Technische Universität München
Department of Civil, Geo and Environmental Engineering
Chair of Cartography
Prof. Dr.-Ing. Liqiu Meng

A Decision Support System for Sales Territory Planning using the Genetic Algorithm

Shahin Sharifi Noorian

Master's Thesis

Submission Date: 16.07.2015

Study Course: Cartography M.Sc.

Supervisor: Dr.-Ing. Christian Murphy

Cooperation: WIGeoGIS GmbH

2015

Acknowledgments

After six months of work at WIGeoGIS, I have finally finished my master's thesis. I would not have finished this project successfully without the help of a number of people. First of all, I would like to thank Dr. Christian Murphy for the unceasing encouragement, support and attention. I am also grateful to Dipl.Geog Michael Steigemann for all his help and support, he was always present to help out whenever I needed it. I want to thank the board of WIGeoGIS for giving me this opportunity to write my thesis in my field of interest. I would like to apologize for not having enough time to spend with my family and my wife Sepideh during this six months and thank them for their understanding. Finally, I would like to thank Prof. Dr.-Ing. Liqiu Meng for accepting to be the examiner of my thesis.

Abstract

Territory Planning is the problem of merging small geographical areas (e.g. municipality, zip code or county) into larger geographical areas (Territories) according to some predefined criteria. Sales territory planning has become one of the most important issues in many selling organizations. The target of sales territory planning is the optimal division of the market among sales team in order to better exploit the potentials of the market and thereby to maximize the total profit. With the aid of territory planning, realistic and fair sales areas can be specified based on the distribution of market potentials. The recent advancement in computer performance provides an opportunity for the high-performance analysis of spatial data. As a step towards creating a new decision support system for the sales territory planning, we propose an efficient and generic method to find the best solution for the three most common scenarios in sales territory planning 3.1.2. In this master thesis, we developed a multi-objective heuristic search method for solving a large-scale sales territory planning problem. The presented method is established upon the genetic algorithm, where a number of candidate solutions compete and cooperate with each other to find the best solution for our problem. In addition, our method is completely independent from the objective function and can be easily configured for a wide variety of multi-criteria problems. The results demonstrate the effectiveness of our proposed technique to find an optimal structure of sales territories in a reasonable time.

Contents

1 Introduction

In recent years along with innovations in the information technology, the decision process and the assessment methodology have evolved rapidly. In the other words, the planning for specific problems is no longer a traditional case based only on the advice of experts or planners, but it has become automated and improved to meet the needs of decision makers. The decision-making process can be complicated because of the multiplicity of data, actors and interests to be considered at each stage of development. One of the most popular and important decision-making processes is Territory Planning. Territory Planning is the problem of merging small Geographical areas (e.g. municipality, zip code or county) into larger Geographical areas (are called Territories) according to some predefined criteria. Territory planning problems are motivated by a wide variety of applications such as political district planning, territory design for schools or social facilities or emergency services, sales territory planning and etc.

1.1 Research motivations

Nowadays there is substantial attention to explore the potential applicability of machine learning and artificial intelligence for spatial data analysis. In this work, we are interested in a proposal of a sales force deployment decision model and more specifically on sales territory alignment, integrating an optimization tool in GIS: the Genetic Algorithm.

There is a tool which is called SalesNet as shown in Fig. 1.1. This tool has been developed by WIGeoGIS GmbH. "SalesNet allows the re-planning of sales areas in record time. By means of area simulations and collaboration options, this tool allows fast decisions to be made in respect of restructuring" [15].

But, there is not any recommender system to simulate and suggest an optimal solution to planners in sales territory alignment. Currently, the planning is done manually by experts based on their expertise and previous experiences. Because the current procedure for planning is somehow trial and error, the result is not totally reliable. On the other hand, there is no evaluation method for those planning results.

1.2 Objectives

For sales territories, well-planned decisions enable an efficient market penetration and lead to decreased costs and improved customer service. In addition, balanced territory design ensures optimal market saturation to improve the success of sales representatives. As

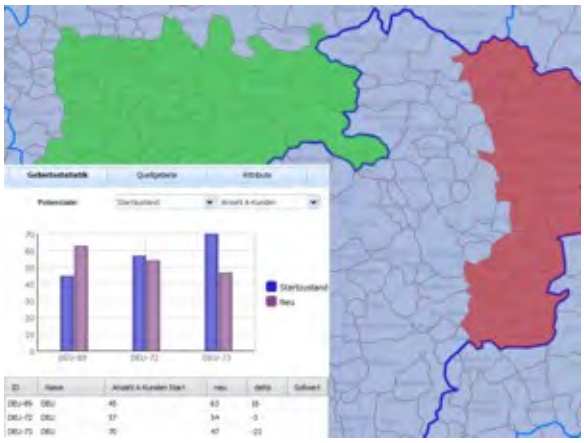

Figure 1.1: *SalesNet from WIGeoGIS GmbH*

we mentioned in the motivation section, proposing a decision-making model and developing a customized genetic algorithm for sales territory planning is the basis of this thesis. In simple words, we aim to eliminate the difficulties of creating the territories manually by decision-makers. The following list describes the objective of this thesis:

- Identifying and reallocating overlapping sale territories.
- Balancing the potential and workload between territories.
- Reducing travel time.
- Improving the performance.

1.3 Outline of this thesis

This thesis is structured as follows. Theoretical Background is provided in Chapter 2, beginning with Geo-marketing and territory planning definitions and proceeding to explain the Genetic Algorithm later in the chapter. In Chapter 3, the territory planning problem is formulated and then several algorithms are explained in the form of a pseudo code, respectively. In Chapter 4, we explained how our decision-making model is implemented and works. Our methodology is evaluated and proved in Chapter 5. Finally we concluded with the summary of results.

2 Theoretical Background

In this chapter we will have a look at some theoretical definitions and different methods of Geo-marketing. To be more focused on the thesis topic, we narrow down the Sales Territory Planning as a method of Geo-marketing. Further we will go through the Genetic Algorithm in details.

2.1 What is Geo-marketing?

The traditional ways of interpreting consumers' behaviors no longer meet the challenge of understanding purchase behaviors. Lifestyles, mobility and Geographical distribution have become dimensions of marketing that cannot be neglected in marketing strategies. If the information about the customers is essential to make decision for marketing approach, Geo-marketing is additional knowledge about where customers are located Geographically [20]. Geo-marketing is a tool marketers use to specify strategies and make decisions. In the other words, Geo-marketing is not a marketing strategy of Geo-data; it is the use of Geo-data to create powerful and actionable insights into the regional market [6]. Geo-marketing is also a key tool for optimizing regional and global markets to obtain the maximum possible turn over [5]. Geo-marketing makes it possible to efficiently increase market penetration, pay more attention to local market characteristics and make smarter decisions. In addition, there is the possibility of integrating information from disparate sources, including internal databases (customers data, sales representative locations, selling agency, locations of competitors) and external databases (market data, residential data, micro-Geographic data).

2.1.1 Geo-marketing Target Groups

Here, we introduce some target groups of Geo-marketing [6]:

- **Local customers:** Geo-marketing can be used as a recommender by those businesses which are looking to open up a new branch in a new area. Geo-marketing can recommend them what areas show the most demand for their products or services and where the competitors are located. It can be also used in Media planning to find out the best location for billboard advertisements by taking different parameters into account such as the residences and traffic patterns of their customer base.
- **Internet customers:** Companies are able to show their online contextual ads to users who are Geographically related to the advertisements.

- **Mobile device users:** User can get some special offers such as discounts, e-coupons, and so on when users mobile device is brought within a certain area in which a shop is sending offers. For example: AT&T shop alert
- **Social media users:** Social media users: Nowadays, users are able to check in to various locations, such as local restaurants or coffee shops and receive some special offers from those businesses.

2.1.2 Geo-marketing Applications

Geo-marketing is an appropriate tool for the clarification of various questions in many fields. Some application fields for Geo-marketing are as follows [4, 9]:

- **Target Audience Analysis:** it is a method of recognizing and exposing regional and local potential customers for a product or service. By means of this analysis, companies can easily identify areas with target groups which are interested in their product or service.
- **Penetration analysis:** this analysis provides essential information to identify the position of company in the market and measures the strengths and weaknesses of company in particular Geographical area.
- **Competitive Analysis:** It is a statement of marketing strategy which determines the strengths and weaknesses of the competitors within the market and helps to obtain the information about potential customers in competitive areas.
- **Site Planning:** One of the crucial problems in the establishment of a business in an area is about its optimum location[25]. Site planning (is also called location analysis) considers existing and potential locations by their realistic parameters and determines how suitable a location is or can be in the future.
- **Marketing strategy:** It is the result of market analysis which empowers the company to establish pricing, distribution, promotional strategies and so on.
- **Media Planning:** This approach finds the most appropriate media outlets to reach the target market during advertising campaigns and makes the media investment more efficient.
- **Sales Territory Planning:** The goal of the sales territory planning is the optimal coverage of the market with sales services and existing resources, in order to better exploit the available potentials and thereby to increase the turnover.

Sales Territory Planning

Sales Territory Planning is considered as the problem of grouping of so-called sales coverage units (SCUs) into larger Geographic clusters called territories in such a way that those clusters are eligible according to the planning criteria and satisfy the conditions[18]. Each

territory must have a responsible sales representative. To be responsible means that the sales representative has to provide service for all (potential) accounts located in the corresponding sales territory. For example, a sales representative who is working for a company that provides materials concerning dental surgery will be responsible for all dentists practicing in her sales territory. She is only allowed to sell products in SCUs that belong to her sales territory [22]. Territory Planning mainly evaluates the sales territory efficiency and the fair distribution of workload and potential among the responsible sales representatives. There are several reasons for aligning existing or designing new territories. Firstly, any changes in the number of sales representatives obviously require modifications in the structure of existing territories. Other reasons are better coverage of market with the existing representatives or equal balancing of workload among the sales representatives. Moreover, customer shifts or the introduction of new products make it necessary to align territories.

There are several commonly used criteria for sales territory design as listed in the following [21]:

Organizational criteria

- *Number of territories:*
  The number of territories to be designed is usually based on the number of sales representatives designated by the company or planner.
- *Basic areas:*
  Sales territories are in most cases not designed based on a single customer. Indeed, customers are usually first aggregated into small areas (SCUs). Then, SCUs will be used as input in the territory planning process. Typical examples for basic areas are zip codes, municipalities or company trading areas.
- *Exclusive assignment of basic areas:*
  In most cases, basic areas must be exclusively assigned to one territory. In other words, only one sales representative should be responsible for one basic area.
- *Locations of sales representatives:*
  The location of sales persons (office or residence) is an important factor to be taken into account in the territory design process. Since they should frequently visit their customers, so the total travel distance for each salesman should be minimized as much as possible.

Geographical criteria

- *Accessibility:*
  Often a good accessibility of territories (e.g. to highways) or within territories (e.g. by means of public transportation) extremely decreases the cost of each sales person.
- *Contiguity:*
  Territories should be Geographically connected.

- *Compactness:*
  It expresses that territories with minimal total travel times are desired.

Activity-related criteria

- *Balance:*
  It expresses the desire of an equal treatment of all sales representatives.
- *Maximizing profit:*
  Profit is the most important aspect in the planning process. Territories should be designed in such a way that the total profit contribution will be maximized and costs should be as low as possible with respect to the limited resource of call time or effort.

2.2 Genetic Algorithm

In 1859, Charles Darwin proposed the theory of natural selection states that the plants and animals that exist today are the result of millions of years of adaptation to the demands of the environment. The organisms that are most capable of acquiring resources and successfully breeding are the ones whose generations will tend to be dominant in the future. Organisms that are less capable will tend to be totally eliminated or have few generations in the future. In nature, an individual in population competes with each other for acquiring resources such as food, shelter, and so on. Weak individuals have less chance to survive and tend to be eliminated in the future. On the other hand, the most adapted or fittest individuals reproduce a relatively large number offspring. Natural selection operates over chromosomes rather than over organisms. It relates chromosomes with the efficiency of the entity they represent. Thus allows that efficient organism which is well-adapted to the environment to reproduce more often than those which are not. The evolutionary process happens within the reproduction phase. There exists a large number of reproductive mechanisms in nature. The most common ones are mutation (which influences the chromosomes of offspring to be different from the parents' chromosomes) and recombination (that combines the chromosomes of the parents to produce the offspring) [27].

In 1975, John Holland developed an idea in his book "Adaptation in natural and artificial systems". He described the first Genetic Algorithm which was inspired from the principal of natural evolution to solve optimization problems. The Genetic Algorithm is a subclass of Evolutionary Algorithms which attempts to mimic some of the processes that occur in natural selection based on "Survival of the fittest" [27, 28].

2.2.1 Search Space

The Genetic Algorithm (GA) always looks for the best solution among a set of possible solutions. The space of all feasible solutions is called *Search Space* which also contains the

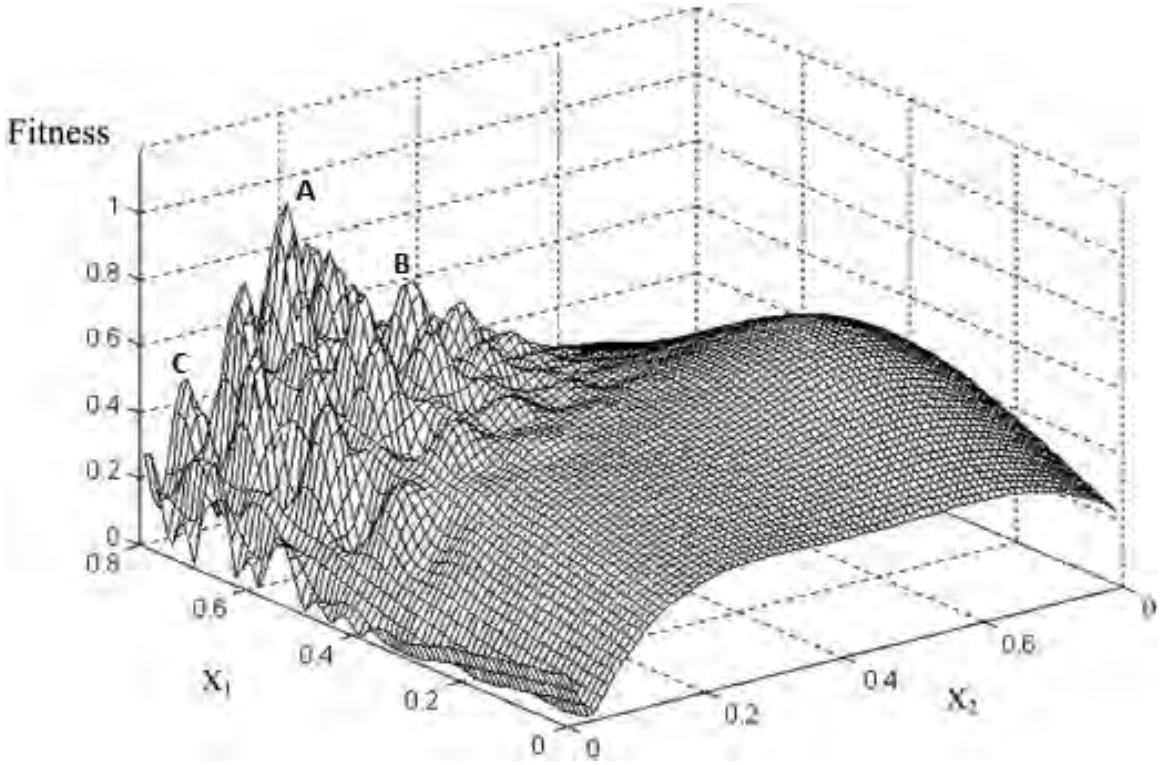

Figure 2.1: *Search space: (A)global optimum (B,C)local optimum (`http://pubs.rsc.org/`)*

global optimum solution. The GA considers each point of the search space as a candidate solution and evaluates its fitness. As the figure 2.1 illustrates, each position in the search space is a solution and has a fitness value (vertical axes). One of the most common issues of the GA is the local optimum. The local optimum is a candidate solution in the search space which has better fitness in comparison to the surrounding solutions, but it is not our desired solution (B and C in figure 2.1). The algorithm can be trapped in the local optima and does not continue to find the global optimum solution (A in figure 2.1). There are several techniques proposed to solve this issue which will be discussed in the methodology chapter in detail [30].

2.2.2 the Essence of the Genetic Algorithm

The GA is a stochastic algorithm. It means that some main operations of the GA such as selection and reproduction are based on random procedures. In addition, GA operates on a population (a set of individuals) in which each individual represents a potential solution to the problem. This feature of the GA makes it possible to execute the calculation concurrently which can significantly improve the efficiency. Genetic Algorithm is an independent optimizer which performs consistently well on a broad range of problem types. In other words, there are no prerequisites on the problem before using the GA. These individuals are randomly generated to make a population (a set of individuals). Every individual is

assigned a degree of its excellence by means of a fitness function according to the problem definition. Fitness function, or objective function, is an evaluator that not only shows how good the solution is, but also measures how close our potential solution is to the target.

One of the most important operators of the GA is *Recombination*. Cross-over is a typical recombination operator in which two parent solutions are cut at a certain point and the halves are connected to each other to make a new child (new candidate solution) which contains the characteristics of the parents. Recombination has a big impact on approaching the algorithm to the target. If parents have some good features, we can expect that all the good features are inherited by the generated child.

Besides the recombination in GAs, *Mutation* is another way of generating new child which can be a little bit different from its parents. Actually, mutation does not happen frequently in nature, but it is a very important operation in the Artificial Genetic Algorithm. The algorithm can explore the search space rapidly by means of few random changes (mutation) in the structure of the generated child after the recombination. The genetic algorithm will perform as a simple random search if the recombination operators produce just a new random solution as a child which does not inherit good characteristics of its parents (last generation). This random search algorithm is inefficient and does not guarantee to find promising solutions in a certain amount of time.

2.2.3 the Simple Genetic Algorithm

The Genetic Algorithm is a heuristic search technique which finds approximate solutions to optimization problems. The GA has two distinguished elements which are individuals and populations. Individual is a single candidate solution to the optimization problem while the population is a set of individuals involved in the optimization process. Observable properties of the individual (Phenotype) should be encoded to the raw genetic information (Genotype). Genotype is also called chromosome in the GA. The mapping between genotype and phenotype is necessary, because the solution set should be converted from the model into the form that the GA can work with, and the new generated chromosome in the GA should be converted into the form that model can evaluate. A chromosome is subdivided to some small parts which are called G̈enes.̈ The genes are the basic component for building a Genetic Algorithm. It is the representation of a single factor for a solution. 2.2 shows the mapping between phenotype and genotypes.

The first step in the GA before running is coding all the possible solutions into chromosomes, which is an abstract representation. This step can also be the most difficult part of a Genetic Algorithm. Secondly, a set of operators should be implemented for a reproduction that performs recombination (cross-over) and mutation over chromosomes. Functionality of GA extremely depends on these two steps. Because, the representation should be based on the structure of the search space and reproduction operators must be relevant according to the properties of the problem. All the possible solutions of the problem have to be mapped to at least one chromosome, to make sure that the algorithm explores the whole search space.

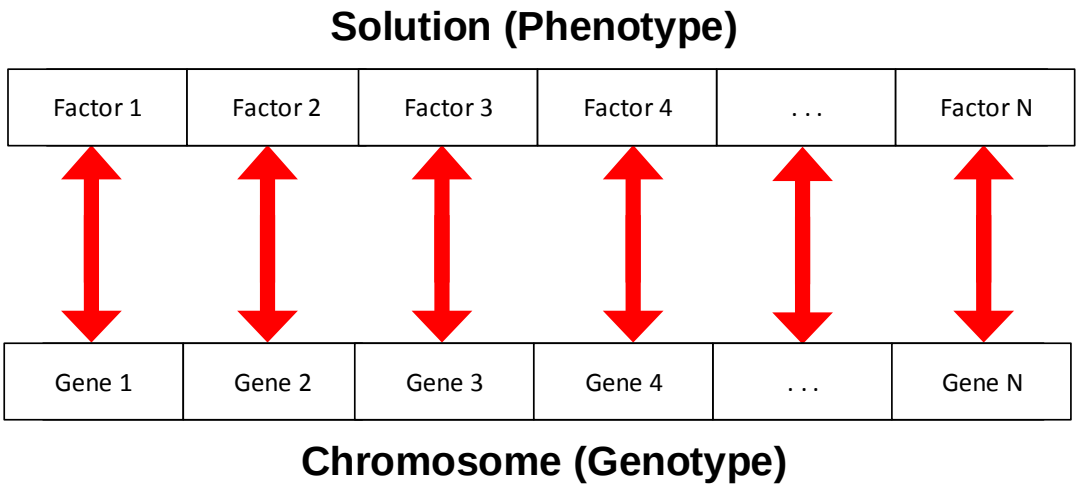

Figure 2.2: *Mapping between phenotype and genotype*

Genetic Algorithm is an iterative method, which loops over the population. Each iteration consists of the following steps:

Selection:

The first step after mapping and determining reproduction operators, consists in selecting individuals for reproduction. Selection is a method that randomly chooses individuals with a probability based on the relative fitness of the individuals in order to select best individuals than poor ones. Selection pressure is a degree to which the better individuals are desired. The higher the selection pressured, the more the better individuals are desired. This selection pressure ensures the improvement of the population fitness after several generations. Convergence rate of GA extremely depends on the selection pressure. If the convergence rate is slow, the GA will take unnecessarily longer time to find the optimal solution. On the other hand, high selection pressure results fast convergence rate that causes premature convergence to an incorrect solution (local optimal). This means that relative best solutions will take over the population and reduce diversity which needed for change and progress. Thus, the algorithm cannot explore whole search space for finding global optimum and will be stopped in the first local optimum it finds.The various selection methods are as follows:

- **Roulette Wheel Selection:**
  The roulette selection is a linear search through a roulette wheel with several slots in the wheel. Each slot is weighted in proportion to the individual's fitness values. This is a very common selection method, in which fit individuals are not guaranteed to be selected, but have more chance than poor ones. But, this method will have

a problem when the fitness values of the best individuals differ very much. For instance, If the best individual fitness is 95%, its slot occupies 95% of roulette wheel. Thus, other chromosomes have too few chances to be selected and it results very quick convergence.

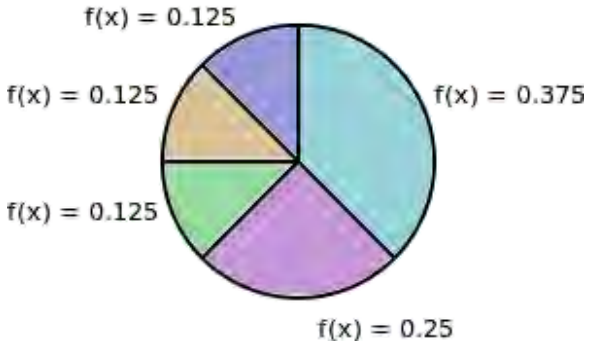

Figure 2.3: *Roulette Wheel Selection (`http://alexanderdbrown.com/`)*

- **Random Selection:**
  This is a stochastic method which randomly selects a parent from the population. The random selection does not take the fitness into account.

- **Tournament Selection:**
  It is the most common and ideal selection strategy which is able to adjust its selective pressure and preserves the population diversity by holding a competition among N individuals of the population. The winner of tournament competition is the best individual with the highest fitness. The winner is selected as a parent for generating new offspring via reproduction process for the next generation.

- **Rank selection:**
  Unlike roulette wheel selection, this method ranks the individuals based on their fitness values. So, each individual receives a rank from 1 to N (size of the population) and the worst individual gets 1 and best gets N. Then, potential parents will be selected via Tournament selection based on theirs ranks.

- **Stochastic Universal Sampling:**
  Stochastic universal sampling provides zero bias and minimum spread. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here equally spaced pointers are placed over the line, as many as there are individuals to be selected. Consider NPointer the number of individuals to be selected, then the distance between the pointers are 1/NPointer and the position of the first pointer is given by a randomly generate number in the range [0, 1/NPointer]. For 6 individuals to be selected, the distance between the pointers is 1/6=0.167. Fig. 2.4 shows the selection for the above example. Sample of 1 random number in the range [0, 0.167] => 0.1. After selection the mating population consists of the individuals: [1, 2, 3, 5, 6, 9]

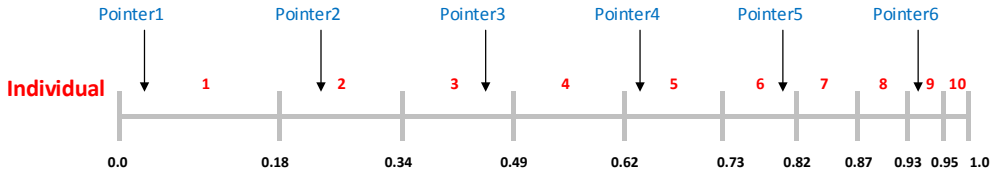

Figure 2.4: *Stochastic Universal Selection [16]*

Crossover (Recombination):

Crossover is the process which takes two parents from the mating pool, which has been created after the selection process, in order to create a better offspring. Crossover consists of three steps which are as follows:

1. Two individuals are randomly selected as the parents from mating pool.
2. A cross-point is randomly selected across each individual at the same position.
3. Finally, values are swapped between the two individuals following the cross-point.

Several crossover techniques are available and some of them are discussed as follows:

- **Single Point Crossover:**
  This method is one of the most common operators is used in traditional genetic algorithm. It uses one random cross-point in which two parents are cut. Then, the genomes next to the crosspoint are exchanged to produce a new child.

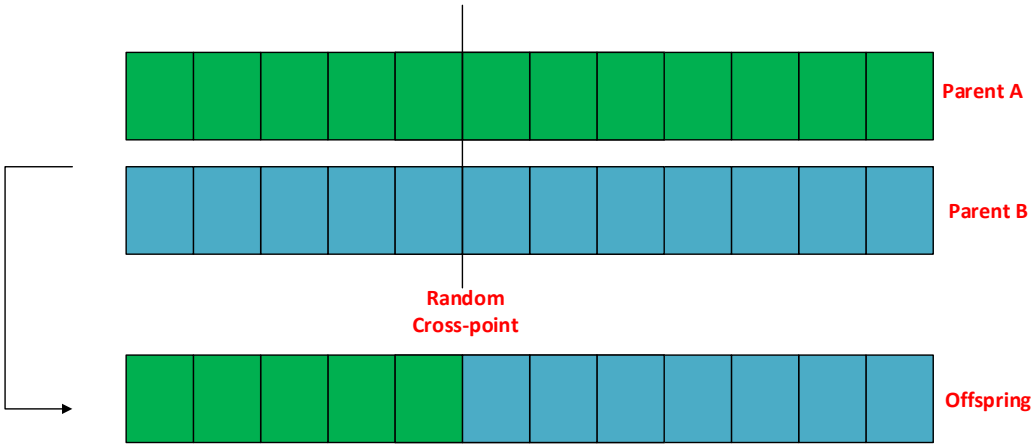

Figure 2.5: *Single-point Crossover*

- **Two Point Crossover:**
  This method uses two random cross-points along the length of each parent. Then, the genomes between two points are exchanged.

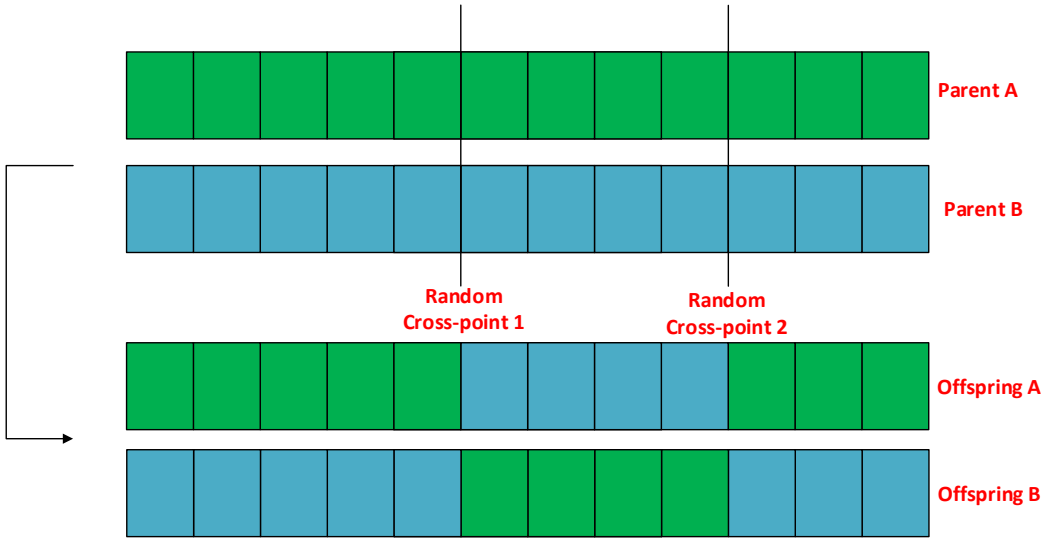

Figure 2.6: *Two-point Crossover*

- **Uniform Crossover:**
  Unlike previous operators that worked by dividing the parents into a number of sections and reassembling them to produce offspring, the uniform crossover considers each gene independently. This makes a random choice as to which parent each gene should be inherited from. Firstly, a string of N (size of the parents) random variables is generated from a uniform distribution between 0 and 1. In each position, if the random value is below a defined parameter (usually 0.5), the gene is inherited from parent A; Otherwise from parent B.

- **Ordered Crossover:**
  Ordered crossover is used when we have an order based permutation problem. In these kinds of problems, the generated offspring should not have duplicate genes after recombination. In addition, the intention is to transmit the relative order of genes from the second parent. For example, in Traveling Salesman Problem, each city cannot be visited more than once except the origin. Moreover, the order of genes in TSP chromosome shows the time that is taken for a complete tour. The Ordered Crossover acts as follows:

  1. Two random cross-points are generated along first parent.
  2. The segment between two cross-points are copied from the first parent (PI) into the first offspring at the same position.

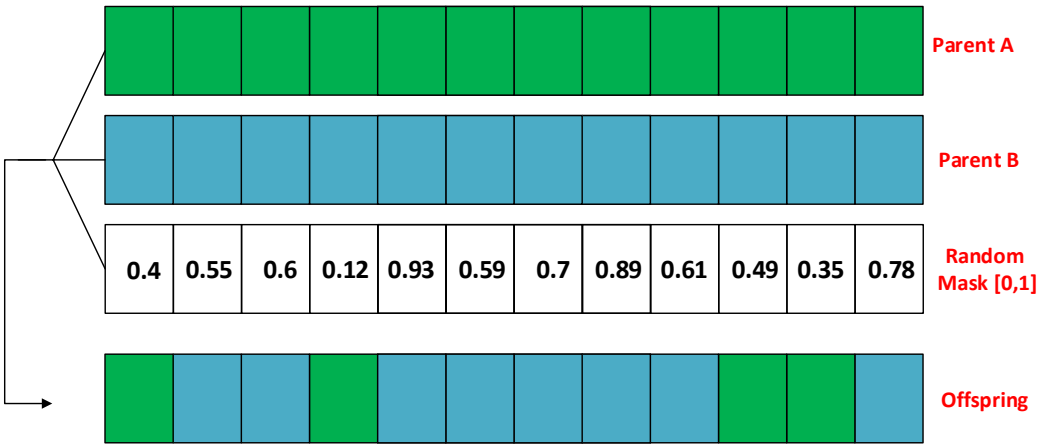

Figure 2.7: *Uniform Crossover*

3. Remaining unused genes from left and right segments of the first parent are copied into the first child based on the order that they appear in parent 2.
4. Create the second offspring in the same way, with the parent roles reversed.

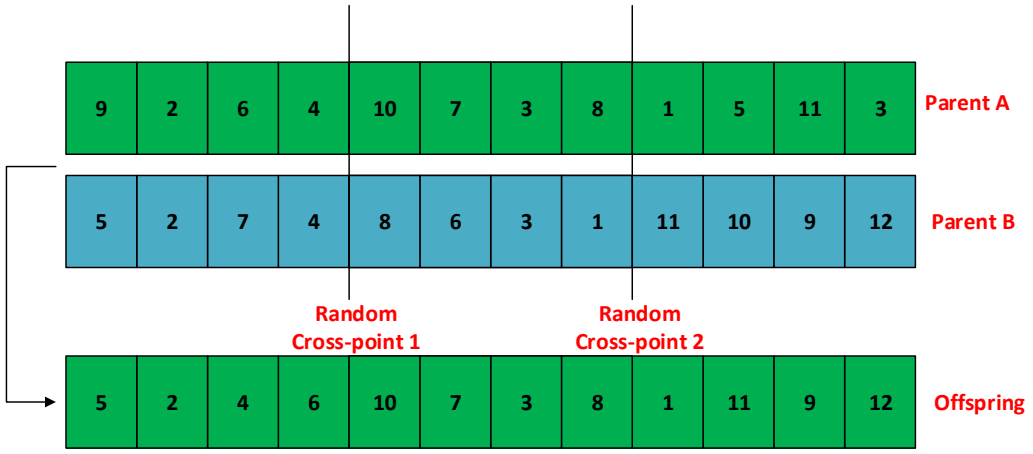

Figure 2.8: *Ordered Crossover*

- **Crossover probability:**
  Crossover probability is a parameter which determines the chance of performing crossover operation on the chosen parents (PC). This parameter is usually determined in the range [0.5,1.0]. After selecting the parents, a random number is generated between 0 and 1. If the number is lower than PC, the crossover operation are not performed and two parents will be copied to the next generation; otherwise, two new offspring are created by means of crossover operator.

Mutation:

Mutation is viewed as a simple search operator which preserves the genetic diversity in the population. This operator uses only one chromosome as the parent and creates the child by making some few random changes in the structure of parent chromosome. As we discussed in the last section, the crossover operator is used for exploiting the current generation of solutions to extract better ones. But, the mutation operator is used to explore the whole search space in order to make the algorithm converge toward the global optimum.

Mutation also holds up the algorithm to be trapped in a local optimum. In short, some of the different forms of mutation are listed as follows:

- **Flip Mutation:**
  This form of mutation is commonly used for binary representation of the solutions in which, each gene in considered separately and can be flipped $(0 \leftrightarrow 1)$ with a small probability.

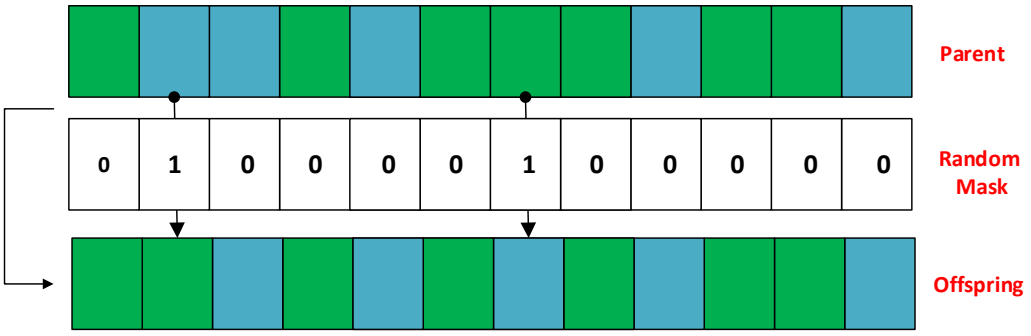

Figure 2.9: *Flip Mutation*

- **Swap Mutation:** In this form, two genes among the chromosome are selected randomly and interchange their values.

- **Inversion Mutation:** This operator works by selecting a subset of the chromosome randomly and reversing the order of genomes in which they appear in the chromosome.

- **Uniform Mutation:** This scheme was designed to consider all the genomes of the parent chromosome independently. It is an iterative method which starts from the

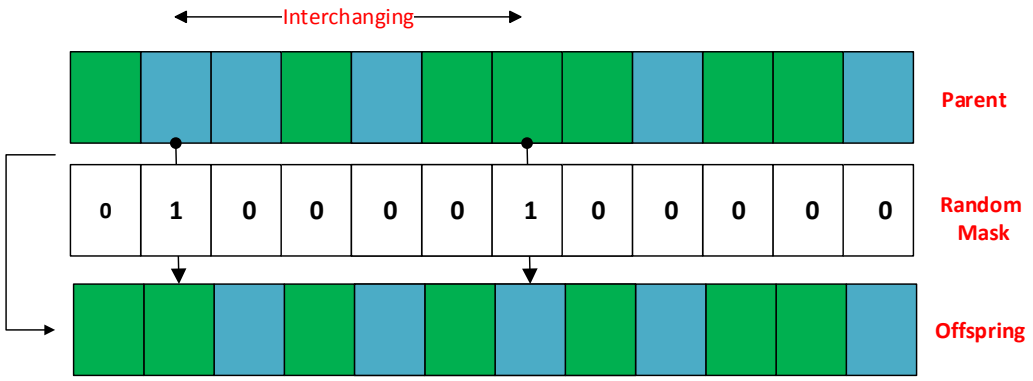

Figure 2.10: *Swap Mutation*

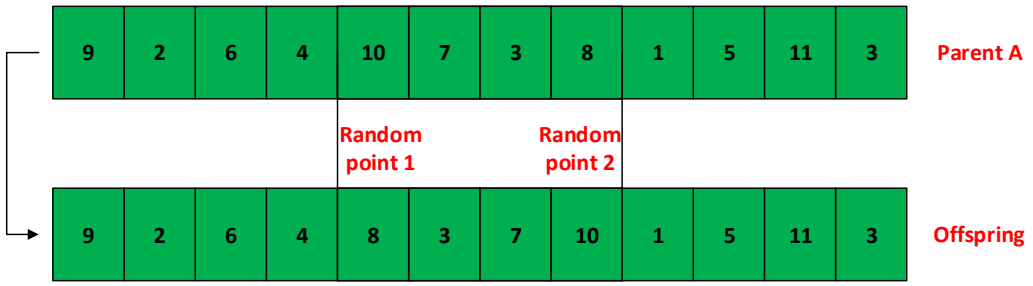

Figure 2.11: *Inverse Mutation*

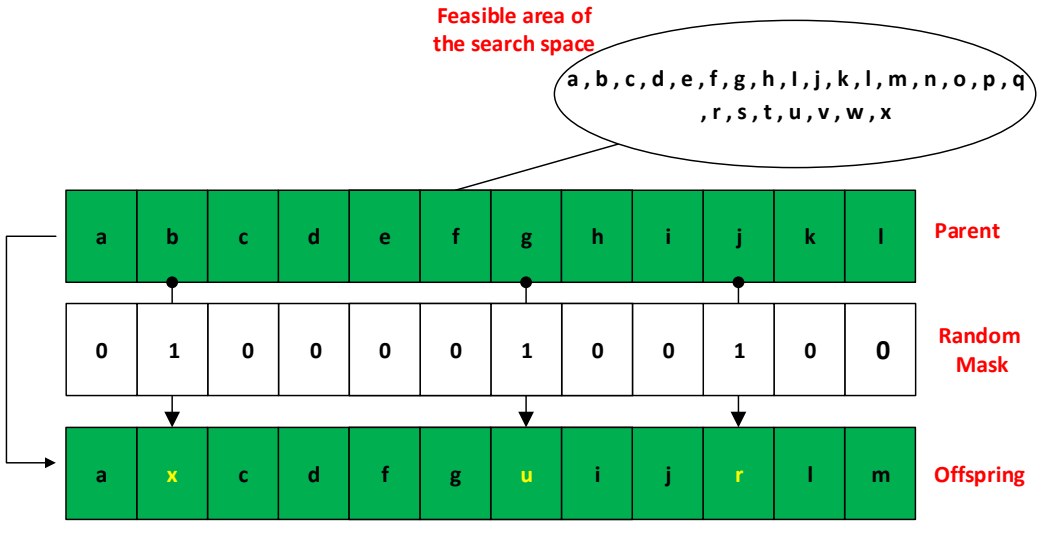

Figure 2.12: *Uniform Mutation*

first genome and traverse the whole length of the chromosome. For each gene, a random number is generated between 0 and 1. If the generated number is lower than the probability of mutation (PM), a new random value can be chosen for the genome from the feasible area of the search space; otherwise, the operator doesnât change the value of the genome. This operator can also be extended to a kind of hillclimbing mutation operators which mutates the genome only if the mutation improves the fitness of chromosome (solution).

Termination Condition:

There are various conditions that can stop the algorithm. The following are the few techniques of termination:

- **Elapsed time:** The evolutionary process will end after a predetermined period of time has elapsed.
- **Generation count:** Terminates evolution after a certain number of generations have passed.
- **Stagnation:** This condition halts the process if no improvement in populationâs best fitness is observed within a predetermined number of generations or during an interval of time.
- **Target fitness:** The algorithm is terminated once at least one candidate in the population has reached to the specified fitness score.
- **User Abort:** This condition is implemented to allow users to terminate the genetic algorithm whenever they want.

Fitness Function:

The fitness function (is also called objective function in optimization problems) is the most important component of genetic algorithm. It measures the goodness of candidate solutions and makes a basis for selection, and thereby it simplifies improvements. In the other words, the fitness function defines what improvement means for a specific problem. For instance, if the algorithm is supposed to maximize the number of 1 in a binary array, the fitness function will be as follows:
As it is shown in the figure above, the solution B is the best candidate for reproduction.

The general flowchart of a simple Genetic Algorithm is the following:

2.2.4 Advantages and Disadvantages of Genetic Algorithm [27]

Here, we mention some of the advantages of genetic algorithm briefly:

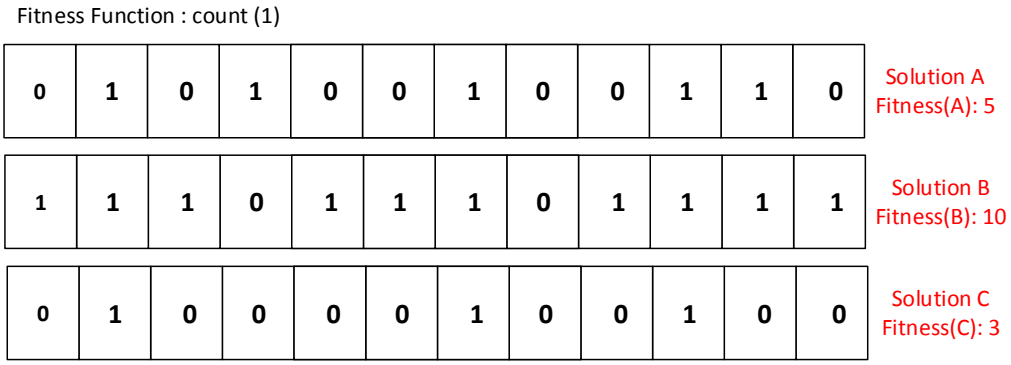

Figure 2.13: *Fitness function*

- It is possible to run the algorithm in parallel
- The algorithm is able to escape from local optimum
- It is able to discover global solution
- It can solve multi-objective problems
- It is totally independent from the problem domain
- The fitness function can be anything which can be evaluated by computers
- It can be easily customized for different problems
- Its concepts are easy to understand
- It always returns an answer for the problem and the answer gets better with time
- It can be very fast for some special problems (e.g. TSP)

, and some limitations of genetic algorithm are:

- Identifying the fitness function can be difficult
- Definition of representation (mapping from Phenotype to Genotype) for the problem is not straightforward
- Premature convergence happens
- It is always difficult to find the best value for various parameters such as: selection pressure, crossover and mutation probabilities, population size and so on
- GA cannot always find the exact solution, but finds the best solution
- It does not assure constant optimization response time

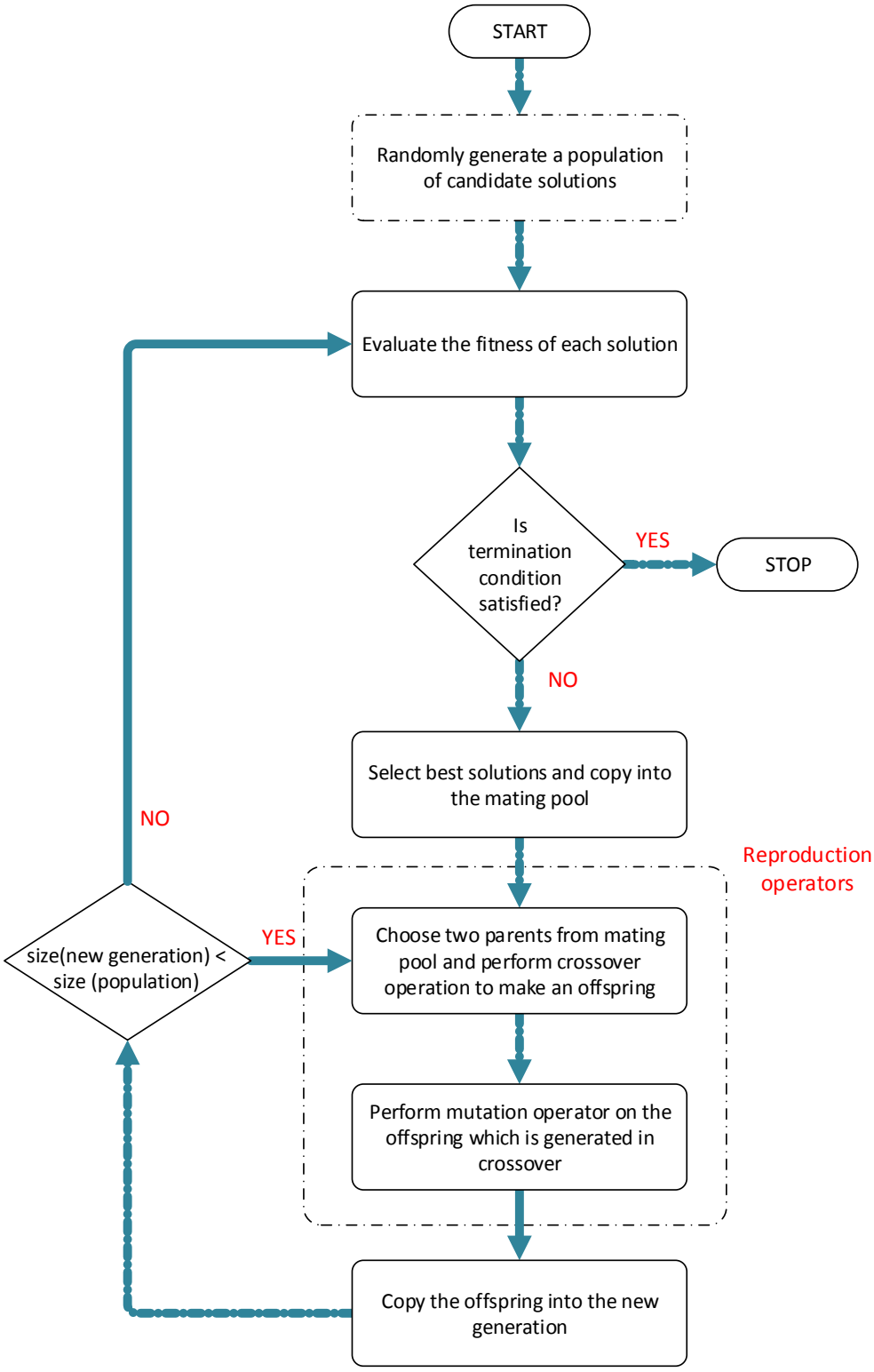

Figure 2.14: *Flowchart of Genetic Algorithm*

2.2.5 Applications of Genetic Algorithm [24, 13, 27]

The Genetic Algorithm is applicable in many fields. Here, we list some real-world uses of Genetic Algorithm.

Automotive Design

Using GA in designing composite materials and aerodynamic shapes for cars results combinations of best materials and best engineering to provide faster, lighter, more fuel efficient and safer vehicles.

Engineering Design

Getting the most out of a range of materials to optimize the structural and operational design of buildings, factories, machines, etc. is a rapidly expanding application of GA. These are being created for such uses as optimizing the design of heat exchangers, robot gripping arms, satellite booms, building trusses, flywheels, turbines and etc.

Robotics

GA can be used to find the optimal designs and components for particular usages, or to return results for entirely new types of robots that can perform multiple tasks and have more general application.

Optimized Telecommunications Routing

Finding dynamic and anticipatory routing of circuits for the optimized telecommunications networks is one of the main applications of GA in recent years.

Trip, Traffic and Shipment Routing

The "Traveling Salesman Problem" or TSP as a well-known application of GA can be used to plan the most efficient routes and scheduling for travel planners, traffic routers and even shipping companies.

Computer Gaming

GA has been programmed to incorporate the most successful strategies from previous games - the programs 'learn' - and usually incorporate data derived from game theory in their design. Game theory is useful in most all GA applications for seeking solutions to whatever problems they are applied to, even if the application really is a game.

Marketing

The world changes all the time, so does the market. Thus dynamically changing the trading systems or strategies is needed to keep their profitability during the time. Genetic Algorithm is used to change the values of the strategyâs parameters or using completely different rules to generate the trading signal.

3 Methodology

In this chapter, we will explain our methodology in more detail. In section 3.1 we will formulate the mathematical model of sales territory planning problem. Then we will depict our algorithm in section 3.2.

3.1 Sales Territory Planning Problem Definition

As we discussed in previous chapters, Sales territory planning can be formulated using an optimization problem in which each basic area (SCU) is assigned to the best candidate sales representative according to certain criteria and constraints. Note, sales force deployment is an aggregate planning problem. The usual planning horizon is one year. So, decisions as how often (e.g., every week, once a month), which days and at what time a customer is to be visited (and the sequence of customers) are the subject of subsequent planning stages [26].

3.1.1 Basic model of sales territory design:

Several researches have been done on territory design problems so far and most of them have the same basic criteria, including the desire for compact, contiguous, and balanced territories. Such a basic model allows a wide applicability of the algorithm and can be considered as a starting point for more complex models which take more constraints and criteria into account [21].

Firstly, we define 'Building Blocks' for territory design problems. Then, we present the mathematical model of territory design problem:

Basic Area:

The basic area (also called Sales Coverage Unit) is considered as a relatively small Geographical area which depend on the specific problem to be solved. Zip code, municipality and trading areas are the most common examples of basic areas [32].

$$\begin{aligned} &C\text{: a set of basic areas} \\ &j \in |C| \text{: index of basic area which is often represented by its central point} \end{aligned} \tag{3.1}$$

Location of sales representatives:

The location of salespersons is considered as the centroid of the basic area where the salesperson lives, because our distance matrix consists of travel time and distance between the centroid of all municipalities in Germany.

$$\begin{aligned} &V\text{: a set of basic areas for locating salespersons } (V \subseteq C) \\ &i \in |V| \text{: index of sales representative} \end{aligned} \tag{3.2}$$

Number of territories:

The number of sales territories is often defined by decision makers before planning. Thus, we assume that the number of territories is N.

Non-overlapping:

Each basic area must be assigned only to one salesperson.

$$\begin{aligned} &X_{ij} \in \{0,1\} \text{: a binary variable equal to } 1 \text{ if salesperson located in SCU } i \\ &\text{is assigned to SCU } j \text{ ; Otherwise equal to } 0 \end{aligned} \tag{3.3}$$

Complete assignment of basic areas:

Every basic area must be assigned to exactly one salesperson (a territory).

$$\begin{aligned} &T_i \subset C\text{: } i\text{-th territory} \\ &\sum_{i=1}^{N} T_i = C \ \ , \ \ T_i \cap T_k = \emptyset \ \ , \ \ \forall\, i \neq k \ \ , \ \ 1 \leq i,k \leq N \end{aligned} \tag{3.4}$$

Contiguity:

Territories should be created in such a way that all basic areas are connected. In order to obtain contiguity, neighborhood information for each basic area is required [18].

$J_i \subseteq C$: a set of SCUs which are assignable to salesperson in SCU i

$A_i \subseteq C$: a set of SCUs are adjacent to SCU j

q_{ivj}: quantity of flow from v to j with origin in location i

$$X_{ij} + \sum_{v \in A_i \cap J_i} (q_{ijv} - q_{ivj}) - \sum_{v \in J_i | i=j} X_{iv} = 0 \qquad i \in I, j \in J_i$$

$$|J_i| . X_{ij} - \sum_{v \in A_i \cap J_i} q_{ijv} \geqslant 0 \tag{3.5}$$

Compactness:

A territory is compact if it is relatively round-shaped and undistorted. In our approach we consider the total travel time from the centroid of territory T_j (usually the basic area where the salesperson lives) to each basic area assigned to territory T_j. Then, the following problem should be solved in order to obtain the compactness:

$$minimize\ f(x) = \sum_{i=1}^{N} K_i \qquad (i = 1, ..., N)$$

$$subject\ to$$

$$K_i = \sum_{j \in T_i} D_{ij} \qquad (i = 1, ..., N) \tag{3.6}$$

$$where:$$

$D_{ij} =$ travel time or travel distance from the salesman basic area

of territory$i$ to SCUj

$T_i =$ a set of basic areas belong to territory i

Objective:

There are several criteria that can be considered to group basic areas into the sales territories. Some of them search to balance income or potential turnover [23]. Some criteria attempt to balance workload between salesmen [19]. Another objective is to maximize the total profit contribution [31]. In our model, we formulate an objective to balance the potential customers and workload among sales representatives. In order to obtain this objective, we should minimize the total standard deviation of potential in each territory from the mean potential value (target value).

$$
\begin{aligned}
&minimize\ f(x) = \sum_{i=1}^{N} |R_i - \bar{r}| \\
&subject\ to \\
&\qquad \bar{r} = \frac{1}{N} \sum_{j=1}^{n} r_j \\
&\qquad R_i = \sum_{j \in T_i} r_j \qquad (i = 1, ..., N) \\
&\qquad |R_i - \bar{r}| \leqslant \delta \bar{r} \qquad (i = 1, ..., N) \\
&where: \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (3.7)
\end{aligned}
$$

n : number of basic areas

N : number of territories

T_i : i-th Territory

r_j : the amount potential contained in SCU j

R_i : the total potential in territory i

δ : is the maximum allowable percentage $(0 \leq \delta \leq 1)$

3.1.2 Different Scenarios in Sales Territory Planning:

Nowadays, marketers and decision makers face three different scenarios in sales territory planning. These scenarios are the following:

1. We have neither the location of sales representatives nor the existing sales territory structure. In this case, the number of needed territories should be determined by decision makers. Then the algorithm consists of two parts, respectively:
   - Finding the best location for each sales representative. The number of sales representatives are equal to the number of territories.
   - Performing a heuristic search algorithm for finding an acceptable sales territory structure.

   This scenario is usually called *Green Field Planning* in Geo-marketing.

2. We have a number of sales representatives with fixed locations, but we do not have any existing territory structure.

3. We have either fixed sales representatives or existing sales territory structure. But, the territories should be optimized due to the new strategy of the sales organization.

For each scenario, a respective strategy is chosen. But, we are going to design a generic approach for all scenarios. So we should have a workflow which supports all scenarios. Fig. 3.1 presents how our model handles all scenarios:

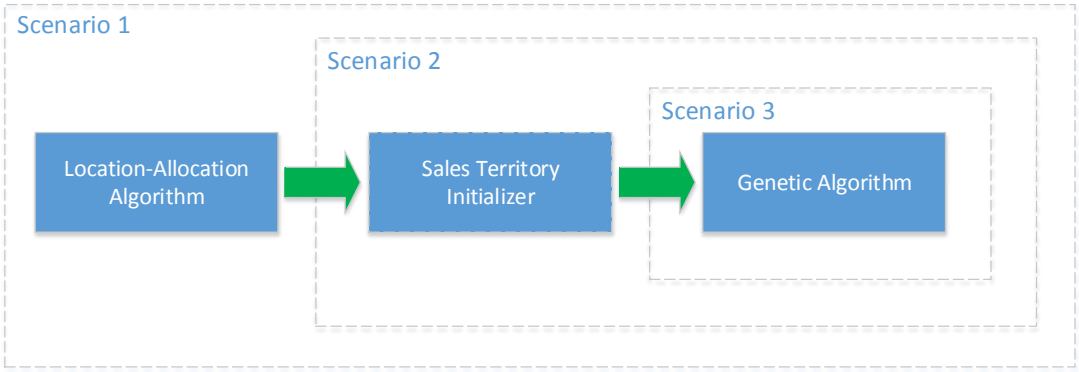

Figure 3.1: *Generic Workflow*

3.2 Algorithm Design

In this section, we firstly introduce a novel approach to handle the location-allocation problem. Secondly, we define an algorithm for initializing a raw structure for sales territories based on the travel time. Ultimately, we design a customized genetic algorithm for solving our multi-objective sales territory design problem. As we discussed before, one of the main advantages of the genetic algorithm is that the genetic algorithm does not depend on the problem type. So, we only have to design a respective fitness function for each scenario and we do not need to change the main body of the algorithm which will be defined later.

3.2.1 A Simple Approach for Location-Allocation

For the first scenario, it is very important to find a set of basic areas which are the starting points for the sales territory planning. These basic areas are also considered as the living points of sales representatives. Finding the best basic area as an input for the genetic algorithm has a dramatic impact on the performance of the GA. In our approach, we use the distribution of potential among basic areas to avoid locating several sales representatives in empty areas. The algorithm 1 classifies the basic areas into several classes and determines how many basic areas must be selected for each class.

3.2.2 Sales Territory Initialization

When the starting points are ready, we can start the GA to find the structure of sales territory. Since the search space can be so large due to the number of basic areas, we have to minimize the search space before running the genetic algorithm. So, we attempt to cluster

the basic areas based on the nearest sales man to make a raw structure for sales territories. The algorithm 2 shows the clustering process of basic areas.

Algorithm 1 Location Allocation Algorithm

1: **Inputs**:
   $Candidates = [c_1, c_2, \ldots, c_N]$ # list of capital municipalities as candidate location
   $Size_s \leftarrow$ size of sales team
2: **Outputs**:
   $Locations \leftarrow$ empty array list # empty array to store the locations of sales team
3: $Classes \leftarrow$ NaturalBreaks($Candidates$) # classifying the candidates by using NaturalBreaks
4: $Size_c \leftarrow$ size of $Classes$ # number of classes
5: **for** i = 1 to $Size_c$ **do**
6: $weight \leftarrow i$ # the weight of class
7: $identifier \leftarrow (Size_c * (2 + (Size_c - 1)))/2$
8: $capacity \leftarrow Round(weight * (Size_s/identifier))$ # the number of persons can be located in class[i]
9: **for** j = 1 to $capacity$ **do**
10: $tmp \leftarrow$ getRandomCandidate($Classes\,[i]$) # get a candidate municipality from class[i] randomly
11: addItemToList($Locations$, tmp) # adding a municipality to Locations array
12: **end for**
13: **end for**
14: **return** $Locations$

3.2.3 The Genetic Algorithm for Sales Territory Planning

The sales territory planning is the problem of grouping basic Geographic areas (SCU) such as counties, zip codes, municipalities into larger clusters called sales territories. The goal of sales territory planning is almost the maximizing of total profit of sales organization. As the sales territory planning is classified as NP-hard problem, using a meta-heuristic search algorithm is beneficial. Moreover, the sale territory planning is a multi-objective problem and is non-continuous and thus cannot be solved via linear programming [8]. The genetic algorithm is a search heuristic which has proven efficient and effective for solving spatial problems [29]. So we have chosen the genetic algorithm for our solution approach. Firstly, The abstract scheme of the genetic algorithm is defined in a pseudo-code fashion. Then, we describe in detail each part of the algorithm that have been designed particularly for the sales territory problem.

Mapping between Phenotype and Genotype

One of the main and also difficult parts of designing a Genetic Algorithm is the mapping between phenotype and genotype. In this step, we are supposed to encode all decision variables of our problem in such a way that can be applied to the genetic algorithm and can be simply decoded in order to be shown inside the application. Since we are going to design an abstract algorithm for any size of problem, the main issue in our approach

Algorithm 2 Clustering Algorithm

Inputs:
  $G = [g_1, g_2, \ldots, g_N]$ # set of basic areas
  $S = [s_1, s_2, \ldots, s_N]$ # set of salesmans basic areas
  DM # distance matrix (SCU_a to SCU_b)
Initialize:
  $Size_1 \leftarrow$ size of G
  $Size_2 \leftarrow$ size of S
  $Chromosome \leftarrow$ empty array list # array list to store territory genomes

for i = 1 to $Size_1$ **do**
  $tmp \leftarrow 0$ # variable to store nearest salesman id
  $gid \leftarrow G[i]$
  for j = 1 to $Size_2$ **do**
    $t \leftarrow$ Maximum value of travel time
    $sgid \leftarrow S[j]$ # basic area of salesman
    $traveltime \leftarrow DM(gid, gsid)$
    if $t > traveltime$ **then**
      $t \leftarrow traveltime$
      $tmp \leftarrow$ getSalesmanId($gsid$)
    end if
  end for
  $Genome \leftarrow$ createGenome(gid , tmp) # instantiating the territory genome object

  addItemToList($Chromosome$, $Genome$) # adding Genome to Chromosome
end for
return $Chromosome$

Algorithm 3 The Genetic Algorithm

$initialize$ a population with random candidate solutions;
while ($termination\ condition$ is satisfied) **do**
  $evaluate$ each candidate;
  $select$ fitter candidates and *copy* into the mating pool;
  while (size of new generation = size of population) **do**
    $select$ pair of parents from mating pool;
    $recombine$ parents to make offspring;
    $mutate$ the generated offspring;
    copy the offspring into the next generation;
  end while
end while
$decoding$ the best found solution

is that the proposed genotype data structure should be as efficient as possible. Moreover, the structure of genotype should be flexible enough to be easily manipulated via the operations of the algorithm. According to the empirical result, our algorithm is capable to handle large scale problems (e.g sales territory in the level of municipalities around more than $10,000$ basic areas) in a promising calculation time and computation effort. Thus, we have chosen a *Generic Array List* data structure to store an arbitrary size of genotype in a array form and manipulate it easily. Generic array list data structure is a collection of items which supports all optional operations such as add, delete, update, etc. The main advantage of a generic array list is that it is capable to store different data types such as Integer, String, Object, Arrays, etc. The figure 3.3 shows a schema of generic array list.

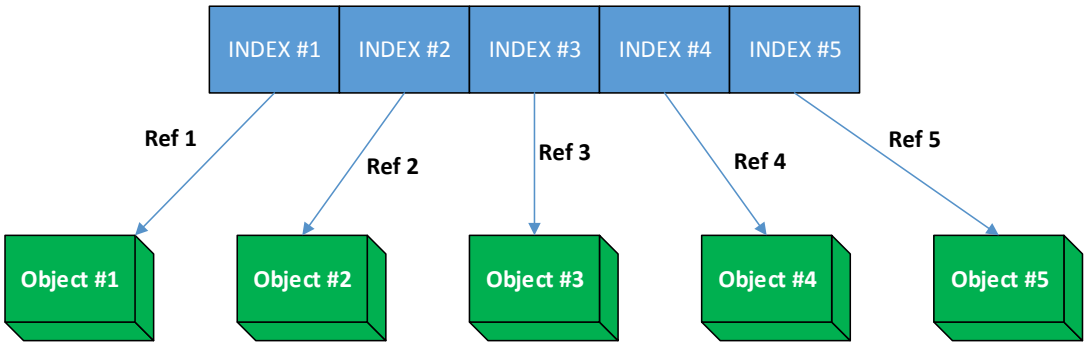

Figure 3.2: *Generic Array List*

Since array list is index-based data structure, iterating through an array of N elements is faster than other data structures with the same size. Moreover, the array list needs less memory than other data structures such as linked list. As we mentioned before, we are able to add and store any type of items into the generic array list. According to our defined territory planning model 3.3, each SCU must be assigned to only one sales representative. So, the pair of basic area (SCU) and sales representative should be unique. The figure 3.3 shows an object which is used as a genome in our genetic algorithm.

Initializing a Random Population

The Genetic Algorithm is a random population-based meta-heuristic algorithm. In the GA, we maintain many individuals (candidate solutions) in a population. The size of population is usually defined before running the algorithm. Each individual (also called *chromosome*) is composed of genomes. As we discussed before, we are able to add an arbitrary number of genomes into the generic array list. In our model, the size of array list shows the number of basic areas. The algorithm 13 creates a population of random candidate solutions by means of an iterative process. The process will be stopped when the number of generated individuals are equal to the size of population which is given by user.

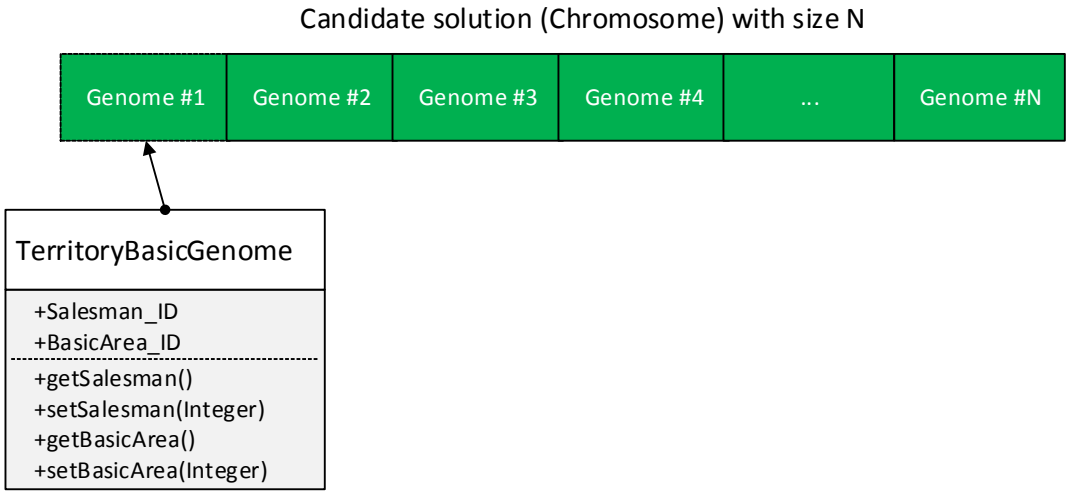

Figure 3.3: *Territory Genome Object*

Algorithm 4 Making Random Population

Inputs:
  $Size \in \mathbb{N}$ # size of the population
  $S = [s_1, s_2, \ldots, s_N]$ # S: set of salesmans
  $G = [g_1, g_2, \ldots, g_L]$ # G: set of basic areas
Initialize:
  $N \leftarrow$ size of S
  $L \leftarrow$ size of G
  $Chromosome \leftarrow$ empty array list
  $Population \leftarrow$ empty array list
for i = 1 to $Size$ **do**
  for i = 1 to L **do**
    $gid \leftarrow G[i]$
    $rnd \leftarrow$ getRandomNum(1 , N) # making random index between 1 and N
    $sid \leftarrow S[rnd]$ # choose one sales man randomly
    $Genome \leftarrow$ createGenome(gid , sid) # instantiating the territory genome object
    addItemToList($Chromosome$, $Genome$) # adding Genome to Chromosome
  end for
  addItemToList($Population$, $Chromosome$) # adding Chromosome to Population
end for
return $Population$

The generated random population by the algorithm above will be used as an input for the selection process that will be discussed in the next section.

Selection Process

At this step, the Genetic Algorithm goes through the main loop. The selection process is supposed to simulate the natural selection by giving fitter individuals a higher chance to survive and make offspring into the next generations. The selected individual will be added into the temporary storage (also called mating pool) to be used for breeding via cross-over and mutation operations. So the selection process is considered as the main component of the GA for converging towards a global optima. In addition, because fitter individuals will be selected via selection process based on their fitness, we also need the fitness function to evaluate how fit each individual is. We will explain our fitness functions in the next section. In our model, we use the *Tournament Selection* as the selection method. This method does not consider the global information of a population (e.g. relative fitness or the ranking of individuals). It just takes the local information into account. It means that best individual will be selected within a small group of individuals instead of the whole population [30]. The *Tournament Selection* gets the tournament size k as the parameter, which controls the selection pressure that is introduced in section 2.2.3. *Tournament Selection* is an iterative process which compares each individual k times with others and chooses the fittest individual in each tournament. If the tournament size is 1, the selection method will imitate the random selection. Because in this case, the selection pressure will be 0. On the other hand, If the tournament size is equal to the size of population, the fittest individual will occupy the whole population because of the high selection pressure. The algorithm 5 shows the Tournament Selection Method.

Algorithm 5 Tournament Selection

Inputs:
$Population = [ind_1, ind_2, \ldots, ind_N]$ # the population of candidate solutions
K # tournament size
Initialize:
$Size \leftarrow$ size of P # size of population
$Selection \leftarrow$ empty array list # array list of selected individuals
for i = 1 to $Size$ **do**
  $List \leftarrow$ empty array list # temporary list for each tournament
  addItemToList($List$, $Population[i]$)
  for j = 1 to $(K - 1)$ **do**
    $indv \leftarrow$ getRandomItem($Population$) # random individual($indv \notin List$)
    addItemToList($List$, $indv$)
  end for
  $fittest \leftarrow$ getFittestIndividual($List$) # fittest individual wins the tournament
  addItemToList($Selection$, $fittest$)
end for
return $Selection$

All selected individuals in the selection process will be copied into the mating pool to be used in the next operations.

Elitism

The generated offspring in the GA are supposed to inherit good characteristics of its parents, but sometimes offspring are weaker than their parents. So good candidate solutions can be lost after cross-over and mutation operations.Although the GA is capable to rediscover the lost candidate solutions again, it can increase the calculation time. For solving this issue, we will use a technique which is called *Elitism* [27].

In this technique, we copy a proportion of the best candidate solution from the mating pool into the next generation without any changes. Most of the time, *Elitism* extremely improves the performance of the GA, because it does not waste the time to discover the lost candidate solutions in the previous generations and helps to navigate the search direction towards the global optimal solution. In addition, unchanged candidate solutions which are copied by elitism are also remained in the mating pool for the reproduction process.

The Fitness Function

The fitness function can be viewed as the heart of the genetic algorithm. It means that the goodness of the candidate solution will be evaluated by the fitness function. In the other words, the fitness function determines how close the candidate solution is to the global optimal solution based on the type of problem. Since the sales territory planning is a multi-criteria problem intrinsically, we should formulate a multi-criteria decision making model to simultaneously minimize the components of criterion vector $F(x)$:

$$\begin{aligned} & minimize \quad F(x) = (f_1(x), f_2(x)) \\ subject\ to & \\ & f_1(x) \ = \ equations\ 3.6 \\ & f_2(x) \ = \ equations\ 3.7 \end{aligned} \tag{3.8}$$

Moreover, each objective is measured on a different scale. In the following model, we provide a mathematical formulation to normalize our objective components and simplify

them into the single objective function:

$$fitness = \left(\left(\frac{f_1(x) - a}{b - a}\right) * w_1\right) + \left(\left(\frac{f_2(x) - c}{d - c}\right) * w_2\right)$$

$$subject\ to$$

$$a \leq f_1(x) \leq b$$

$$c \leq f_2(x) \leq d$$

$$where:$$

$f_1(x)\ ,\ f_2(x)$: control variables

$w_1\ ,\ w_2$: weights of control variables

$a\ ,\ c$: lower bounds of control variables

$b\ ,\ d$: upper bounds of control variables

(3.9)

In this part, we only consider the first objective that we have defined in the objective section 3.1.1. The algorithm 6 shows how to calculate the fitness of a candidate solution.

The Cross-Over Operation

The first operation of reproduction process is *Cross-Over*. This operation picks up two parents from the mating pool and combines them in such a way that the generated offspring inherits the characteristics of its parents. This operation gets the probability value $0 \leq P \leq 1$ as the first parameter and the number of cross-over points as the second parameter. In our model, we use a *one point cross-over*. The algorithm 7 depicts how the cross-over operation makes two offspring from two parents:

Algorithm 6 Fitness Function

Inputs:
  $Chromosome = [genome_1, \ldots, genome_N]$ # candidate solutions
  $S = [s_1, s_2, \ldots, s_N]$ # S: set of salesmans
  DM # distance matrix (SCU_a to SCU_b)
  W_1, W_2 # criteria weights
  a, c # lower bounds
  b, d # upper bounds
Initialize:
  $Size_1 \leftarrow$ size of S
  $Size_2 \leftarrow$ size of $Chromosome$
  $potentials \leftarrow$ empty array list # array list to store potential values
$dist \leftarrow 0$ # total distance (second critera: $f_2(x)$)
$fitness \leftarrow 0$
for i = 1 to $Size_1$ **do**
  $potential \leftarrow 0$
  $sid \leftarrow S[i]$
  for j = 1 to $Size_2$ **do**
    $Genome \leftarrow Chromosome[j]$
    $gsid \leftarrow$ salesman_id of $Genome$
    if $gsid = sid$ **then**
      $potential \leftarrow potential +$ getPotential($Genome$)
      $SCU1 \leftarrow$ get SCU of $S[i]$
      $SCU2 \leftarrow$ get SCU of $Genome$
      $dist \leftarrow dist + DM(SCU1, SCU2)$ # distance from $SCU1$ to $SCU2$
    end if
  end for
  addItemToList($potentials$, $potential$)
end for
$sd \leftarrow$ calculateDeviation($potentials$) # standard deviation (first criteria: $f_1(x)$)

$fitness \leftarrow$ calculateFitness($sd, dist, w_1, w_2, a, b, c, d$) # equation 3.2.3

return $fitness$

Algorithm 7 Cross-Over Operation

Inputs:
$Chromosome1 = [genome_1, \ldots, genome_N]$ # a candidate solutions as parent1
$Chromosome2 = [genome_1, \ldots, genome_N]$ # a candidate solutions as parent2
P # cross-over probability: $0 \leq P \leq 1$
Initialize:
$Size_1 \leftarrow$ size of $Chromosome1$
$Size_2 \leftarrow$ size of $Chromosome2$
$Result \leftarrow$ empty array list # an array list to store 2 offspring
$rnd \leftarrow$ getRandomNum(0 , 1) # random number between 0 and 1
if $rnd \geq P$ **then**
  $Max \leftarrow$ Min($Size_1$, $Size_2$)
  if $Max > 1$ **then**
    $index \leftarrow$ getRandomNum(1 , Max) # random index between 1 and Max
    for j = 1 to $index$ **do**
      $Genome \leftarrow Chromosome1[j]$
      $Chromosome1[j] \leftarrow Chromosome2[j]$
      $Chromosome2[j] \leftarrow Genome$
    end for
  end if
end if
addItemToList($Result$, $Chromosome1$) # adding first offspring
addItemToList($Result$, $Chromosome2$) # adding second offspring
return $Result$

The Mutation Operation

After the cross-over operation, we should make very little changes in the generated offspring to attempt to explore the whole search space. In our model, this operation has a very important role in converging the search direction to the global optimum. Since we have already created a raw structure of sales territories, we just mutate the offspring on the border of existing territories. It means that a basic area which is located on the border can be randomly assigned to the neighbor territory based on a very small probability. In this approach, we also maintain the compactness and contiguity of the raw structure, because we only reassign the basic areas on the border and those inside the territories will be remained unchanged.

Algorithm 8 Mutation Operation

```
1: Inputs:
   Chromosome = [genome_1, ..., genome_N] # offspring
   P                                      # mutation probability: 0 ≤ P ≤ 1
   NM ← Neighborhood matrix               # NM[i, j] = 1 if SCU_i is neighbor of SCU_j; Otherwise 0
2: Initialize:
   Size ← size of Chromosome
3: for i = 1 to Size do
4:    genome ← Chromosome[i]
5:    list ← get the neighbors of genome # temporary salesman ID

6:    rnd ← getRandomNum(0 , 1)  # random number between 0 and 1
7:    if rnd ≥ P then
8:       nbr ← getRandomGenome(list)  # selecting random neighbor

9:       nsid ← getSalesmanID(nbr)  # salesman Id of neighbor genome
10:      sid ← getSalesmanID(genome)  # salesman Id of genome

11:      gid ← getScuID(genome)      # SCU Id of genome
12:      if nsid ≠ sid then
13:         genome ← createGenome(gid , sid)
14:         Chromosome[i] ← genome
15:      end if
16:   end if
17: end for
18: return Chromosome
```

Minimizing the Search Space

An we explained in chapter 2, the Genetic Algorithm is a non-deterministic searching method. It means that the whole search space should be explored by the algorithm to find the best solution. Thus, exploring of a larger search space is likely to take longer time. Although we have discussed about minimizing the search space before running the genetic algorithm in the section 3.2.2, it is also possible to minimize the search space inside the genetic algorithm by using evolutionary operators that avoid generating useless or invalid solutions. We can avoid useless solutions by defining some constraint in mutation operation algorithm. A useless solution is a valid candidate solution which is generated by the algorithm after several generations, but does not lead the search direction to the global optimal solution. Since our goal is the balancing of the potential between territories (sales persons), it will not be beneficial to mutate a candidate solution in mutation operation by taking a basic area from a territory with less potential and assigning it to a territory with more potential, because the weak territory will get weaker. The figure 3.4 shows the useless solution:

In addition, an invalid solution is a candidate solution which is generated by the algorithm, but it is not in the feasible area of search space and violates the territory planning

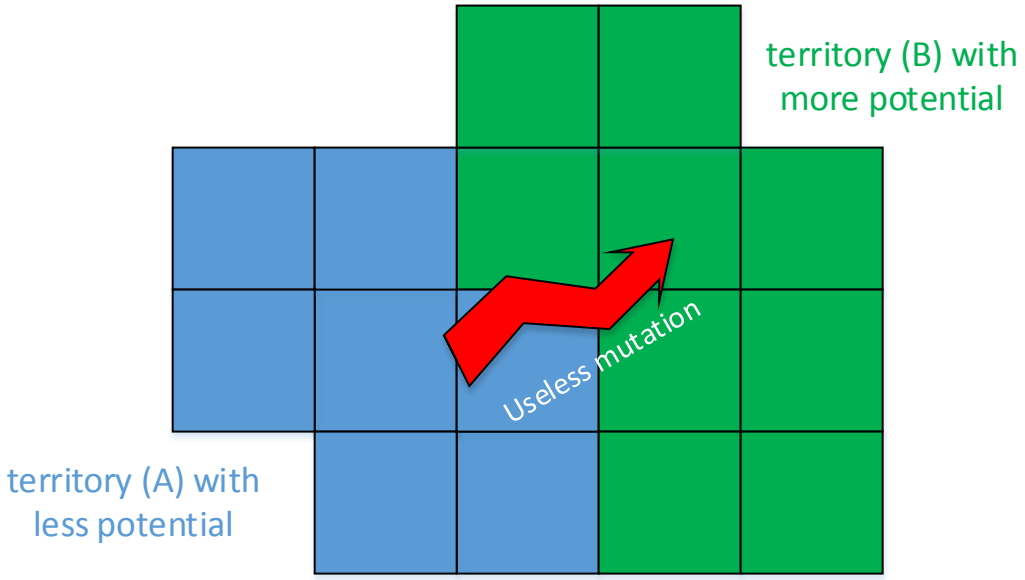

Figure 3.4: *Useless Solution*

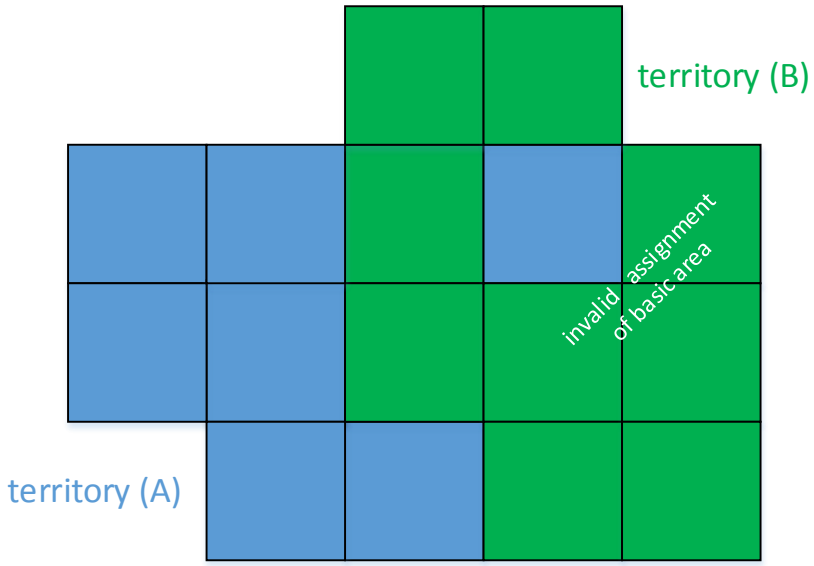

Figure 3.5: *Invalid Solution*

contiguity constraint which is defined in section 3.5. The figure 3.5 shows an invalid solution in which a basic area is disconnected from its territory:

For the treatment of an invalid solution, we designed algorithm 9 to repair the invalid solutions. This algorithm detects the invalid assignments of basic areas and assigns them to the surrounding territory.

Algorithm 9 Treatment Operation

Inputs:
$Chromosome = [genome_1, \ldots, genome_N]$ # offspring
$NM \leftarrow$ Neighborhood matrix # $NM[i,j] = 1$ if SCU_i is neighbor of SCU_j; Otherwise 0
Initialize:
$Size1 \leftarrow$ size of $Chromosome$
for i = 1 to $Size1$ **do**
  $genome \leftarrow Chromosome[i]$
  $list \leftarrow$ get the neighbors of $genome$ # temporary salesman ID
  $Size2 \leftarrow$ size of $list$
  for j = 1 to $Size2$ **do**
    $nsid \leftarrow$ getSalesmanID($list[nbr]$) # salesman Id of neighbor genome
    $sid \leftarrow$ getSalesmanID($genome$) # salesman Id of $genome$
    $flag \leftarrow 0$
    if $(nsid = sid)$ **then**
      $flag \leftarrow 1$ # basic area is not island
      break; # stop the loop
    end if
  end for
  if $(flag = 0)$ **then**
    $gid \leftarrow$ getScuID($genome$) # SCU Id of $genome$
    $genome \leftarrow$ createGenome(gid , sid)
    $Chromosome[i] \leftarrow genome$ # marge the island into the surrounding territory
  end if
end for
return $Chromosome$

Escaping from the Local Optima

As we discussed in chapter 2, the genetic algorithm can be easily trapped in local optima. Thus, the GA should be capable to escape from the local optimum and converge to global optimum. For solving this issue, the cross-over rate and specially mutation rate should be adaptive. It means the mutation probability should be increased when the diversity of candidate solutions is decreasing and should be decreased when the diversity is increasing. In our model, we increase the mutation rate dynamically when the decrement of standard

deviation becomes too slow. On the other hand, we decrease the mutation rate when the decrement of standard deviation is too fast.

Termination conditions

There are a some reasons which can stop the genetic algorithm. Since the genetic algorithm is a non-deterministic, we are not able to specify the global optimal solution so precisely. Figure 3.6 presents the progress of the genetic algorithm in minimizing the *standard deviation* of potential among territories as much as possible, but the progressive decrement of standard deviation is stopped after 500 generations. If a certain number of generations pass without any improvement, the algorithm will be terminated. The algorithm 10 shows the stagnation termination.

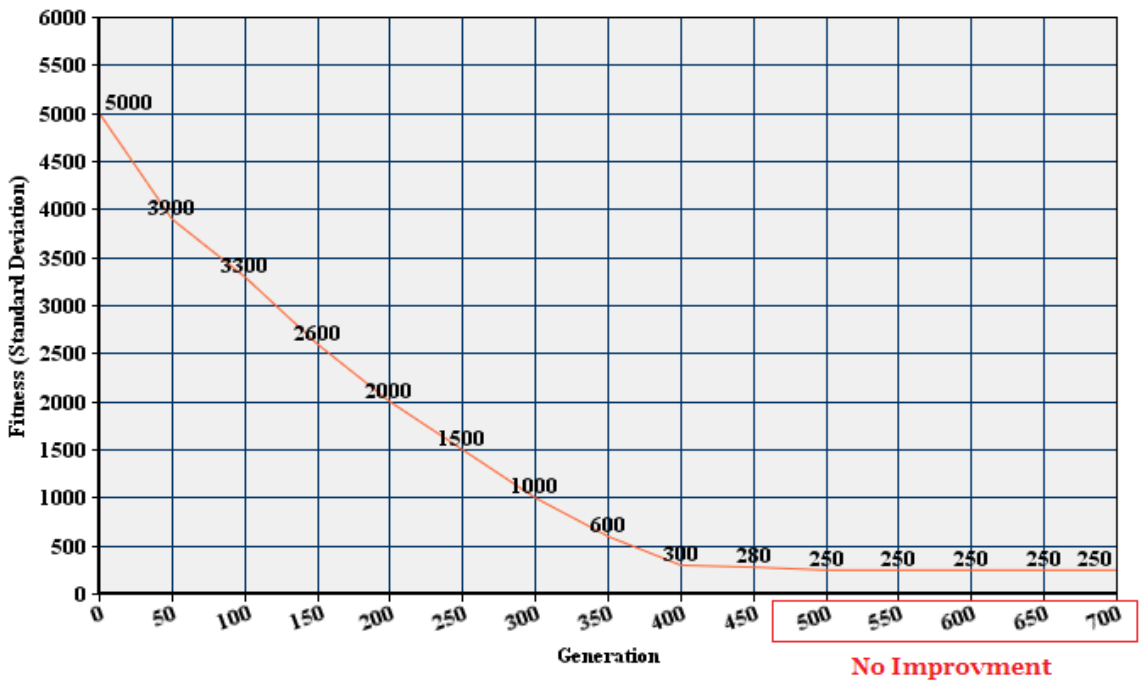

Figure 3.6: *Stagnation Termination*

Algorithm 10 Termination Algorithm

Inputs:
  $generationNumber$ # generation number
  $generationLimit$ # limit of generations without improvement
$fitness \leftarrow$ getFitness();
if ($generationNumber = 0$) || hasFitnessImproved($fitness$) **then**
  $fittestGeneration \leftarrow generationNumber$
end if
if ($generationNumber - fittestGeneration \geq generationLimit$) **then**
  return $true$
else
  return $false$
end if

Decoding and Visualizing

Ultimately, when the termination criteria is satisfied, the algorithm stops and returns the best solution as an array list. Each item in the array list is a specific object that we have already defined it as territory genomes in figure 3.3. The territory genome determines a basic area which belongs to a certain sales representative. At this step, we have to decode our result such that can be visualized in the user interface. The algorithm 11 describes how the array list will be decoded:

Algorithm 11 Decoding Final Result

Inputs:
  $T = [genome_1, \ldots, genome_N]$ # T: candidate solution as an array list
  $S = [s_1, s_2, \ldots, s_N]$ # S: set of salesmans
Initialize:
  $Tl \leftarrow$ size of T # Tl: length of solution array
  $Sl \leftarrow$ size of S # Sl: number of salesmans
for i = 1 to Sl **do**
  $Poly \leftarrow$ empty array list # creating empty polygon list for salesman i
  for j = 1 to Tl **do**
    $Genome \leftarrow T[j]$
    $S\_id \leftarrow$ get S_id from $Genome$
    $G\_id \leftarrow$ get G_id from $Genome$
    if $S\_id = S[i]$ **then**
      $Poly \leftarrow$ ($Poly$ + get polygon from $Genome$) # adding SCU into territory i
    end if
  end for
  Draw($Poly$) # drawing the generated territory on the map
end for

4 Implementation

In this chapter, we are going to illustrate how the system is implemented. In the first section, we present an overview of the system structure and describe the system workflow. Secondly, the database schema will be shown. Then the main core of our decision support system is described. Ultimately, the graphical user interface will be presented.

4.1 System Architecture and Workflow:

Fig. 4.1 shows an overview of the system architecture:

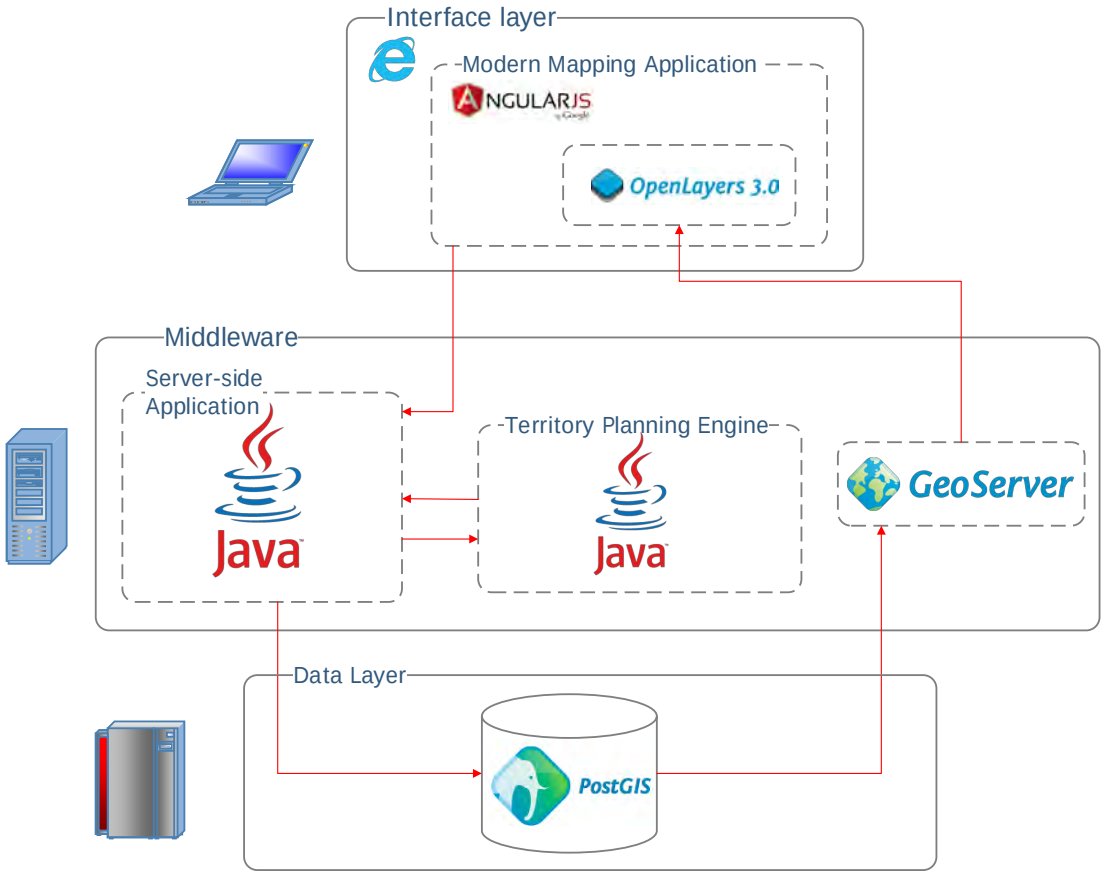

Figure 4.1: *System Architecture*

As the figure 4.1 shows, our system architecture consists of three different layers. We have designed a web application for running the territory planning engine and also for visualizing the result. As the first layer, our web application consists of a client-side application and a server-side application. The client-side application has been developed based on *AngularJS* framework [1]. A map component is also embedded into the client-side application by using the *OpenLayers3* library [10] to visualize the result of the territory planning process on the map. The users (decision makers or marketers) send a request to the server via our client-side application to start the territory planning process. The second layer contains a server-side application which has been written in the Java Programming Language. We have chosen the java programming language because of its good performance that is almost equal to native programming languages such as *C* and *Fortran* [7]. The server application reads all needed data from the database (e.g. distance matrix, location of sales people, basic areas and etc) and passes it to the territory planning engine which is also included in the second layer. The territory planning engine has been developed as an independent module which gets some inputs and returns a java object as the result. After starting the engine by the server-side application, all data should be loaded on the main memory (RAM). When the planning algorithm is terminated, the engine sends the result back to the server-side application and the result will be inserted into the database 4.2 via SQL queries. In addition, we used the *GeoServer* for visualizing our geospatial data such as basic areas, sales territory structures and etc. The embedded map component in our user interface sends a WMS request to the geoserver. Then the geoserver reads the data from the database and sends back the generated map to the client-side application as a PNG file. Our dataset consists of some spatial data about municipalities, states, cities, roads network, districts, etc. which are stored as geometry in our database. Moreover, we have additional data related to our case study that we explain later in the next chapter.

4.2 Database Design

In this thesis, we used *PostgreSQL* as the database management system [12]. In addition, we used *PostGIS* as a spatial extension for our database [11]. We store all needed data for territory planning in the database. Our basic areas for our case study are all municipalities of Germany. Since the calculation of travel time and distance in the scale of whole Germany during the processing is not efficient and needs much resources, we have calculated the travel time and distance in advance. The travel time is calculated from the center of each municipality to the center of all others based on the roads network by mean of the ArcGIS Network Analyst extension. This extension allows users to build a network dataset and perform analyses on a network dataset [2]. The result of travel time calculation has been inserted into the distance table in our database. This table contains approximately 120 million rows which have been indexed by BTREE [3] to be retrieved quickly. The figure 4.2 illustrates our database schema. The structure of territories is stored in the tables *territory_initialized* (before optimization) and *territory_optimized* (after optimization). These tables only consist of a pair of IDs. The first ID (*municipaliy_id*) identifies the municipality and the second ID (*salesman_id*) shows the sales representative who is responsible for the municipality. All municipalities that are assigned to a specific salesman ID make a territory

of that salesman.

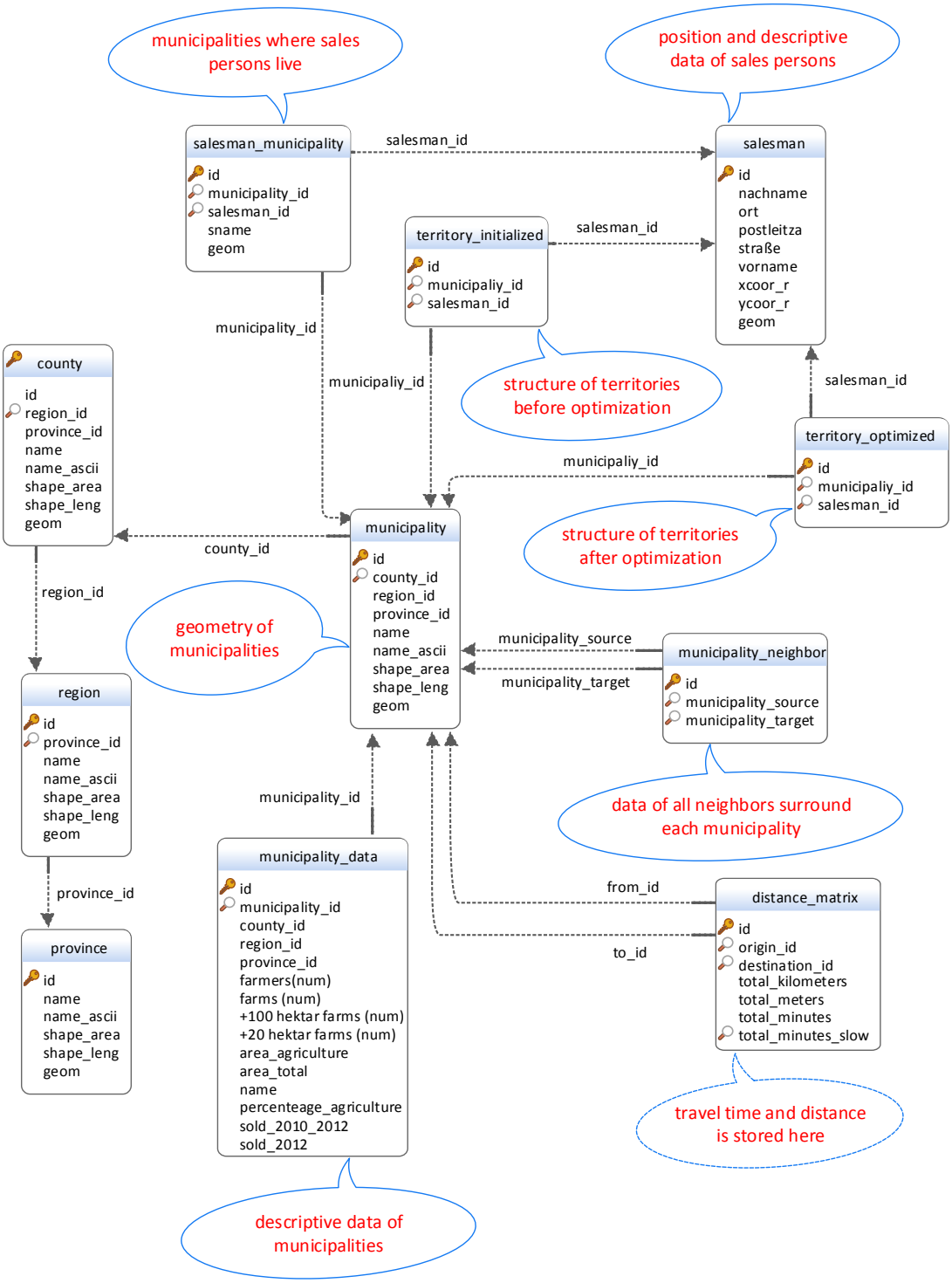

Figure 4.2: *Database Schema*

Since we need to consider the neighbors of each municipality in our algorithm 8, we have calculated the neighbors of each municipality by using the *st_intersects()* function of PostGIS and store the result in the table *municipality_neighbor*. In our database schema, the tables which have the *geom* column such as *municipality*, *county*, *region* and *province* are mostly used for visualization of the result. But we use the data of the municipality_data table directly in our algorithm.

The table *municipality_data* is the most important data table which we used as the main input for our algorithm. Although our algorithm is independent from the input and the objective function, any descriptive data can be used for the optimization process, but we have taken a simple agricultural data of municipalities into account for the optimization of territories according to our case study which will be explained in the next chapter.

4.3 The Territory Planning Engine

As we discussed before, territory planning involves great computational complexity because of a large number of alternative solutions. This complexity can be expanded when the objectives contain some specials constrains such as contiguity and compactness. Because the number of basic areas involved in the sales territory planning is high in our case study, we need a heuristic approach like the genetic algorithm to find the optimal solution. On the other hand, GA is a non-deterministic population-based search algorithm. Because of exploring the whole search space, the calculation process often takes long time depends on the size of search space. Thus, for finding the optimal solution in a reasonable time, the territory planning engine has been developed based on the multi-core parallelism concept by using the standard concurrent programming package included in the java platform. It means that we split the population of candidate solutions into several groups. Then each group will be assigned to the one core (CPU). Each CPU acts like a worker which performs the genetic operations on a certain group of solutions. But the parallel processing is not possible for all operations. In the other words, some operations cannot be executed concurrently, because the whole population should be taken into account at the same time. For instance, our selection operation cannot be performed concurrently. Because for the selection process, each candidate solution should compete with all other candidates to be selected for the next generation. The figure 4.3 presents our parallelism approach.

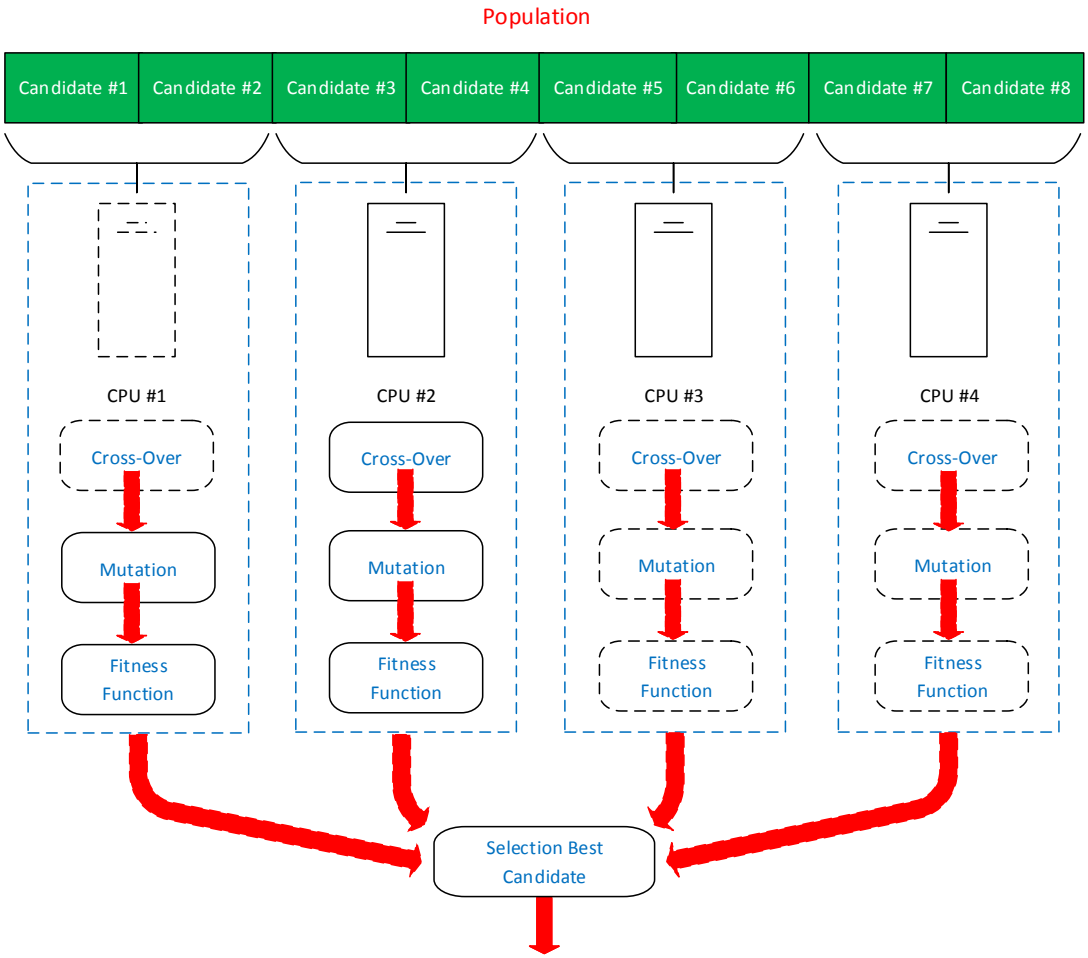

Figure 4.3: *Parallelism Approach*

4.4 Graphical User Interface

As it is shown in the figure 4.1, we have prepared a simple browser-based user interface for interaction with territory planning engine and also for visualizing raw data and the generated sales territories. We used AngularJS as a MVC framework to implement client-side application. The core idea behind MVC is that you have clear separation in your code between managing its data (model), the application logic (controller), and presenting the data to the user (view) [17]. The view gets data from the model to display to the user. When a user interacts with the application by clicking or typing, the controller responds by changing data in the model. Finally, the model notifies the view that a change has occurred so that it can update what it displays. We used this feature to update our map component when the territory planning process is finished and the result should be visualized. For instance the client-side application always checks for any updates in the

database concerning the result of territory planning and visualizes the structure of generated territories as soon as it is found by the genetic algorithm. Fig. 4.4 below show our simple interactive user interface (client-side application):

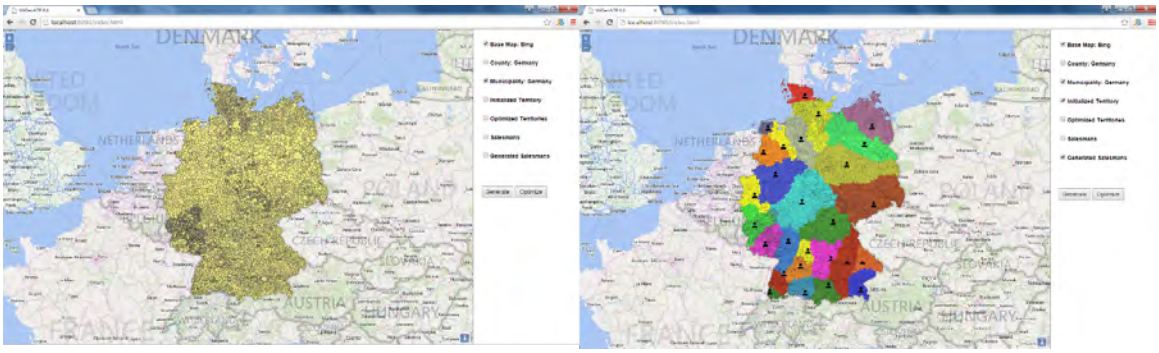

Figure 4.4: *Client-side application*

5 Results and Analysis of Experiment

To assess the correctness and effectiveness of our method, we performed some experiments for both evaluating the balance of potential among sales representatives and evaluating the travel time improvement. The benchmarks have been done on a system with Intel Core i7-4770 3.40 GHz (Quad core CPU), 8 Gigabytes of Memory (RAM) and on Hard Disk with 7200 RPM.

Our step by step experiment starts by a brief explanation of the dataset we used. In section 5.1.2 we setup the parameters of our genetic algorithm by testing all possibilities and choosing the best parameters. In section 5.1.3 we explained our innovative method to find the best position for locating sales representatives in the first scenario (Green Field Planning). In section 5.1.4 and 5.1.5 we compared the results of our method before and after optimization. Then we checked contiguity and compactness constraints in section 5.1.6. We finally discussed the obtained results in section 5.2.

5.1 Experiment

5.1.1 The dataset

As the case study, we considered a situation where an agricultural machinery manufacturer is selling a certain product (e.g. tractor) in Germany. The sales team consists of 27 representatives who must cover the whole Germany. Each sales representative is supposed to be responsible for only one territory. Territory planning is carried out in the level of municipalities which include 11,212 basic areas as SCUs. Although our method is completely independent from inputs and objectives, we considered the number of farms as the available potential inside each municipality. The more the number of farms inside the SCU, the higher is the chance of selling the agricultural machinery. However, the calculation of the available potential can be more complicated by taking more inputs into account such as percentage of agricultural land, number of farmers, purchasing power, number of sold machines and etc.

5.1.2 Genetic Algorithm Parameters Setup

The Genetic Algorithm is very sensitive to parameters such as population, elitism percentage, mutation rate (probability) and cross-over rate (probability). After several experiments, we investigated the best configuration for setting up the Genetic Algorithm. Since the calculation time depends on the specifications of the computer on which the planning is being done, we tested the parameters based on the number of generations. According to

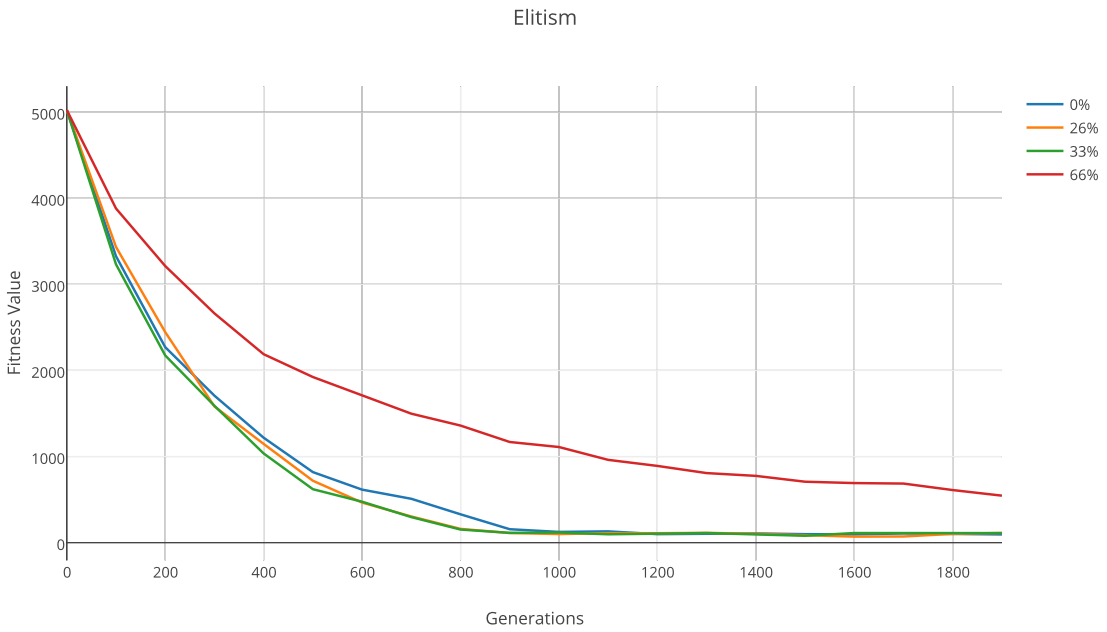

Figure 5.1: *Comparing elitism rate*

the case study, our goal is to decrease the difference of potential values among salesmen. Thus, we considered the standard deviation of each solution as the fitness value which should decrease after several generations. In this case, the less fitness value addresses better solution.

For the first step, we evaluated four different elitism rates. The elitism rate identifies the percentage of individuals that should be transferred to the next generation without any changes. As depicted in figure 5.1, the fitness value has progressively decreased after some generations. But the decrease of the fitness value is faster when the elitism rate is set to 26%.

As the second parameter, we investigated the probability of the cross-over operation. We compared three cross-over probabilities. The result is shown in figure 5.2. Although the progress of all three cross-over probabilities is almost same, the cross-over rate of 0.9 (green line) has a better effect on the performance of the algorithm.

The Genetic Algorithm is very sensitive to the mutation rate. After several experiments we realized that very high and very low mutation rates can lead the genetic algorithm to the local optimum. As it is shown in the figure 5.3, the mutation rate of 0.5 (orange line) extremely decreased the fitness value after 400 generations, while other mutation rates are not as efficient as the mutation rate of 0.5.

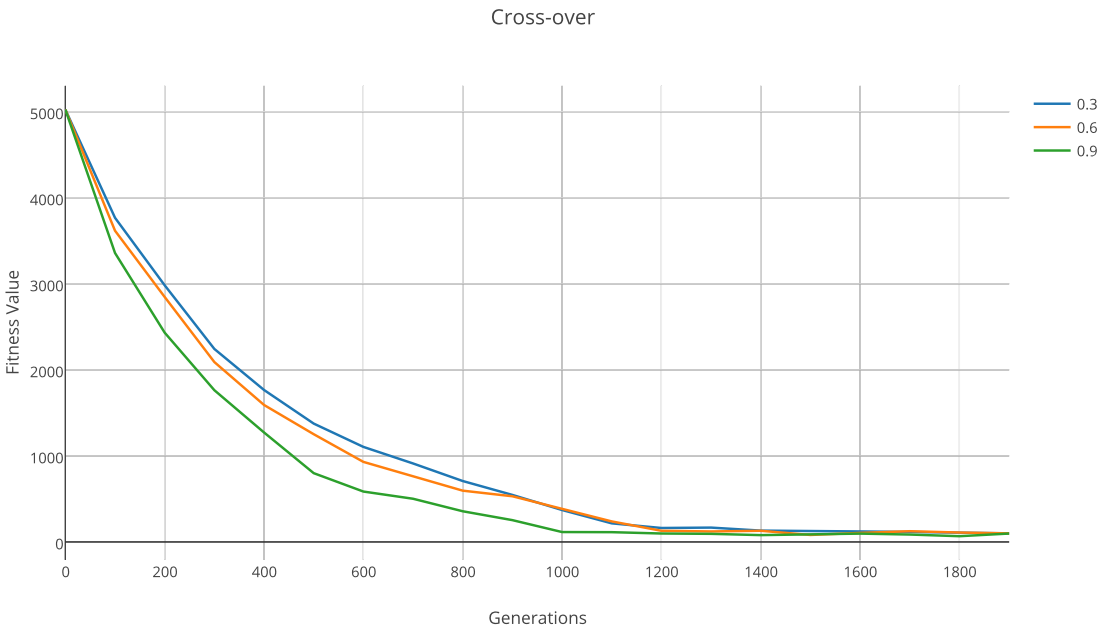

Figure 5.2: *Comparing Cross-over rate*

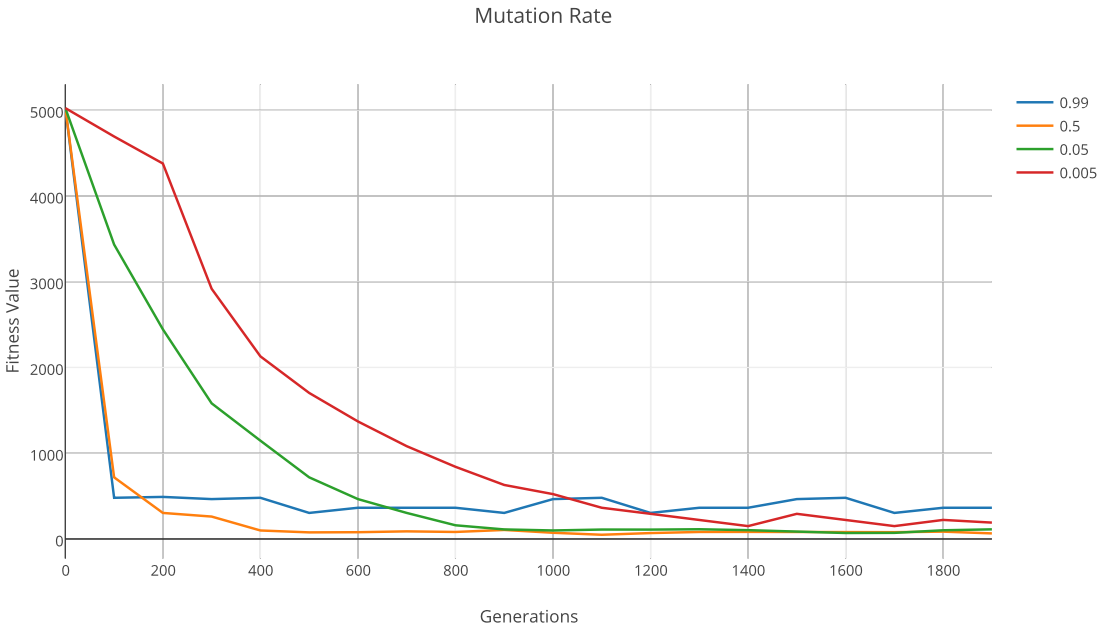

Figure 5.3: *Comparing mutation rate*

5.1.3 Evaluating the Location-Allocation and Initializing the Territories

As we explained in section 3.1.2, we have three scenarios. Since the first scenario (*Green Field Planning*) includes other scenarios, we evaluated our method only for the *Green Field Planning*.
We defined three steps for solving the Green Field Planning problem. Firstly, we found the best positions for locating salesmen. Then we created the first version of territories by using the clustering algorithm 2 based on the travel time from the salesman's living location to the center of municipalities. According to clustering algorithm, we considered each municipality and assigned it to the nearest salesman. The location of salesmen and generated territories in algorithm 2 are temporary and used as input for the next step (Genetic Algorithm).

Since the living location of salesmen is an input for sales territory planning, finding the best location for sales teams is crucial. As the location-allocation problem is not the main focus of this thesis, we designed a relatively simple algorithm to find the proper locations for the sales team. In this step, we selected the capital municipality of each district (landkreis), because these capital municipalities have enough requirements such as accessibility, amenities and etc. for locating the sales team. Thus our candidates locations are limited to 402 municipalities that are equal to the number of districts (German Landkreis). The figure 5.4 illustrates all candidate municipalities and their accessibility.

After several experiments, we have realized that finding a proper location for our sales representatives extremely improves the performance of the territory planning algorithm. The reason is that the existing potential is not distributed homogeneously in the whole Germany. As it is shown in figure 5.5, in our case study, the available potential value (number of farms) of the municipalities are aggregated to district level and the districts are classified into 5 classes from red (poor area) to green (rich area). The eastern part has less potential than western or southern part of Germany. It means that eastern territories should be much larger than southern territories to have almost equal potential. To solve this problem, we first used the Natural Breaks algorithm which is a method of manual data classification that seeks to partition data into classes based on natural groups in the data distribution. In the second step, we assigned a weight in the range of 1 to 5 to each class based on its value. Then we calculated the capacity of each district class (number of salesman that can be located in the class) according to its weight. For instance, 140 districts belong to *Class1* and only one sales representative can be located in this class. On the other hand, 26 districts belong to *Class5* and 10 sales representatives can be located in these districts.
As it is shown in figure 5.6, eastern territories are much larger than southern territories.

5.1.4 Evaluating the Balance of Potential among Salesmen

Since balancing the potential or the workload among sales representatives (territories) is one of the main objectives in the sales territory planning, we want to prove the efficiency of our algorithm for this objective. After initializing the structure of territories, the potential distribution is not balanced. Thus, the standard deviation of the potential is relatively

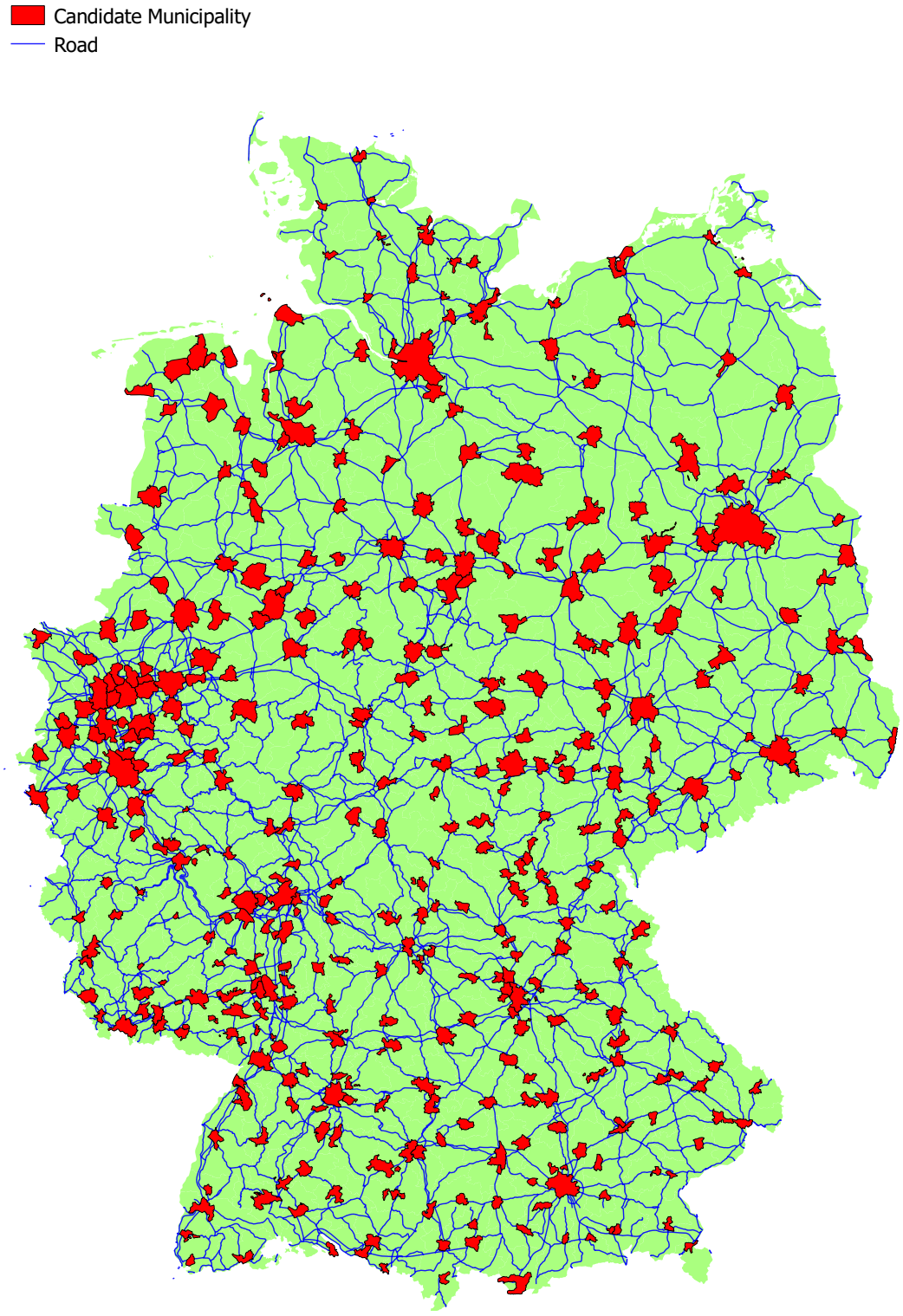

Figure 5.4: *Accessibility of candidate municipalities*

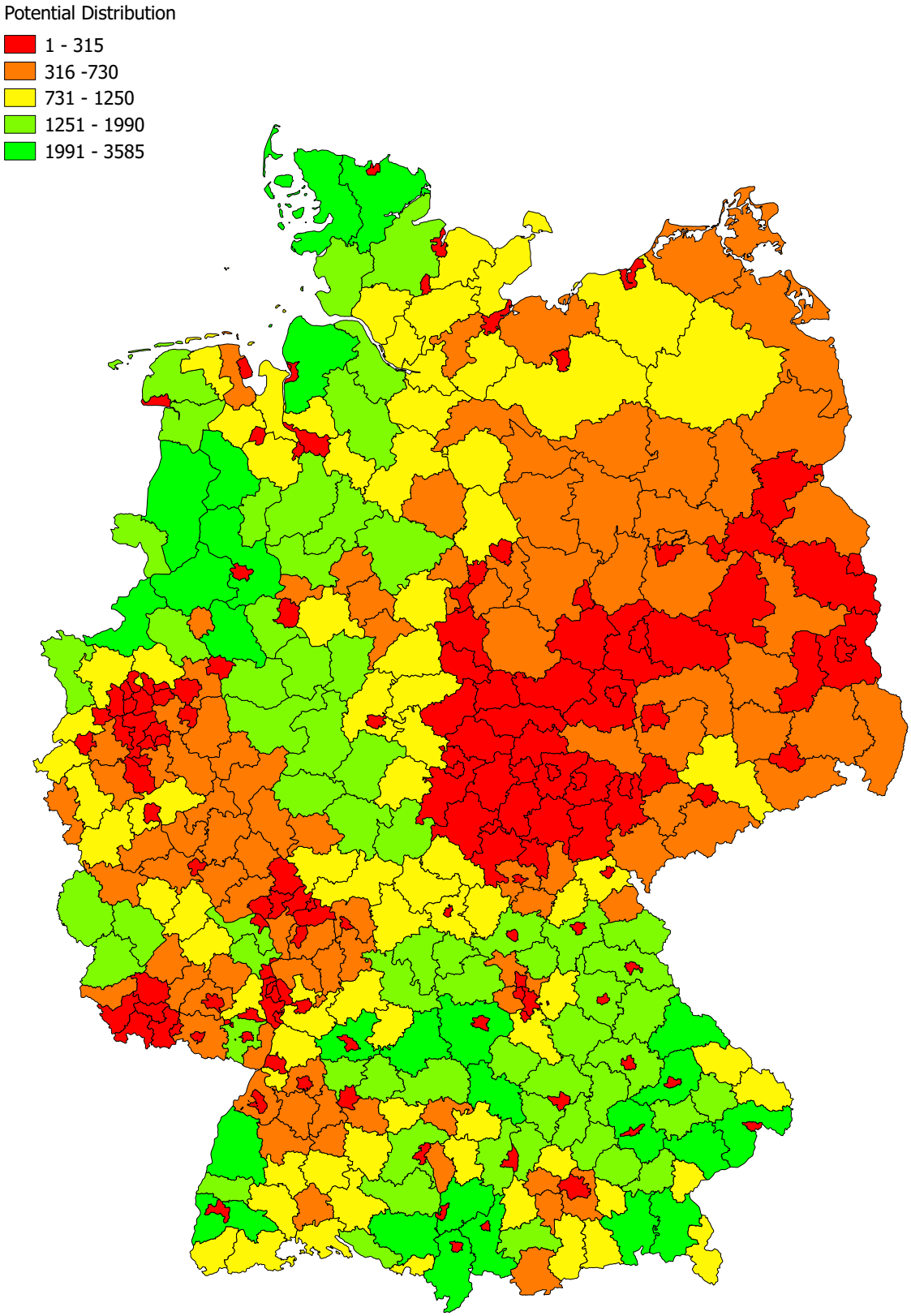

Figure 5.5: *Distribution of potential (number of farms)*

Class	Range of potential value	Weight	Candidate districts	Capacity(number of salesmen)
Class 1	1 - 315	1	140	1
Class 2	316 - 730	2	96	3
Class 3	731 - 1250	3	75	5
Class 4	1251 - 1990	4	65	8
Class 5	1991 - 3585	5	26	10
Total			402	27

Table 5.1: Classification of districts based on potential value

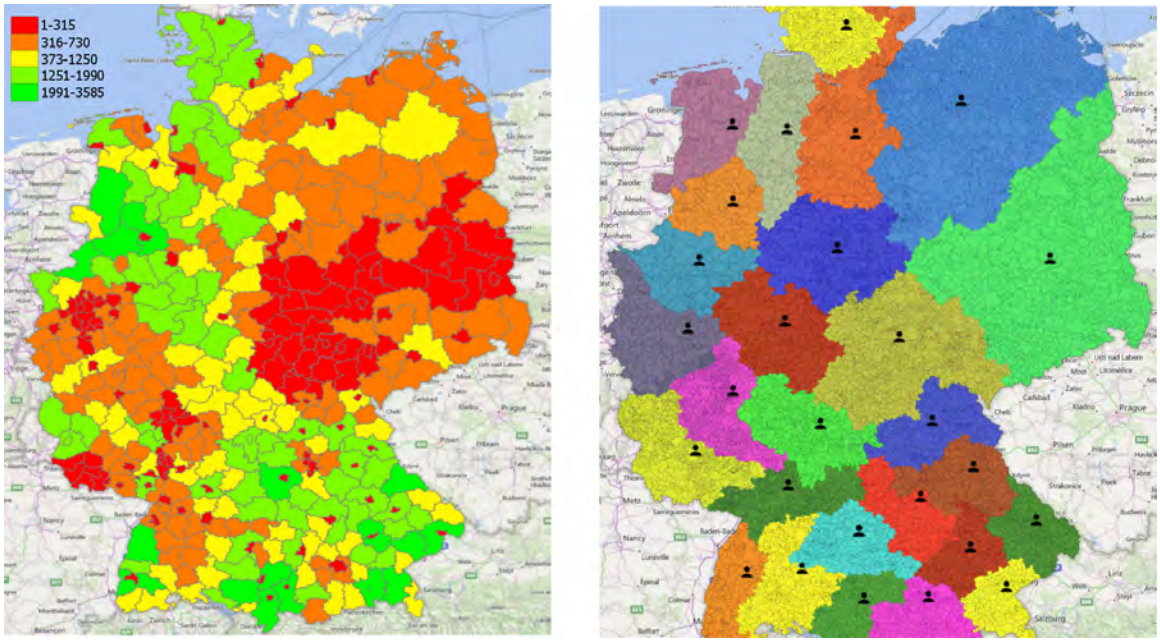

Figure 5.6: *Distribution of Potential*

Salesman	Potential Value	Deviation From Mean	Mean
person #1	6202	-4875.62	11077.62
person #2	16646	5568.38	11077.62
person #3	8002	-3075.62	11077.62
person #4	12349	1271.38	11077.62
person #5	11666	588.38	11077.62
person #6	13153	2075.38	11077.62
person #7	9983	-1094.62	11077.62
person #8	14269	3191.38	11077.62
person #9	14776	3698.38	11077.62
person #10	11978	900.38	11077.62
person #11	22938	11860.38	11077.62
person #12	8744	-2333.62	11077.62
person #13	17345	6267.38	11077.62
person #14	14888	3810.38	11077.62
person #15	11449	371.38	11077.62
person #16	14291	3213.38	11077.62
person #17	8034	-3043.62	11077.62
person #18	9353	-1724.62	11077.62
person #19	5893	-5184.62	11077.62
person #20	3963	-7114.62	11077.62
person #21	9625	-1452.62	11077.62
person #22	11488	410.38	11077.62
person #23	6205	-4872.62	11077.62
person #24	10608	-469.62	11077.62
person #25	9299	-1778.62	11077.62
person #26	9555	-1522.62	11077.62
person #27	6394	-4683.62	11077.62
Total Potential	299096		

Table 5.2: Potential Distribution Before Optimization

high. Table 5.2 represents the potential distribution among salesmen after initializing the territories and before running the optimization process.

Figure 5.7 also represents the deviation of potential value from the mean in each territory. The numbers on the horizontal axis show the ID of salesmen in table 5.2.

After initializing the territories, we must balance the potential among the territories according to the defined objective. It means that the rich territory (high potential territory) should shrink and the poor territory should expand. We have run the optimization process and it has finished after 369 seconds which is a promising computational time for the scale of our problem. Figure 5.8 represents the progress of our algorithm in which the standard deviation has progressively decreased during the time until improvement is not possible any more.

Table 5.3 and Figure 5.9 show the result of our optimization algorithm.

As it is shown in figure 5.9, the generated territories are relatively balanced. After bal-

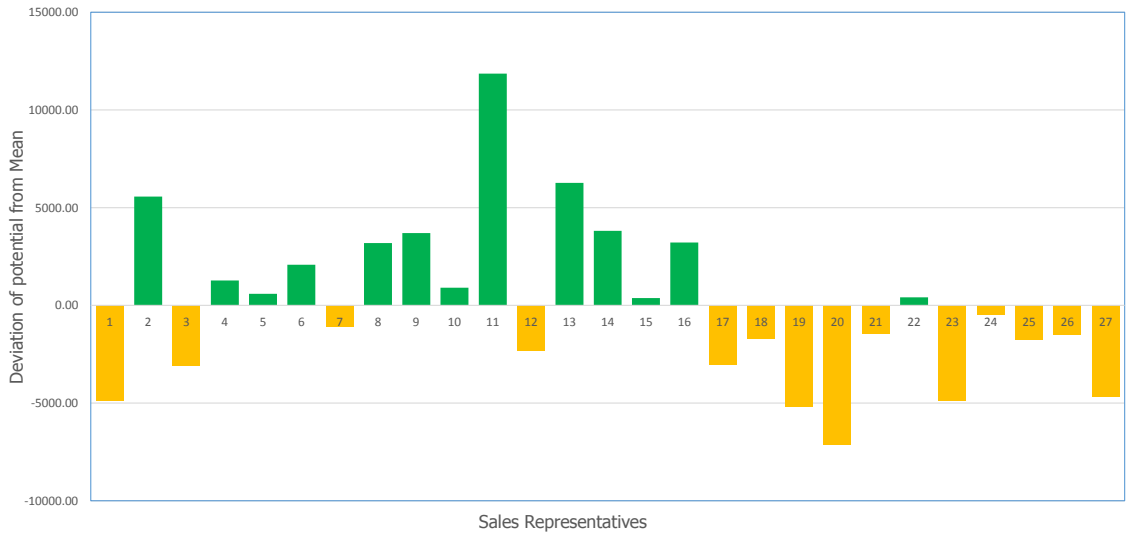

Figure 5.7: *Before Optimization*

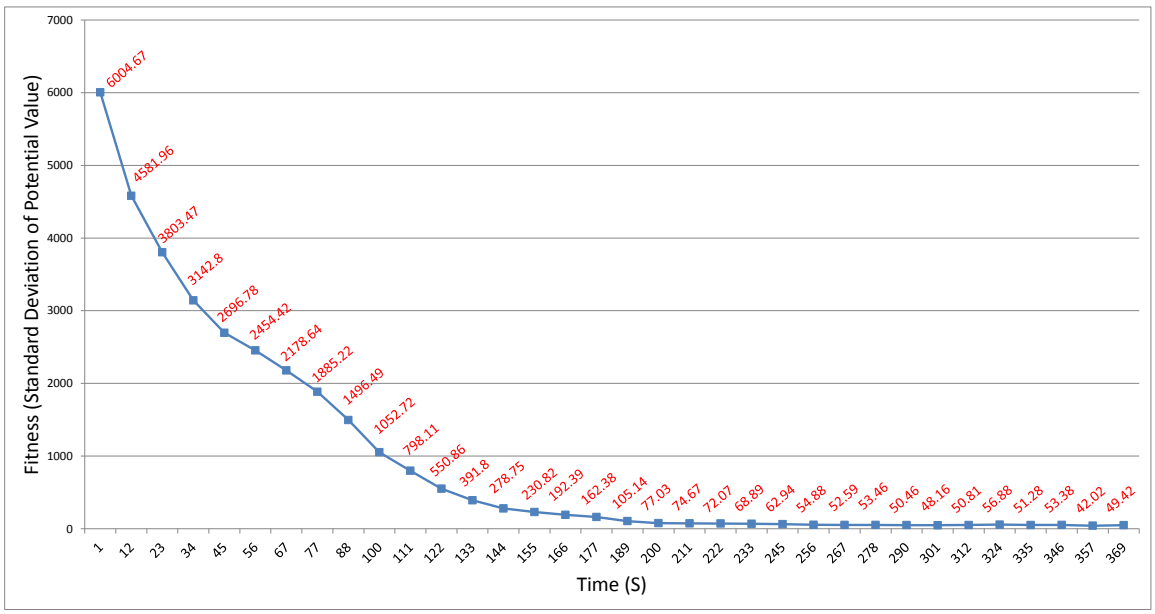

Figure 5.8: *The Progress of Genetic Algorithm*

Salesman	Potential Value	Deviation From Mean	Mean
person #1	11102	24.38	11077.62
person #2	11050	-27.62	11077.62
person #3	11025	-52.62	11077.62
person #4	11080	2.38	11077.62
person #5	11004	-73.62	11077.62
person #6	11127	49.38	11077.62
person #7	11046	-31.62	11077.62
person #8	11080	2.38	11077.62
person #9	11097	19.38	11077.62
person #10	11106	28.38	11077.62
person #11	11094	16.38	11077.62
person #12	11124	46.38	11077.62
person #13	11086	8.38	11077.62
person #14	11074	-3.62	11077.62
person #15	11048	-29.62	11077.62
person #16	11121	43.38	11077.62
person #17	11102	24.38	11077.62
person #18	11183	105.38	11077.62
person #19	11072	-5.62	11077.62
person #20	11031	-46.62	11077.62
person #21	11067	-10.62	11077.62
person #22	11071	-6.62	11077.62
person #23	10945	-132.62	11077.62
person #24	11052	-25.62	11077.62
person #25	11145	67.38	11077.62
person #26	11020	-57.62	11077.62
person #27	11144	66.38	11077.62
Total Potential	299096		

Table 5.3: Potential Distribution After Optimization

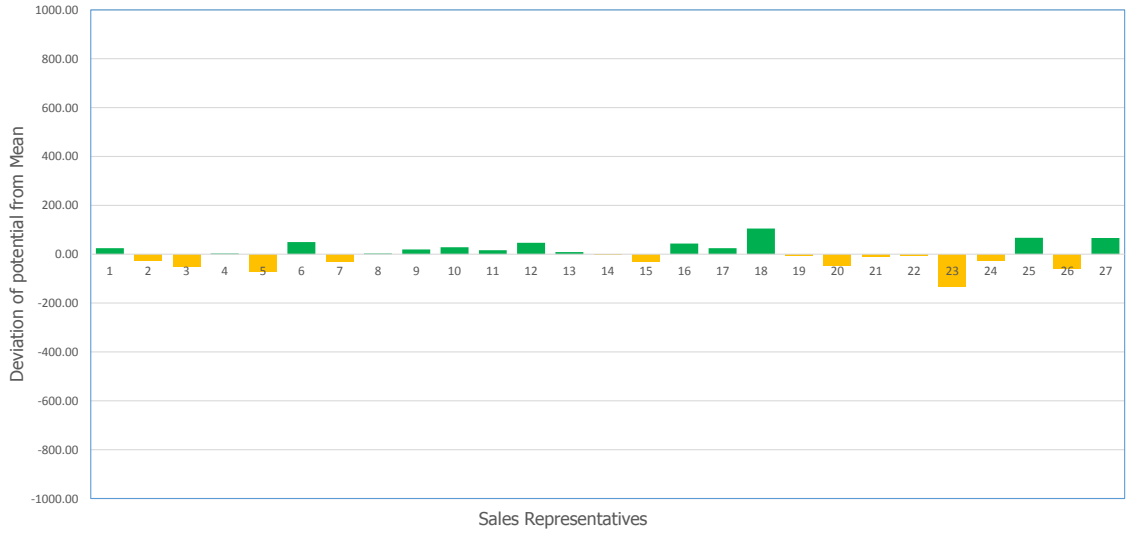

Figure 5.9: *After Optimization*

ancing the potential between territories, the location of sales representatives should be updated according to the new structure of territories, because the territories have been resized or moved. Thus ,we use the algorithm 1 to find the best location for the sales persons based on the current structure of territories. Figure 5.10 clearly illustrates the structure of territories before and after the optimization.

It is also obvious that the number of territories in the eastern part of Germany is low, but these territories are larger than others. On the other hand, the number of territories in the southern part and the western part of Germany is high, but these territories are smaller than in the eastern territories. The reason is that the distribution of potential is not equal in the whole Germany. We will discuss this in section 5.2.

5.1.5 Evaluating the Travel Time Improvement

Since the sales territory planning is classified as a multi-objective problem, we developed a method which is capable to fulfill more than one objective simultaneously. As the travel cost is one of the important parameters that must be taken into account in sales territory planning, we have defined the minimizing of travel time and distance as an objective in our algorithm beside the balancing of potential. In our case study, minimizing the travel cost is not the first priority in comparison to balancing the potential. So we defined a lower weight for minimizing the travel cost than balancing the potential among salesmen.

The the calculation of travel cost depends on several parameters and can vary among different sales organization. For instance, a salesman visits the potential customers in several neighbor municipalities in one journey and return to his living location (figure 5.11). But another salesman only visits one municipality in each journey (figure 5.12).

In our algorithm, we assumed that the salesmen only visit the potential customers of one municipality in each journey. In addition, we did not take the empty municipalities into

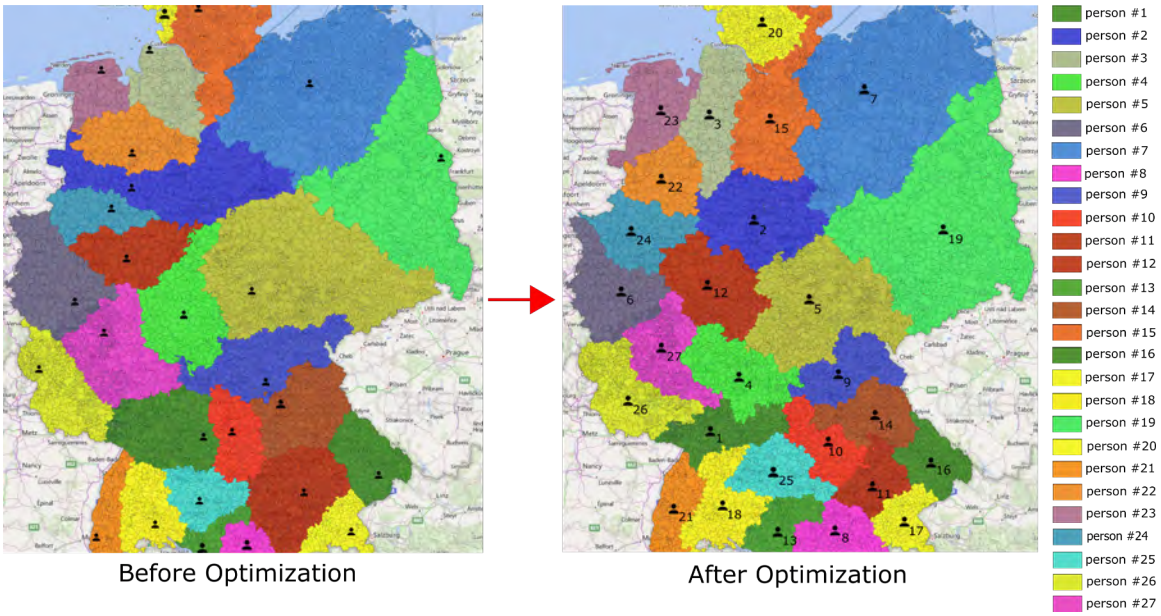

Figure 5.10: *Territories Before and After Optimization*

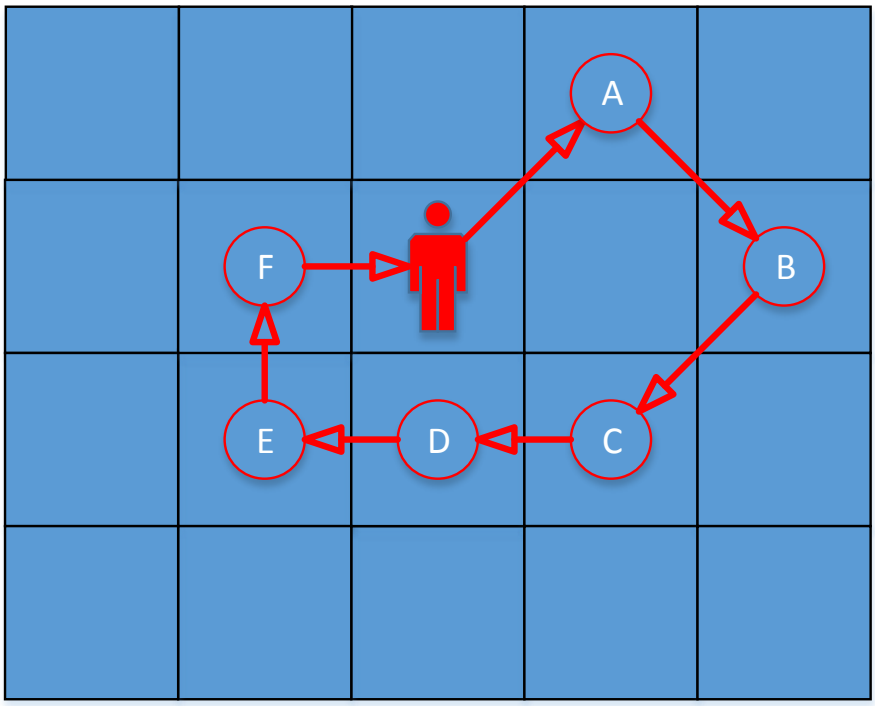

Figure 5.11: *Visiting several municipalities in one journey*

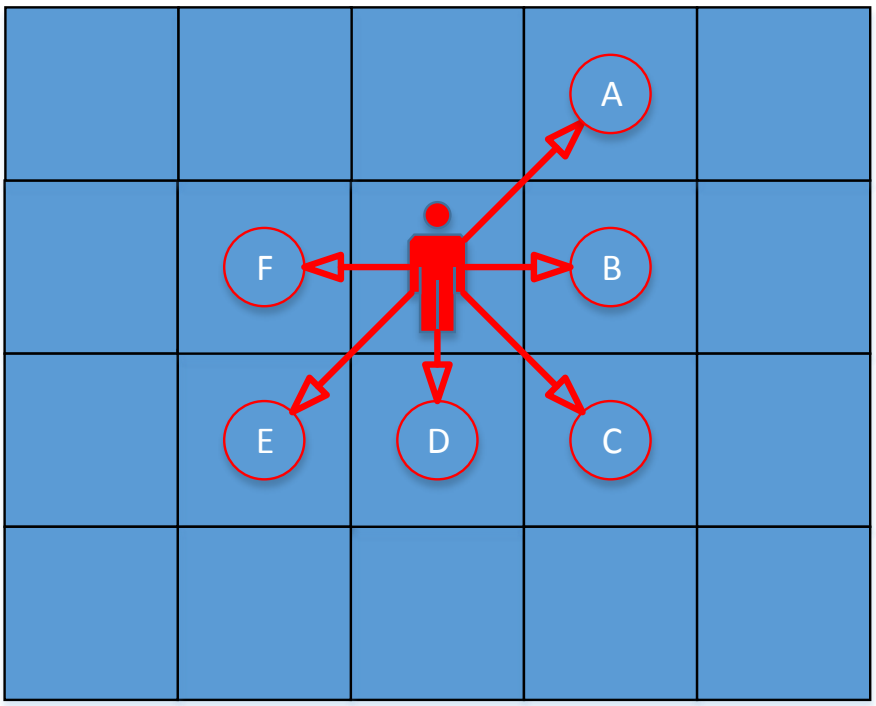

Figure 5.12: *Visiting one municipality in one journey*

account. In other words, the salesman only visits the municipalities where the potential customers exist. So the total travel cost for each salesman is the sum of all travel costs from the living location to the municipalities (not empty) belonging to the salesman. Table 5.4 shows, the total travel time before and after the optimization process, which has decreased around 2539 hours. For example, if the company pays 10 € per hour as the travel cost, it will save $25,390$ € after optimization.

In our case study, we only consider travel time as the cost of travel, but we also show the improvement in travel distance in table 5.5.

5.1.6 Evaluating the Contiguity and Compactness

In addition to the balance potential and the minimum travel cost, we also have other objectives which are round-shape territories (compactness) and contiguity. This is why we performed the second steps (clustering). Although we can also guarantee the contiguity and compactness in the Genetic Algorithm by defining some constraints, it extremely affects the performance of the algorithm. Because our search space is too big, the chance of finding a solution which does not violate our constraints is very low. Thus, we firstly create a temporary structure of territories which are compact and totally connected, then we start balancing the potential among salesmen from the borders of territories. In other

Salesman	TravelTime Before Optimization(Hour)	TravelTime After Optimization(Hour)
person #1	45.8	112.6
person #2	2177.9	2767.00
person #3	2423.4	1775.7
person #4	359.5	126.8
person #5	479.6	169.5
person #6	732.0	525.6
person #7	212.6	204.2
person #8	748.8	533.1
person #9	440.7	256.6
person #10	1093.2	1086.8
person #11	137.7	217.8
person #12	639.6	466.8
person #13	157.7	140.9
person #14	1402.2	590.7
person #15	738.2	631.6
person #16	124.7	398.6
person #17	1513.6	464.5
person #18	467.2	257.2
person #19	203.07	144.3
person #20	97.5	148.2
person #21	24.4	195.2
person #22	415.04	711.2
person #23	78.9	83.6
person #24	201.6	314.6
person #25	108.4	137.6
person #26	542.3	481.0
person #27	164.2	248.2
Total TravelTime	15730.975	13191.324

Table 5.4: TravelTime Optimization

Salesman	Distance Before Optimization(KM)	Distance After Optimization(KM)
person #1	3614.013	8751.412
person #2	161389.439	200823.250
person #3	173538.811	122370.882
person #4	25927.688	8179.686
person #5	34006.933	11248.511
person #6	55277.859	39641.800
person #7	14218.206	13724.962
person #8	59266.471	41007.961
person #9	35299.776	19322.427
person #10	78467.847	77605.997
person #11	8610.650	13765.838
person #12	50804.065	37313.380
person #13	11100.342	9550.289
person #14	111448.792	43759.152
person #15	46821.120	38567.696
person #16	8163.978	25921.241
person #17	102443.046	30819.484
person #18	34742.107	16913.352
person #19	13975.581	10352.322
person #20	6142.961	8977.485
person #21	2015.275	14040.252
person #22	26066.628	45305.106
person #23	5100.494	5157.610
person #24	16161.968	21007.215
person #25	7611.667	9693.158
person #26	42971.005	35676.592
person #27	11828.277	17458.074
Total Travel Distance	1147015.013	926955.1481

Table 5.5: Travel Distance Optimization

words, we randomly select some municipalities on the border of territories and assign them to the neighbor territories. If this operation decreases the standard deviation value, we will transfer the new structure of territories as a good solution to the next generation. Moreover, we defined some operations to detect and reject the invalid solutions to keep compactness and contiguity of the temporary structure of territories. We always guarantee the compactness of territories by minimizing the travel time which is also one of our main objectives.

5.2 Analysis of Results

In this chapter we will discuss the results of our evaluation in general.

5.2.1 Location-Allocation and Initializing the Territories

After several experiments, we have realized that finding a proper location for our sales representatives extremely improves the performance of the territory planning algorithm. The reason is that the existing potential is not distributed homogeneously in the whole Germany. As it is shown in figure 5.5, in our case study, the southern part of Germany has more potential than other parts. While the eastern part is relatively empty of potential customers. It means that eastern territories should be much larger than than southern territories to have almost equal potential among salesmen. For instance, if the number of sales persons are more than the capacity in the eastern part, it will make much effort for the optimization algorithm to expand, re-arrange and shift the generated territories to find out the balance structure of territories. As it is shown in figure 5.6,the eastern territories are much larger than the southern territories. In other words, the southern part of Germany has been divided into several territories, because it has a high density of farms. Furthermore, big territories do not mean much more travel cost than small territories, because in big territories there are a lot of municipalities that are empty. In these territories, salesmen do not travel to those municipalities for visiting potential customers.

5.2.2 Balance of Potential among Salesmen

Ranking of Territories

One of the main points we understood during our experiments was that mutating the territories based on their ranking improves the performance of our Genetic Algorithm very much. When the Genetic Algorithm starts to balance the potential value among 27 salesmen, each territory (salesman) has a different number of potential customers depending on the temporary structure of the generated territories in step 2. Since the optimization algorithm is supposed to expand the poor territories (less potential) and shrink the rich territories (high potential), assigning a municipality from a poor territory to the surrounding rich territories in the mutation operation only increases the calculation time of our algorithm. From an algorithmic point of view, the search direction can be guided to the global optimum by excluding useless explorations of search space.

Dynamic Mutation Rate

As the Genetic Algorithm is likely to be trapped in local optima, a dynamic mutation rate helps it to jump from local optima. When the progress of optimization slows down after several generations, high mutation rates can make a big change in the population, thus the algorithm jumps from local optima. As soon as the progress is normal, the mutation rate decreases.

Contiguity and Compactness Constraints

As we discussed in the previous section, we also have to care about the constraints such as contiguity and compactness. The travel time optimization guarantees that each municipality is assigned to the nearest salesman if it fulfills our first objective (potential balancing). This is why the balanced territories will be compact as much as possible. Moreover, our graph tester guarantees contiguous territories after optimization, since the graph tester component makes an undirected graph from all municipalities belonging to a territory and checks the connectivity of this graph. As soon as the graph is disconnected, the algorithm denies the individual solution and does not transfer it to the next generations. Therefore, the mutation process never produces non-contiguous territories. Figure 5.13 shows how the undirected graph inspects the contiguity constraints.

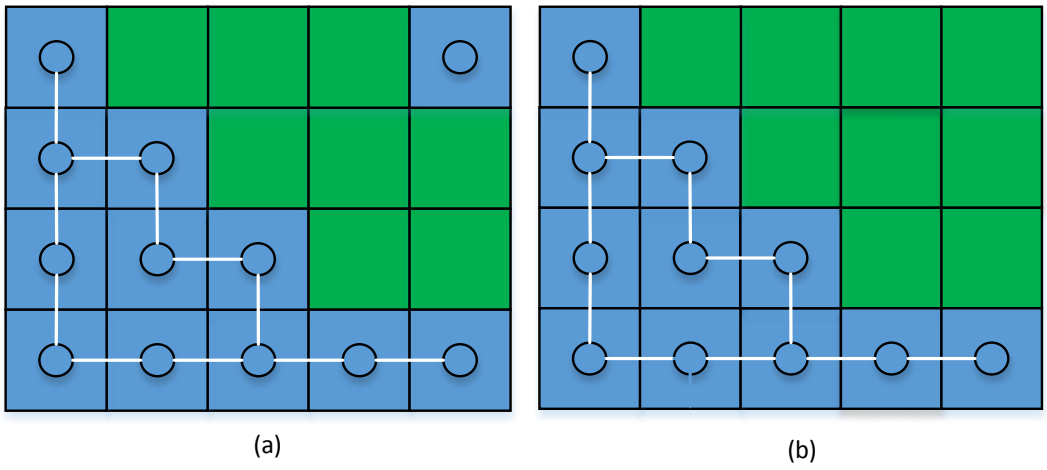

Figure 5.13: *(a) the contiguity constraint for the territory is violated; (b) the territory is totally connected*

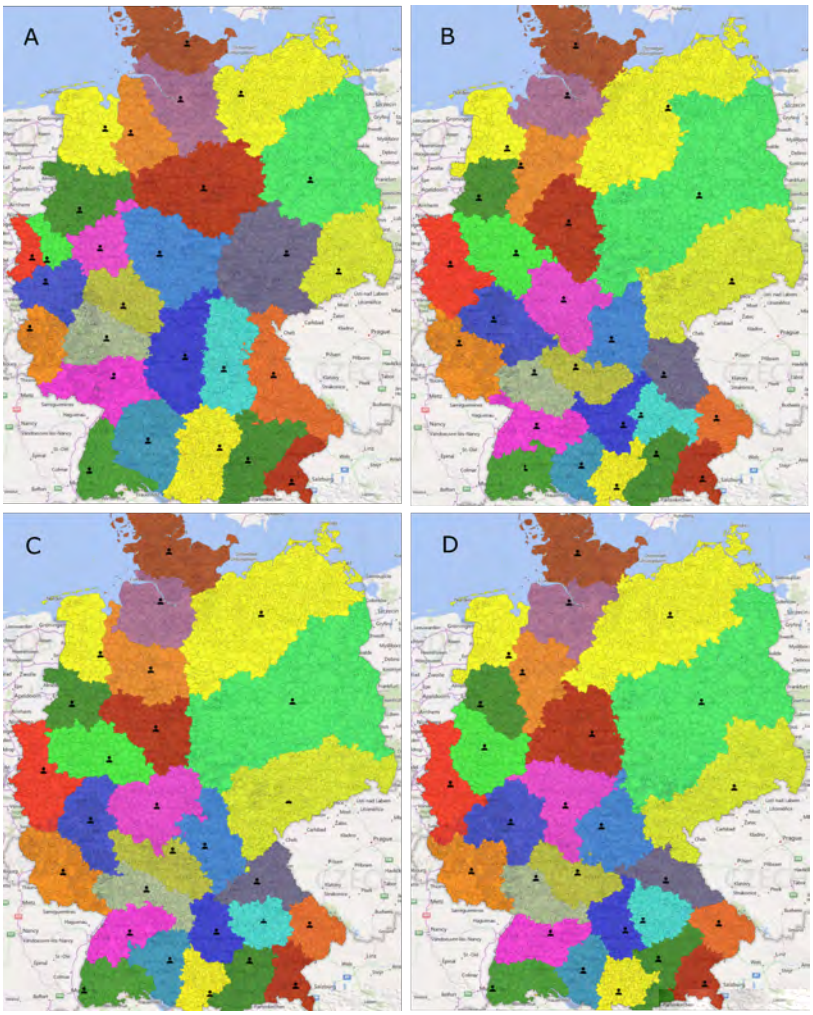

Figure 5.14: *(A) before optimization; (B) first optimization; (C) second optimization; (D) third optimization*

5.2.3 Random Results of Genetic Algorithm

As we explained in chapter 2, Genetic Algorithm is a stochastic optimization method. The Genetic Algorithm explores the whole search space to investigate the global optimum. Since our search space is too big, there are so many possible solution for our problem. If the algorithm discovers the best solution which fulfills our objectives, the optimization process will be terminated. So we performed our optimizations algorithm three time and the results are shown in figure 5.14 and 5.15. Various structures of the big territories in the eastern part of Germany are easily recognizable. Moreover, the southern territories look completely different in each optimization.

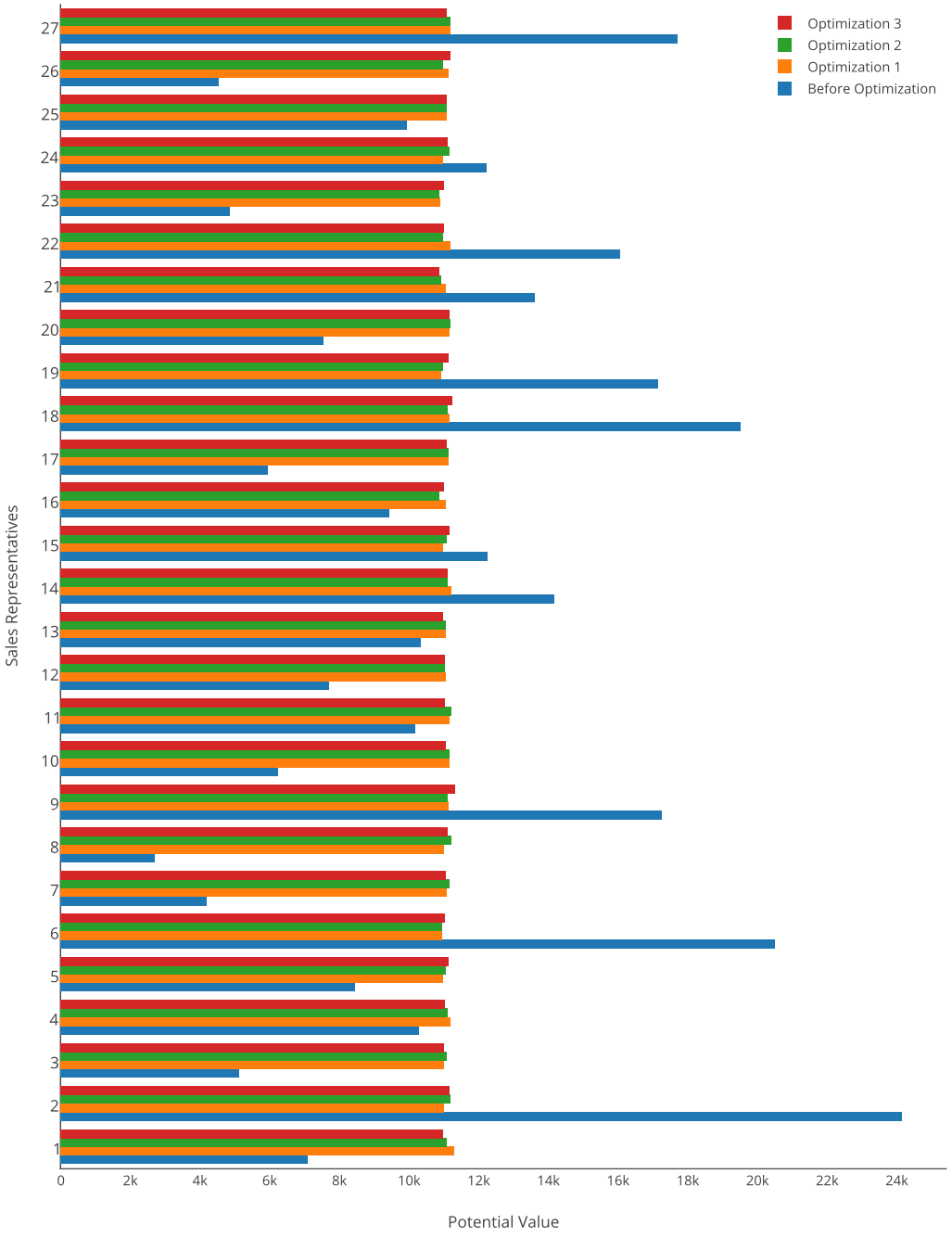

Figure 5.15: *Results of all experiments*

6 Conclusion and Future work

6.1 Conclusion

Since the sales territory planning is one of the most important issues for a selling organization, an intelligent decision support system for sales territory planning is highly needed. One major benefit of a decision support system is the ability to plan the outcomes of various scenarios before making a real investment.
In this thesis, we presented a decision support system based on the Genetic Algorithm as an effective approach for optimizing sales territories in large scale. The Genetic Algorithm (GA) is a stochastic search algorithm which mimics the process of natural selection. The GA always looks for the best solution among a set of possible solutions. The use of a dynamic mutation rate makes the GA more robust to local optima. The performance issue has been improved by means of the parallel processing architecture. In addition, the ranking of territories based on their potential values has guided the search direction towards the global optimum. Due to the independent nature of the GA, our method is easily configured to any objective in territory planning and is scalable to large search spaces similar to what we considered in this thesis. Although our method is focused on sales territory alignment, we proposed a solution for the location-allocation problem as well. According to our results, locating the sales team based on the distribution of potential, extremely improves the total calculation time of Green Field Planning. Territory Planning is classified as NP-Hard problem. To the best of our knowledge, this is the first thesis to apply the GA in Sales Territory Planning and demonstrates higher flexibility for optimization over existing alternative methods [22, 32].

In short, the contributions of this master thesis include:

1. A fast and simple computer-aided method for finding best locations for salesmen.
2. An independent optimization method based on the GA in order to balance the potential among sales representatives and minimize the travel cost automatically.
3. Implementation of these methods by using java programming language.
4. Designing a web based user interface as the client.

6.2 Future Work

Future work can be done in a number of different directions. In a practical sense, the efficiency of our territory planning engine can be improved and some bugs should be fixed.

Future research could include improvements to our GA to improve the computational cost. Our future work will include the acceleration of the computation time, and investigating a generic model for our optimization engine that can be applied to a large number of optimization problems without loss in accuracy. Also, it might be useful to modify the algorithm to be more flexible for adding user defined search criteria. This might allow marketers and decision makers to give weights to the desired objectives or make trade-off between different objectives. It is also possible to implement a web service [14] for our territory planning engine to serve various mobile and desktop applications.

Bibliography

[1] Angularjs mvc framework. 2015. https://angularjs.org/.

[2] Arcgis network analyst. 2015. http://www.esri.com/software/arcgis/extensions/networkanalyst.

[3] B-tree indexing in postgresql. 2015. http://www.postgresql.org/docs/9.2/static/indexes-types.html/.

[4] Geomarketing gis - the applications are numerous. 2015. https://www.wigeogis.com/en/solutions.

[5] Geomarketing: Know "where". 2015. http://www.gfk.com/solutions/geo-marketing/Pages/default.aspx.

[6] Geomarketing marketing, explore the strategy of geomarketing marketing. 2015. http://www.marketing-schools.org/types-of-marketing/geomarketing.html.

[7] Java and native programming languages. 2015. https://blogs.oracle.com/swdeveloper/entry/java_and_native_programming_languages/.

[8] Linear programming. 2015. http://en.wikipedia.org/wiki/Linear_programming/.

[9] Marketing strategies. 2015. http://www.entrepreneur.com/topic/marketing-strategies.

[10] Openlayers 3. 2015. http://openlayers.org/.

[11] Postgis: Spatial and geographic objects for postgresql. 2015. http://www.postgis.net/.

[12] Postgresql. 2015. http://www.postgresql.org/.

[13] Real-world uses of genetic algorithms. 2015. http://brainz.org/15-real-world-applications-genetic-algorithms/.

[14] Restful web service. 2015. http://www.ibm.com/developerworks/library/ws-restful/.

[15] Sales territory planning. 2015. http://www.wigeogis.com/en/sales_territory_planning.

[16] Stochastic universal sampling. 2015. `http://www.pg.gda.pl/~mkwies/dyd/geadocu/algselct.html`.

[17] Shyam Seshadri Brad Green. *AngularJS*. O'Reilly, 2013.

[18] Andreas Drexl and Knut Haase. Fast approximation methods for sales force deployment, 1999.

[19] C. Easingwood. Heuristic approach to selecting sales regions and territories. *Oper. Res. Quart.*, 24(4):527–534, 1973.

[20] Cliquet G. *Geomarketing - Methods and Strategies in Spatial Marketing*. Iste, 2002.

[21] Jörg Kalcsics, Stefan Nickel, and Michael Schröder. Towards a unified territorial design approachâapplications, algorithms and gis integration. *Top*, 13(1):1–56, 2005.

[22] Sven Mueller Knut Haase. Upper and lower bounds for the sales force deployment problem with explicit contiguity constraints. *European Journal of Operational Research*, 237(2):677–689, 2014.

[23] L.M. Lodish. Vaguely right approach to sales force allocations. *Harvard Business Rev.*, pages 119–124, 1974.

[24] Kwong Man, Tang. Genetic algorithms: concepts and applications. *Industrial Electronics, IEEE Transactions on*, 43(19):519 – 534, 1996.

[25] F MEHDIPOUR, MS MESGARI, B GOLCHIN, and MR AKBARI. Facility sitting using gis and genetic algorithms.

[26] Mark H. Karwan Mohsen Golalikhani. A hierarchical procedure for multi-skilled sales force spatial planning. *Computers Operations Research*, 40(5):1467–1480, 2013.

[27] Deepa Sivanandam. *Introduction to Genetic Algorithms*. Springer, 2008.

[28] Eiben Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[29] Ningchuan Xiao. Geographic optimization using evolutionary algorithms. In *8th International Conference on GeoComputation*. Citeseer, 2005.

[30] Mitsuo Gen Xinjie Yu. *Introduction to Evolutionary Algorithms*. Springer, 2010.

[31] A. Zoltners. A unified approach to sales territory alignment. *Sales Management*, pages 360–376, 1979.

[32] Andris A Zoltners and Prabhakant Sinha. Sales territory alignment: A review and model. *Management Science*, 29(11):1237–1256, 1983.