanding, and soft skills. These values are separated by a delimiter. The data for the stations and workers are saved in different tables. Examples can be seen in Table 5.3. The values are in the range between zero and 100. The data were created randomly by a random number generator. The values are then normalized to the range between zero and one to simplify the evaluation later.

The random actions in the algorithm are derived from a random object. Each problem has it is own random object. This object is initialized with a seed given by a unique global random object. The seed of the global random was over the whole evaluation the same.

If we calculate the satisfaction as described in Section 4.4 we can represent the different satisfaction values by a matrix. The matrix is shown in Figure 5.1 is an example of the problem where 16 stations need to be served and the staff consists of ten workers. The color spectrum of the matrix goes from white to green. White values indicate a small violation of the requirements whereas green values represent a huge requirement violation. Therefore values that are highlighted in white should be preferred, values in green should be avoided. With this matrix, it is possible to show how good a worker performs if he is assigned to the task at the station. The third worker with index two is the best worker of the staff which is displayed by the color range of the third column which is in the worst case light green. Whereas column five should be strictly avoided because of the vibrant green color spectrum which is occasionally broken by a few slight green values.



Violation of Worker

Station	Worker 0	1	2	3	4	5	6	7	8	9
0	-10	-21	-21	-63	-96	-89	-59	-41	-17	0
1	-16	-9	0	-44	-93	-63	-36	-32	-39	-14
2	-48	-50	-25	-94	-143	-113	-86	-82	-71	-46
3	-54	-22	-8	-55	-128	-88	-79	-34	-52	-27
4	-28	-21	0	-26	-75	-45	-19	-33	-51	-26
5	-69	-72	-47	-116	-165	-135	-108	-104	-92	-67
6	-82	-51	-33	-74	-127	-81	-102	-63	-81	-56
7	-11	-4	0	-27	-76	-47	-19	-16	-34	-9
8	-64	-27	-15	-56	-109	-63	-84	-39	-61	-32
9	-78	-37	-32	-92	-165	-125	-116	-53	-75	-38
10	-35	-28	-3	-30	-67	-37	-26	-40	-58	-33
11	-94	-53	-45	-86	-146	-106	-114	-69	-91	-54
12	-7	0	0	-2	-48	-9	-25	-12	-30	-5
13	-48	0	0	-1	-39	-15	0	-11	-29	-4
14	-75	-42	-26	-67	-120	-74	-95	-54	-72	-47
15	-54	-47	-22	-49	-95	-65	-45	-59	-77	-52

Figure 5.1: Resulting matrix of the problem instance C16W10

This matrix is a good representation of what happens internally when the algorithm optimizes the problem. The algorithm tries to find an allocation of stations and workers so that only whitish values of the matrix are used and the constraints are fulfilled as well.

5.3 Results

The results of a benchmark instance are summarized in a table. The results of the instance where both punishment weights were set to 0.5, 16 stations had to be served and the staff consisted of 10 employees can be seen in Table 8.1.

Benchmark	Problem	Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C16-W10				0.5	0.5		
	BruteForce	15.147	8.051			6	15.147
	Crisp_No_Constraints	15.147	8.04	5	0	6	15.147
	Crisp_Constraint_WH	14.793	8.043	0	0	3	14.793
	Crisp_Constraint_CP	15.147	8.04	5	0	6	15.147
	Crisp_Both	14.78	8.04	0	0	4	14.78
	Fuzzy_No_Constraint	12.427	8.156	0.5	1	0	8.156
	Fuzzy_Constraint_WH	12.853	8.156	0	1.5	0	8.156
	Fuzzy_Constraint_CP	12.013	8.156	1.5	0	0	8.156
	Fuzzy_Both	12.697	8.156	0	0	1	8.156

Table 5.4: Results of the benchmark instance with 16 stations and 10 worker with both punishment weights set to 0.5

The first row of the table contains general information on the benchmark instance. In the benchmark column, the problem which was evaluated is indicated by an abbreviation of the problem. The other values in this row are values that let us estimate how good the solutions are that we obtained.

First, there is the crisp satisfaction value which is obtained by a heuristic. This method assigns to each station the best worker possible corresponding to the task. If we recall Figure 5.1 with the matrix related to that problem. The method searches in each row the data field with the highest value. This list of numbers is the best allocation of workers to tasks possible without regarding the constraints. This value will help us to interpret the results of the evolutionary algorithm correctly.

An estimation for the fuzzy solution is given by the fuzzy satisfaction. The method which obtains this value minimizes the difference between the sum of all properties of the worker and the sum of all properties of the station. In the end, the resulting value corresponds to an allocation of workers to the tasks with the minimal amount of "over-fitting" or "under-fitting".

The next two values are the punishment weights which influence the calculation of the fitness if the constraints are not fulfilled.

The number given for the completed task outlines how many tasks can be completely fulfilled. That means that each requirement of the task is met with an equal or higher skill level in that area by the worker. This value allows us later on to how applicable the used approach is.

Last but not least the fitness column shows the different fitness values used by the evolutionary algorithm.

All other values in the table are medians of the given variables over 31 runs.

5.4 Evaluation

The first problem instance (Crisp-No-Constraints) is used to verify if the algorithm can find the optimal solution. Therefore we compare the resulting crisp satisfaction with the resulting value of the benchmark method. All resulting tables can be find in Chapter 8

In Tables 8.1, 8.5, 8.9 and 8.13 we can see that with an increase of complexity the distance towards the optimal solution increases. Also, the number of completed tasks is decreasing. The algorithm can find the optimal solution for the smaller problem instances (C16-W10, C50-W20). The number of function evaluations for the other problems is insufficient to obtain the optimal solution. This can be seen in the discrepancy of the values in the problem C500-W75. The discrepancy shows as well that the algorithm has not converged yet. The huge gap between the possible number of completed tasks and the reached one is a result of missing influence. The fitness value which is used for the optimization does not consider the number of completed tasks. In section Section 4.4 the composition of the fitness value was visualized. It shows that the fitness value solely depends on the crisp and fuzzy satisfaction as well as both constraints values.

The constraints influence the number of completed tasks hence the algorithm wants to avoid a violation of the constraints. This leads to other individuals which are conform to the constraints. Resulting in a lower fitness value because the quality of the solution was traded for the fulfillment of the constraints. No adjustment would let to an even lower fitness value because the loss due to the constraints would be greater than the loss of the trade-off.

Furthermore, the results show that the fuzzy approach has an even lower number of completed tasks. The fuzzification of the values leads to a greater interval of solutions which were not possible with the crisp values. The aggregation of the different fuzzy sets to a single one and the subsequent defuzzification have also their influence on the result. This process allows the algorithm to compensate bad allocations with good allocations which leads to a smaller number of completed tasks. Additionally, with an increase in the problem complexity and therefore in the search space, we can determine a decrease of the gap of the number of completed tasks between the crisp and fuzzy problem formulations relative to the problem complexity. At the problem C50-W20 the fuzzy approach could only fulfill a fifth of the tasks completed in comparison to the crisp solution whereas the fulfillment increased to almost two-thirds of the value of the crisp solution in the problem C500-W75.

The solutions to the different problems differ from each other which is proof of the adaptation of the algorithm to the different compositions of the fitness value. The adaptation is shown by the influence of the constraint values to the fitness values.

Tables 8.9 to 8.12 showcase the results of the problem instance C100-W25 corresponding to the different constraint setups. The results differ marginally hence the initialization of the global random variable was overall evaluations the same. The slight changes in the result can be attributed to changes in the constraint weights which influenced the decisions made by the algorithm. A change of the weights only affected the evaluation of the crisp values. The values for the fuzzy satisfaction are unchanged over all the different problems. The CrispConstraintCP and CrispBoth instance shows the marginal distinctions of the fitness values corresponding to the change of the constraints weights.

Let us take a detailed look at the problem where the first constraint factor was set to one and second to five. The results where the constraints were not part of the fitness values showcase that there a few workers who can be assigned to all tasks. It is represented by the high value for the first constraint. The value meant that for the resulting allocation the assigned employees took 18 hours overtime in total. Further five additional employees are needed for the allocation. We now look at the development of these constraint values over the different problems by iterating through the rows. The next instance accounts for violations of the first constraints. If the first constraint would be disregarded the fitness would decrease relative to the extent of the violation. The algorithm adapts the solution using an additional worker for the allocation which increases the value to 30 in comparison to the problem earlier. With the assignment of another employee, the algorithm avoided the negative influence of the first constraint which is now zero on the fitness value completely. We go on to the problem where the second constraint has an impact on the fitness value. In comparison to the first results, we can see that the algorithm avoided the majority of the negative consequences by reducing the value to 5. The last problem instance that we analyze in more detail calculates the fitness value regarding both constraints. So both constraint values influence the fitness value. Here the algorithm has to find a trade-off between the constraints. Although the first constraint has a smaller influence on the calculation of the fitness the algorithm reduced the violation towards zero. Consequently exceeding the threshold of the second constraint by assigning two additional workers.

Concluding we want to compare the results of the crisp approach with the ones of the fuzzy approach. Overall it can be stated that the crisp approach fulfills more tasks completely than the fuzzy approach. Also, the fuzzy satisfaction values are almost the same even when the algorithm was not optimizing on this value. This is a result of the defuzzification of the resulting fuzzy variable. The fuzzy variable is the union of the three fuzzy variables which are related to the different skill areas. This combination allowed that allocations that were "over-fitted" were compensated by worse ones. Therefore the values are around the optimum. In future works, the defuzzification method should be reworked that an approach to compensate "over-fitted" allocations with "under-fitted" ones results in negative consequences for the fitness value.

5.5 Summarization

In the end, we can summarize that the prototype can find an allocation of workers to tasks which is near the optimal solution. Furthermore, the prototype can adapt to different constraint setups. The fuzzy approach used by the algorithm remained behind its potential because of design decisions in the modeling which influenced the results negatively. Nonetheless, we are convinced that with modifications the fuzzy approach is competitive.

6 Conclusion & Future Works

In this thesis, evolutionary algorithms and fuzzy techniques were used to solve an allocation based on the VRP. One use case is the planning of tours in a company that provides maintenance services for different industrial machines. The related work was examined in Chapter 3. In Chapter 4, the concept of a framework was introduced which can solve a variety of allocation problems in different approaches. The different components are explained in detail. A prototype framework was provided which solves allocation problems based on the VRP for a variety of problem sizes. Although the crisp approach of the prototype solves the problems well the fuzzy concept left room for improvements.

Future Works

The fuzzy approach can be improved by modifying the defuzzification. Here a weighting method can be introduced which prevents the compensation. Another point is the used t-norm and co-norms which can be changed with different measurements for the minimum and maximum.

The VRP which the prototype is based on can also be easily extended. The thesis assumed that the distance values were already encoded in the requirements. The detachment of the different objectives would transform the problem in a multi-objective optimization here the allocation of the worker to the tasks has to be optimized as well as the distances of the different tours.

7 Acknowledgements

I would first like to thank my thesis supervisor Prof. Dr.-Ing. habil. Sanaz Mostaghim, for accepting my proposed topic. Further, I want to thank her for the continuous encouragement and interest in the progress which allowed this paper to be my work.

I would also like to acknowledge Dr.-Ing. Heiner Zille and Dr.-Ing. Christoph Steup as my advisors for this thesis. I am thankful and indebted to them for their patience and sharing their knowledge as well as providing guidance through the process of writing this thesis. At this point, I wish the best for your young family Heiner.

Another acknowledge goes to the employees of the company of Silver Seed Games UG. The topic of this thesis is the continuation of my task and ideas during the internship. Especially I want to thank Friedrich Lüder for his constant helpfulness.

I also want to mention my companions who have accompanied me all the way here. They also provided important inspiration to solve the task as well as the energy to recover from hard times.

Finally, I must express my gratitude to my family who got my back all the time. Throughout my years of study, I did not have to fear anything. This accomplishment would not have been possible without them. Thank you.

8 Appendix

Benchmark	Problem	Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C16-W10				0.5	0.5		
	BruteForce	15.147	8.051			6	15.147
	Crisp_No_Constraints	15.147	8.04	5	0	6	15.147
	Crisp_Constraint_WH	14.793	8.043	0	0	3	14.793
	Crisp_Constraint_CP	15.147	8.04	5	0	6	15.147
	Crisp_Both	14.78	8.04	0	0	4	14.78
	Fuzzy_No_Constraint	12.427	8.156	0.5	1	0	8.156
	Fuzzy_Constraint_WH	12.853	8.156	0	1.5	0	8.156
	Fuzzy_Constraint_CP	12.013	8.156	1.5	0	0	8.156
	Fuzzy_Both	12.697	8.156	0	0	1	8.156

Table 8.1: Results of the benchmark instance with 16 stations and 10 workers with both punishment weights set to 0.5

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C16-W10	BruteForce		15.147	8.051	1	5	6	15.147
	Crisp_No_Constraints		15.147	8.04	10	0	6	15.147
	Crisp_Constraint_WH		14.79	8.043	0	0	4	14.79
	Crisp_Constraint_CP		15.147	8.04	10	0	6	15.147
	Crisp_Both		14.783	8.044	0	0	3	14.783
	Fuzzy_No_Constraint		12.427	8.156	1	10	0	8.156
	Fuzzy_Constraint_WH		12.853	8.156	0	15	0	8.156
	Fuzzy_Constraint_CP		12.013	8.156	3	0	0	8.156
	Fuzzy_Both		12.65	8.156	0	0	0	8.156

Table 8.2: Results of the benchmark instance with 16 stations and 10 workers the punishment weights are set to one for the WH constraint and five for the WH constraint

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C16-W10	BruteForce		15.147	8.051	5	1	6	15.147
	Crisp_No_Constraints		15.147	8.04	50	0	6	15.147
	Crisp_Constraint_WH		14.79	8.043	0	0	4	14.79
	Crisp_Constraint_CP		15.147	8.04	50	0	6	15.147
	Crisp_Both		14.79	8.04	0	0	4	14.79
	Fuzzy_No_Constraint		12.427	8.156	5	2	0	8.156
	Fuzzy_Constraint_WH		12.853	8.156	0	3	0	8.156
	Fuzzy_Constraint_CP		12.013	8.156	15	0	0	8.156
	Fuzzy_Both		12.48	8.156	0	0	0	8.156

Table 8.3: Results of the benchmark instance with 16 stations and 10 workers the punishment weights are set to five for the WH constraint and one for the CP constraint

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C16-W10	BruteForce		15.147	8.051	10	10	6	15.147
	Crisp_No_Constraints		15.147	8.04	100	0	6	15.147
	Crisp_Constraint_WH		14.79	8.043	0	0	4	14.79
	Crisp_Constraint_CP		15.147	8.04	100	0	6	15.147
	Crisp_Both		14.79	8.04	0	0	4	14.79
	Fuzzy_No_Constraint		12.427	8.156	10	20	0	8.156
	Fuzzy_Constraint_WH		12.853	8.156	0	30	0	8.156
	Fuzzy_Constraint_CP		12.013	8.156	30	0	0	8.156
	Fuzzy_Both		12.697	8.156	0	0	1	8.156

Table 8.4: Results of the benchmark instance with 16 stations and 10 workers with both punishment weights set to 10

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C50-W20	BruteForce		48.573	25.015	0.5	0.5	32	48.573
	Crisp$_{N}o_{C}onstraints$		48.573	25.006	7.5	0	32	48.573
	Crisp$_{C}onstraint_{W}H$		48.207	25.006	0	0	25	48.207
	Crisp$_{C}onstraint_{C}P$		48.573	25.006	7.5	0	32	48.573
	Crisp$_{B}oth$		48.153	25.006	0	0	25	48.153
	Fuzzy$_{N}o_{C}onstraint$		40.99	25.033	0	3	5	25.033
	Fuzzy$_{C}onstraint_{W}H$		40.72	25.033	0	3	5	25.033
	Fuzzy$_{C}onstraint_{C}P$		40.833	25.033	1	0	5	25.033
	Fuzzy$_{B}oth$		40.577	25.033	0	0	5	25.033

Table 8.5: Results of the benchmark instance with 50 stations and 20 workers with both punishment weights set to 0.5

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C50-W20	BruteForce		48.573	25.015	1	5	32	48.573
	Crisp$_{N}o_{C}onstraints$		48.573	25.006	15	0	32	48.573
	Crisp$_{C}onstraint_{W}H$		48.167	25.006	0	0	25	48.167
	Crisp$_{C}onstraint_{C}P$		48.553	25.006	12	0	31	48.553
	Crisp$_{B}oth$		48.11	25.006	0	0	24	48.11
	Fuzzy$_{N}o_{C}onstraint$		40.99	25.033	0	30	5	25.033
	Fuzzy$_{C}onstraint_{W}H$		40.72	25.033	0	30	5	25.033
	Fuzzy$_{C}onstraint_{C}P$		40.833	25.033	2	0	5	25.033
	Fuzzy$_{B}oth$		41.103	25.033	0	0	5	25.033

Table 8.6: Results of the benchmark instance with 50 stations and 20 workers the punishment weights are set to one for the WH constraint and five for the CP constraint

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C50-W20	BruteForce		48.573	25.015	5	1	32	48.573
	Crisp$_{N}o_{C}onstraints$		48.573	25.006	75	0	32	48.573
	Crisp$_{C}onstraint_{W}H$		48.167	25.006	0	0	25	48.167
	Crisp$_{C}onstraint_{C}P$		48.563	25.006	75	0	31	48.563
	Crisp$_{B}oth$		48.16	25.006	0	0	25	48.16
	Fuzzy$_{N}o_{C}onstraint$		40.99	25.033	0	6	5	25.033
	Fuzzy$_{C}onstraint_{W}H$		40.72	25.033	0	6	5	25.033
	Fuzzy$_{C}onstraint_{C}P$		40.833	25.033	10	0	5	25.033
	Fuzzy$_{B}oth$		40.367	25.033	0	0	5	25.033

Table 8.7: Results of the benchmark instance with 50 stations and 20 workers the punishment weights are set to five for the WH constraint and one for the CP constraint

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C50-W20	BruteForce		48.573	25.015	10	10	32	48.573
	Crisp$_{N}o_{C}onstraints$		48.573	25.006	150	0	32	48.573
	Crisp$_{C}onstraint_{W}H$		48.167	25.006	0	0	25	48.167
	Crisp$_{C}onstraint_{C}P$		48.553	25.006	120	0	31	48.553
	Crisp$_{B}oth$		48.117	25.006	0	0	24	48.117
	Fuzzy$_{N}o_{C}onstraint$		40.99	25.033	0	60	5	25.033
	Fuzzy$_{C}onstraint_{W}H$		40.72	25.033	0	60	5	25.033
	Fuzzy$_{C}onstraint_{C}P$		40.833	25.033	20	0	5	25.033
	Fuzzy$_{B}oth$		40.577	25.033	0	0	5	25.033

Table 8.8: Results of the benchmark instance with 50 stations and 20 workers with both punishment weights set to 10

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C100-W25	BruteForce		98.96	50.052	0.5	0.5	70	98.96
	Crisp_No_Constraints		97.88	50.03	9	2.5	46	97.88
	Crisp_Constraint_WH		97.07	50.03	0	3	44	97.07
	Crisp_Constraint_CP		97.287	50.03	7.5	0.5	43	96.55
	Crisp_Both		96.547	50.03	0	1	41	95.443
	Fuzzy_No_Constraint		82.5	50.087	0	7	13	50.087
	Fuzzy_Constraint_WH		82.297	50.087	0	6.5	13	50.087
	Fuzzy_Constraint_CP		82.487	50.087	0	3	13	47.079
	Fuzzy_Both		82.73	50.087	0	3	15	47.079

Table 8.9: Results of the benchmark instance with 100 stations and 25 workers with both punishment weights set to 0.5

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C100-W25	BruteForce		98.96	50.052	1	5	70	98.96
	Crisp_No_Constraints		97.88	50.03	18	25	46	97.88
	Crisp_Constraint_WH		97.067	50.03	0	30	43	97.067
	Crisp_Constraint_CP		96.853	50.03	15	5	40	91.317
	Crisp_Both		95.927	50.03	0	10	37	86.003
	Fuzzy_No_Constraint		82.5	50.087	0	70	13	50.087
	Fuzzy_Constraint_WH		82.297	50.087	0	65	13	50.087
	Fuzzy_Constraint_CP		82.487	50.087	0	30	13	20.079
	Fuzzy_Both		82.73	50.087	0	30	15	20.079

Table 8.10: Results of the benchmark instance with 100 stations and 25 workers the punishment weights are set to one for the WH constraint and five for the CP constraint

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C100-W25	BruteForce		98.96	50.052	5	1	70	98.96
	Crisp_No_Constraints		97.88	50.03	90	5	46	97.88
	Crisp_Constraint_WH		97.08	50.03	0	6	44	97.08
	Crisp_Constraint_CP		97.177	50.03	75	1	41	95.947
	Crisp_Both		96.317	50.03	0	2	39	94.227
	Fuzzy_No_Constraint		82.5	50.087	0	14	13	50.087
	Fuzzy_Constraint_WH		82.297	50.087	0	13	13	500.887
	Fuzzy_Constraint_CP		82.487	50.087	0	6	13	44.079
	Fuzzy_Both		82.73	50.087	0	6	15	44.079

Table 8.11: Results of the benchmark instance with 100 stations and 25 workers the punishment weights are set to five for the WH constraint and one for the CP constraint

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C100-W25	BruteForce		98.96	50.052	10	10	70	98.96
	Crisp_No_Constraints		97.88	50.03	180	50	46	97.88
	Crisp_Constraint_WH		97.08	50.03	0	60	44	97.08
	Crisp_Constraint_CP		96.78	50.03	150	10	39	86.317
	Crisp_Both		95.943	50.03	0	20	37	75.943
	Fuzzy_No_Constraint		82.5	50.087	0	140	13	50.087
	Fuzzy_Constraint_WH		82.297	50.087	0	130	13	50.087
	Fuzzy_Constraint_CP		82.487	50.087	0	60	13	-9.921
	Fuzzy_Both		82.73	50.087	0	60	15	-9.921

Table 8.12: Results of the benchmark instance with 100 stations and 25 workers with both punishment weights set to 10

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C500-W75	BruteForce		498.893	250.226	0.5	0.5	438	498.893
	Crisp_No_Constraints		441.193	250.074	0	22	76	441.193
	Crisp_Constraint_WH		440.97	250.059	0	22	81	440.97
	Crisp_Constraint_CP		440.253	250.057	0	20.5	78	419.677
	Crisp_Both		439.833	250.065	0	20	78	420.153
	Fuzzy_No_Constraint		404.98	250.6	0	21	48	250.6
	Fuzzy_Constraint_WH		405.687	250.616	0	21	48	250.616
	Fuzzy_Constraint_CP		406.303	250.54	0	20	48	230.681
	Fuzzy_Both		405.337	250.594	0	20	49	230.597

Table 8.13: Results of the benchmark instance with 500 stations and 75 workers with both punishment weights set to 0.5

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C500-W75	BruteForce		498.893	250.226	1	5	438	498.893
	Crisp_No_Constraints		441.193	250.074	0	220	76	441.193
	Crisp_Constraint_WH		440.97	250.059	0	220	81	440.97
	Crisp_Constraint_CP		437.87	250.067	0	200	75	237.633
	Crisp_Both		437.493	250.063	0	200	74	238.177
	Fuzzy_No_Constraint		404.98	250.6	0	210	48	250.6
	Fuzzy_Constraint_WH		405.687	250.616	0	210	48	250.616
	Fuzzy_Constraint_CP		406.303	250.54	0	200	48	50.681
	Fuzzy_Both		405.337	250.594	0	200	49	50.597

Table 8.14: Results of the benchmark instance with 500 stations and 75 workers the punishment weights are set to one for the WH constraint and five for the CP constraint

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C500-W75	BruteForce		498.893	250.226	5	1	438	498.893
	Crisp$_{N}o_{C}onstraints$		441.193	250.074	0	44	76	441.193
	Crisp$_{C}onstraint_{W}H$		440.97	250.059	0	44	81	440.97
	Crisp$_{C}onstraint_{C}P$		439.45	250.061	0	41	79	399.39
	Crisp$_{B}oth$		439.34	250.057	0	40	77	399.383
	Fuzzy$_{N}o_{C}onstraint$		404.98	250.6	0	42	48	250.6
	Fuzzy$_{C}onstraint_{W}H$		405.687	250.616	0	42	48	250.616
	Fuzzy$_{C}onstraint_{C}P$		406.303	250.54	0	40	48	210.681
	Fuzzy$_{B}oth$		405.337	250.594	0	40	49	210.597

Table 8.15: Results of the benchmark instance with 500 stations and 75 workers the punishment weights are set to five for the WH constraint and one for the CP constraint

Benchmark	Problem		Crisp Satisfaction	Fuzzy Satisfaction	ConstraintWH	ConstraintCP	Completed Tasks	Fitness
C500-W75	BruteForce		498.893	250.226	10	10	438	498.893
	Crisp$_{N}o_{C}onstraints$		441.193	250.074	0	440	76	441.193
	Crisp$_{C}onstraint_{W}H$		440.97	250.059	0	440	81	440.97
	Crisp$_{C}onstraint_{C}P$		437.11	250.062	0	400	73	37.153
	Crisp$_{B}oth$		437.197	250.064	0	400	74	37.493
	Fuzzy$_{N}o_{C}onstraint$		404.98	250.6	0	420	48	250.6
	Fuzzy$_{C}onstraint_{W}H$		405.687	250.616	0	420	48	250.616
	Fuzzy$_{C}onstraint_{C}P$		406.303	250.54	0	400	48	-149.319
	Fuzzy$_{B}oth$		405.337	250.594	0	400	49	-149.403

Table 8.16: Results of the benchmark instance with 500 stations and 75 workers with both punishment weights set to 10

Bibliography

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[2] B. M. Baker and M. A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, Apr. 2003.

[3] R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, Apr. 2012.

[4] J. Bezdek. Fuzzy models—what are they, and why? [editorial]. *Fuzzy Systems, IEEE Transactions on*, 1:1 – 6, 03 1993.

[5] J. Brito, F. J. Martínez, J. A. Moreno, and J. L. Verdegay. Fuzzy approach for Vehicle Routing Problems with fuzzy travel time. In *International Conference on Fuzzy Systems*, pages 1–8, July 2010.

[6] J. Brito, J. A. Moreno-Pérez, and J. L. Verdegay. Fuzzy Optimization in Vehicle Routing Problems. In *IFSA/EUSFLAT Conf.*, 2009.

[7] E. Cao and M. Lai. The open vehicle routing problem with fuzzy demands. *Expert Systems with Applications*, 37(3):2405–2411, Mar. 2010.

[8] G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, Oct. 1959.

[9] N. A. El-Sherbeny. Imprecision and flexible constraints in fuzzy vehicle routing problem. *American Journal of Mathematical and Management Sciences*, 31(1-2):55–71, 2011.

[10] F. Herrera, M. Lozano, and A. Sánchez. Hybrid crossover operators for real-coded genetic algorithms: an experimental study. *Soft Computing*, 9(4):280–298, Apr. 2005.

[11] J. H. Holland and Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* MIT Press, 1992.

[12] Hyun-Sook Yoon and Byung-Ro Moon. An empirical study on the synergy of multiple crossover operators. *IEEE Transactions on Evolutionary Computation*, 6(2):212–223, 2002.

[13] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher. *Computational Intelligence: A Methodological Introduction.* Texts in Computer Science. Springer-Verlag, London, 2 edition, 2016.

[14] G. Laporte and Y. Nobert. Exact Algorithms for the Vehicle Routing Problem. In S. Martello, G. Laporte, M. Minoux, and C. Ribeiro, editors, *North-Holland Mathematics Studies*, volume 132 of *Surveys in Combinatorial Optimization*, pages 147–184. North-Holland, Jan. 1987.

[15] E. H. Mamdani and S. Assilian. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. In D. Dubois, H. Prade, and R. R. Yager, editors, *Readings in Fuzzy Sets for Intelligent Systems*, pages 283–289. Morgan Kaufmann, Jan. 1993.

[16] S. Mostaghim. Evolutionäre Algorithmen Kapitel 3: Kodierung, Fitness, Selektion. 2017.

[17] S. Mostaghim. Intelligente Systeme Kapitel 6: Fuzzy-Systeme (FS). 2017.

[18] V. P. Patil and D. D. Pawar. The optimal crossover or mutation rates in Genetic algorithm : A Review. *International Journal of Applied Engineering and Technology*, 5:38–41, 2015.

[19] M. Srinivas and L. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667, Apr. 1994.

[20] P. Toth and D. Vigo, editors. *The vehicle routing problem.* SIAM monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics, Philadelphia, 2002.

[21] L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.

Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Lars Wagner Magdeburg, 30.06.2020