UNIVERSITY OF AGDER

Metaheuristics Applied to the Optimization of Continuous Functions

Atle Hellvik Havsø
Per-Steinar W. Karlsen

Supervisors
Noureddine Bouhmala, PhD
Bjørn Jensen, PhD

This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder, 2013
Faculty of Engineering and Science
Department of ICT

Abstract

Optimization is a field of mathematics which studies and develops mathematical methods with the aim of optimizing a wide range of problems. In physics these methods are central. Essentially all the dynamical equations in physics can be expressed as a series of optimization problems in terms of action integrals. Optimization can better be explained as finding the optima, also known as extremes, of a mathematical object. Such object may be a continuous function, as the case of this thesis. The approaches for solving optimization problems are generally divided into two categories, deterministic optimization and stochastic optimization. The main difference is that the deterministic approach applies calculus and the stochastic approach applies a search technique. For solving complex optimization problems, the stochastic approach has long proven to be most efficient.

This thesis focuses on improving the two stochastic search methods: Simulated Annealing and the Genetic Algorithm. This is performed by implementing two newly developed methods. The first method is the Tangent-based Evaluation method, which is better suited to detect abnormalities in continuous functions than the common one-point evaluation method. The other method is the Analytic Swap method for generation of solutions. Solution generation is an important part of any stochastic algorithm. Usually the new solutions generated by a random function, but the Analytic Swap method combines randomness with analytics to generate better solutions.

Preface

This master thesis is written as a part of the master program in Information and Communication Technology at the University of Agder. The thesis project is organized as course IKT590, Master's Thesis and makes up 25 percent of the two-year master program.

The main object of the master thesis it to give students an opportunity to perform an independent work of an academic character. The approach has to be scientific and should contain elements of research and present new ideas and contribute with new knowledge. The results are to be presented in this report, as well as through a public presentation and on a poster displayed on the university campus. The completion of the master thesis also marks an ending to the two-year master program.

This thesis is the result of the joint effort between Atle Hellvik Havsø and Per-Steinar W. Karlsen. Both students have previously completed the bachelor program in computer engineering at the University of Agder, with specialization respectively in software development; and network management and security. The supervisors of this thesis are Noureddine Bouhmala, PhD at Univerity of Agder and Bjørn Jensen, PhD at Vestfold University College.

Acknowledgements

We would like to express our gratitude to our supervisors, Noureddine Bouhmala, PhD and Bjørn Jensen, PhD for their ideas and guidance through our work with this thesis. Without their help and assistance, this thesis could not have been completed.

We also would like to thank Morten Goodwin, PhD at Univerity of Agder for his teaching and guidance during the planning and writing process.

Atle: I would like to thank my girlfriend for her support during this project and also my friends and family.

Per-Steinar: I would like to thank my wife for her support en encouragement during my studies. This could not have been accomplished without you. I also wish to thank my family and friends, especially my brother Bjørnar for his assistance and our enlightening discussions.

Atle Hellvik Havsø & Per-Steinar Karlsen
Grimstad, Norway
June 3rd, 2013

Content

1 Introduction

Optimization is a field of mathematics which studies and develops mathematical methods with the aim of optimizing a wide range of problems. In physics these methods are central. Essentially all the dynamical equations in physics can be expressed as a series of optimization problems in terms of action integrals. The Merriam-Webster's Collegiate Dictionary defines **Optimization** generally as:

> *"[A]n act, process, or methodology of making something (as a design, system, or decision) as fully perfect, functional, or effective as possible;" [1]*

Generally, optimization is regarded as another word for improving or making better. Within the field of mathematics, it has a more specific meaning. The same dictionary defines mathematical optimization as:

> *"[T]he mathematical procedures (as finding the maximum [or minimum] of a function) involved in this" [1]*

Optimization can better be explained as finding the optima, also known as extremes, of a mathematical object. The term object within mathematics has a wide definition. Some common types of objects within number theory are numbers, functions, functionals, series, matrices and sets. Objects are also found within other branches of mathematics like algebra, logic and algebra. The form of the extremes to an object depends of the object-type. For example, two dimensional functions have two types of optima, minima and maxima. Other types of objects may have more optima. A three dimensional geometrical figure has six optima types. An important subject, when discussing optima, is the notion of global and local optima. A global optimum is an optimum for any part of an object, but a local optimum is an optimum for a subsection of an object.

Mathematical optimization problems are generally divided into two categories: continuous optimization problems and discreet optimization problems. The main difference is that continuous optimization problems have an infinite number of possible solutions, whereas discrete optimization problems have a finite number of possible solutions. In both cases, the goal of optimization is to find the optima of the problem. Continuous optimization problems often takes form as a function, and the optima is to be found where the corresponding graph turns from going up, to turning down, or vice versa.

The approaches for solving optimization problems are generally divided into two categories, deterministic optimization and stochastic optimization. The main difference is that stochastic optimization always implies some degree of randomness and uncertainty, while deterministic methods do not. The consequence of this is that if it is possible to find a solution deterministically, it will always be correct and a repetition of the procedure will always yield the same result. However, stochastic optimization will always have a statistical error and it is likely that on gets a different solution by repeating the procedure. To describe the statistical variation in the results, one tends to use a probability distribution. This makes it possible to describe the accuracy of a stochastic method and the likelihood of the actual solution being within a given range around a solution achieved stochastically. The rest of this thesis is generally conserving stochastic optimization of continuous functions.

1.1 Continuous Optimization

The main concern in this thesis is the solution of continuous optimization problems by applying stochastic search algorithms. To understand stochastic optimization, it is important to know the differences between deterministic optimization and stochastic optimization. This can be demonstrated by optimizing the following continuous function. $f(x) = x^4 + x^3 - 4\,x^2$. The goal is to find the global minimum by using the derivative of the function. $f'(x) = 4x^3 + 3x^2 - 8x$. This is the most common method used to optimize relatively simple functions. The procedure is as following:

1. Find the derivative of $f(x) = x^4 + x^3 - 4\,x^2$.
2. Find all optima by setting: $f'(x) = 0 \implies x_1 \approx -1.84 \wedge x_2 = 0 \wedge x_3 \approx 1.09$
3. Find the corresponding $f(x)$ values: $f(-1.84) \approx -8.31, f(0) = 0, f(1.09) = -2.05$
4. Determine the global minimum: $-8.31 < -2.05 < 0$. The global minimum is $x \approx -1.84$.
5. Additionally one may also determine that function's extremes divergences to infinity: $\lim_{x\to\infty} f(x) = \infty \wedge \lim_{x\to-\infty} f(x) = \infty$

The procedure used above to find the global minimum can be illustrated by the following two graphs. The first graphs, Figure 1, shows the function, $f(x)$, with the three optima marked. It has one local minimum in addition to the global minimum and one local maximum. It has no global maximum, since it approaches infinity, when it divergences. The second graph, Figure 2, shows the derivative of the function and the optima are located where graph crosses the y-axis.

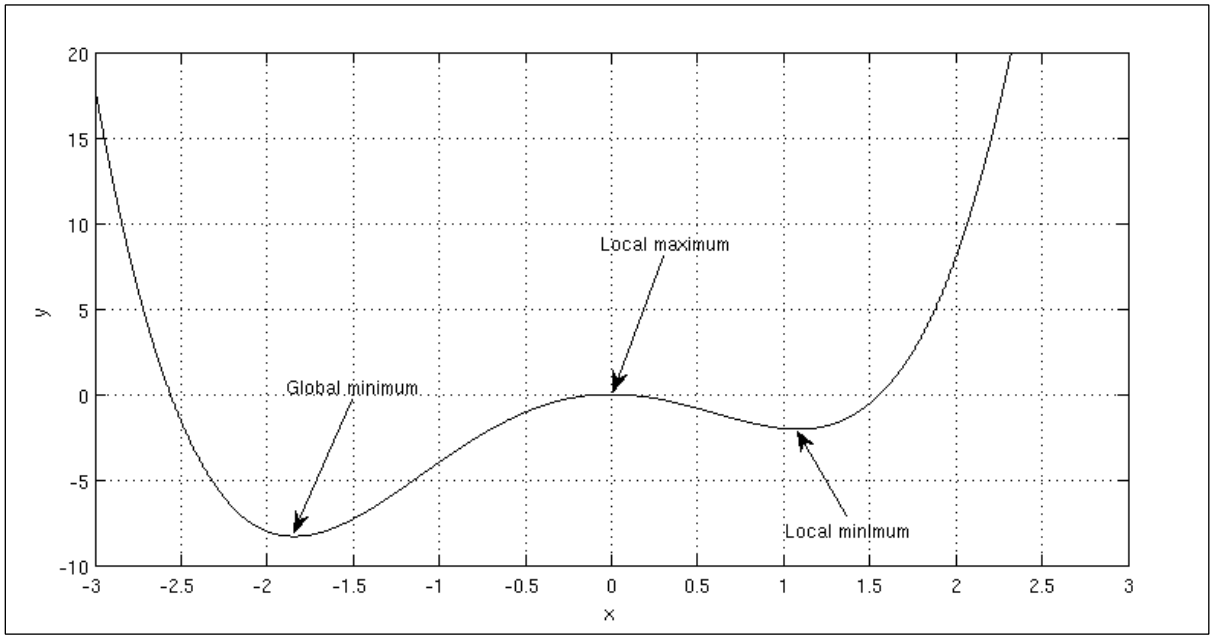


Figure 1 - Graph showing the locations of the optima of f(x). The lowest local minimum is also the global minimum.

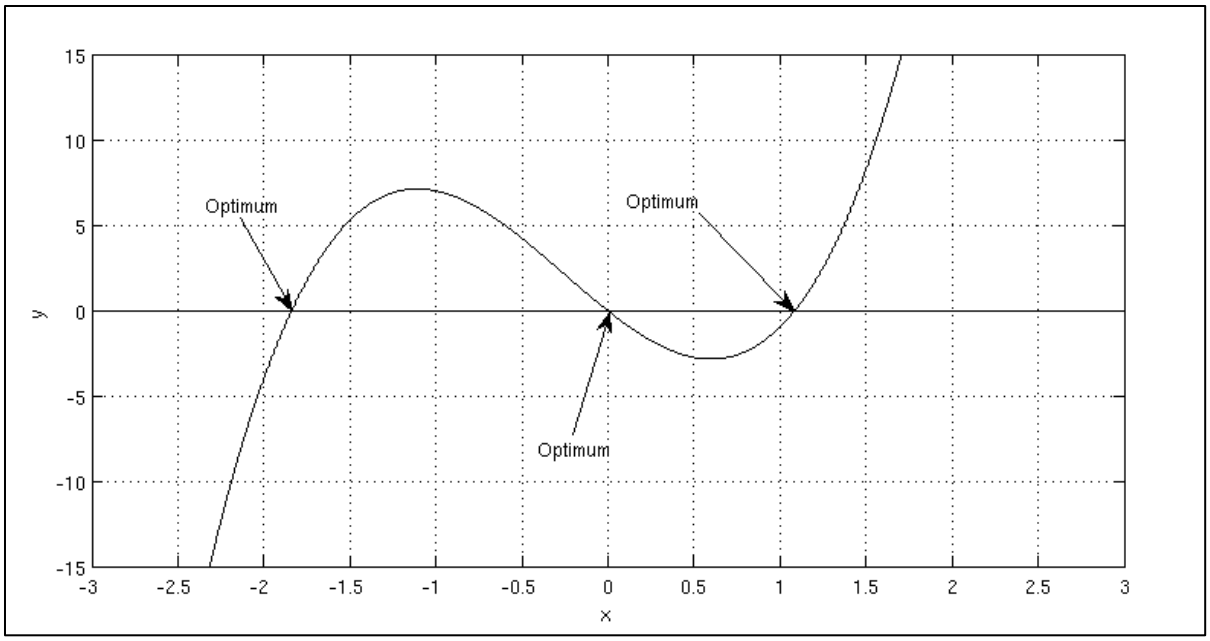


Figure 2 - Graph showing the derivative of f(x), f'(x). The optima of f(x) is located where f'(x) crosses the y-axis, f'(x)=0.

Even though solutions derived by deterministic optimization is mathematical accurate, not all problems are possible to be solved with this approach. Common reasons are that it requires too much computation or that the calculus becomes too complex. For these types of problems, a stochastic optimization may be more appropriate. Stochastic optimization often performed by using a search technique. The simplest form of search is to just selecting possible solutions by random, and return the best solution as the global minimum or maximum. This method is neither efficient nor reliable. To increase efficiency and at the same time ensure a satisfying outcome of a search, it is important to apply some kind of search strategy. Common for many search strategies are that they uses previous explores solutions to determine where to search next. This also applies to stochastic search algorithms, but they also incorporate some random mechanism. The goal is to combine the benefits of random search and more analytic search.

One simple method is to perform an extensive search[1]. This is done by exploring every possible solution. For optimizing a continuous function, an extensive search is performed by exploring every possible solution with a given interval, since there are an infinite number of possible solutions. Its main advantage is that it always finds the correct solution. For simple problems with a small search range, this search method may be suitable. But for more complex problems, it may be too time consuming or require too much memory.

Another search method is descent search. It is performed by random selecting one possible solution, x_n. Then another solution, x_{n+1}, in the neighborhood of the first solution, $N(x)$, is selected. The best solution is then selected, and the procedure is repeated until no further improvements are made. The main advantage is that it is little computational capacity and it is likely to find the closest optimum. The graph in Figure 3 shows how this method may be applied for solving the example above.

[1] Extensive Search is not defined as a stochastic search method since it does not include any randomness.

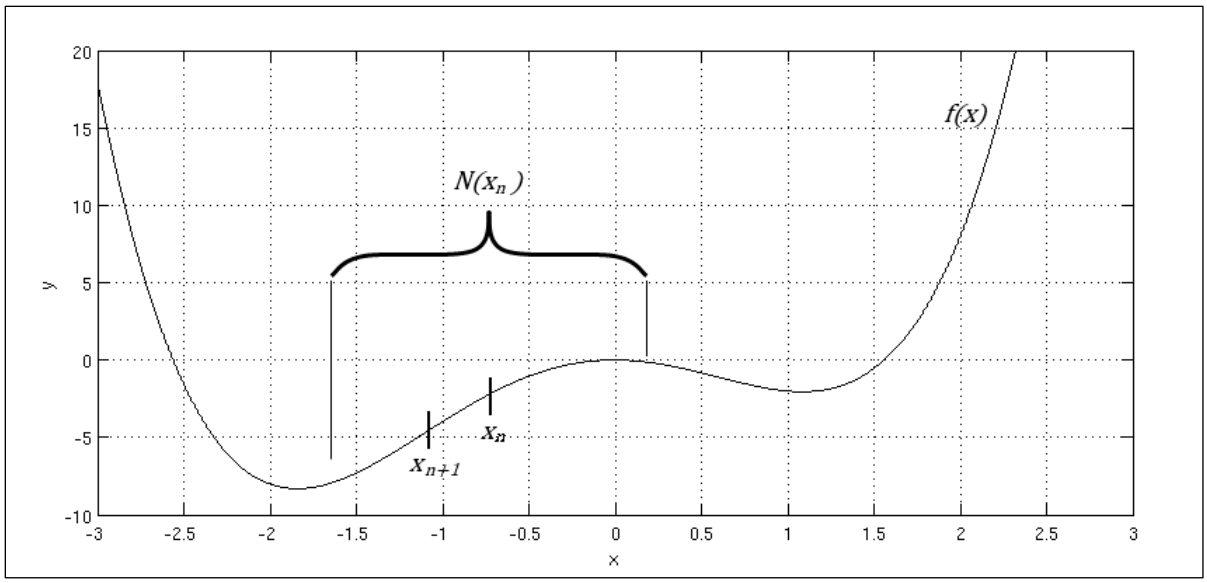


Figure 3 - Graph showing one round of the Descent Algorithm. The x selected here will lead to the discovery of the global minimum.

Descent search has one major drawback, regarding global optimization, which is that it may easily be trapped in a local optimum, as showed in Figure 4. This makes this search strategy more suited for finding a local optimum, rather than a global optimum. Instead, more advanced stochastic search strategies has to be applied. In the next chapter, some of the more common strategies are presented.

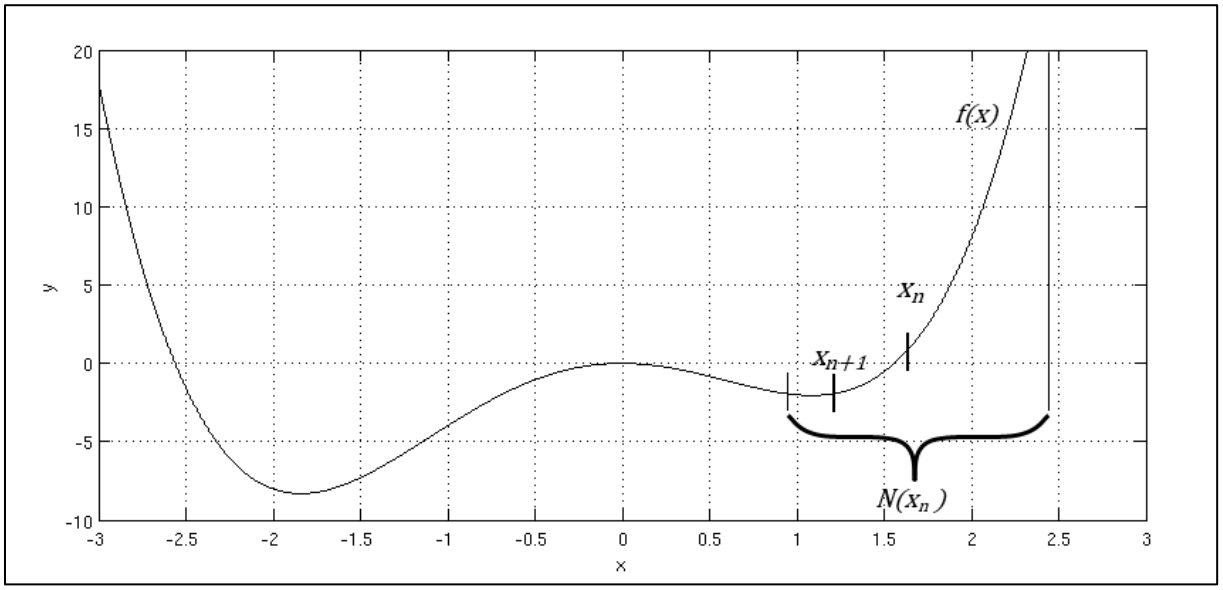


Figure 4 - Graph showing one round of the Descent Algorithm. The x selected here will NOT lead to the discovery of the global minimum.

1.2 Stochastic Search

Stochastic search methods are known for being efficient for solving complex optimization problems. However, this efficiency comes with the price of reduced statistical accuracy. For example, if one applies a stochastic search method on the same problem twice, without changing any parameters, it is very likely that one will get two close, but still different, results. By performing a search on the same problem multiple times, it is possible to calculate statistical accuracy. Usually, the Normal distribution is used for calculating the mean value and variance.

For more than sixty years, research has been done to improve stochastic search, with the goals of improving efficiency and increasing accuracy. Another direction of research has been to expand the number of problems that stochastic search methods can be applied on. As research has progressed, several different algorithms have been proposed and further developed. Following are some of the more well-known methods presented.

- Genetic Algorithm

  The Genetic Algorithm was first proposed by John Holland in his book "Adaptation in Natural and Artificial Systems" published in 1975. [6] This version of the algorithm is today known as the Simple Genetic Algorithm. The Genetic Algorithm is an evolutionary algorithm and it is inspired by the process of procreation and evolution in the biology.
- Simulated Annealing

  Simulated Annealing was first introduced in 1983 by Kirkpatrick, Gelatt and Vecchi and it was initially intended to solve combinatorial optimization problems. [2] The name and concept are borrowed from metallurgy, where annealing is the process occurring when warm metals cools and settles.
- Tabu Search

  Tabu Search was first described by Glover and McMillan in 1986 and was initially used to solve the general employee scheduling problem. [3] Today, it is applied to a wide-range of optimization problems, including global optimization of continuous functions. The main concept of Tabu Search is a list of solutions which are taboo. Solutions are added to the list as they are explored, this to avoid previous explored solutions from being re-explored. Thus, avoid the problem of circling between solutions within a local optimum.
- Variable Neighborhood Search

  The key concept of Variable Neighborhood Search is to find multiple local optima by varying the search range, also known as neighborhood, hence the name. It was introduced by Hansen and Mladenovic in 1997. [4] Variable Neighborhood Search can be explained simply as: Applying a local search algorithm, for example the Descent Algorithm, systematically on multiple parts of the function and then selecting the best local optimum as global optimum, that is: the lowest local minimum as global minimum and the highest local maximum as global maximum.
- Ant Colony Optimization

  Ant Colony Optimization was first presented by Colorni, Dorigo and Maniezzo in 1991 at the European Conference on Artificial Life in Paris. [5] Originally it was used for solving path optimization problems, but it has also been successfully applied on continuous optimization problems as well. [6] It mimics the mechanisms used by an ant colony to determine shortest path between two points. The basic concept is that each ant acts autonomously when trying to find a path between the points. Each ants has a certain amount of pheromone and it is distributed evenly long the path. If

an ant finds a short path, the pheromone is distributed with a high concentration, than if the ant finds a long path. By using a large swarm of ants and the shortest path is the path with highest level of pheromone.

- Greedy Randomized Adaptive Search Procedure

  The Greedy Randomized Adaptive Search Procedure was developed by Feo and Resende and first publicized in 1989. [7] It was first applied on set cover problems, but has later been modified for solving continuous optimization problems as well. [8] It uses randomization and greediness to generate possible solutions, which are further used as starting-point for a series of local searches. This process is repeated until a predefined stop criterion is met.

1.3 Problem and Hypothesis

Within the fields of engineering and science, solving optimization problems is of great importance. There are many different optimization problems. Some of the more well-known are combinatorial, discreet, scheduling and continuous optimization problems. For this thesis, continuous optimization problems are the main concern. One of the most used approaches to solve optimization problems is to apply a stochastic search method, of which the Genetic Algorithm and Simulated Annealing are two. Both require a method for evaluation potential optima. Previous, Jensen, Bouhmala and Nordli have proposed a new tangent based method for evaluation of potential global optima. It has been presented in their paper [9], where it is combined with the Simple Genetic Algorithm. With this as a background, the following three research questions were created:

- Is it possible to further increase the performance of combination of the tangent-based evaluation method and the Genetic Algorithm?
- Is it possible to increase the performance of Simulated Annealing by applying the tangent-based evaluation method?
- Can the performance of the Genetic Algorithm and Simulated Annealing be enhanced by introducing an Analytic Swap method for crossover/generation of new solutions?

The following hypotheses have been derived for the research questions:

- Hypothesis #1: It is possible to improve the algorithm used in [9] by applying various strategies for crossover and by introducing elitism-based selection.
- Hypothesis #2: It is possible to combine the tangent based method with Simulated Annealing.
- Hypothesis #3: It is possible to improve the Genetic Algorithm and Simulated Annealing by introducing a new a sequential-swap based method for crossover/generation of new solutions.

These hypotheses are to be proven by performing experiments and analysis of the results. The experiments consist of using the optimization method in various configurations to optimize a set of continuous functions.

1.4 Importance of Topic

Optimization is useful for solving many different types of mathematical problems and this thesis is focusing upon the optimization of continuous functions. Today there are two main approaches to optimization, the deterministic approach and the stochastic approach. The deterministic approach always provides a solution which is mathematically correct, if it is able to solve the problem. The main drawback is that it can be very computational intensive and thereby slow to solve complex problems. For these kinds of problems, it may be beneficial to apply the stochastic approach. This because, stochastic search methods are generally faster, compared to the deterministic methods. However, stochastic search methods do not provide a mathematically correct solution. It may not event provide the exact same solution if it is repeated. Nevertheless, stochastic search methods are in many cases be accurate enough and their accuracy and efficiency are being improved as research progress. Altogether, the efficiency and high enough accuracy of the stochastic approach may in many cases outweigh the mathematical precision of the deterministic approach.

Optimization plays an important part within many different fields of science and beyond. A few examples are: economics, engineering, physics, chemistry and biology. Today, there are a huge number of different ways of applying stochastic optimization and new applications emerges as research progresses. We will therefore present three possible applications.

- Economic analysis. Imagine a case where an owner of a factory which can manufacture four different products. Previous analyzes of the production have resulted in one mathematical function for each product, which describes the cost of making each product based on the number of produced products. Marked research has also been performed and resulted in functions describing the expected sales income based on the number of products in the marked. By combining these functions and applying optimization, one can calculated how many of each product the factory should manufacture to ensure the highest total income.
- Industrial engineering. Another case may be to design a pipe line system for transportation of some fluid or gas, the system may have several links to the outside and have variable load. To ensure that the system is designed to be able to withstand a worst case scenario, but also able to distribute the load and not to be more expensive than necessary, optimization may be applied.
- Chemical processes. Many processes in the chemical industry are very complex and contain several stages. They may be energy intensive, require expensive additives and enzymes, and produce toxic waste products and other types of emissions. For both economic and environmental reasons, it may be of interest to simplify processes. One possible and often efficient approach is to apply optimization. For example a process may require an additive to produce a certain chemical product, but the additive may also result in a highly toxic by-product, which may also be expensive to handle. The temperature may also be an important factor. By applying optimization, on may determine the best combination to ensure a high quality product and the lowest possible level of waste, but simultaneously ensuring high level of revenue.

These are just three practical applications where optimization may be of great use. For many optimization problems, the deterministic approach may be too complicated to use. For such problems, the stochastic approach has proven be more suitable approach. We hope that our research will enhance the accuracy and usefulness of stochastic optimization.

1.5 Limitations and Key Assumptions

In this chapter, the limitations and assumptions for this thesis are discussed. These are defined to avoid unnecessary extensions to the thesis. The main contributions are the further exploration of the Tangent-based method for evaluation of potential optima and the introduction of the Analytic Swap method for generation of new potential optima. The Tangent-based method is limited to be applied to functions with one variable, as it has not yet been expanded to handle functions with multiple variables. Thus, makes it not possible to apply this method on functions with multiple variables and only experiments with functions with one variable can be performed. The Analytic Swap method is however not depending on the number of variables a function has. Therefor it may be applied to generate new solutions for any function, regardless of the number of variables the function has. In this thesis, the experiments are limited to only consider functions with one and two variables. The experiments are also limited to a set of benchmark functions and the experiments may only prove that our contribution can benefit the optimization of these functions. However, one may assume that our contributions may be successfully on other continuous optimization problems, if the experiments show a successful outcome. For full list of functions used for the experiments, see chapter 6.

The implementation of the Genetic Algorithm and Simulated Annealing in this thesis is done in Python and the experiments have been performed on a regular workstation-type computer. Since the computer is not dedicated to the experiments and has to share resources with other processes, it was not possible to assure that the available computational capacity was constant. Also, the programing environment and implementation of the search methods may not be the most efficient one. Thus implying that, the measure of the efficiency may not be completely accurate, especially when it comes to time consumptions. Therefor the efficiency is measured in number of rounds/iterations, rather than time. However, this is of less importance since the main goals of this thesis are to improve the accuracy of the optimization and to further explore solution of new continuous problems.

1.6 Main Contributions of Thesis

If the hypotheses show to be correct, the potential outcome of the research will be:

- Expanding the knowledge of the accuracy of the optimization method presented in [9] by applying their method on additional problems and by applying it together with Simulated Annealing. This will show that the Tangent based method for evaluation of potential optima may be able to optimize any function with one variable.
- Improving the accuracy of the Genetic Algorithm when optimizing functions with one variable by introducing other strategies. In [9], it is showed that after a certain number of generations, the best solution in a generation is worse than, the best solution in the previous generation. The cause of this may be use of random selection of solutions or that the strategy paring and/or crossover are not good enough. By applying other, it may be possible to avoid this decline and possibly increase the accuracy of the optimization.
- Proving that the new Analytic Swap method may be better than the traditional methods for generating new solutions by increasing efficiency and/or accuracy of the optimization.

However, if the hypotheses show to not be correct, the results of the research will regardless contribute to the general knowledge about applying the Tangent based method and the Analytic Swap method and their appliance together with the Genetic Algorithm and Simulated Annealing.

1.7 Literature review

Various stochastic search methods have successfully been applied on many types of optimization problems.

S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi were the first to describe the Simulated Annealing algorithm in their article “Optimization by Simulated Annealing” [2]. However, a similar algorithm was independently developed by V. Černý and presented in the paper “Thermo dynamical approach to the traveling salesman problem: An efficient simulation algorithm”. [10]

The Genetic Algorithm was first described in the book “Adaptation in Natural and Artificial Systems” published by John Holland. [11] It describes the basic concepts of the algorithm and how it is derived for the field of biologically. Another important publication regarding optimization by applying the Genetic Algorithm is the book “Genetic Algorithms in Search, Optimization, and Machine Learning” by David. E. Goldberg. [12]

The tangent-based method for evaluation of potential optima was developed by Bjørn Jensen, Noureddine Bouhmala and Thomas Nordli and presented in their paper: “A novel tangent based framework for optimizing continuous functions” [9], where it is combined whit the Genetic Algorithm.

1.8 Thesis Report Outline

This rest of the thesis report has the following structure:

Chapter 2. Titled “The Genetic Algorithm”. Gives an introduction to the Genetic Algorithm and how it has been implemented in this thesis. It also contains a short summary of recent research and development of the algorithm.

Chapter 3. Titled “Simulated Annealing”. Introduces Simulated Annealing in a similar fashion as in chapter 2.

Chapter 4. Titled “The Tangent-based Evaluation method”. Describes the evaluation method developed by Jensen, Bouhmala and Nordli. It also presents a mathematical proves of its accuracy.

Chapter 5. Titled “The Analytic Swap Method”. Presents the new Analytic Swap method for the generation of new solutions to be used in a stochastic search.

Chapter 6. Titled “Experiments and Results”. Presents the experiment setup and a summary of the experiment results.

Chapter 7. Titled “Discussion”. Discusses the results of the experiments in detail. It focuses on the trends in the results and performance in general.

Chapter 8. Titled “Conclusion”. Gives a short conclusion of the thesis in the light of the hypothesis in chapter 1.3. It also discusses possible future work briefly.

2 The Genetic Algorithm

The Genetic Algorithm was first proposed by John Holland in his book "Adaptation in Natural and Artificial Systems" published in 1975 [11]. This version of the algorithm is today known as the Simple Genetic Algorithm. The Genetic Algorithm is an evolutionary algorithm and it is inspired by the process of procreation and evolution in the biology. Thus, much of the terminology used to explain this algorithm has it origin in biology. The idea behind it is that one may combine two solutions, called parents, into new solutions, called children. [13] This can be compared with how a child gets some features from the mother, and other features from the father. If this is done enough times, some of the children solutions will inheritance the best parts of both parents and one may end up with a better solution than the ones started with, usually known as optimization. One of the key differences, when GA is compared to the other the strategies, is that GA handles multiple solutions within every cycle (in the Genetic Algorithm known as generation).

2.1 The Algorithm

To be able to describe the algorithm, the following key-concepts have to be introduced. The algorithm works in cycles, which is known as ***Generations***. Each generation contains a set of solutions, known as ***Individuals***. Each individual has a ***Genome***, which is the value of the individual's solution. The smallest entity a genome may be divided into is a ***Gene***. For genomes containing a numerical value, a gene is usually bits or digits. In addition of the genome, each individual also has a value which denotes how well it solves the given problem, known as the ***Fitness***. The fitness is usually calculated before selection operation is performed. This is because many selection strategies rely on the fitness to decide which individuals which are to be brought forward to the next generation. The fitness may also be used by some pairing strategies.

In each generation, there are two population sets, the ***Parent Population*** and the ***Children Population***. In the first generation, the parent population is generated randomly. In the following generations, the parent population is the solutions which are brought forward from the previous generation. In each generation a new set of individuals is created. This is the children generation.

The children population is created by applying a set of genetic operators on the parent population. They are always performed in the following order and they simulate the process of biological procreation.

1. Pairing – Each individual in the parent generation is put into pairs. This can either happen randomly or by using a predefined strategy. The various pairing strategies used in this thesis, is described in chapter2.2.1.
2. Crossover – The process of combining parent pairs into new children by combining the genome of both parents. The various crossover strategies used in this thesis, is described in chapter2.2.3.
3. Mutation – It is when changes are performed on the genome of an individual and this may happen randomly or by following a strategy. The usage of mutations is optional and is not a part of the Simple Genetic Algorithm.
4. Selection – It is the process of selecting which individuals, which is to become the next parent generation. When selecting individuals, one may either only take the whole children population, or from both the parent and the children population. The various selection strategies used in this thesis, is described in chapter 2.2.4.

As for any stochastic search algorithm, The Genetic Algorithm reaches a state when it terminates and the genome of the individual with the best fitness is returned as the solution. This happens when the predefined conditions of termination is met. Common example of such conations may be when the best fitness converges or that a predefined number of generations have been created.

The Genetic Algorithm:

```
1. Initiation:
2.      Generate population.
3.      Evaluate the fitness of each individual in population.
4. While: (Best fitness is being improved) Do:
5.      {
6.      Create parents pairs by applying a paring strategy.
7.      Create new children population by applying crossover on
   parent pairs.
8.      Perform mutations (Not mandatory).
9.      Evaluate the fitness of individuals in new population.
10.     Perform selection for next generation of parents.
11.     }
12. Finish:
13.     Return the individual with best fitness as the ultimate
        solution.
```

This thesis is based on a previous implementation of the genetic Algorithm, presented in [9]. In this implementation, the Simple Genetic Algorithm is used, which does not apply the mutation operation. It also uses random paring and uniform crossover and only the children population is selected to be carried forward to the next generation. In this thesis, one of the hypothesis is that is it possible to improve the implementations of the Genetic Algorithm. This is to be explored by applying various combinations of the four operation types.

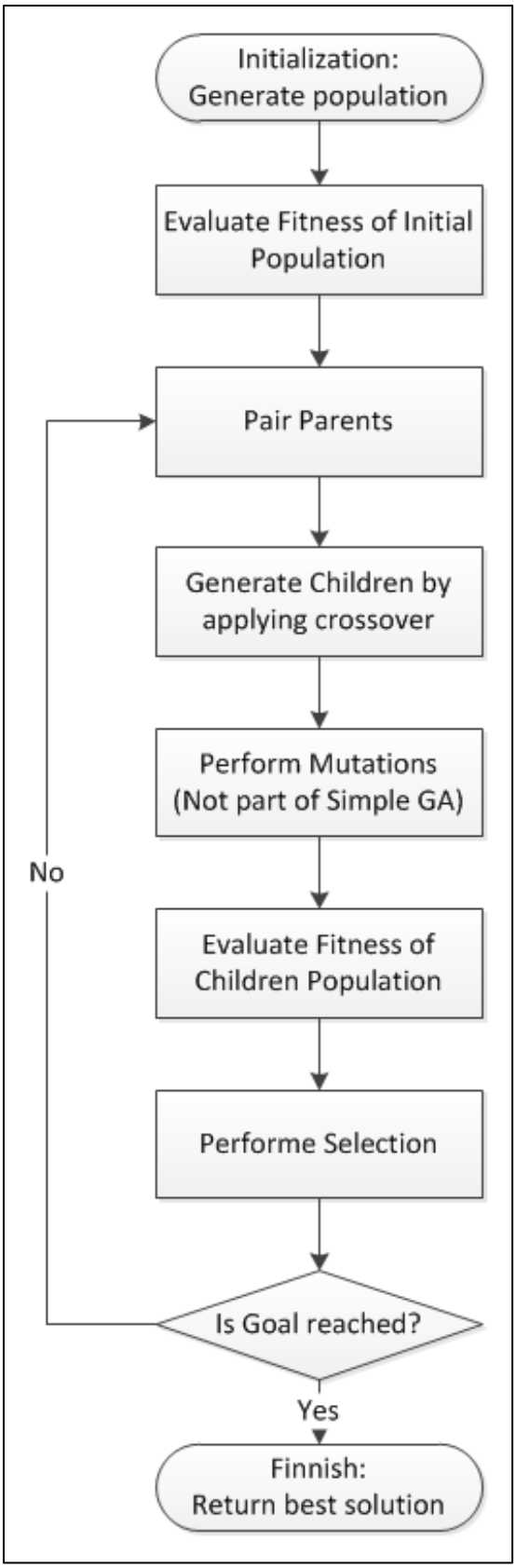


Figure 5 – Flow chart showing the structure of GA. The Goal is usually that the best solution has not been improved for a certain number of rounds.

2.2 Implementation

The implementation of the Genetic Algorithm requires one parameter, the size of the population, and minimum one strategy for each of the three operators, pairing, crossover and selection. All the three strategies may be selected individually. In addition, the mutation operator may be implemented, which requires its own strategy. The selection of strategies may have a great impact on the performance of the algorithm. It is thereby important the correct strategies are selected. Since one of the goals of this thesis is to explore how well different combinations of strategies performer, multiple variants of the various strategies were implements. The concept of mutations is not addressed in this thesis. All the implemented strategies are presented in this sub-chapter.

2.2.1 Size of Population

Through all the experiments in this thesis, the same population size has been used. This is to reduce the complexity when the results are analyzed. The population size was determined by performing a limited number of test experiments. The selection has to take two factors into account. Firstly, it is important that the population is large enough to provide a sufficient gene-pool. If the population is too small, many solutions may not be explored. Secondly, the population cannot be too large, because the required computational resources are proportional to the population size. The result of a too large population may usually reduce efficiency and theoretically, it may exhaust the computational resourced. Efficiency is especially in this thesis, because the experiments consist of a long series of optimizations. By balancing the two factors against each other, a population size of 100 was selected for all experiments.

2.2.2 Pairing Strategies

The pairing operator is the first operator to be applied in every generation and is the process where the parent population is divided into pairs, which later are used to generate a new children population. There exist many different strategies to use for selecting pairs. The pairing strategy used in this thesis is based on a roulette algorithm. It selects pairs by random, but the algorithm weights the fitness of each parent. This makes it more likely that both parents in a pair have the same fitness level.

2.2.3 Crossover Strategies

The crossover operator is the second operator and it combines two parents to create two new children. The genomes, also known as the solution, of both parents are divided into the same number of sections. The children are then created by taking sections from both parents. It is important to notice that a section can only be a part of one of the children and that all sections have to be used. There are several strategies that can be used for performing a crossover. The selection of strategies is depending on multiple factors:

- Static or variable length of genome.
- Dependency between genes.
- Number of parents used in crossover.

In this thesis, all genomes are of same length and only two parents are used when performing crossover. Also, genes, in this case: bits, in the solution, may occur multiple times in each solution. Based on this, the following crossover strategies have been implemented.

Crossover Strategy	Description
Single-Point	The simplest crossover strategy. A single point is selected randomly and both parents are divided into two sections. The children are then created by combining the first section from one parent and the second section from the other parent.
Uniform	Crossover is applied on gene level (e.g. bit or digit level), instead of on sections. For each gene, it is selected by random if it is to be switched or not. This is done by generate a random binary value, also known as a mask, with the same length as the parents' genome, where 1 represents switch and 0 represents not switch.
Analytic Swap	Consists of a series of gene-swaps like in uniform crossover, but the order is random. Between each swap, the fitness of the new interim solution is evaluated. In the end, the interim solution with the highest fitness is selected. For a detailed description, see chapter 5.

Table 1 – Crossover strategies implemented in this thesis.

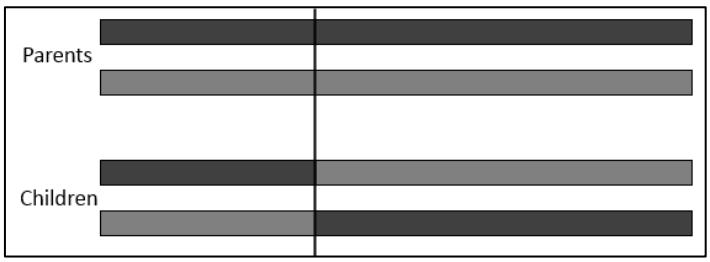


Figure 6 – Visual representation of single point crossover. The horizontal line represents the crossover points.

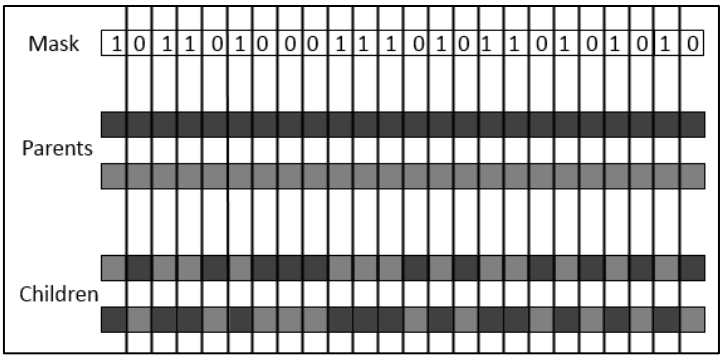


Figure 7 – Visual representation of uniform crossover. The horizontal lines represent the division into genes (e.g. bits) and the mask determines which genes are to be switched.

2.2.4 Selection Strategies

The selection operator is the last to be applied and it is used for selecting the individuals, which are to be brought forward into the next generation. There are many different selection strategies, but common to all is that the number of selected individual always is the same as the number of parents. This is in contrast to the situation in biology, where a generation may be larger than the previous generation. This limitation is crucial to avoid uncontrollable growth of individuals. The most extreme scenario is where the initial population, p_0, and both parents and children are brought forward into next generation. After n generations, the total population, p_n, will be ${p_0}^n$. The following selection strategies are the ones implemented as a part of this thesis.

Selection Strategy	Description
Only Children	The simplest selection strategy. As the name says, only the newly generated children are brought into the next generation. Since the number of children is equal to the number of parents, none of the children are discarded.
Elitism	All children are first sorted by fitness level, and then the two children with the lowest fitness level are discarded and replaced by the two parents with the highest fitness level.

Table 2 – The main classes of selection strategies implemented in this thesis.

2.2.5 Mutation

Crossover is the common operator used to create new solutions or children. It requires two parents, in some cases more, and all the parents' genes are used when creating the children. Thou, no genes are discarded during crossover, no new genes are introduced. In nature, errors, or mutations, may occur when an offspring is created. The concepts of mutations can also be implemented in the Genetic Algorithm. This is usually implemented by using a random function with low probability, as it is done in this thesis. It selects if one or more genes are to be changed, or mutated. If the genome is divided into bits, as in this thesis, the selected bits are switched. In other cases, where the genome is divided into digit, the selected digits are discarded and new digit is selected by random. The mutation operator is not implemented as a part of this thesis.

2.3 Prior Research

The Genetic Algorithm has it origin in the research of evolutionary programing in the 1960's. [14] One of the pioneers was John Holland at the University of Michigan, who was the first to use the term "Genetic Algorithm" in 1975 in his book "Adaptation in Natural and Artificial Systems". [11] This implementation is today known as the Simple Genetic Algorithm. The main advantage of this algorithm was that it was problem independent, compared to previous evolutionary algorithms, which usually were more problem specific. Since then, many variants of the Genetic Algorithm have been developed and genetic algorithms have been applied to a wide range of optimization problem, as well as to other problem types.

In attrition to widen the possible appliances for the algorithm, much research has also been put into improving the algorithm by increasing the diversity among the individuals, thus increase the number of explored solutions. One of the latest trends in this research is to apply various continuous probability distributions to create new crossover strategies. Both Laplace and Gaussian distribution have been applied successfully for solving optimization problems. [15; 16]Increased diversity is also being achieved by applying various mutation strategies, with one of the latest being Adaptive Directed Mutation proposed by Tang and Tseng in 2013. [17] The development of new pairing and selection strategies has lately been of less important.

Research in genetic algorithms is not only focused on how to improve the performance of the algorithm alone. Much effort has also been devoted to creating hybrid algorithms. Three of the lasted hybrids, which have been applied to solve continuous optimization problems is the, previously mentioned, Gauss-VNS-GA-IPA, the GAAPI algorithm and Tabu Search-Genetic Algorithm. GAAPI is developed by Ciornei and Kyriakides and first published in 2012. [18] It is a hybrid between the Genetic Algorithm and a variant of Ant Colony Optimization for continuous functions called API. The Genetic Algorithm is used for fining possible starting points, also known as nests. Ant Colony Optimization is then used to explore the starting point with the highest fitness. This is repeated until the termination criteria are met.

The Tabu Search-Genetic Algorithm hybrid was proposed by Ramkumar, Schoen and Lin in 2010. [19] It applies Tabu Search for the approximation of ranges, which contains a local optima, and the Genetic Algorithm used within these rages for determine the accurate location of the optima.

3 Simulated Annealing

Simulated Annealing is inspired by annealing within metallurgy, which is a process used to modify the ductility and hardness of metal by heating and regulating the cooling process. It was first described by Kirkpatrick, Gelatt and Vecchi in 1983. [2] The idea behind Simulated Annealing is that in the same way that the metal atoms tries to achieve an ideal state when the metal cools, it will provide an ideal solution. It is very similar to the simpler Descent Algorithm, but instead of discarding all inferior solutions, it uses a probability function to determine if an inferior solution is to be accepted or not. When metal is hot, the metal atoms are more likely to move into a less ideal state. As the metal cools, it is becoming less likely that atoms may move into a less ideal state. In Simulated Annealing, this is represented by the probability function, which by time makes it less likely that inferior solutions are to be selected. By continuing the usage of terminology from metallurgy, one may say that the algorithm starts out to be hot, and are cooling by time.

3.1 The Algorithm

In this chapter, the basic operation of Simulated Annealing is explained, for a close description of our implementation, see next chapter. Simulated Annealing operates with three solutions simultaneously: the current solution, x_n, the accepted solution, x^*, and the best solution, $\overline{x}$. The current solution is the solution which is generated in a round, either randomly or by modifying the accepted solution. The accepted solution is a solution generated in a previous round. It is always replaced with the new solution, if it is better, but is there is also a probability of it being replaced if the new solution is inferior. The best solution is always the best solution yet explored and can only be improved. The algorithm consists of three phases:

The first phase of the algorithm is the initiation phase. During this phase the first solution is generated randomly and predefined parameters, like the initial temperature, are set.

The second phase is the search phase and it is in this phase solutions are explored, in this case potential optima. This phase consists of two loops, the outer and the inner loop. The outer loop handles the decrease in temperature and it ends when the stop criterion is met. Possible stop criteria are number of iterations, convergence in best solution or that temperature has reached a predefined minimum. Within the outer loop, the inner loop is also found. The inner loop handles the exploration of possible solutions and starts with the generation of a new solution, x_n,. If the new solution is better than the accepted solution, the accepted solution is replaced. If not, a number, p, between 0 and 1 is generated randomly and compared to the probability function, P. If the p is lower than the value of P, the new solution replaces the previous accepted solution, even though it being inferior. If the new solution also is the best solution yet explored, it also replaces the current best solution.

The last phase is the finishing phase, where the best solution, $\overline{x}$, is presented. For a continuous optimization problem, the best solution is the global minimum or maximum. Other data may be returned as well. During this phase no calculations are performed.

```
1.  Initiation:
2.      Select a random solution, x_1 → x* → x̄.
3.      Calculate F* = F̄ = f(x_1).
4.  While:(Stop criterion NOT reached) Do:
5.      {
6.      Set i = 0
7.      While:(i < I) Do:
8.            {
9.            Generate new solution, x_n
10.           Calculate F_n = f(x_n).
11.           If: (F_n is better than F*) Then:
12.                 {
13.                 x_n → x*, and F_n → F*.
14.                 Set i = 0
15.                 }
16.           Else:
17.                 {
18.                 Generate random number, p.
19.                 If: (p < P) Then:
20.                     {
21.                     x_n → x*, and F_n → F*.
22.                     }
23.                 }
24.                 i++.
25.           If: (F_n is better than F̄) Then:
26.                 {
27.                 x_n → x̄, and F_n → F̄.
28.                 }
29.           }
30.     Reduce temperature.
31.     }
32. Finish:
33.     Return x̄ as the best solution.
```

Symbols used in the algorithm:

- n: Denotes the current round.
- x_n: The current solution.
- $\overline{x}$: Best solution, $\overline{x} \in X$.
- $f(x)$: The function which the algorithm is applied to.
- F_n: The function value given by the current solution, $F_n = f(x_n)$.
- $\overline{F}$: The function value given by the best solution, $\overline{F} = f(\overline{x})$.
- P: The probability function used to decide if an inferior solution is to be accepted. Various functions may be used, but it is required that the probability decreases by time and $P_n \in [0,1]$
- p: A random number, $p \in [0,1]$
- x^*: Accepted solution.
- F^*: The function value given by the accepted solution, $F^* = f(x^*)$.
- I: Length of inner loop.

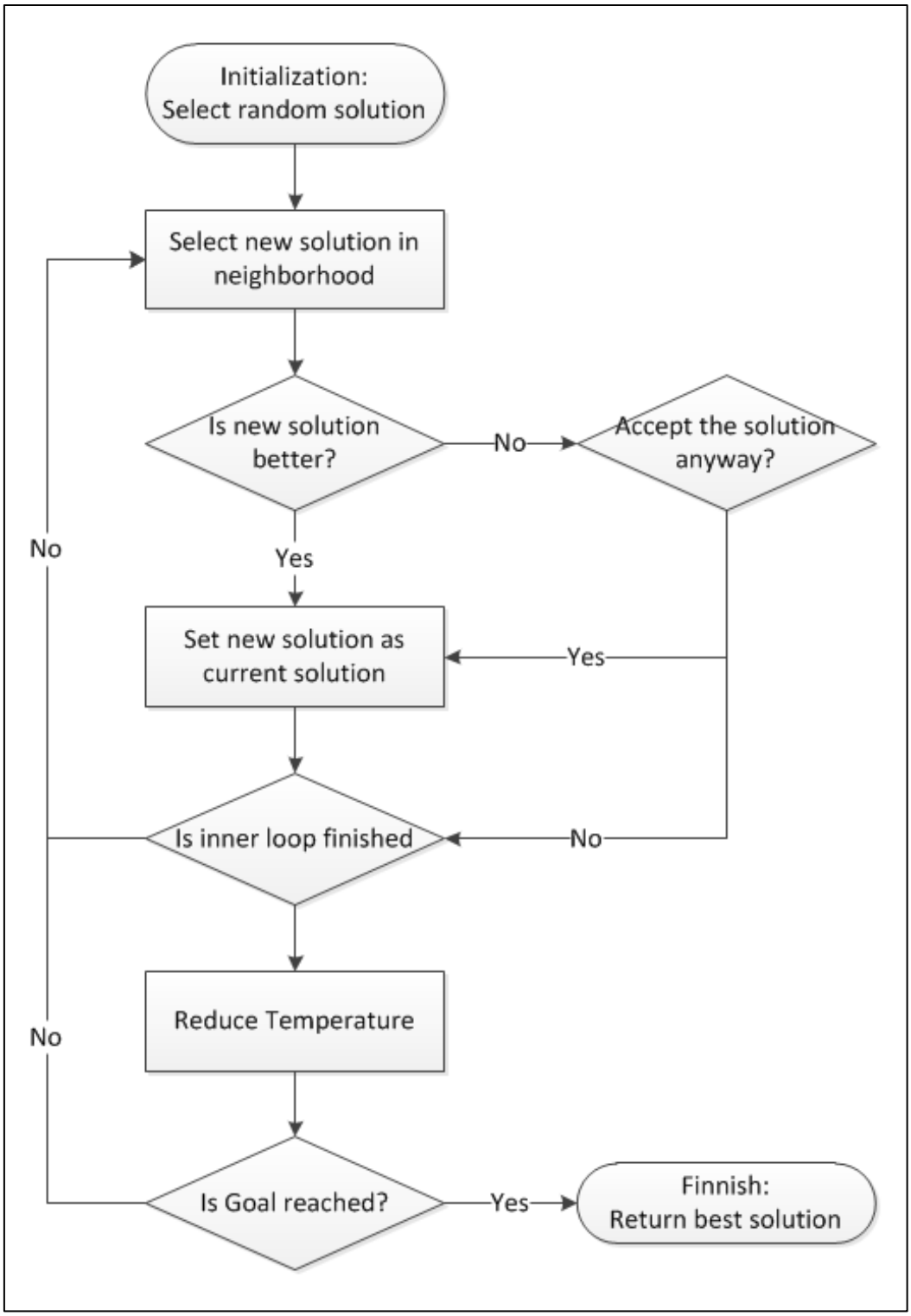


Figure 8 – Simplified flowchart showing the structure of SA. In the beginning, it is hot and any solution may be accepted, even though they are inferior. The probability for accepting an inferior solution is decided by a probability function and the probability is reduced as time passes. This is known as the cooling period. Usual goals for this algorithm are either that the solution converges or that the temperature has reached a certain level.

3.2 Implementation

Our implementation of Simulated Annealing is based upon the standardized version, which includes the usage of a probability function based on the Boltzmann distribution. Main contribution to this algorithm is the appliance of the tangent-based method for evaluation of potential optima and the introduction of the new Analytic Swap method for solution generation.

3.2.1 Solution Generation

In this implementation of Simulated Annealing, two methods for generating new solutions are used. The first method is standard bit inversion. One bit is selected randomly and inverted. The second method is a new method and is based on a sequence of bit-swapping operations. For further information about this method, see chapter 5.

3.2.2 Initial temperature

Selecting an appropriate initial temperature is an important part of configuring Simulated Annealing. A common method is to define it a static constant with a value between 0.5 and 1. To reduce the time required for parameter tuning, "Partitioning of Unstructured Meshes for Parallel Processing" was used to determine the initial temperature automatically and individually for each problem. This algorithm was developed by Bouhmala as a part of his doctoral thesis and first publicized in 1998. [20] The algorithm has the following structure:

1. **Initiation:**
2. Generate initial random solution, x_0.
3. Set: $x^* = x_0$
4. **Set:** $global\_increase = 0, Counter = 0$
5. **For:** (n=1; n<100'000; n++) **Do:**
6. {
7. Generate random solution, x_n.
8. Calculate difference, $\Delta x = x_n - x^*$.
9. **If:** $(x_n > x^*)$ **Then:**
10. {
11. $x^* = x_n$
12. $global\_increase = global\_increase + \Delta x$
13. $Counter++$
14. }
15. }
16. $Average = \frac{global\_increase}{conter}$
17. $T = -\frac{Average}{\log 0{,}8}$
18. **Finish:**
19. Return T as the initial temperature solution.

3.2.3 Probability Function

The acceptance probability function is a variant of the Boltzmann distribution. This distribution was used when Simulated Annealing was first presented and is today very common among implementations of Simulated Annealing. [2] This distribution takes two variables, the current temperature and the difference between the new solution and the accepted solution. The current temperature is calculated by multiplying the old temperature with a predefined constant.

$$P(T, \Delta F) = e^{\left(-\frac{1}{T}\Delta F\right)}$$

$$\Delta F = \bar{F} - F_n$$

$$T_{new} = T_{old} * r$$

It is important to notice that the above calculation of ΔF is only correct when solving minimization problems, when solving maximization problems, the signing has to be inverted.

3.3 Prior Research

Simulated Annealing was first introduced in 1983 by Kirkpatrick, Gelatt and Vecchi and it was initially intended to solve combinatorial optimization problems. [2] It has later been applied to solve many different optimization problems, including discrete and global continuous optimization problems. For global optimization of functions with multiple variables, the Enhanced Simulated Annealing algorithm presented by Siarry, Berthiau, Durbin and Haussy has been an important contribution. [21]

The main advantage is that it can be applied without any prior knowledge about the solution. [22] Nevertheless, it has two major drawbacks: it is computational intensive [22; 23] and requires that the length of the cooling period is scheduled correctly. [24] Since it was first publicized, many variations have been developed to minimize the drawbacks, as well as enhancing the accuracy of the algorithm. One important milestone was the introduction of parallelism done by Ram, Sreenivas and Subramaniam in 1996 [23] and one of the latest contributions to parallelism and optimization of continuous functions is the Temperature Parallel Simulated Annealing proposed by Cai, Ewing and Ma in 2012. [22]

A lot of research on Simulated Annealing has been on how to schedule the cooling period. To reduce the need of manually adjusting the scheduling of the cooling period, Adaptive Simulated Annealing was introduced by Ingber in 1987.[2] [24] Several improvements to the Adaptive Simulated Annealing have been proposed. One of the later is the Fuzzy-based Adaptive Sample-sort Simulated Annealing proposed by Shukla, Sun and Tiwari in 2007. [25] It combines Adaptive Simulated Annealing with parallelism and fuzzy logic. A similar algorithm was presented by Oliveira and Petraglia in 2011, when they combined Simulated Annealing and fuzzy logic with Particle Swarm Optimization. [26] It has showed great results, when used to solve global optimization problems.

[2] In the original publication from 1987, it was known as Very Fast Simulated Re-annealing. It was renamed in 1993 by Ingber. [25]

4 The Tangent-based Evaluation method

In the first chapter of this thesis, the concept of optimization of continuous functions was introduced. An important part of any optimization process is the evaluation of potential global optima. The common method to evaluate a potential global optimum is to insert it into the function, $f(x)$ and a value is returned. This value is then compared to the value of other potential global optima. This is called one-point evaluation. For deterministic optimization, evaluation of potential global optima only has to be performed once, since the set of potential optima is known to be all the local optima of the function.

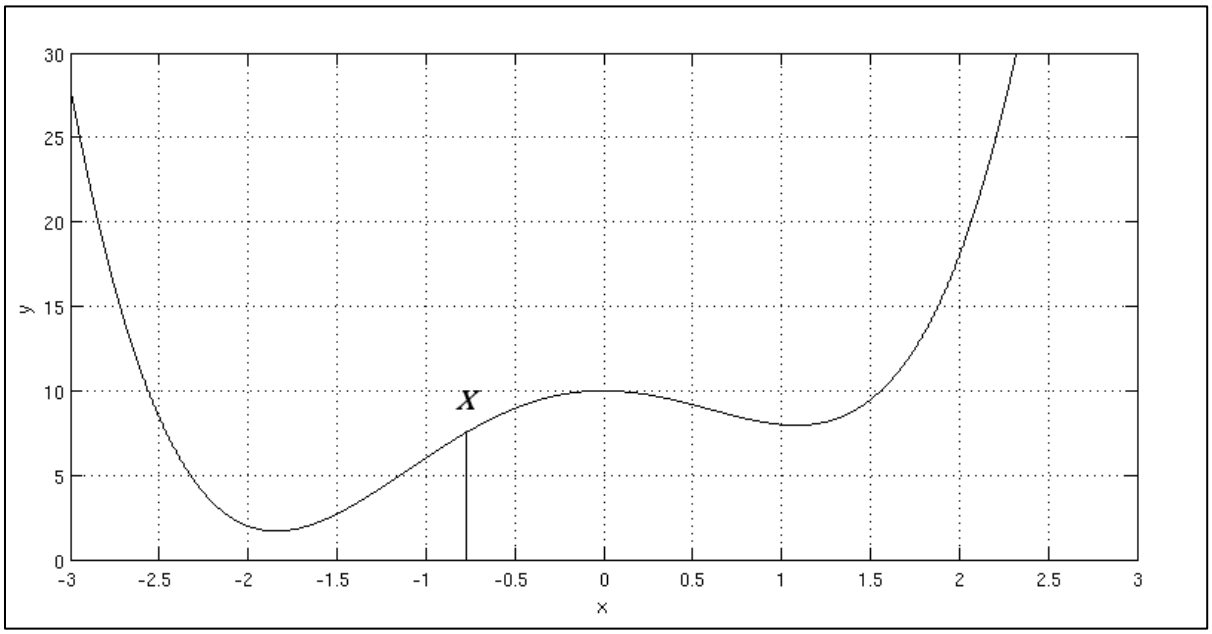


Figure 9 – By applying one-point evaluation, only f(x) is evaluated and the evaluation is valid for this x-value. To evaluate nearby x-values, new evaluations have to be performed.

The same method for evaluation is also often used when preforming stochastic optimization. Stochastic optimization usually involves the usage of a search algorithm, which returns possible solutions to the problem. These solutions are then evaluated and the results are then used in the next round of the searching process. When the stop criterion for the search is achieved, the best solution is returned and is considered the global optimum. When using stochastic optimization, some degree of uncertainty always has to be expected. In some cases the difference between the result and the actual global optima may be relatively large. This applies particular for complex functions with many abnormalities. Abnormalities may for example be spices, which may be hard to detect by a stochastic search algorithm. This is especially, if the abnormality only affects a small portion of a function. In these cases, it is likely that points on both sides of an abnormality are evaluated, but the abnormality is not detected.

To increase the probability of detecting such abnormalities, it would be beneficial to evaluate a small portion of the function, instead of just a point. This is known as two-point evaluation. The simplest method used to perform two-point evaluation, is to apply a modified version of the derivation operator. By definition, any function is the derivative of another function:

$$f(x) = \frac{dg(x)}{dx} = \lim_{h \to 0} \frac{g(x+h) - g(x)}{h}$$

Two-point evaluation is achieved by choosing a small h-value, but still larger than zero. This will provide an evaluation of the range between x and $x + h$. This will reduce the need of evaluation nearby x-values. However, by using this is an approximation, a mathematical error is introduced and the accuracy is reduced.

$$f(x) = \frac{dg(x)}{dx} \simeq \frac{g(x+h) - g(x)}{h}$$

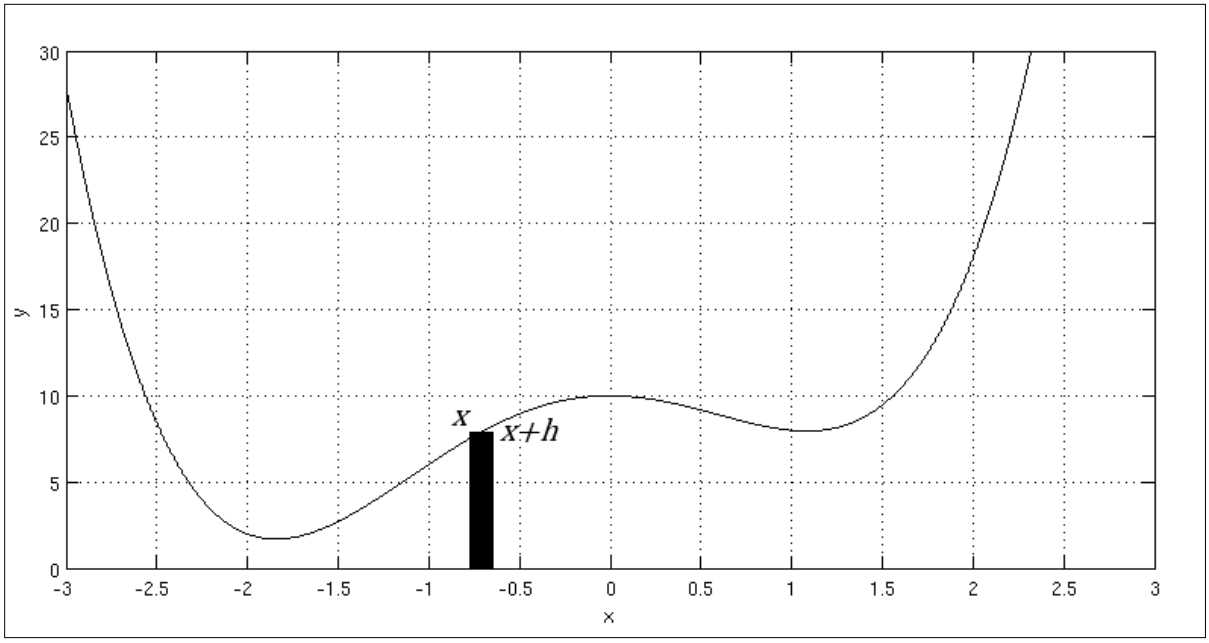


Figure 10 – By applying two-point evaluation, on does not only evaluate the points x and x+h, but also the range between. Since this is an approximation, it will introduce an error. The degree of error is dependent on the size of h and the development of the curve.

Using the formula for two-point evaluation, one can reduce the number of x-values to be evaluated. This increases the efficiency, to the cost of the introduction of a small mathematical error. This is because this approximation seldom equals the one-point evaluation and that the evaluation is the asymmetric, regarding point x. This error can be reduced by introducing a symmetric method for evaluation. This can be performed by evaluating a range on both sides of the point x. It is important that the ranges on both sides of x are of equal size, and suitable ranges are $[x - h, x\rangle$ and $[x, x + h\rangle$.

Evaluating the range $[x, x+h\rangle$:

$$f(x) = \frac{dg(x)}{dx} = \lim_{h \to 0} \frac{g(x+h) - g(x)}{h}$$

Evaluating the range $[x-h, x\rangle$:

$$f(x) = \frac{dg(x)}{dx} = \lim_{h \to 0} \frac{g(x) - g(x-h)}{h}$$

Combining the evaluation of whole range:

$$f(x) + f(x) = \frac{dg(x)}{dx} + \frac{dg(x)}{dx} = \lim_{h \to 0} \frac{g(x+h) - g(x)}{h} + \lim_{h \to 0} \frac{g(x) - g(x-h)}{h}$$

$$f(x) + f(x) = \frac{dg(x)}{dx} + \frac{dg(x)}{dx} = \lim_{h \to 0} \left(\frac{g(x+h) - g(x)}{h} + \frac{g(x) - g(x-h)}{h} \right)$$

$$2f(x) = 2\frac{dg(x)}{dx} = \lim_{h \to 0} \frac{g(x+h) - g(x-h)}{h}$$

$$f(x) = \frac{dg(x)}{dx} = \lim_{h \to 0} \frac{g(x+h) - g(x-h)}{2h}$$

By applying the same approximation as used on the regular derivation formula, the following symmetric two-point evaluation formula is created:

$$f(x) = \frac{dg(x)}{dx} \simeq \frac{g(x+h) - g(x-h)}{2h}$$

This makes it possible to perform a symmetric two-point evaluation of a given x-value. Thus, making the evaluation more accurate, than if the asymmetric two-point evaluation formula is used. The two points used in this evaluation also forms a tangent to the point x, hence the name of this method.

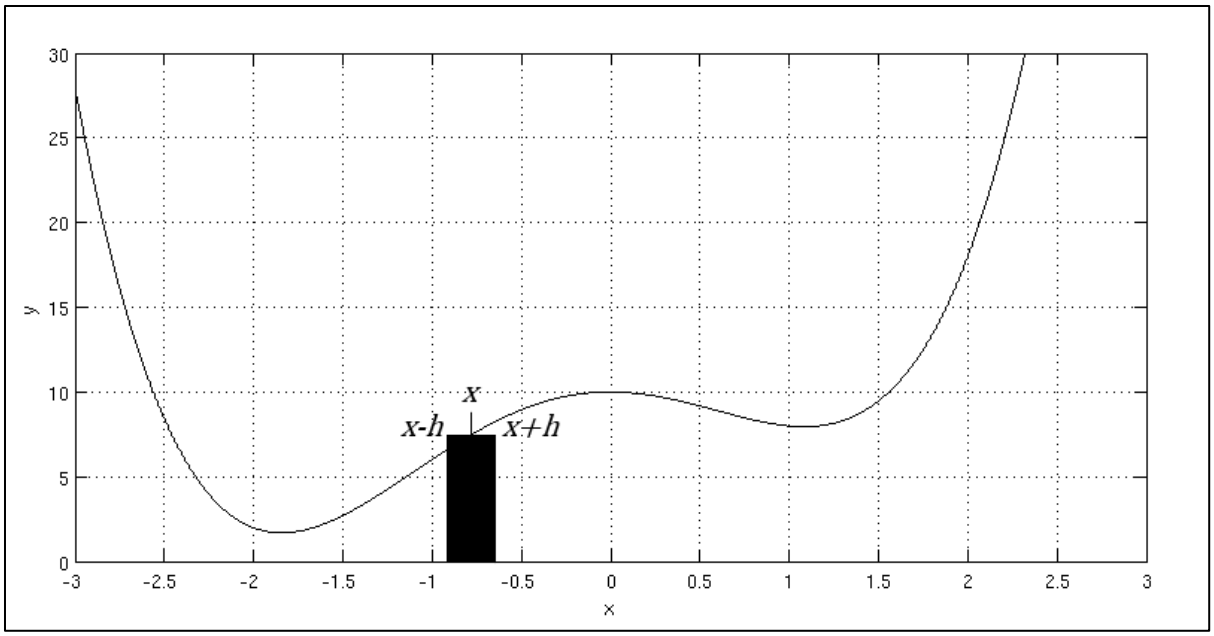


Figure 11 – Symmetric two-point evaluation proves to be more mathematical accurate. However, the error introduced by applying an h-value greater than zero is not eliminated, only reduced.

4.1 Estimation of Accuracy

As mentioned earlier, the approximation in the Tangent-based Evaluation method introduced a mathematical inaccuracy. The order of the inaccuracy can be estimated by combining two Tailor series. This is performed as following:

Step1: We start by introducing the standard definition of a Tailor series.

$$g(x) = \sum_{k=0}^{\infty} \frac{g^{(k)}(a)}{k!}(b-a)^k$$
$$= g(a) + g'(a)(b-a) + \frac{g''(a)}{2}(b-a)^2 + \frac{g'''(a)}{6}(b-a)^3 + \cdots$$

Step 2: To evaluation of the left side of x, a and b is set as $a = x$ and $b = x - h$. This creates the following series:

$$g(x-h) = g(x) + g'(x)(x-h-x) + \frac{g''(x)}{2}(x-h-a)^2 + \frac{g'''(x)}{6}(x-h-x)^3 + \cdots$$
$$= g(x) - g'(x)h + \frac{g''(x)}{2}h^2 - \frac{g'''(x)}{6}h^3 + \cdots$$

Step 3: To evaluation of the right side of x, a and b is set as $a = x$ and $b = x + h$. This creates the following series:

$$g(x+h) = g(x) + g'(x)(x+h-x) + \frac{g''(x)}{2}(x+h-x)^2 + \frac{g'(x)}{6}(x+h-x)^3 + \cdots$$

$$= g(x) + g'(x)h + \frac{g''(x)}{2}h^2 + \frac{g'(x)}{6}h^3 + \cdots$$

Step 4: The two series from step 2 and 3 is then combined into one series to create a symmetric evaluation.

$$g(x+h) - g(x-h) =$$

$$\left(g(x) + g'(x)h + \frac{g''(x)}{2}h^2 + \frac{g'(x)}{6}h^3 + \cdots\right)$$

$$-\left(g(x) - g'(x)h + \frac{g''(x)}{2}h^2 - \frac{g'''(x)}{6}h^3 + \cdots\right)$$

$$g(x+h) - g(x-h) = 2g'(x)h + \frac{g'''(x)}{3}h^3 + \cdots$$

Step 5: To reduce the complexity of the series, it can be simplified by defining the terms containing third and higher derivatives of $g(x)$ as the reminder of third order.

$$\frac{g(x+h) - g(x-h)}{2h} = g'(x) + R_3(x), \qquad R_3(x) = O(h^2)$$

Step 6: The remainder is then to be omitted, which introduces an error of $O(h^2)$, and the following approximation is made:

$$\frac{g(x+h) - g(x-h)}{2h} \cong g'(x)$$

Step 7: By definition, the derivative of one function may be viewed as another function.

$$f(x) = g'(x) = \frac{dg(x)}{dx} \cong \frac{g(x+h) - g(x-h)}{2h}$$

These calculations proves that the error introduced by applying the Tangent-based Evaluation method can be estimated to be no more than of $O(h^2)$.

5 The Analytic Swap Method

The Analytic Swap method is a new method for generating new solutions. It combines randomness with analytics, which together have showed to be very powerful. Due to the differences between various stochastic search algorithms, it has to be adapted individually. It is bit-based and to generate a new solution, it requires that an initial solution is provided. The method consists of a series of bit-swap operations. A bit-swap operation may only be performed once on a bit and stops when all bits are switched. Each swap creates an interim solution, which consist of all previous swaps. After each swap, an evaluation is performed and in the end, the interim solution with the best score is selected. In the end when all swaps are performed, the interim solution with the highest score is selected.

Due to the differences between the Genetic Algorithm and Simulated Annealing, the Analytic Swap method has to be adapted differently. For other appliances, different adaptations may be required. In the Genetic Algorithm, new solutions are generated during crossover. It always generates two new solutions, known as children, with the basis in two initial solutions, known as parents. The bit-swap operation used in this case is that the selected bit is interchanged between the two solutions and the evaluation of each interim solution is performed individually. After all bit-swaps are performed, the best interim solution for the first and the second child is selected individually, even though the best interim solution may be worse than the initial solution. The following implementation is used in this thesis, when applying the Analytic Swap method to the Genetic Algorithm:

1. **Initiation:**
2. **Input:** Two initial solution, x_0^1 and x_0^2.
3. Generate one empty solution table for each initial solution with layout: {solution, fitness}
4. **For:** (i=1; i<Max number of swaps; i++) **Do:**
5. {
6. Select random bit/digit not previously selected.
7. Generate new interim solutions, x_i^1 and x_i^2, by interchanging selected bit/digit between previous interim solutions, x_{i-1}^1 and x_{i-1}^2.
8. Evaluate fitness of each new interim solution, $F(x_i^1)$ and $F(x_i^2)$, individually.
9. Add interim solution, x_i^1 and x_i^2, and corresponding fitness value, $F(x_i^1)$ and $F(x_i^2)$, to their respective solution table.
10. }
11. **Finish:**
12. Return the interim solution with highest fitness value from each solution table.

Simulated Annealing requires a different adaptation. The generation of a new solution starts with one initial solution and the bit-swap operation used is bit inversion. After all bit-swaps have been performed, the best interim solution is selected. If the best interim solution is worse than the initial solution, the probability function in Simulated Annealing decides if the best interim solution is to be selected or not. If the probability function decides that the new solution is not to be selected, the Analytic Swap method is re-applied on the initial solution. The random order of bit-swaps will ensure that the new solution will be different than the previous generated solution. The following implementation is used in this thesis, when applying the Analytic Swap method to Simulate Annealing:

1. **Initiation:**
2. **Input:** One initial solution, x_0.
3. Generate empty solution table with layout: {solution, fitness}
4. **For:** (i=1; i<Max number of swaps; i++) **Do:**
5. {
6. Select random bit/digit not previously selected.
7. Generate new interim solution, x_i, by inverting selected bit/replacing selected digit with random digit in x_{i-1}.
8. Evaluate fitness of new interim solution, $F(x_i)$.
9. Add interim solution, x_i, and corresponding fitness value, $F(x_i)$, to solution table.
10. }
11. **Finish:**
12. Return the interim solution with highest fitness value.

6 Experiments and Results

The main objective of the experiments is to explore the performance of the two new methods presented in this thesis. The experiment setup is created by applying these methods together with the Genetic Algorithm and Simulated Annealing. For reference, other experiments without these methods have also been set up. The experiments were conducted by optimizing a set of ten well-known benchmark functions. Of these functions, six are functions with one variable and four are functions with two variables. Because the Tangent-based Evaluation method only supports functions with one variable, two different sets of experiments have been created. For more details, see introduction to chapter 6.1 and 6.2.

The efficiency of the optimization is measured by the number of iterations required to achieve convergence. One iteration is one cycle of the algorithm and is defined different for the Genetic Algorithm and Simulated Annealing. In the Genetic Algorithm, one iteration is the same as one generation. It includes all the steps from pairing to evaluation and selection. Thus, one iteration includes the generation of a set of new solution, known as children. In Simulated Annealing, one iteration is defined as one cycle of the inner loop. This always includes the generation of one new solution, evaluation and the accepting/discarding of the new solution. It is also important to notice that the temperature is not reduced in every iteration.

All experiments were performed by using Python as programing environment. For the implementation, some third-party libraries were used, with NumPy being the most important in terms of the mathematical calculations. Python was selected because of previous experience and for being well-suited for solving mathematical problem. Our implementation is based on an earlier implementation created by Thomas Nordli at the Vestfold University College and was used as a part of the research performed by Jensen, Bouhmala and Nordli. [9] The original implementation was a single file of Python-code and implemented The Tangent-based Evaluation method together with the Simple Genetic Algorithm with random pairing, uniform crossover and only bring the children population forward. Our implementation uses a modular design, which makes it easier to add new elements, such as new strategies for pairing, crossover, mutation and selection. We have also implemented additional strategies for exploring the different combinations, as described in chapter 2.2.

6.1 Functions with one variable

For functions with one variable, ten different experiment sets have been created, where six are based on the Genetic Algorithm and four are based on Simulated Annealing. For the experiments with Tangent-based Evaluation, six different h-values where used. These values are: 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}, 10^{-11} and 10^{-13}. Each experiment where perform 100 times for every function. Based on the results, the average result, excess and sample-variance were calculated and the minimum and maximum values were determined.

Following experiment setups were performed with the Genetic Algorithm and functions with one variable:

Setup	Crossover Strategy	Selection Strategy	Evaluation
GA1	Single Point	Elitism	One-point
GA2	Single Point	Elitism	Tangent
GA3	Uniform	Elitism	One-point
GA4	Uniform	Elitism	Tangent
GA5	Analytic Swap	Elitism	One-point
GA6	Analytic Swap	Elitism	Tangent

Table 3 – Experiment set with the Genetic Algorithm for optimizing one-variable functions.

Following experiment setups were performed with the Simulated Annealing and functions with one variable:

Setup	Solution Generation	Evaluation
SA1	Replace Random Digit	One-point
SA2	Replace Random Digit	Tangent
SA3	Analytic Swap	One-point
SA4	Analytic Swap	Tangent

Table 4 – Experiment set with the Simulated Annealing for optimizing one-variable functions.

Due to the large amount of results, only the most important results are presented in this chapter. A complete set of results of the experiments with one-variable functions is presented in Appendix C.

6.1.1 Function 1.1

$$f_{1.1}(x) = e^x$$

$$g_{1.1}(x) = e^x$$

Optimal solution: $f_{1.1}(1) = e \sim 2.718$

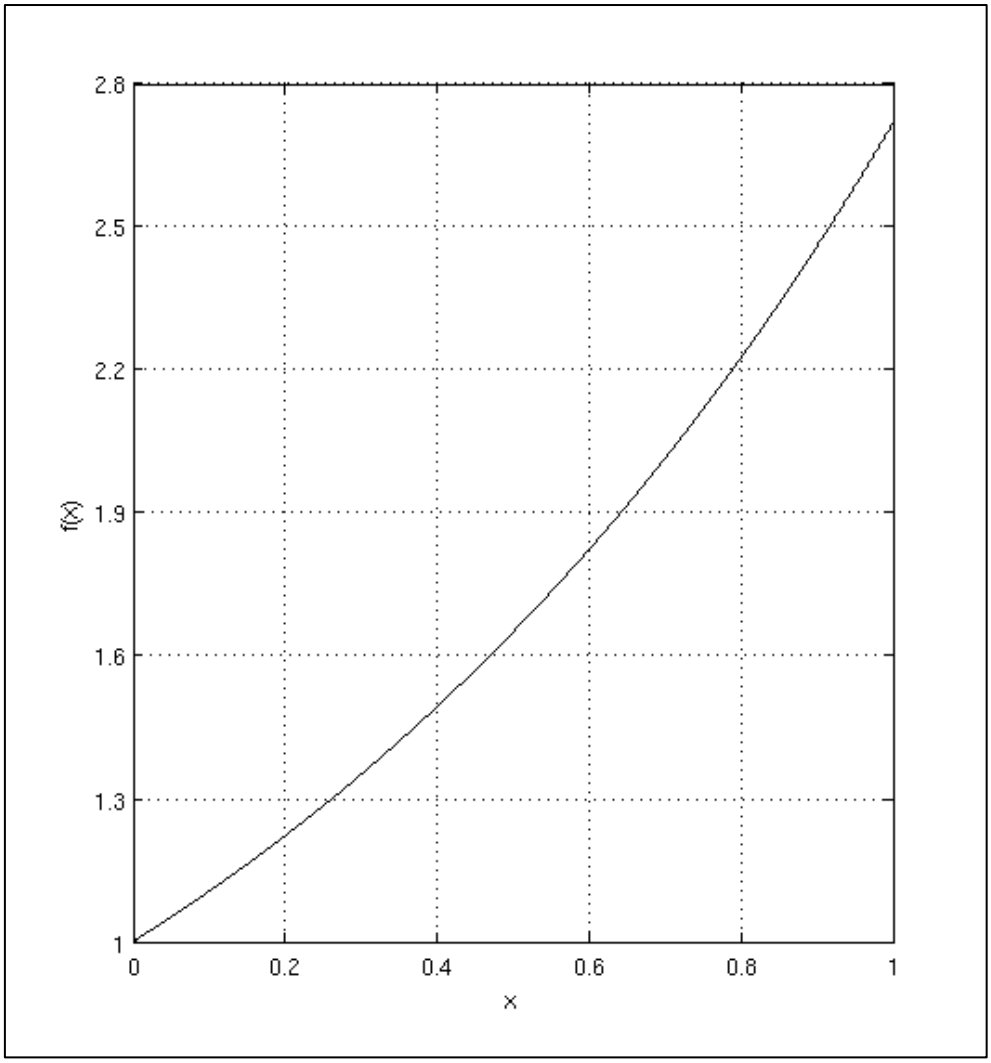


Figure 12 – Plot of function 1.1 for x ∈ [0,1].

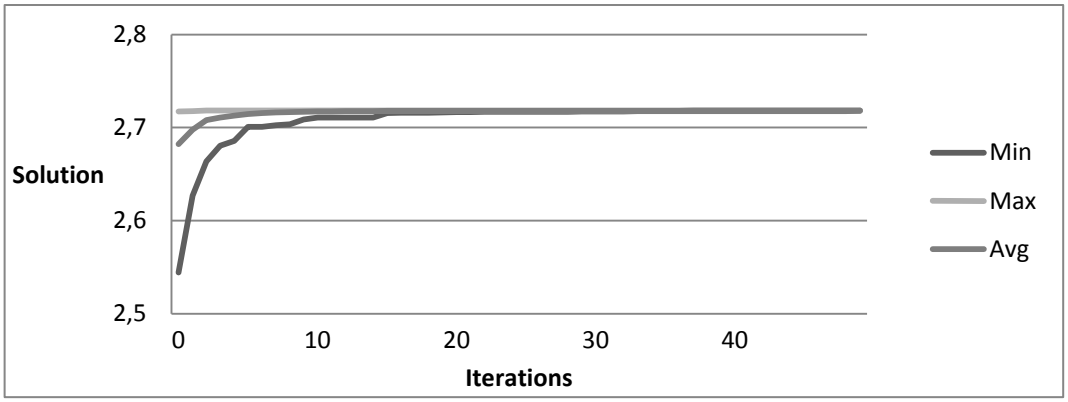


Figure 13 – Experiment 1.1_GA3, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and One-point Evaluation.

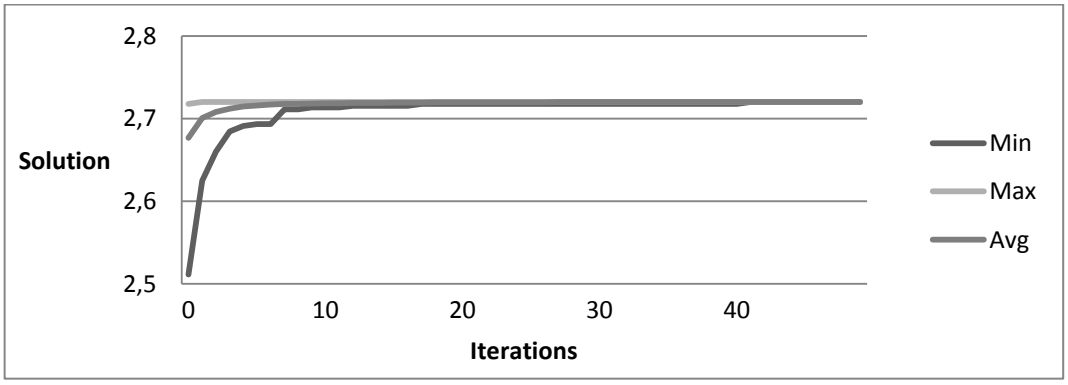


Figure 14 – Experiment 1.1_GA4, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and Tangent-based Evaluation, $H=10^{-13}$.

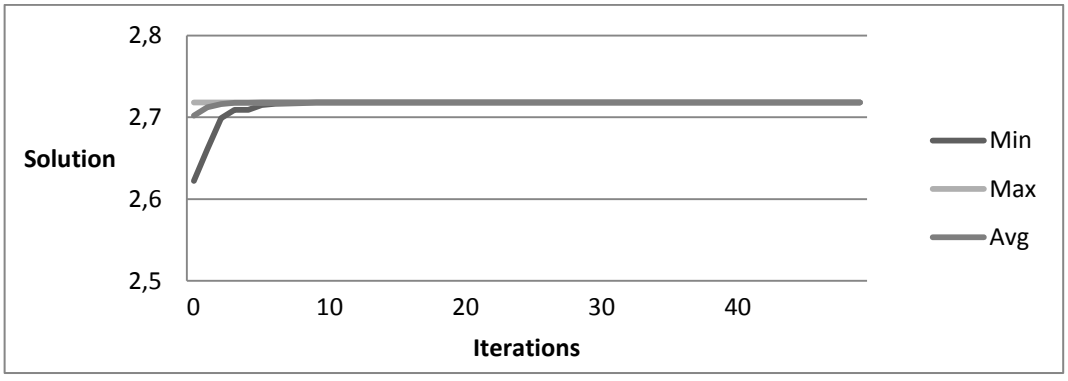


Figure 15– Experiment 1.1_GA5, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and One-point Evaluation.

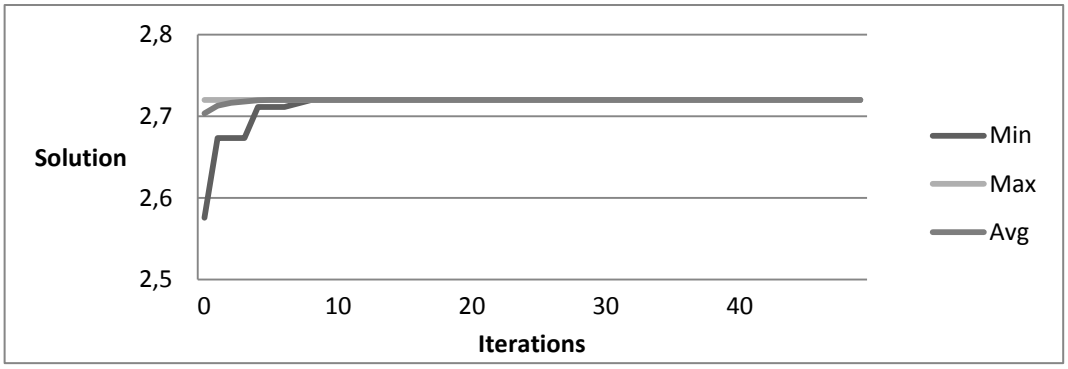


Figure 16 – Experiment 1.1_GA6, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and Tangent-based Evaluation, H=10^{-13}.

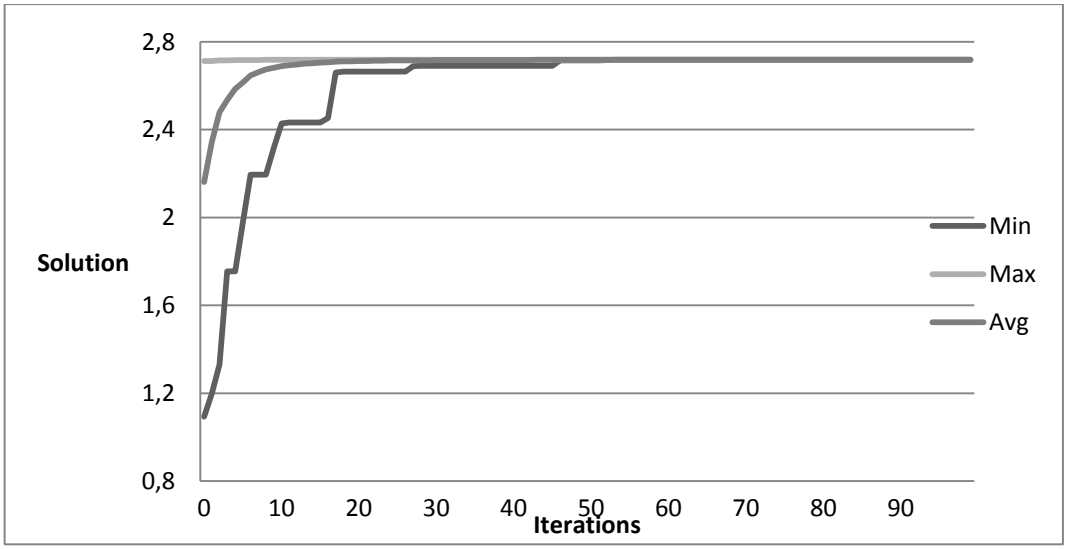


Figure 17 – Experiment 1.1_SA1, Optimizing with Simulated Annealing and One-point evaluation.

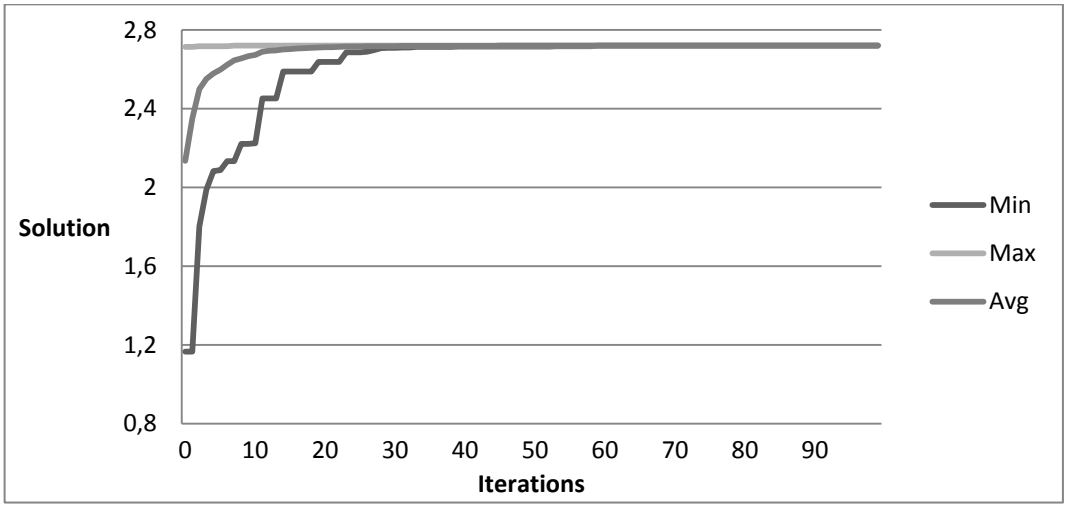


Figure 18 – Experiment 1.1_SA2, Optimizing with Simulated Annealing and Tangent-based Evaluation. H=10^{-13}.

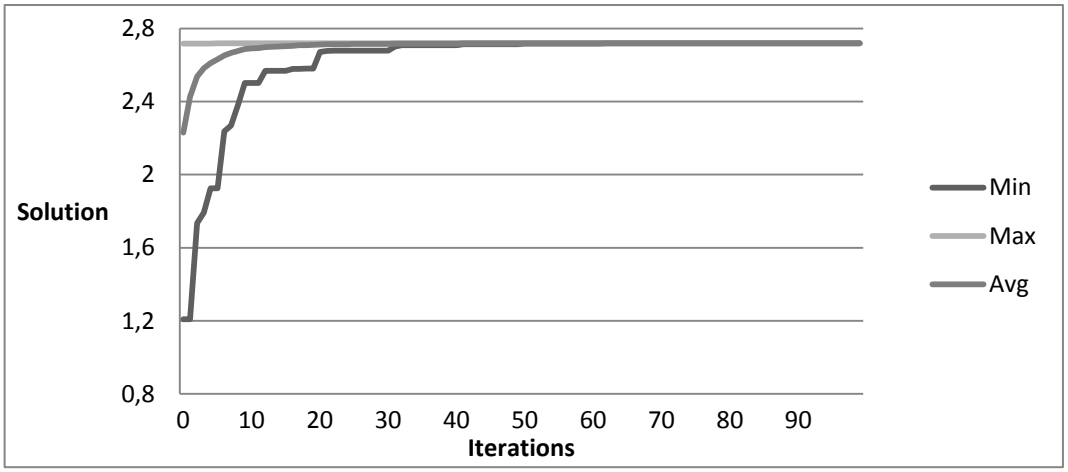


Figure 19 – Experiment 1.1_SA3, Optimizing with Simulated Annealing, Analytic Swap and One-point evaluation.

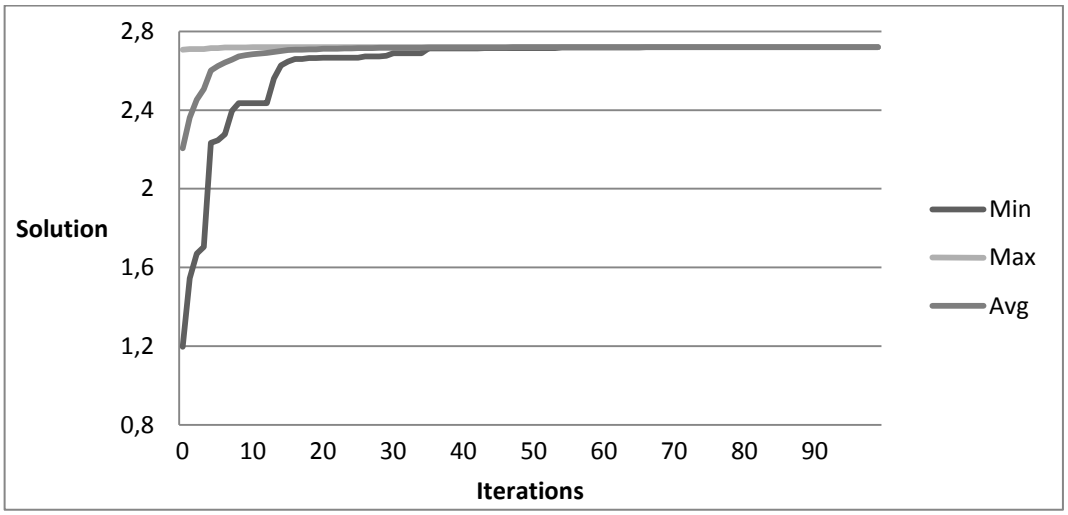


Figure 20 – Experiment 1.1_SA4, Optimizing with Simulated Annealing, Analytic Swap and Tangent-based Evaluation. H=10^{-13}.

	Setup	h	Maximum	Minimum	Average	Variance	Excess
1.1_GA1	GA1	–	2,71799138	2,59694631	2,70089926	0,00043407	0, 17382568
1.1_GA2	GA2	10^{-13}	2,72004641	2,62234678	2,699018786	0,00047665	0, 19263042
1.1_GA3	GA3	–	2,718281269	2,717804741	2,718197598	1,01109E-08	8,423E-5
1.1_GA4	GA4	10^{-13}	2,72004641	2,72004641	2,72004641	2,86859E-29	-0,001764581
1.1_GA5	GA5	–	2,718281828	2,718194845	2,718280697	7,76222E-11	1,135E-6
1.1_GA6	GA6	10^{-13}	2,72004641	2,72004641	2,72004641	2,86859E-29	-0,001764582
1.1_SA1	SA1	–	2,718281824	2,718277567	2,718281167	6,91991E-13	6,615E-7
1.1_SA2	SA2	10^{-13}	2,72004641	2,72004641	2,72004641	2,86859E-29	-0,001764581
1.1_SA3	SA3	–	2,718281827	2,71827087	2,718280584	2,71713E-12	1,245E-6
1.1_SA4	SA4	10^{-13}	2,72004641	2,72004641	2,72004641	2,86859E-29	-0,001764581

Table 5 – Results of optimizing Function 1.1.

6.1.2 Function 1.2

$$f_{1.2}(x) = \frac{1}{1+x^2}$$
$$g_{1.2}(x) = \tan^{-1}(x)$$

Optimal solution: $f_{1.2}(0) = 1$

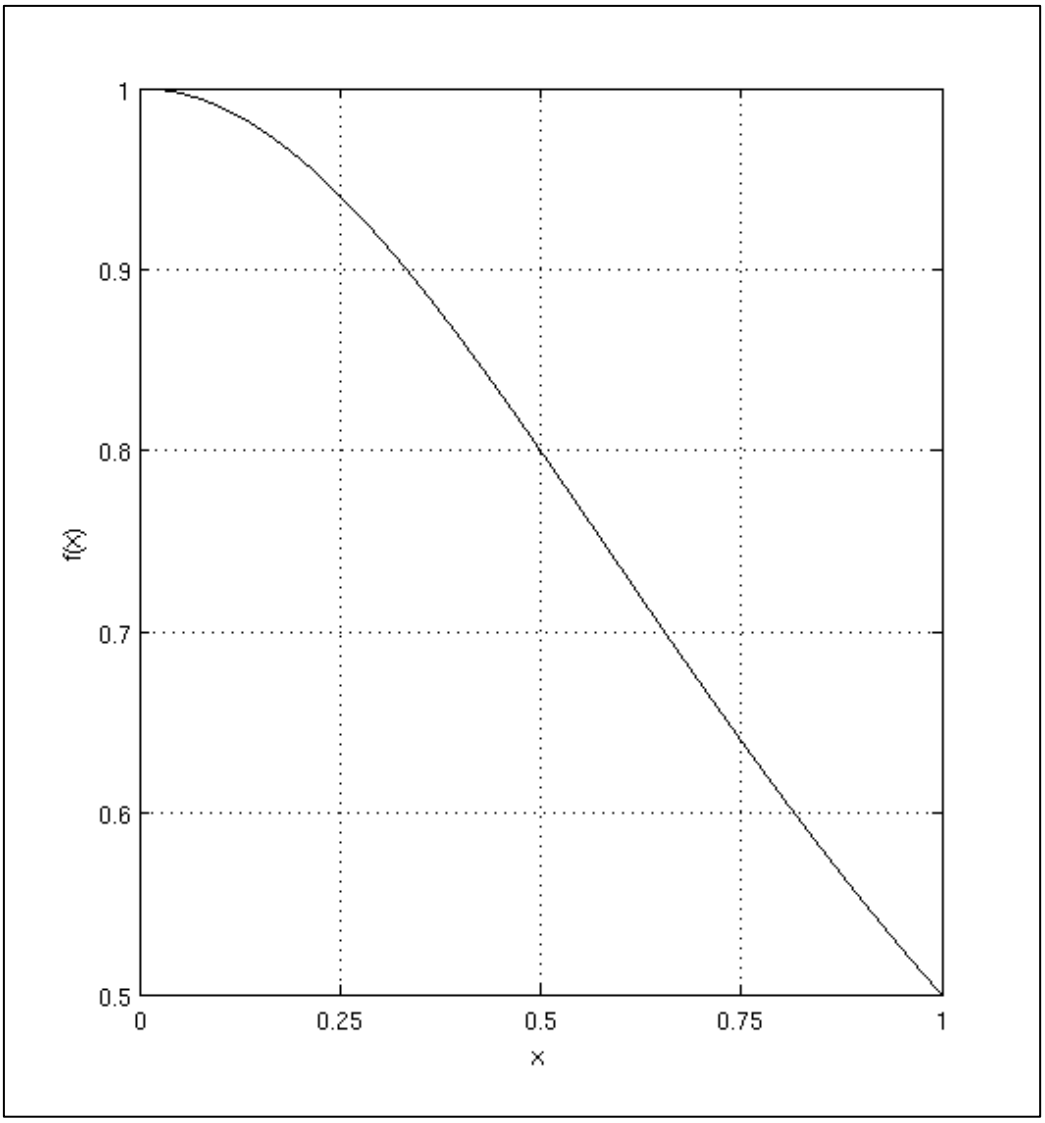


Figure 21 – Plot of function 1.2 for x ∈ [0,1].

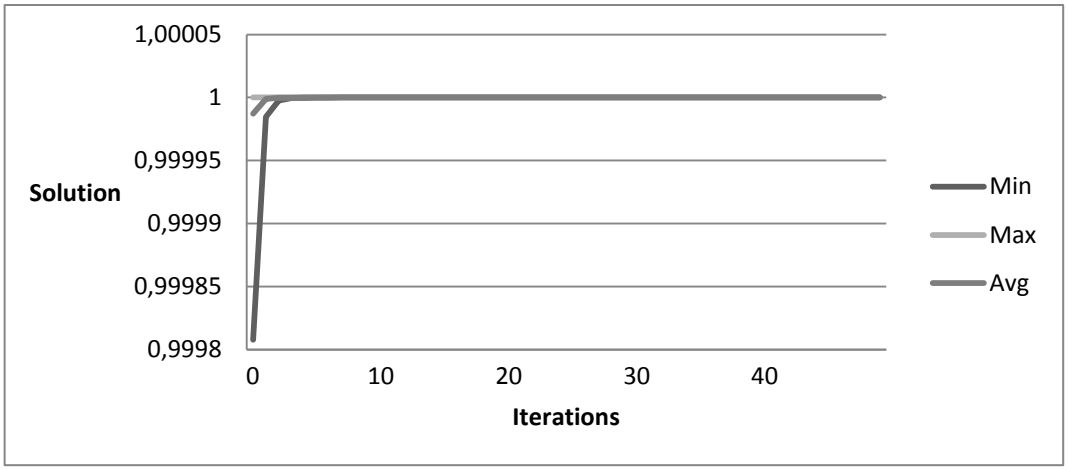


Figure 22 – Experiment 1.2_GA3, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and One-point Evaluation.

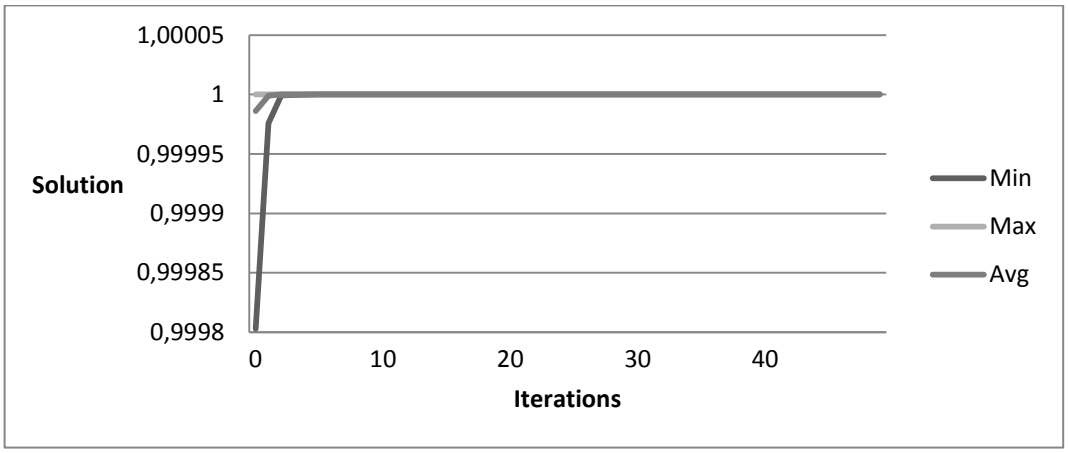


Figure 23 – Experiment 1.1_GA4, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and Tangent-based Evaluation, $H=10^{-5}$.

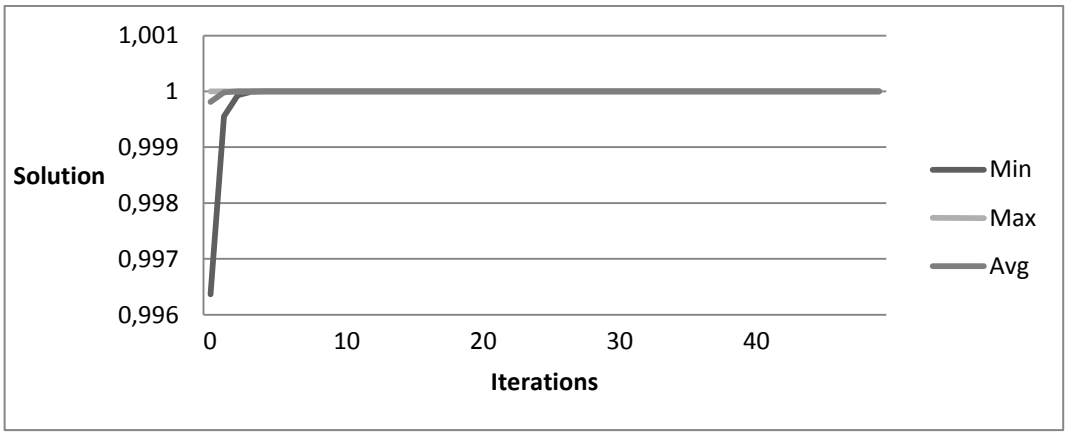


Figure 24– Experiment 1.2_GA5, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and One-point Evaluation.

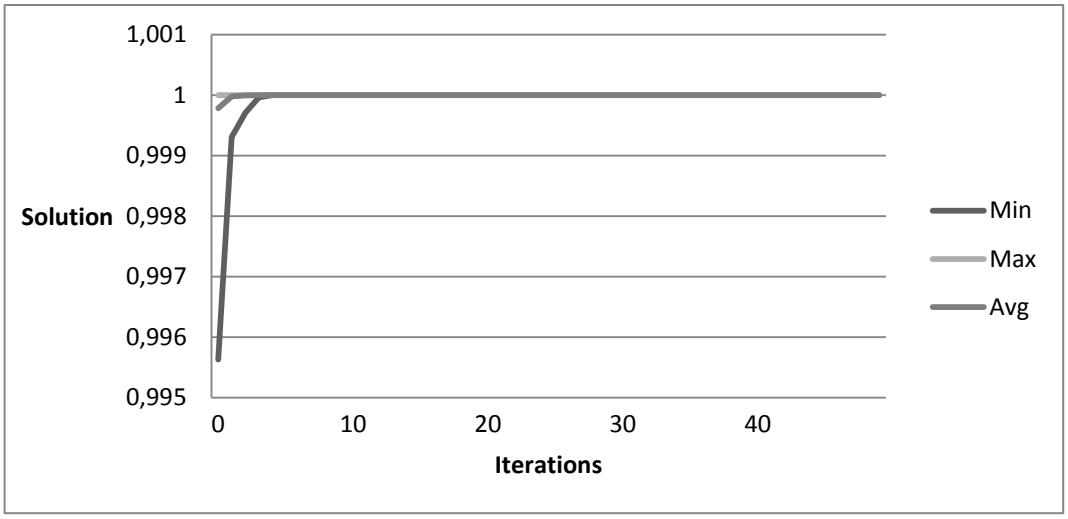


Figure 25 – Experiment 1.2_GA6, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and Tangent-based Evaluation, H=10^{-5}.

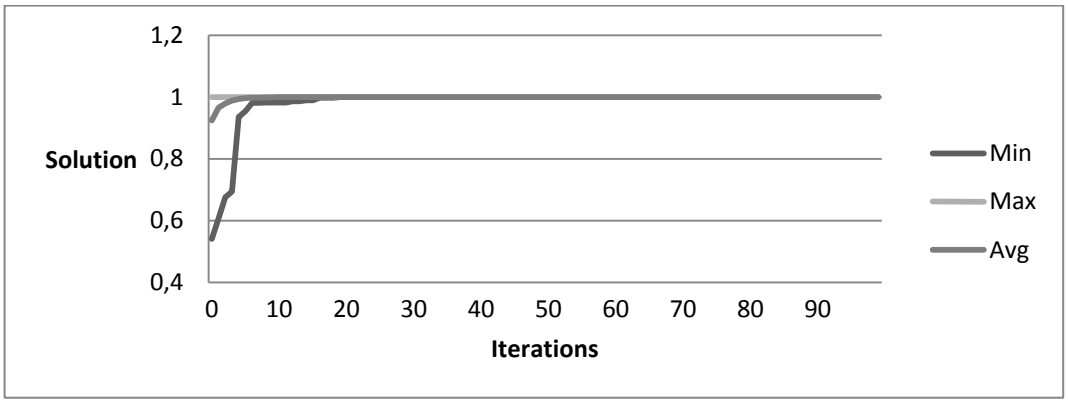


Figure 26 – Experiment 1.2_SA1, Optimizing with Simulated Annealing and One-point evaluation.

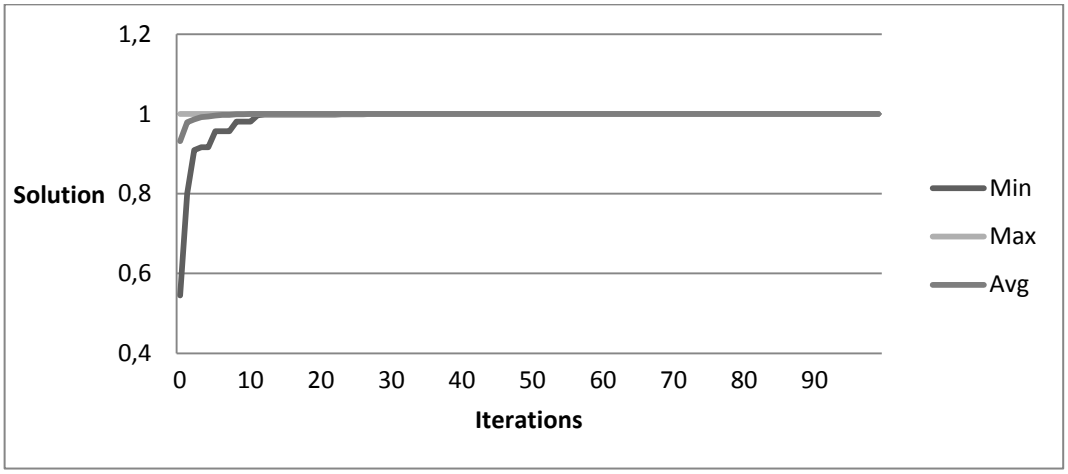


Figure 27 – Experiment 1.2_SA2, Optimizing with Simulated Annealing and Tangent-based Evaluation. H=10^{-5}.

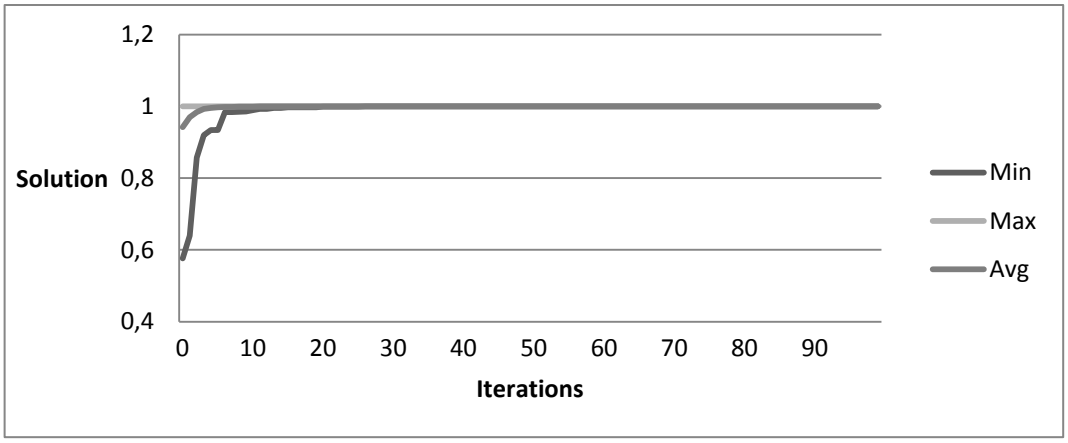


Figure 28 – Experiment 1.2_SA3, Optimizing with Simulated Annealing, Analytic Swap and One-point evaluation.

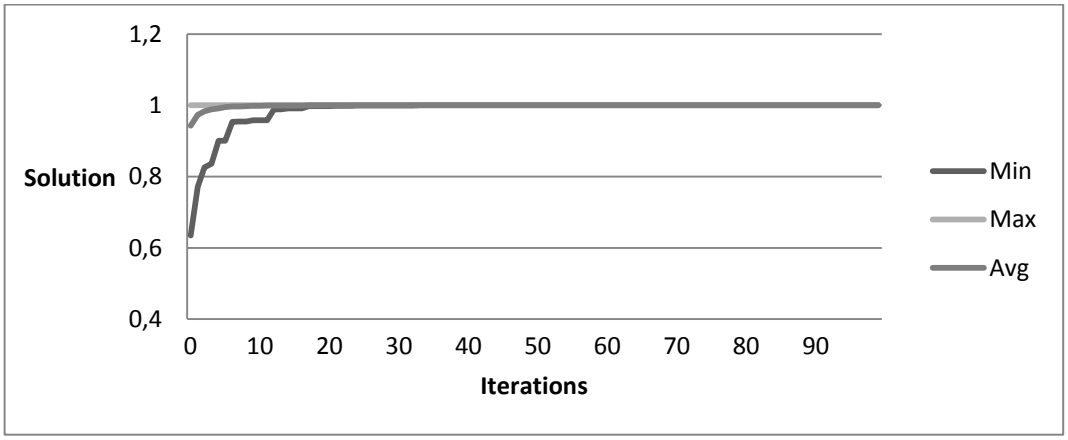


Figure 29 – Experiment 1.2_SA4, Optimizing with Simulated Annealing, Analytic Swap and Tangent-based Evaluation. H=10^{-5}.

	Setup	h	Maximum	Minimum	Average	Variance	Excess
1.2_GA1	GA1	–	1	0,99749277	0,99986237	9,70299E-08	0,00013763
1.2_GA2	GA2	10^{-13}	1,00000086	0,998368055	0,99988152	8,46176E-08	0,00011848
1.2_GA3	GA3	–	1	1	1	–	–
1.2_GA4	GA4	10^{-5}	1	1	1	–	–
1.2_GA5	GA5	–	1	1	1	–	–
1.2_GA6	GA6	10^{-5}	1	1	1	–	–
1.2_SA1	SA1	–	1	0,99999996	0,999999996	–	4E-8
1.2_SA2	SA2	10^{-5}	1	0,999999972	1	2,99238E-17	–
1.2_SA3	SA3	–	1	0,99999988	0,99999999	2,93548E-16	1E-7
1.2_SA4	SA4	10^{-5}	1	0,99999994	0,99999999	1,86744E-16	1E-8

Table 6 – Results of optimizing Function 1.2.

6.1.3 Function 1.3

$$f_{1.3}(x) = \sum_{n=1}^{20} \left(\left(x - \frac{1}{10} \right)^n - 10 \cos\left(2\pi n \left(x - \frac{1}{10} \right) \right) \right)$$

$$g_{1.3}(x) = \sum_{n=1}^{20} \left(\frac{\left(x - \frac{1}{10} \right)^{n+1}}{n+1} - \frac{5}{\pi n} \sin\left(2\pi n \left(x - \frac{1}{10} \right) \right) \right)$$

Optimal solution: $f_{1.3}(0{,}134) \approx 49{,}595$

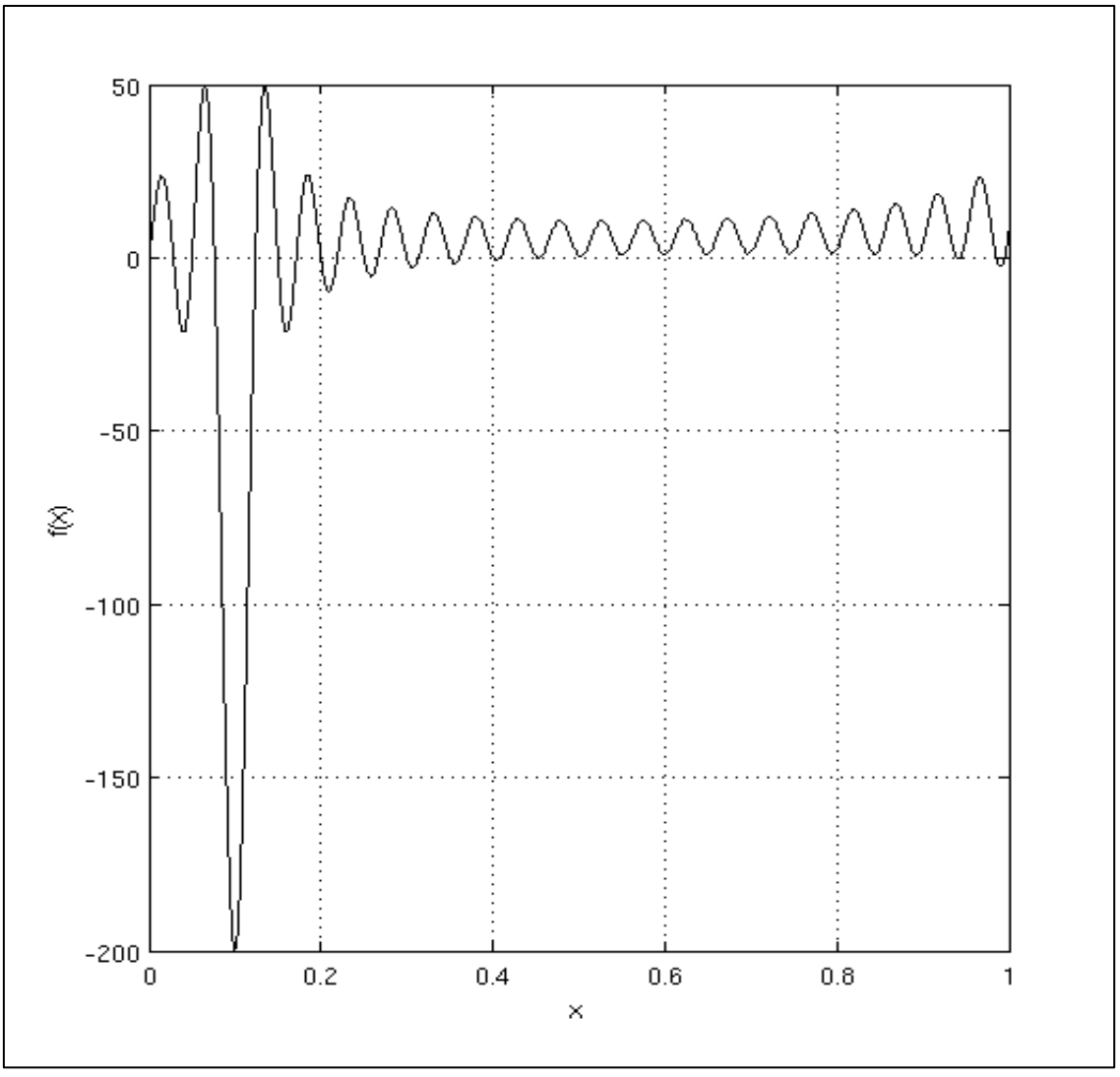


Figure 30 – Plot of function 1.3 for x ∈ [0,1].

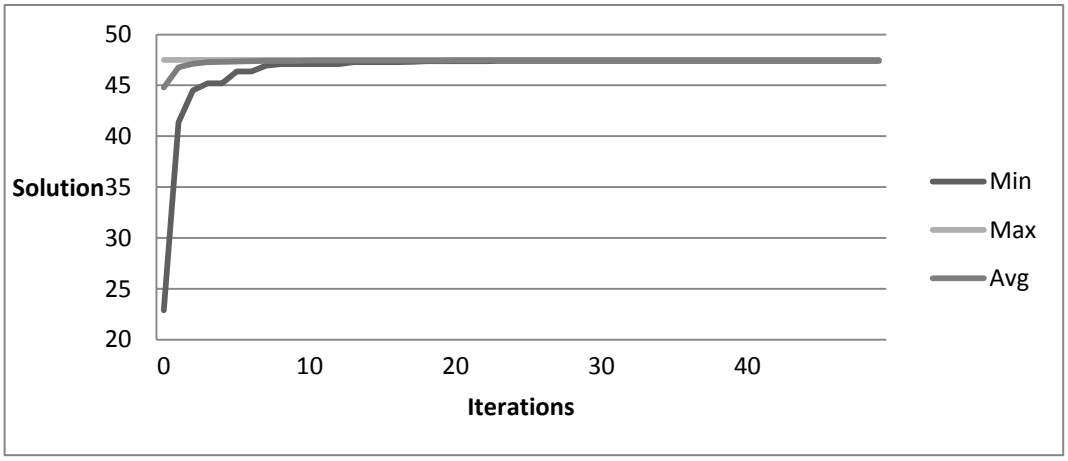


Figure 31 – Experiment 1.3_GA3, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and One-point Evaluation.

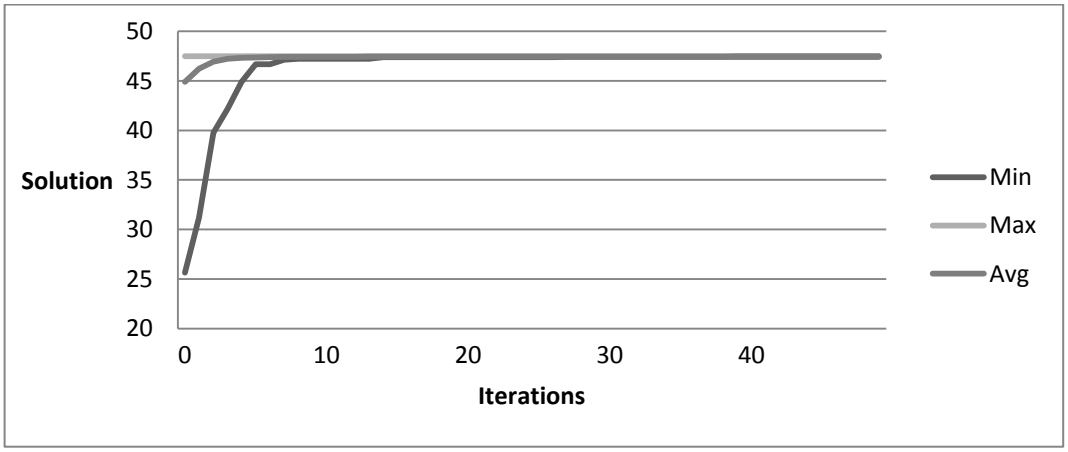


Figure 32 – Experiment 1.3_GA4, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and Tangent-based Evaluation, $H=10^{-5}$.

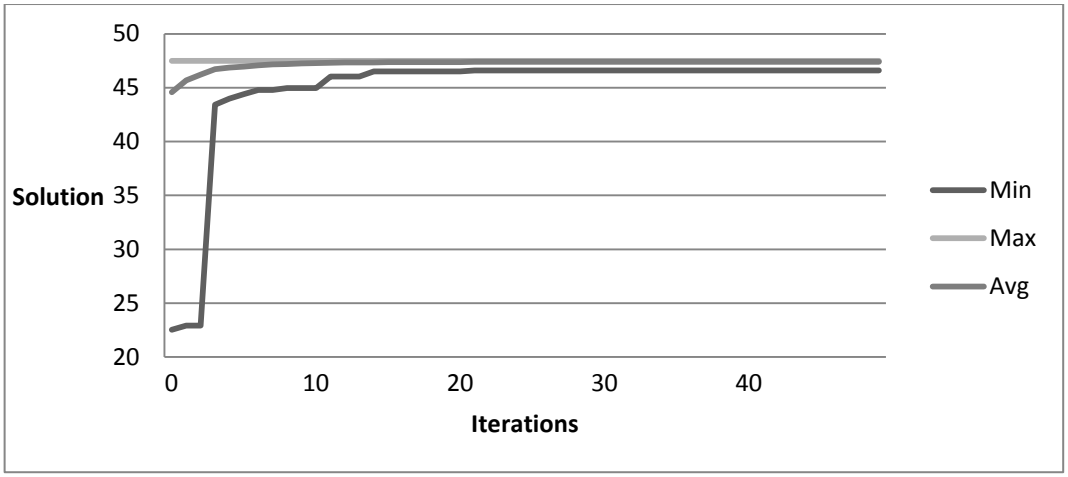


Figure 33 – Experiment 1.3_GA5, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and One-point Evaluation.

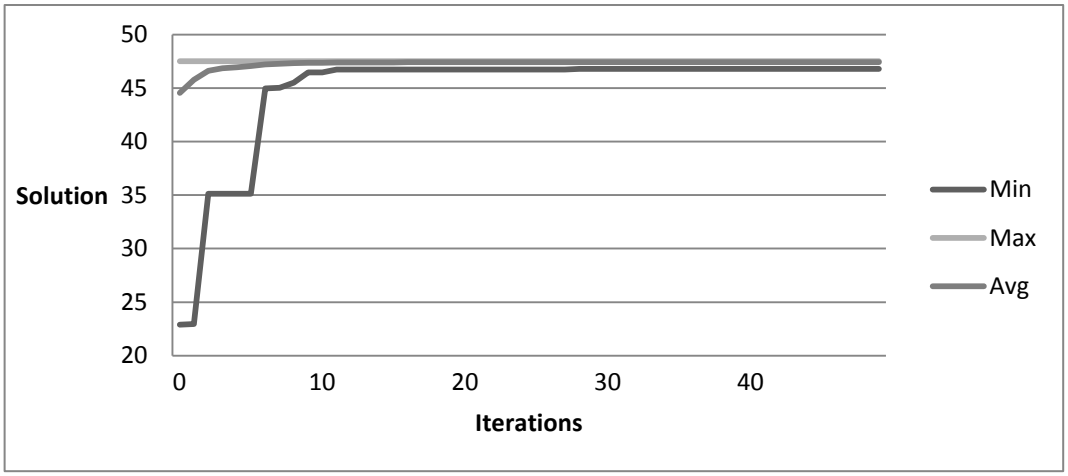


Figure 34 – Experiment 1.3_GA6, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and Tangent-based Evaluation, H=10^{-13}.

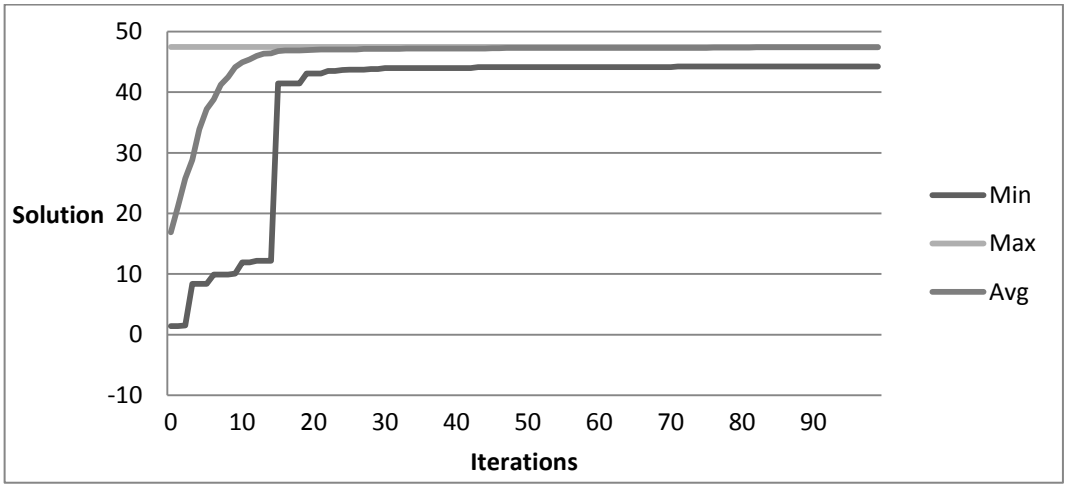


Figure 35 – Experiment 1.3_SA1, Optimizing with Simulated Annealing and One-point evaluation.

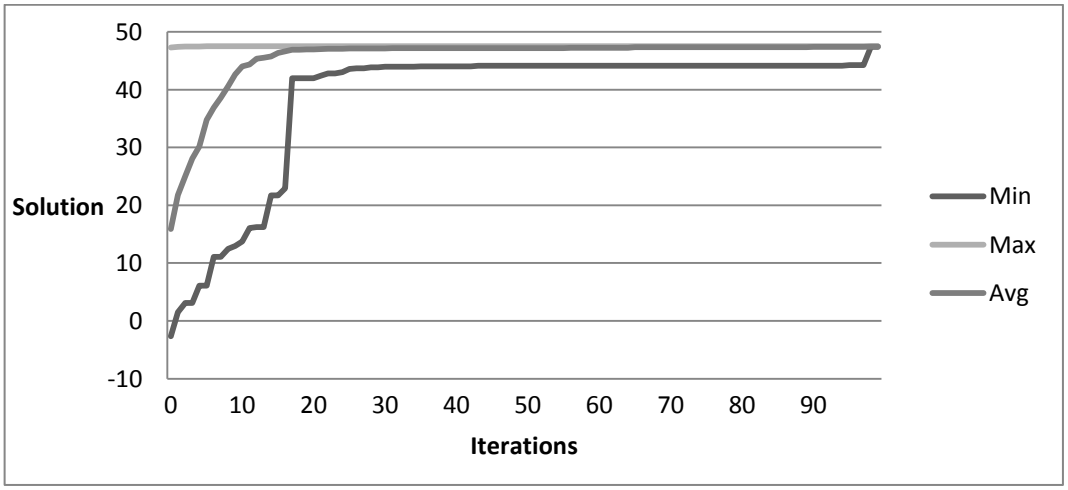


Figure 36 – Experiment 1.3_SA2, Optimizing with Simulated Annealing and Tangent-based Evaluation. H=10^{-5}.

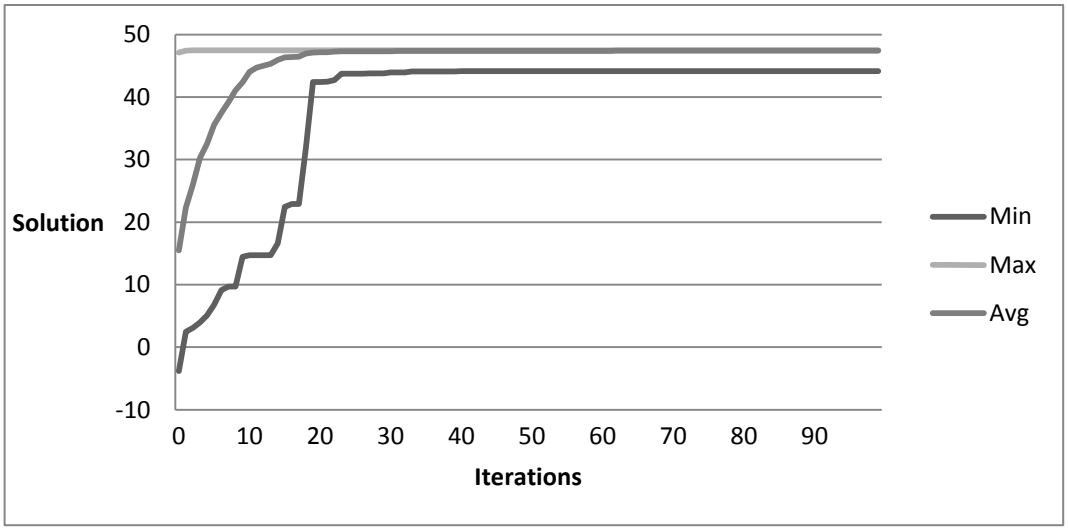


Figure 37 – Experiment 1.3_SA3, Optimizing with Simulated Annealing, Analytic Swap and One-point evaluation.

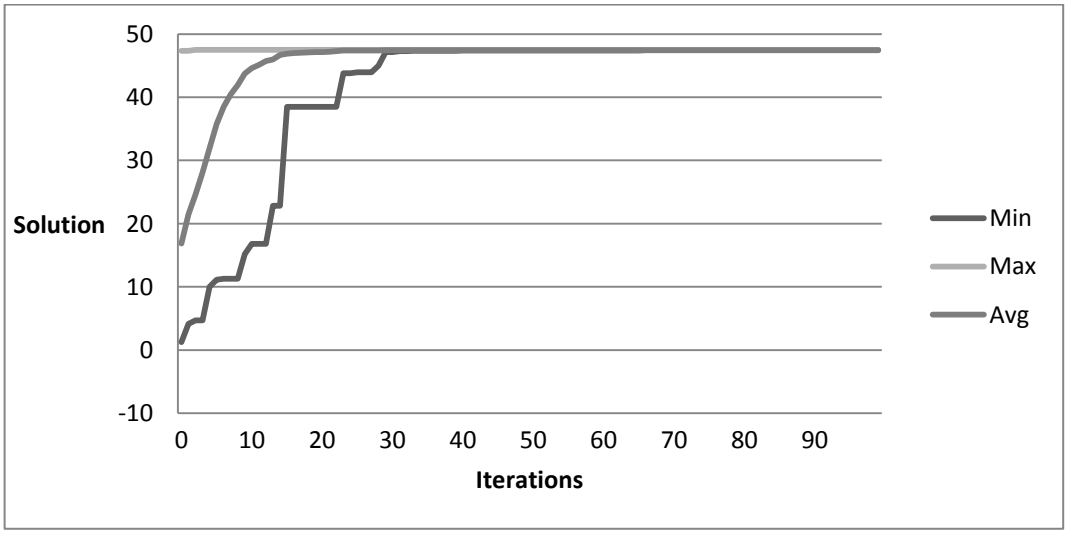


Figure 38 – Experiment 1.3_SA4, Optimizing with Simulated Annealing, Analytic Swap and Tangent-based Evaluation. H=10^{-5}.

	Setup	h	Maximum	Minimum	Average	Variance	Excess
1.3_GA1	GA1	–	47,4925022	23,0082964	45,0965458	18,375547	4,4984542
1.3_GA2	GA2	10^{-11}	47,492521	22,4448016	45,732363	11,9377222	3,862637
1.3_GA3	GA3	–	47,4925237	47,4166783	47,4754847	0,00065416	2,1195153
1.3_GA4	GA4	10^{-5}	47,4925131	47,4168554	47,4776476	0,00052161	2,1173524
1.3_GA5	GA5	–	47,4925237	46,600904	47,4040929	0,02263649	2,1909071
1.3_GA6	GA6	10^{-13}	47,5064432	47,4282169	47,4282169	0,01976401	2,1667831
1.3_SA1	SA1	–	47,4925237	44,2286558	47,3894619	0,20726851	2,2055381
1.3_SA2	SA2	10^{-5}	47,4925131	47,4190441	47,4599743	0,00130498	2,1350257
1.3_SA3	SA3	–	47,4925237	44,1485092	47,4186748	0,1104435	2,1763252
1.3_SA4	SA4	10^{-13}	47,5064432	47,4265072	47,4634249	0,00135318	2,1315751

Table 7 – Results of optimizing Function 1.3.

6.2 Functions with two variables

For functions with two variables, five different experiment sets have been created, where three are based on the Genetic Algorithm and two are based on Simulated Annealing. Each experiment where perform 100 times for every function. Based on the results, the average solution, excess and sample-variance were calculated and the minimum and maximum values were determined.

Following experiment setups were performed with the Genetic Algorithm and functions with two variables:

Setup	Crossover Strategy	Selection Strategy	Evaluation
GA1	Single Point	Elitism	One-point
GA2	Uniform	Elitism	One-point
GA3	Analytic Swap	Elitism	One-point

Table 8 – Experiment set with the Genetic Algorithm for optimizing two-variable functions.

Following experiment setups were performed with Simulated Annealing and functions with two variables:

Setup	Solution Generation	Evaluation
SA1	Replace Random Digit	One-point
SA2	Analytic Swap	One-point

Table 9 - Experiment set with the Simulated Annealing for optimizing two-variable functions.

6.2.1 Function 2.1

$$f_{2.1}(x, y) = x^2 + y^2$$

Optimal solution: $f_{2.1}(1,1) = 2$

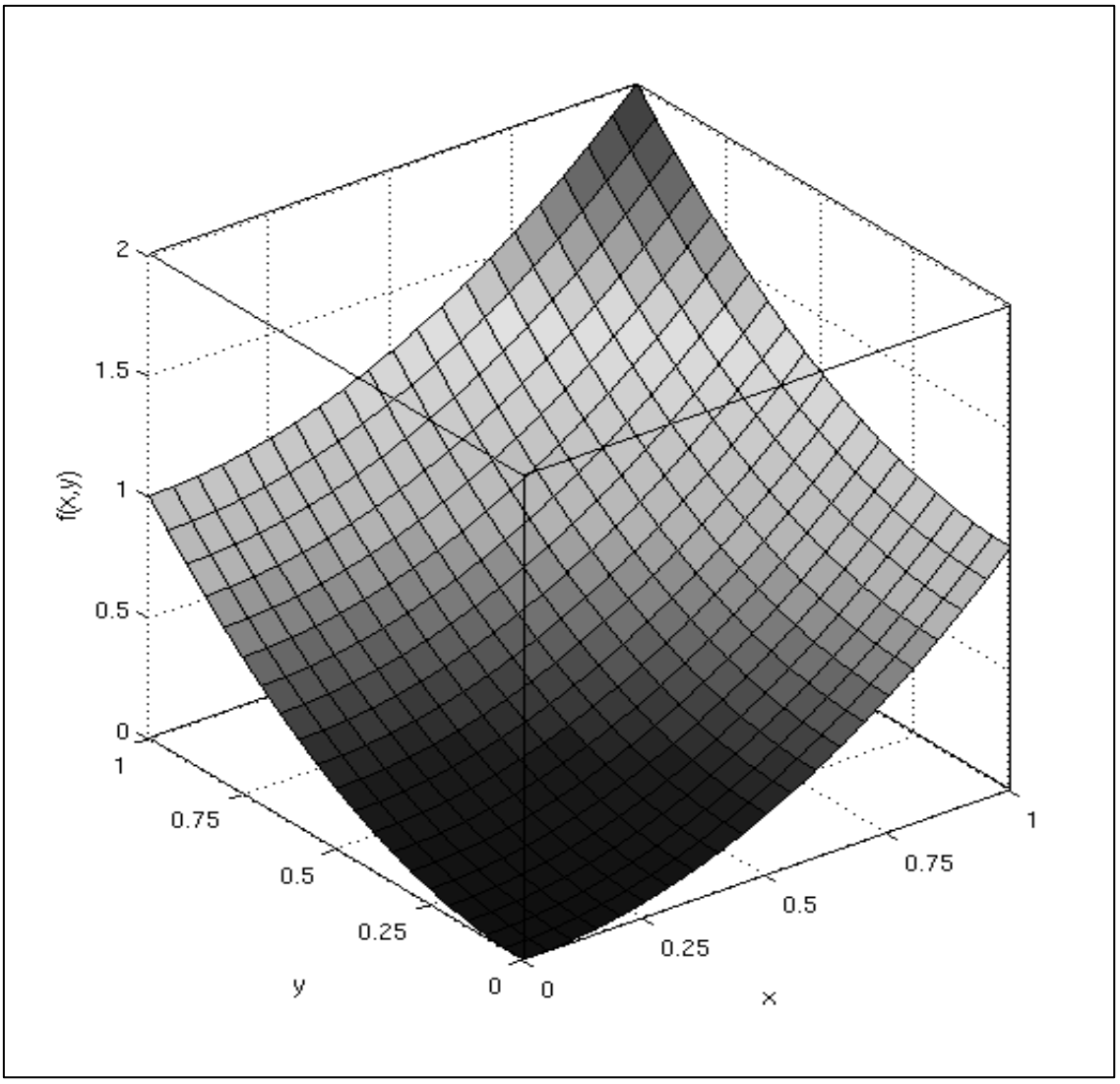


Figure 39 – Plot of function 2.1 for x ∈ [0,1] and y ∈ [0,1].

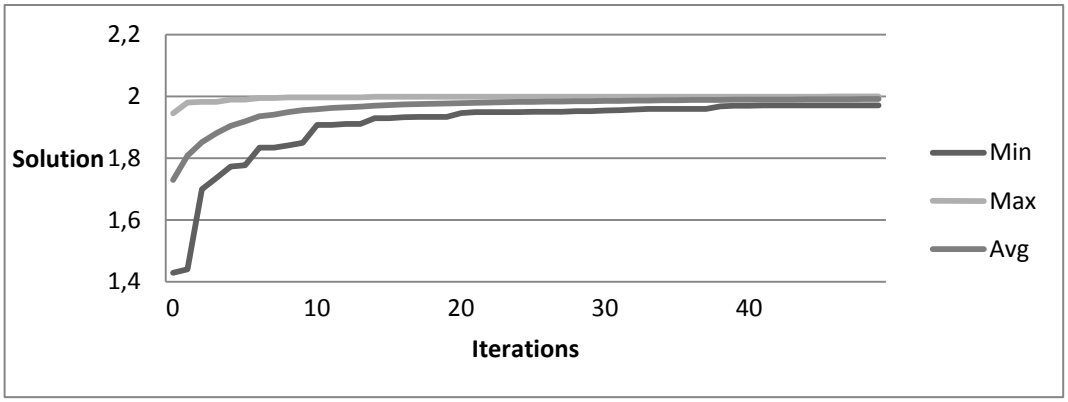


Figure 40 – Experiment 2.1_GA2, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and One-point Evaluation.

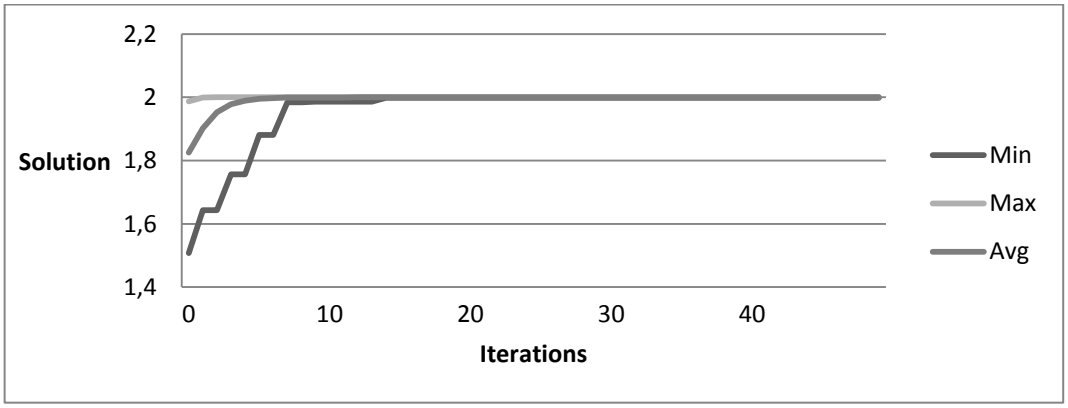


Figure 41 – Experiment 2.1_GA3, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and One-point Evaluation.

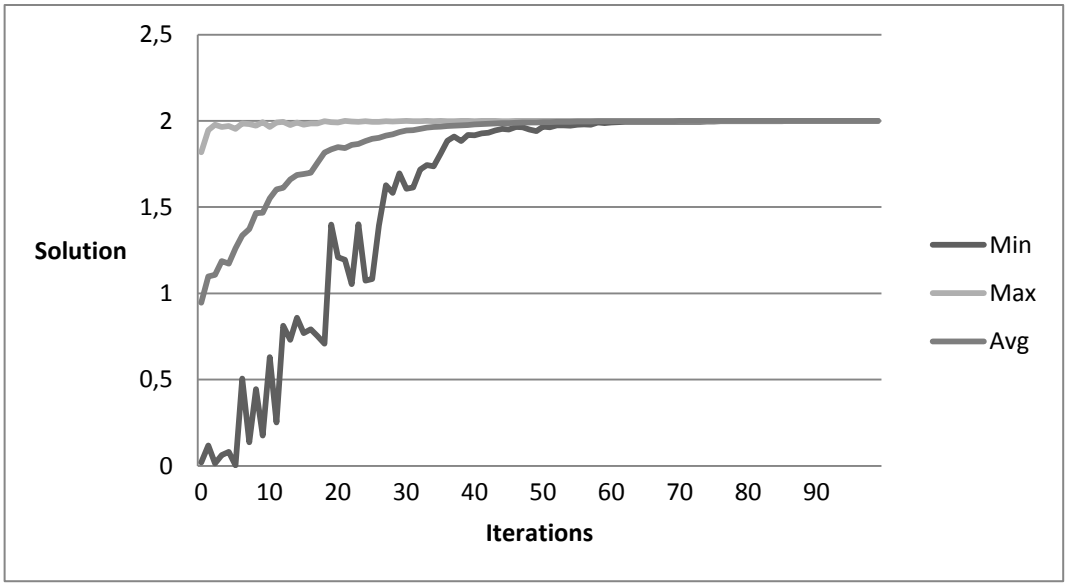


Figure 42 – Experiment 2.1_SA1, Optimizing with Simulated Annealing and One-point evaluation.

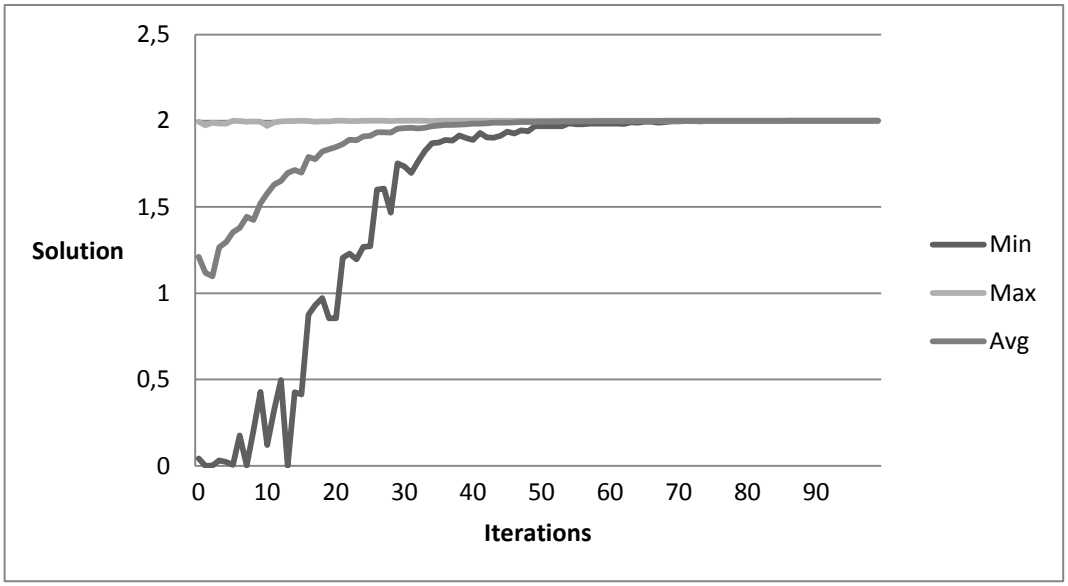


Figure 43 – Experiment 2.1_SA2, Optimizing with Simulated Annealing, Analytic Swap and One-point evaluation.

	Setup	Maximum	Minimum	Average	Variance	Excess
2.1GA1	GA1	1,99432772	1,58187959	1,88544878	0,00662649	0,11455122
2.1GA2	GA2	1,99979533	1,97048235	1,99124449	4,1053E-05	0,00875551
2.1GA3	GA3	2,00000000	1,99859847	1,99998347	1,9784E-08	1,653E-5
2.1SA1	SA1	1,99999678	1,99972327	1,99995906	1,85591E-09	4,094E-5
2.1SA2	SA2	1,99999892	1,99967220	1,99994566	4,423E-09	5,454E-5

Table 10 – Results of optimizing Function 2.1.

6.2.2 Function 2.2

$$f_{2.2}(x, y) = x^2 + 2y^2$$

Optimal solution: $f_{2.2}(1,1) = 3$

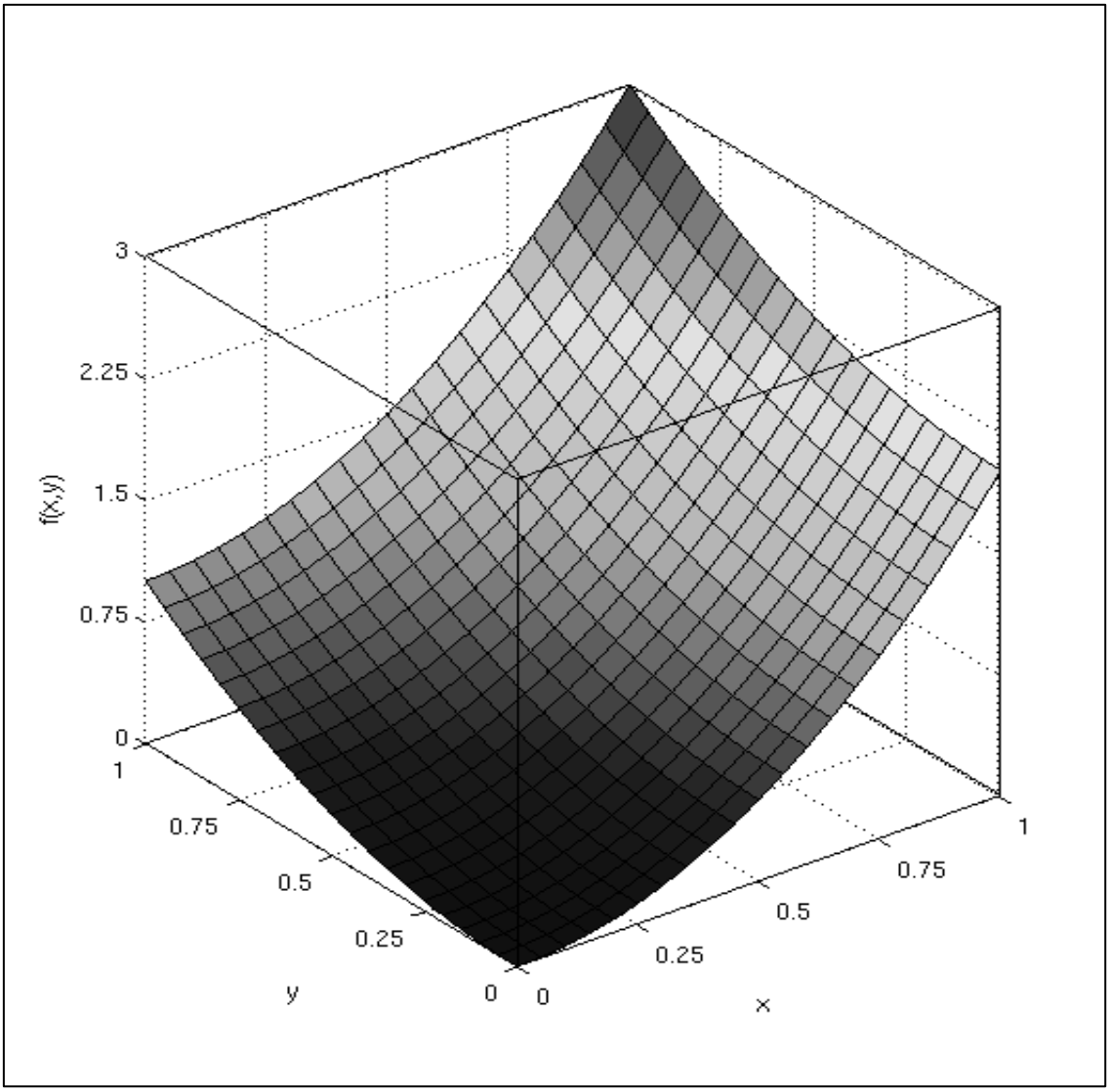


Figure 44 – Plot of function 2.2 for x ∈ [0,1] and y ∈ [0,1].

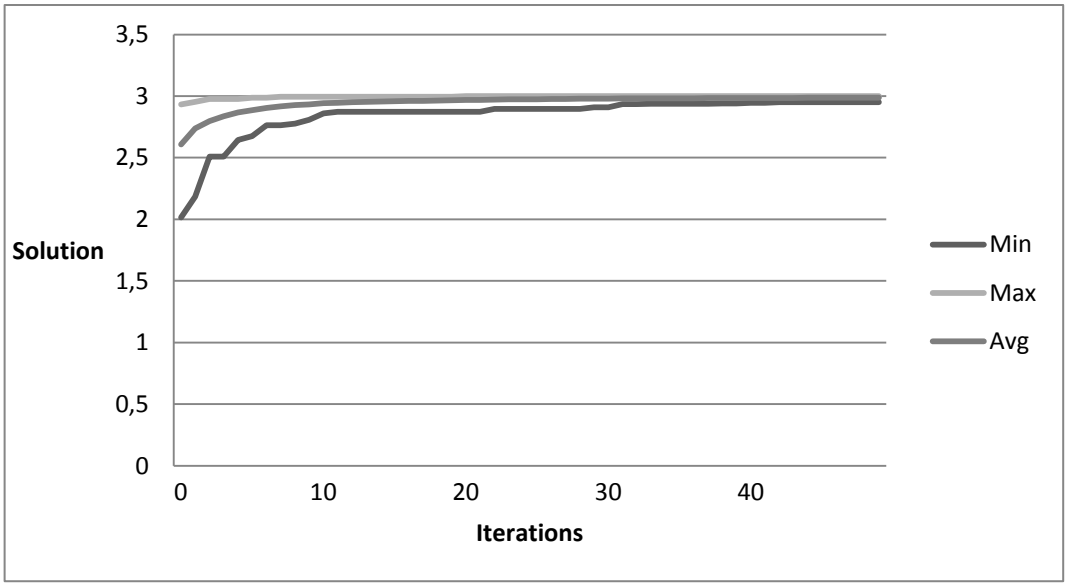


Figure 45 – Experiment 2.2_GA2, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and One-point Evaluation.

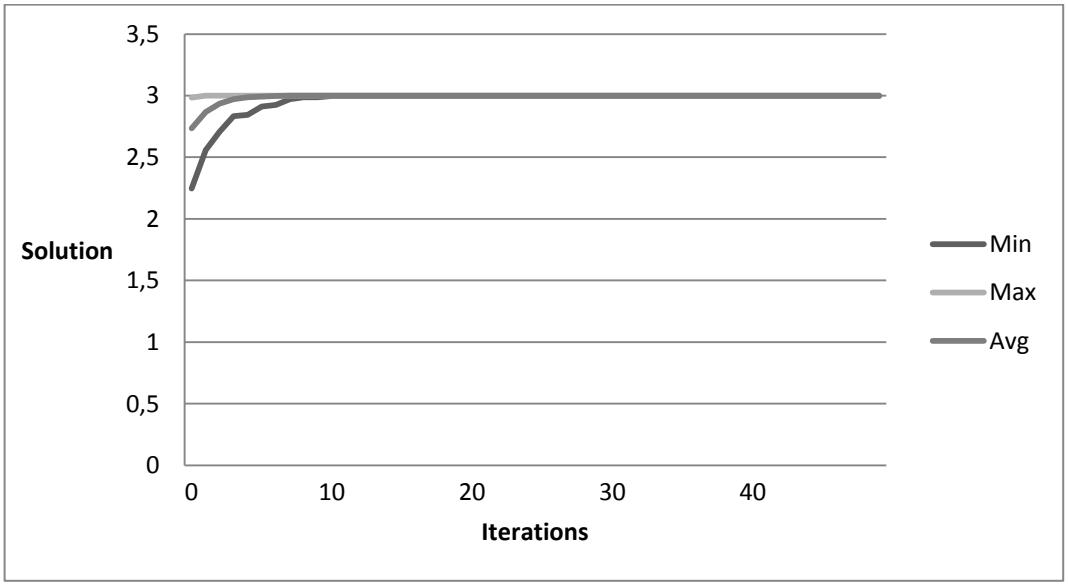


Figure 46 – Experiment 2.2_GA3, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and One-point Evaluation.

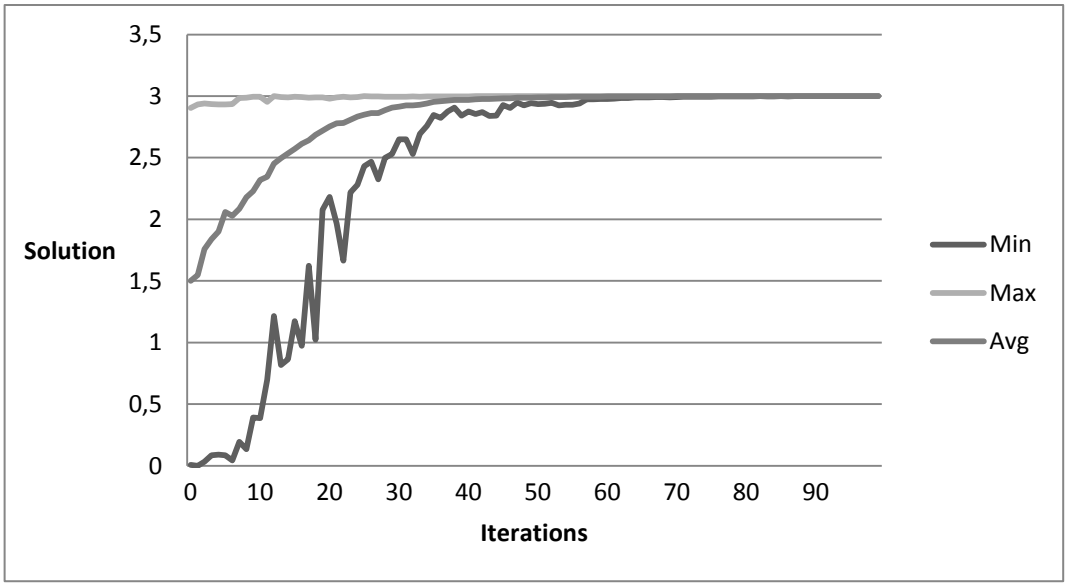


Figure 47 – Experiment 2.2_SA1, Optimizing with Simulated Annealing and One-point evaluation.

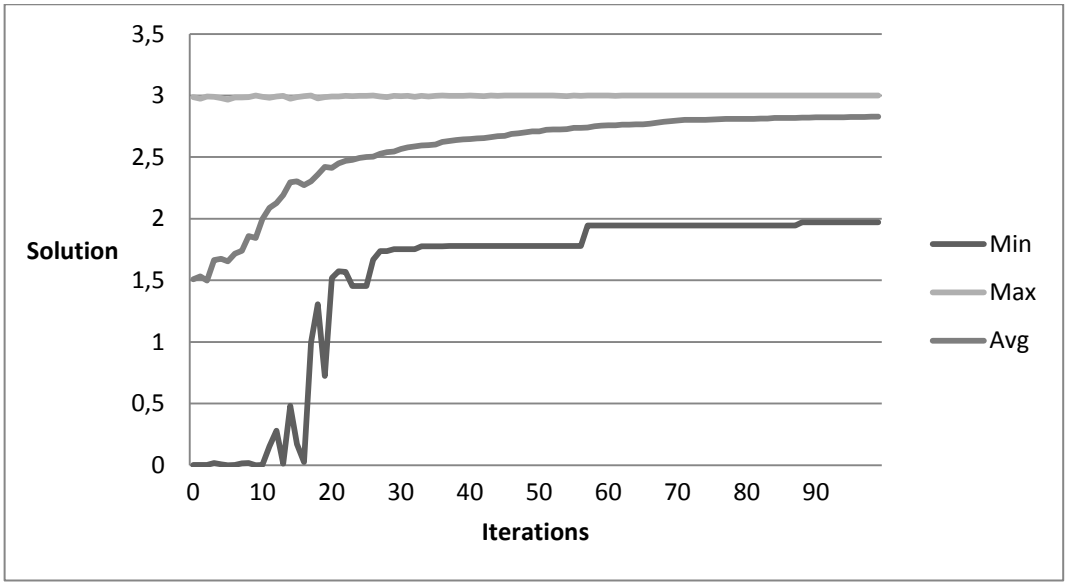


Figure 48 – Experiment 2.2_SA2, Optimizing with Simulated Annealing, Analytic Swap and One-point evaluation.

	Setup	Maximum	Minimum	Average	Variance	Excess
2.2GA1	GA1	2,99361365	2,39503481	2,85645164	0,0138519	0,14354836
2.2GA2	GA2	2,99870218	2,95113875	2,98678531	9,5841E-05	0,1321469
2.2GA3	GA3	3	2,998001	2,99997008	4,2061E-08	2,992E-5
2.2SA1	SA1	2,99999957	2,99957144	2,99993612	4,807E-09	6,388E-5
2.2SA2	SA2	2,99999085	1,97074912	2,82764683	0,06086346	0,17235317

Table 11 – Results of optimizing Function 2.2.

6.2.3 Function 2.3

$$f_{2.3}(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

Optimal solution: $f_{2.3}(0,1) = 101$

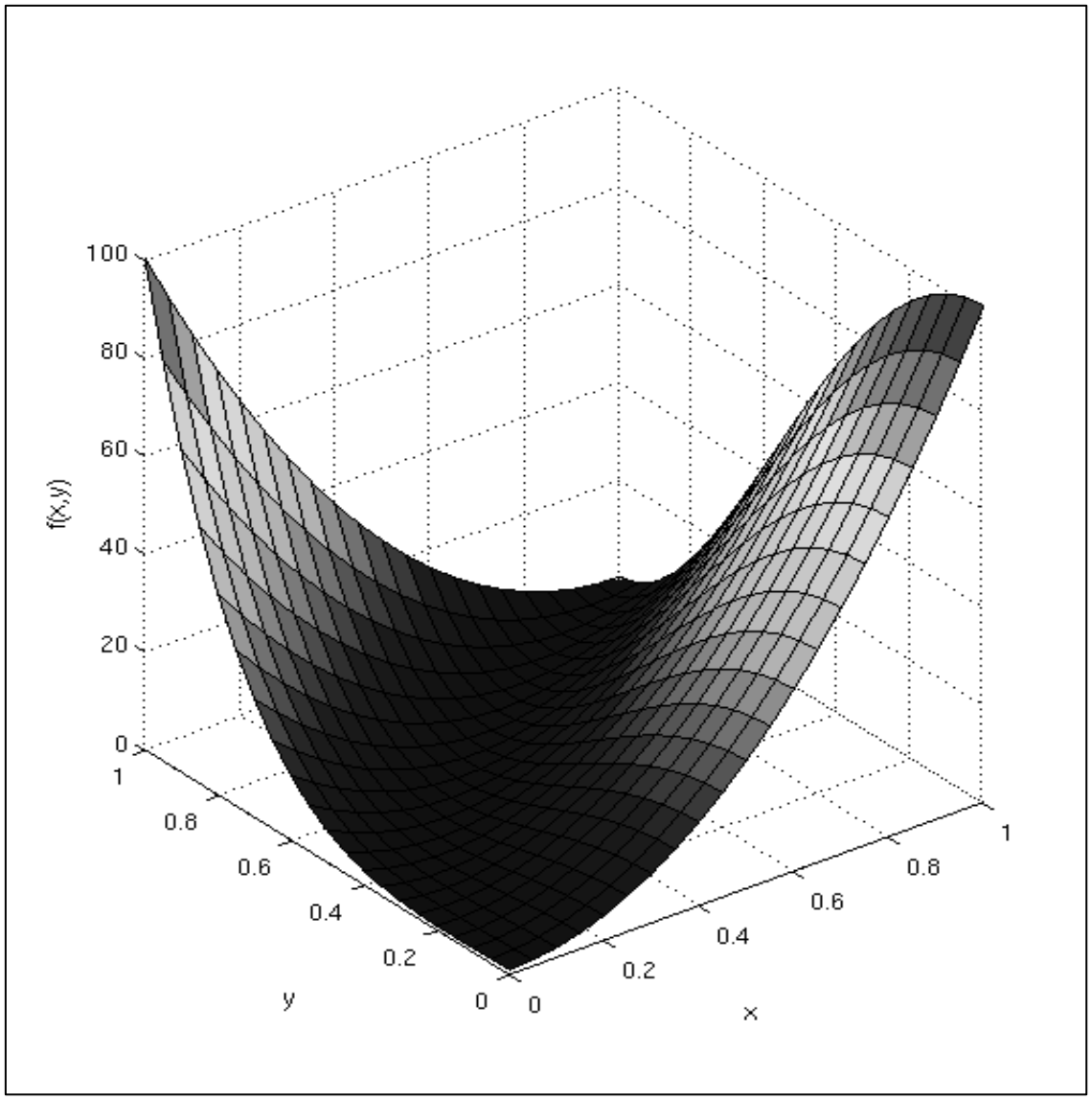


Figure 49 – Plot of function 2.3 for x ∈ [0,1] and y ∈ [0,1].

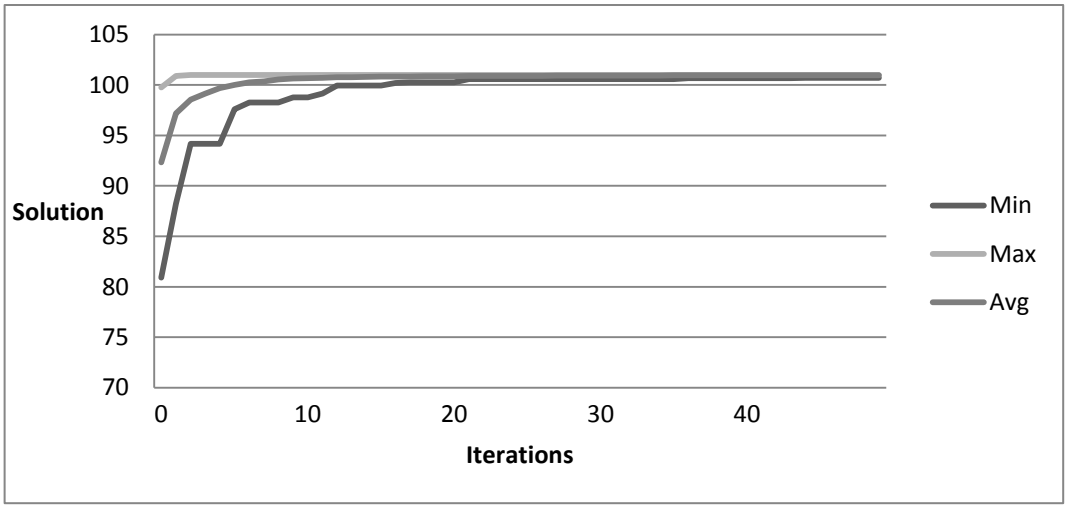


Figure 50 – Experiment 2.3_GA2, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and One-point Evaluation.

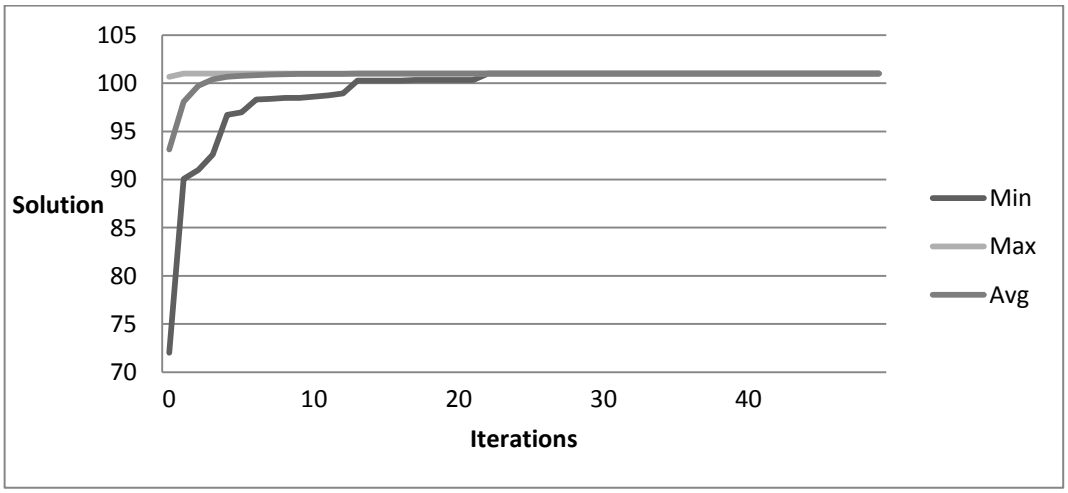


Figure 51 – Experiment 2.3_GA3, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and One-point Evaluation.

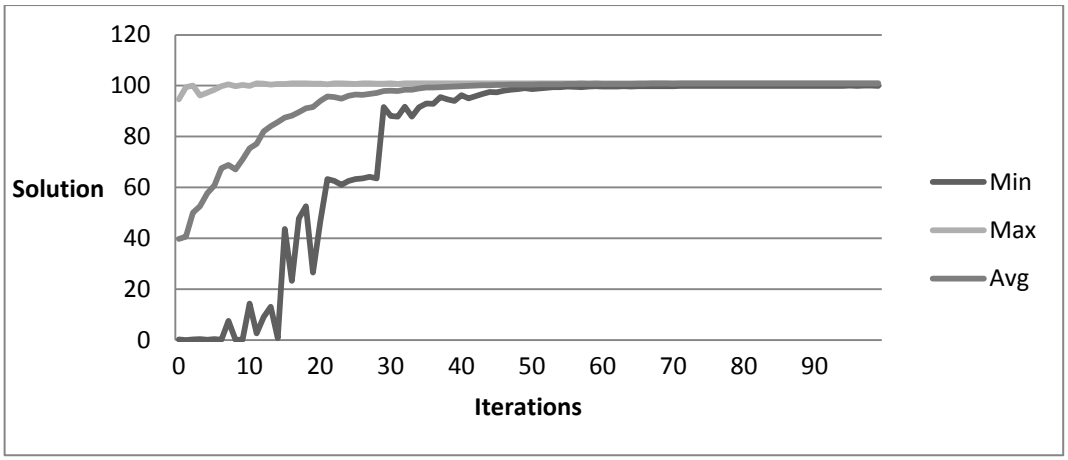


Figure 52 – Experiment 2.3_SA1, Optimizing with Simulated Annealing and One-point evaluation.

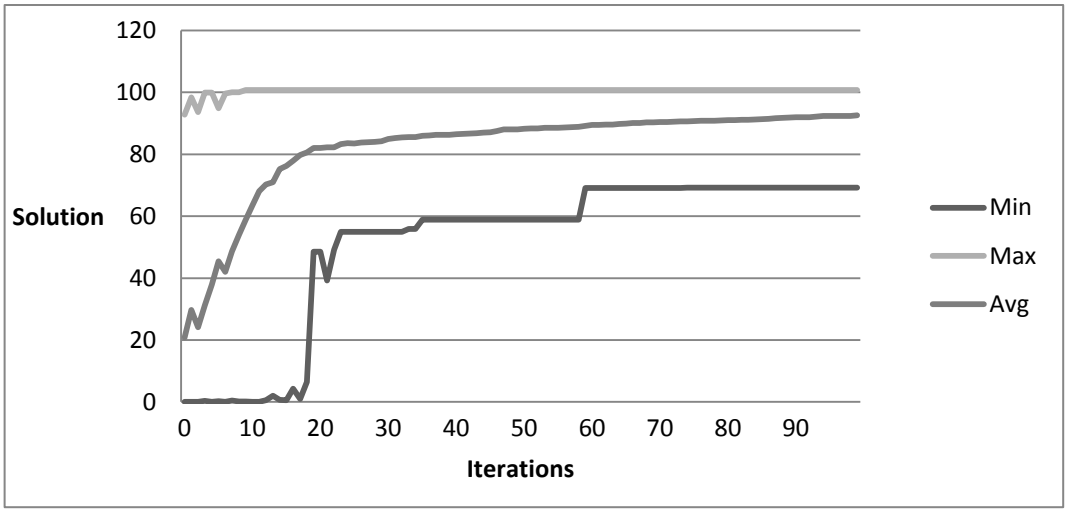


Figure 53– Experiment 2.3_SA2, Optimizing with Simulated Annealing, Analytic Swap and One-point evaluation.

	Setup	Maximum	Minimum	Average	Variance	Excess
2.3GA1	GA1	100,761458	68,2123372	94,8639171	24,2914233	6,1360829
2.3GA2	GA2	100,998567	100,687837	100,964754	0,00191909	0,035246
2.3GA3	GA3	101	100,999879	100,999997	2,7744E-10	3,0E-6
2.3SA1	SA1	100,999884	99,9899641	100,815655	0,14911073	0,184345
2.3SA2	SA2	100,727279	69,1863689	92,5463029	28,4804767	8,4536971

Table 12 – Results of optimizing Function 2.3.

6.2.4 Function 2.4

$$f_{2.4}(x, y) = 20 + (x^2 - 10\cos(2\pi x)) + (y^2 - 10\cos(2\pi y))$$

Optimal solution: $f_{2.4}\left(\frac{1}{2}, \frac{1}{2}\right) = 40{,}5$

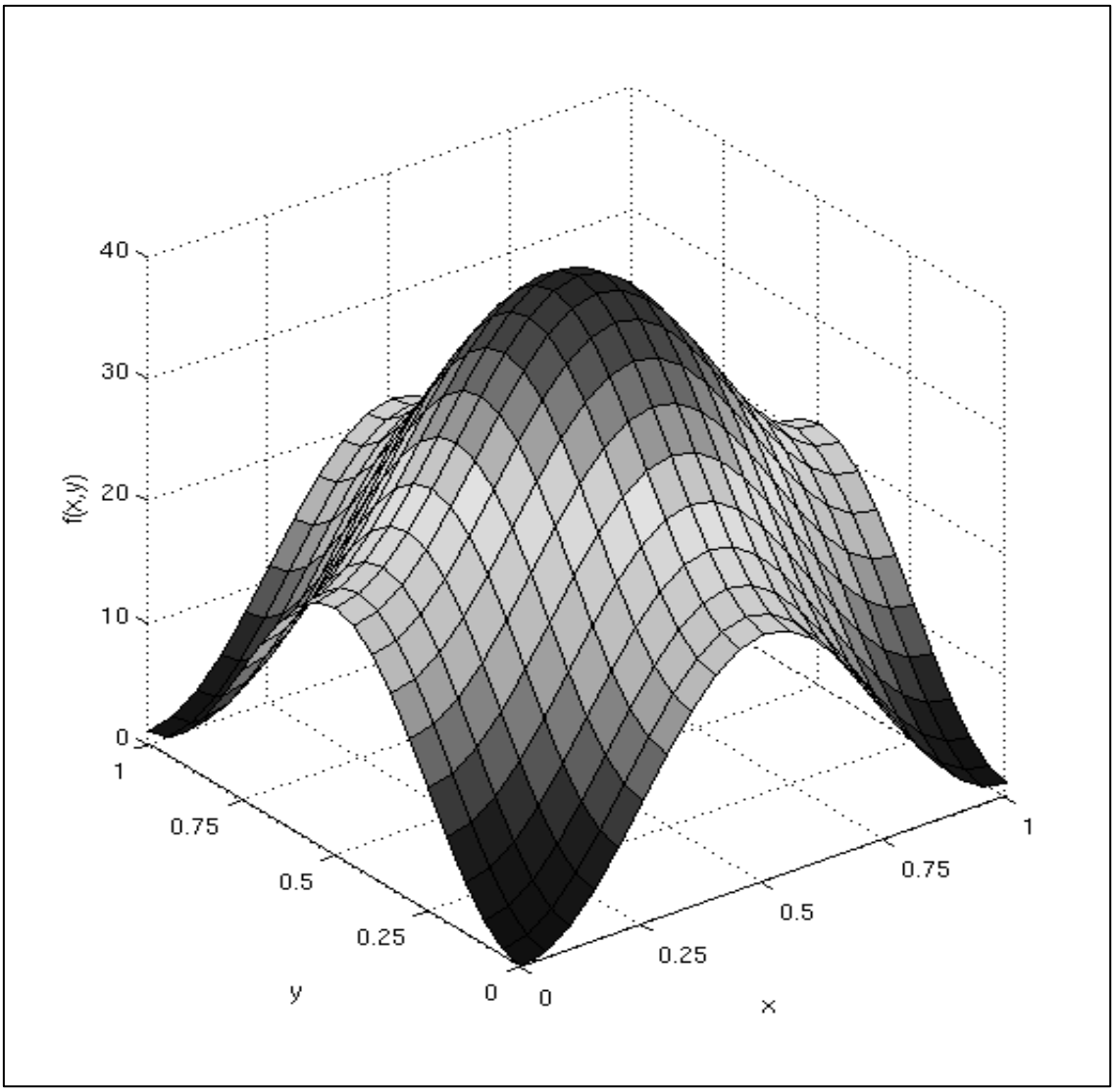


Figure 54 – Plot of function 2.4 for x ∈ [0,1] and y ∈ [0,1].

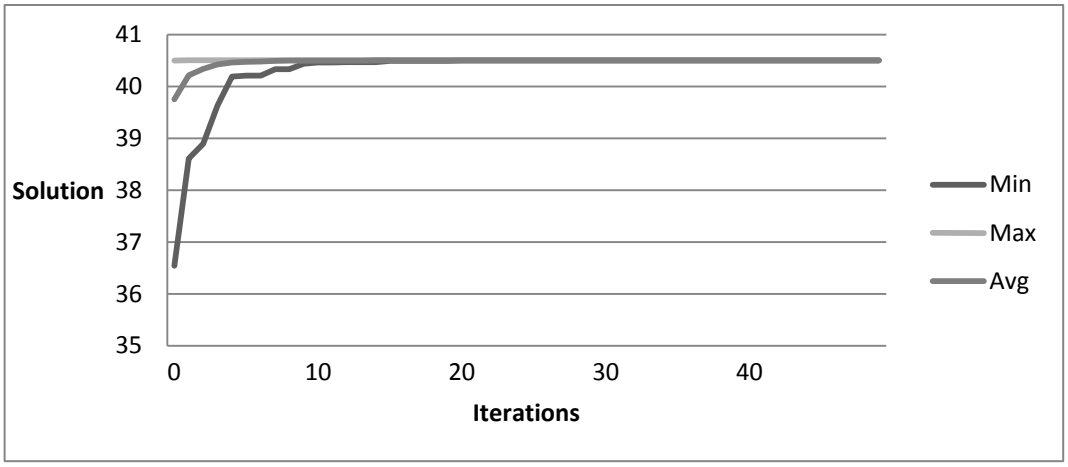


Figure 55 – Experiment 2.4_GA2, Optimizing with Genetic Algorithm, Uniform Crossover, Elitism and One-point Evaluation.

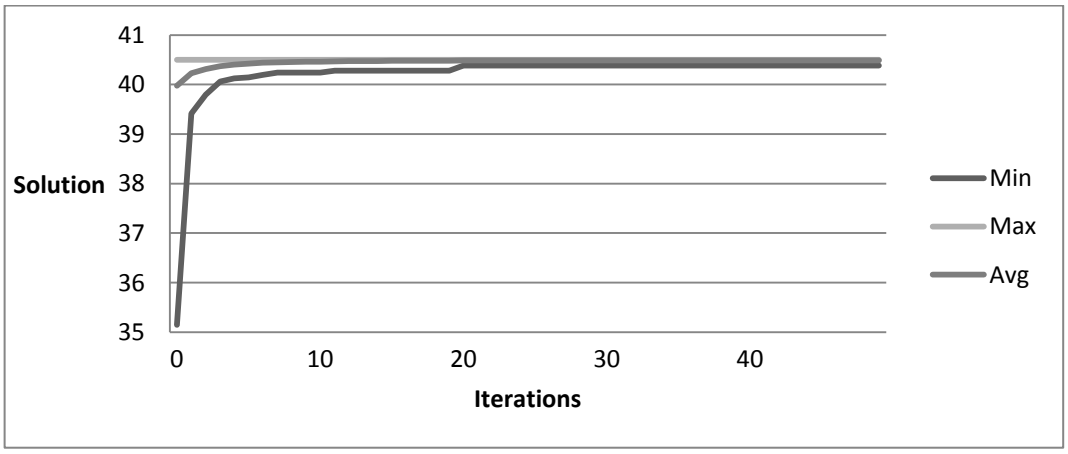


Figure 56 – Experiment 2.4_GA3, Optimizing with Genetic Algorithm, Analytic Swap, Elitism and One-point Evaluation.

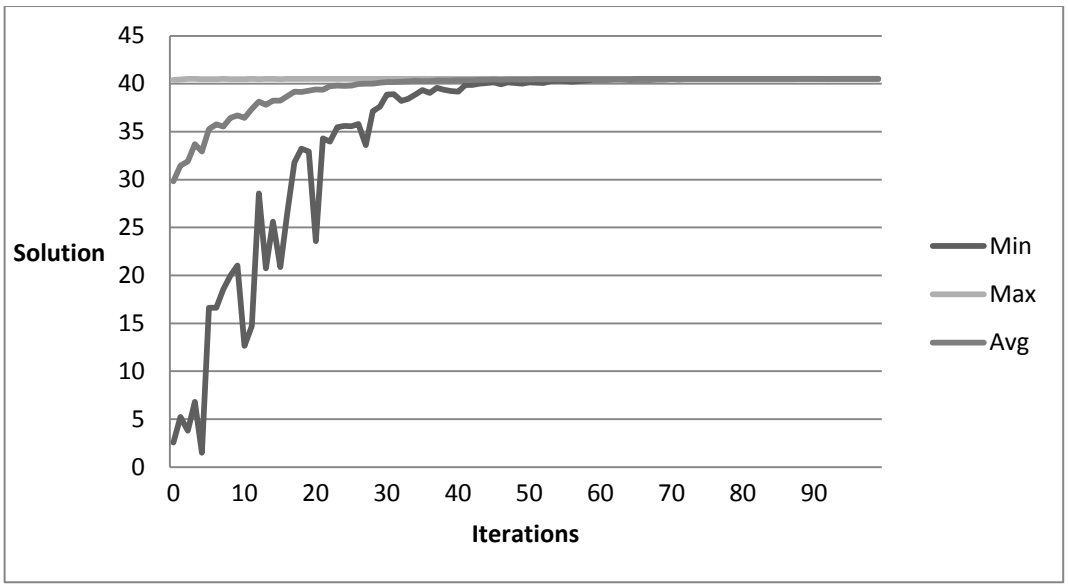


Figure 57 – Experiment 2.4_SA1, Optimizing with Simulated Annealing and One-point evaluation.

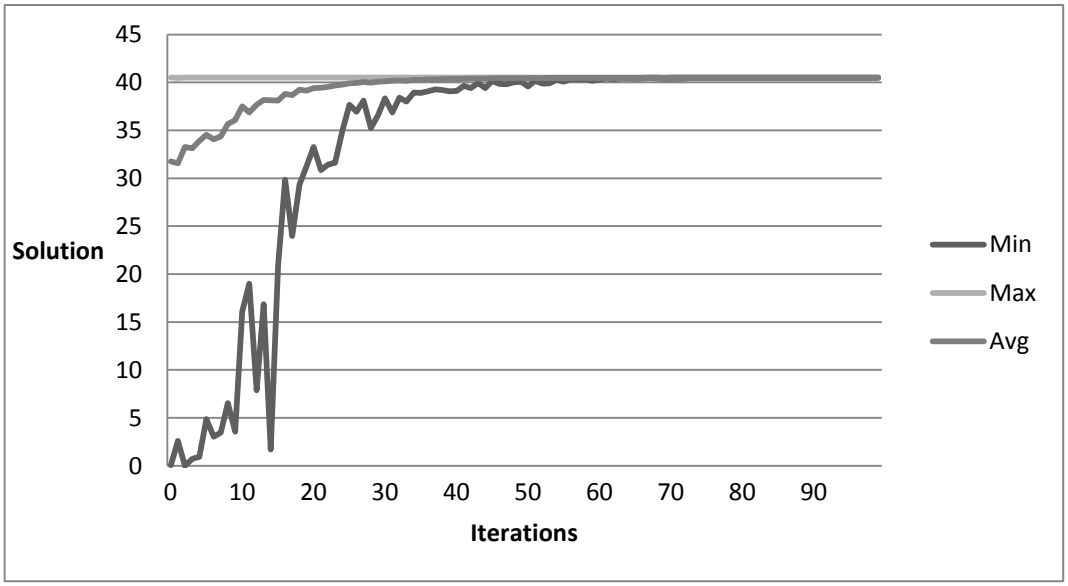


Figure 58 – Experiment 2.4_SA2, Optimizing with Simulated Annealing, Analytic Swap and One-point evaluation.

	Setup	Maximum	Minimum	Average	Variance	Excess
2.4GA1	GA1	40,5025439	39,1135102	40,2416442	0,07548008	0,2583558
2.4GA2	GA2	40,5025431	40,4976201	40,5023065	2,6998E-07	-0,0023065
2.4GA3	GA3	40,5023399	40,3809523	40,4868265	0,0006447	0,0131735
2.4SA1	SA1	40,502541	40,49848	40,5009668	1,1673E-06	-0,0009668
2.4SA2	SA2	40,5025457	40,4843549	40,5007987	5,1816E-06	-0,0007987

Table 13 – Results of optimizing Function 2.4.

7 Discussion

In this thesis two new methods for improving stochastic optimization of continuous functions have been explored. The first method is the Tangent-based Evaluation method. It was first presented in [9] where it was applied on the Simple Genetic Algorithm. In this thesis, this implementation has been improved by introducing elitism. Also, Tangent-based Evaluation has for the first time been applied together with Simulate Annealing.

The second method is the Analytic Swap method. This is a new method for generating new solutions to be used by a stochastic algorithm. This method has been applied to both the Genetic Algorithm and Simulated Annealing, and opposed to Tangent-based Evaluation; it can be used when optimizing functions with any number of variables.

These two methods affect the optimization process differently. The difference is in general that the Tangent-based Evaluation method increases the accuracy of the optimization, whereas the Analytic Swap method causes faster convergence. Thus, the further discussion of these methods is done separately.

7.1 Tangent-based Evaluation

This method was presented in [9]. It was proven by applying Tangent-based Evaluation, the accuracy of stochastic optimization was improved, compared to traditional one-point evaluation. It also stated that this requires a correct tuning of the parameter ***h***. However, it also showed that the accuracy decreased after a certain amount of iterations. In this thesis, the results and conclusion in [9] are confirmed. Also, the decrease in accuracy is eliminated by applying elitism to the Genetic Algorithm.

In addition to the experiments with Tangent-based Evaluation and the Genetic Algorithm, Tangent-based Evaluation has also been applied to Simulated Annealing. The results showed the same improvements in accuracy. They also showed that most suitable value for parameter h is the same for a function, regardless of which stochastic search algorithm being used. The usage of Tangent-based Evaluation also showed one major challenge. It is that before it can be used, one has to determine the anti-derivative of the function one is to optimize. Especially when optimizing advanced functions, this proved to be a major obstacle.

7.2 Analytic Swap

The obtained results show that by applying this method to both The Genetic Algorithm and Simulated Annealing, fewer iterations are required to achieve convergence, when compared to other methods for solution generation/crossover. We believe the main reason is the performance interim evaluations between each swap when a new solution is generated. This reduces the probability of selecting a poor solution. However, during the experiments it was also discovered that the Analytic Swap method is very time consuming. Whereas experiments without this method could be perform in one day, the experiments with Analytic Swap took almost a whole week. This can be remedied by either reducing the total number of iterations before ending the search or by defining a stop-criterion not based on the total number of iteration. For example, one possible stop-criterion is to end the search when a pre-defined number of iterations have been performed without any

improvements. It can also be researched into its own independent search technique, as proposed in chapter 8.1.

7.3 Efficiency

In this thesis, the efficiency of the optimizations has been measured by the number of iterations required to achieve convergence. However, this method may not always be accurate enough. This is because the amount of calculation required for each iteration depends on the complexity of the algorithm. Thus, this makes it not fully accurate to compare the efficiency of different experiments, when they use different algorithms. For example, Simulated Annealing only generates and evaluates one solution for each iteration, but the Genetic Algorithm generates and evaluates hundred solutions. Another method for calculating efficiency is to measure time used on each run. However, it was not possible to perform accurate measure of time used by each experiment. This was due to time constraints, which required multiple experiments to be performed simultaneously on the same computer to achieve maximum utilization of the computer's capacity and it was not possible to ensure an even distribution of capacity.

In the implementation used the experiments, there are in general two areas where there may be potential for increasing the efficiency. Firstly, it may be possible to optimize the code and the computer environment. For example optimization of methods and implementation of more efficient libraries may be implemented. Secondly, it may be possible to increase efficiency by using another programing language than Python. It is generally, interpreted programing languages, like Python, are regarded as less efficient than compiled languages as for example C\C++. For applications where a high level of efficiency is required, it may be beneficial to port the code into a compiled language.

8 Conclusion

In this thesis, main objective has been to solve continuous optimization problems by applying two newly developed techniques on the Genetic Algorithm and Simulated Annealing. The goal when solving a continuous optimization problem is to find the global minimum or global maximum of a continuous function. The first new method applied in this thesis a Tangent-based Evaluation, which purpose is increase the probability of detecting abnormalities on a curve compared to the traditional one-point evaluation method. The second method is the Analytic Swap method for generation of new solutions. Solution generation is a crucial part of any stochastic algorithm. The objective for this method is to ensure higher quality of the generated solutions, thus ensuring faster convergence of the optimization and reducing the number of iterations required.

In the beginning of this thesis, three hypotheses were presented. The first is: It is possible to improve the algorithm used in [9] by applying various strategies for crossover and by introducing elitism-based selection. This hypothesis is confirmed. By introducing elitism-based selection, the decline observed in [9] is avoided. In addition, the introduction of Analytic Swap did reduce the number of iterations required before convergence is achieved.

The second hypothesis is: It is possible to combine the tangent based method with Simulated Annealing. This hypothesis is also confirmed. By applying the Tangent-based Evaluation method to Simulated Annealing, more accurate solutions are obtained, but there is no general improvement to the efficiency.

The third hypothesis is: It is possible to improve the Genetic Algorithm and Simulated Annealing by introducing a new sequential-swap based method for crossover/generation of new solutions. At first glance, it may appear that this hypothesis is confirmed. The number of iterations required to obtain convergence is significantly lower when this method is used, compared to other methods for generating solutions/performing crossover. However, by using this method, each iteration consumes considerably more time, compared to when other methods are used. Thus, this hypothesis may not be considered to be confirmed.

The final conclusions to this thesis are that the combination of Tangent-based Evaluation and the Genetic Algorithm may be improved by introducing elitism; the accuracy of Simulated Annealing is improved by applying Tangent-based Evaluation instead of traditional one-point evaluation and that the current version of Analytic Swap method is too computational intensive. Based on this, some possible subjects for further research are discussed in the following chapter.

8.1 Further Research and Work

Based on the experiences and results we have achieved during this thesis, we would like to propose the following subjects for future research:

- Expansion of the Tanget-based Evaluation Method to be able to handle continuous functions with multiple variables. The experiments with the Tangent-based Evaluation and functions with one variable have showed interesting results. We think that a symmetric multipoint evaluation method with similar propertied as the Tangent-based Evaluation method, may be beneficial for evaluation of functions with multiple variables.
- Creating an adaptive version of the Tangent-based Evaluation method. The experiments in this thesis has showed that the ideal h-parameter different for each continuous function. Previous research has showed success in creating adaptive versions of algorithms which previous required parameters to be tuned correctly. Thus, we believe that it is possible to create an adaptive version of the Tangent-based Evaluation method as well.
- Develop the Analytic Swap method into a separate optimization method. The experiments showed that this method is very inefficient when it is combined with the Genetic Algorithm or Simulated Annealing. The main source for inefficiency was that within each iteration, a high number of new solutions was generated and evaluated. Since this is two main component of a search method, we would recommend that the Analytic Swap Method is developed into its own search method. Another possible approach may be to simplify method to make it less computational intensive.

By this, we conclude out master thesis. We hope that our research may inspire other to carry this effort forwards.

References

1. **Merriam-Webster, Inc.** Optimization. *Merriam Webster.* [Online] Merriam-Webster, Inc. [Cited: 4 28, 2013.] http://www.merriam-webster.com/dictionary/optimization.

2. *Optimization by Simulated Annealing.* **Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P.** 4598, s.l. : American Association for the Advancement of Science, 1983, Science, Vol. 220, pp. 671-680. ISSN: 00368075.

3. *The general employee scheduling problem: an integration of MS and AI.* **Glover, Fred and McMillan, Claude.** 5, Great Britain : Pergamon Journals Ltd., 1986, Computers and Operations Research, Vol. 13, pp. 563-573. ISSN: 0305-0548.

4. *Variable neighborhood search.* **Hansen, Pierre and Mladenovic, Nenad.** 11, Montreal, Canada : Computers and Operations Research, 1997, Vol. 24. ISSN: 0305-0548.

5. *Distributed Optimization by Ant Colonies.* **Colorni, Alberto, Dorigo, Marco and Maniezzo, Vittirio.** [ed.] Francisco J. Varela and Paul Bourgine. Paris, Fance : MIT Press/Bradford Books, 1991. Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life. pp. 134-142. ISBN: 978-0262720199.

6. *The Ant Colony Metaphor for Searching Continuous Design Spaces.* **Bilchev, George. and Parmee, Ian C.** London, United Kingdom : Springer-Verlag, 1995. Proceeding Selected Papers from AISB Workshop on Evolutionary Computing. pp. 25-39. ISBN: 3-540-60469-3.

7. *A probabilistic heuristic for a computationally difficult set covering problem.* **Feo, Thomas A. and Resende, Maurico G. C.** 2, s.l. : Elsevier B.V., April 1989, Operations Research Letters, Vol. 8, pp. 67-71. ISSN: 0167-6377.

8. *Global optimization by continuous grasp.* **Hirsch, M.J., et al., et al.** 2, s.l. : Springer-verlag, March 2007, Optimization Letters, Vol. 1, pp. 201-212. ISSN: 18624-472.

9. *A novel tangent based framework for optimizing continuous functions.* **Jensen, Bjørn, Bouhmala, Noureddine and Nordli, Thomas.** Transactions on Evolutionary Computation : IEEE, 2012. TEVC-00346-2012.

10. *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm.* **Černý, V. and Dreyfus, S. E.** 1, s.l. : Kluwer Academic Publishers-Plenum Publishers, 1985, Journal of Optimization Theory and Applications., Vol. 45, pp. 44-51. ISSN: 0022-3239.

11. **Holland, John H.** *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* s.l. : University of Michigan Press, 1975. ISBN: 0472084607.

12. **Goldberg, David Edward.** *Genetic Algorithms in Search, Optimization, and Machine.* New York, NY, USA : Addison-Wesley Publ. Co, 1989. ISBN: 0201157675.

13. *A genetic algorithm tutorial.* **Whitley, Darrell.** 2, Fort Collins, Colorado, USA : Kluwer Academic Publishers, 1994, Statstics and Computing, Vol. 4, pp. 65-85. ISSN: 1573-1375.

14. **Booker, Lashon, et al., et al.** *Perspectives on Adaptation in Natural and Artificial Systems.* New York, NY, USA : Oxford University Press, IInc., 2005. ISBN: 0-19-516292-7.

15. *A new crossover operator for real coded genetic algorithms.* **Deep, Kusum and Thakur, Manoj.** 1, s.l. : Elsevier Inc., May 2007, Applied Mathematics and Computation, Vol. 188, pp. 895-911. ISSN: 0096-3003.

16. *Optimization of multimodal continuous functions using a new crossover for the real-coded genetic algorithms.* **Tutkun, Nedim.** 4, s.l. : Elsevier Ltd., 2009, Expert Systems with Applications, Vol. 36, pp. 8172-8177. ISSN: 0957-4174.

17. *Adaptive directed mutation for real-coded genetic algorithms.* **Tang, Ping-Hung and Tseng, Ming-Hseng.** 1, s.l. : Elsevier Science B.V., January 2013, Applied Soft Computing, Vol. 13, pp. 600–614. ISSN: 1568-4946.

18. *Hybrid Ant Colony-Genetic Algorithm (GAAPI) for Global Continuous Optimization.* **Ciornei, Irina and Kyriakides, Elias.** 1, s.l. : IEEE, February 2012, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 42. ISSN: 1083-4419.

19. *Hybrid enhanced continuous tabu search and genetic algorithm for parameter estimation in colored noise environments.* **Ramkumar, Barathram, Schoen, Marco P. and Lin, Feng.** 4, s.l. : Eliviser Science, April 2011, Expert Systems with Applications, Vol. 38, pp. 3909-3917. ISSN: 0957-4174.

20. **Bouhmala, Noureddine.** *Partitioning of Unstructured Meshes for Parallel Processing, PhD Thesis.* Switzerland : University of Neuchatel, 1998.

21. *Enhanced Simulated Annealing for Globally Minimizing Functions of Many-Continuous Variables.* 2, New York, New York, USA : Association for Computing Machinery, June 1997, ACM Transactions on Mathematical Software, Vol. 23, pp. 209-228. ISSN: 0098-3500.

22. *Investigation of temperature parallel simulated annealing for optimizing continuous functions with application to hyperspectral tomography.* **Cai, Weiwei, Ewing, David J. and Ma, Lin.** 12, Clemson, USA : Elsevier Science, 2011, Applied Mathematics and Computation, Vol. 217, pp. 5754-5767. ISSN: 0096-3003.

23. *Parallel Simulated Annealing Algorithms.* **Janaki Ram, D., Sreenivas, T. H. and Ganapathy Subramaniam, K.** 2, Madras, India : Elsevier Science, 1996, Journal of Parallel and Distributed Computiong, Vol. 37, pp. 207-212. ISSN: 0743-7315.

24. **Ingber, Lester.** *Adaptive simulated annealing (ASA): Lessons learned.* McLean, Virginia, USA : Lester Ingber Research, 1995.

25. *Fuzzy-based adaptive sample-sort simulated annealing for resource-constrained project scheduling.* **Shukla, Sanja, Son, Young and Tiwari, M.** 9, London : Springer-Verlag, 2008, International Journal of Advanced Manufacturing Technology, Vol. 36, pp. 982-995. ISSN: 0268-3768.

26. *Global optimization using dimensional jumping and fuzzy adaptive simulated annealing.* **Oliveira, Hime A. and Petraglia, Antonio.** 6, Rio de Janeiro, Brazil : Elisevier Science, 2011, Applied Soft Computing, Vol. 11, pp. 4175-4182. ISSN: 1568-4946.

Figures

Tables

Appendix A – Functions with One Variable

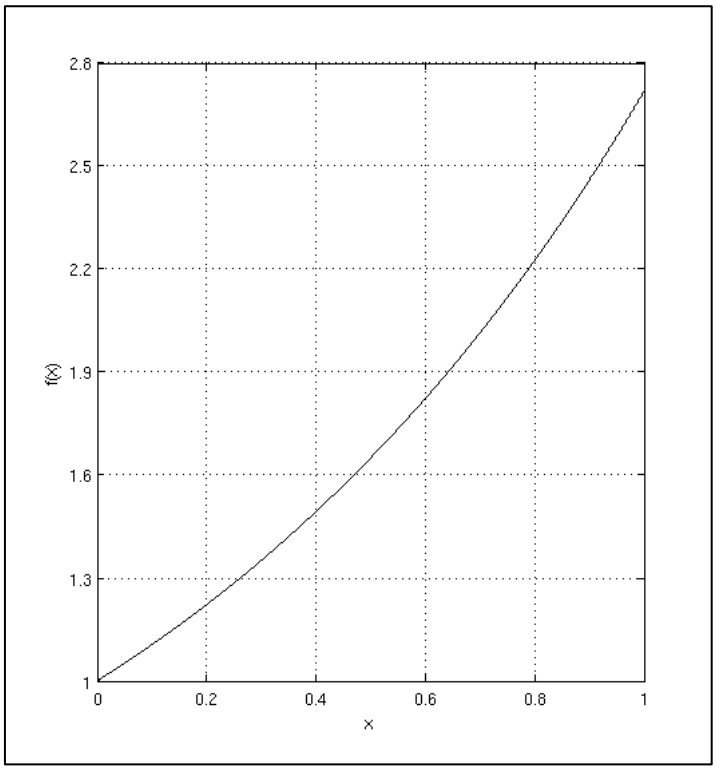
	Function	Optimal Solution
1.1:	$f_{1.1}(x) = e^x$ $g_{1.1}(x) = e^x$	$f_{1.1}(1) = e \sim 2.718$
1.2:	$f_{1.2}(x) = \frac{1}{1+x^2}$ $g_{1.2}(x) = \tan^{-1}(x)$	$f_{1.2}(0) = 1$
1.3:	$f_{1.3}(x) = \sum_{n=1}^{20}\left(\left(x-\frac{1}{10}\right)^n - 10\cos\left(2\pi n\left(x-\frac{1}{10}\right)\right)\right)$ $g_{1.3}(x) = \sum_{n=1}^{20}\left(\frac{\left(x-\frac{1}{10}\right)^{n+1}}{n+1} - \frac{5}{\pi n}\sin\left(2\pi n\left(x-\frac{1}{10}\right)\right)\right)$	$f_{1.3}(0{,}134) \approx 49{,}595$
1.4:	$f_{1.4}(x) = x^2$ $g_{1.4}(x) = \frac{1}{3}x^3$	$f_{1.4}(1) = 1$
1.5:	$f_{1.5}(x) = e^{-x}$ $g_{1.5}(x) = -e^{-x}$	$f_{1.5}(0) = 1$
1.6:	$f_{1.6}(x) = e^{-2\pi x}\cos(2\pi x)$ $g_{1.6}(x) = \frac{1}{4\pi}e^{-2\pi x}(\sin(2\pi x) - \cos(2\pi x))$	$f_{1.6}(0) = 1$

Appendix B – Functions with Two Variables

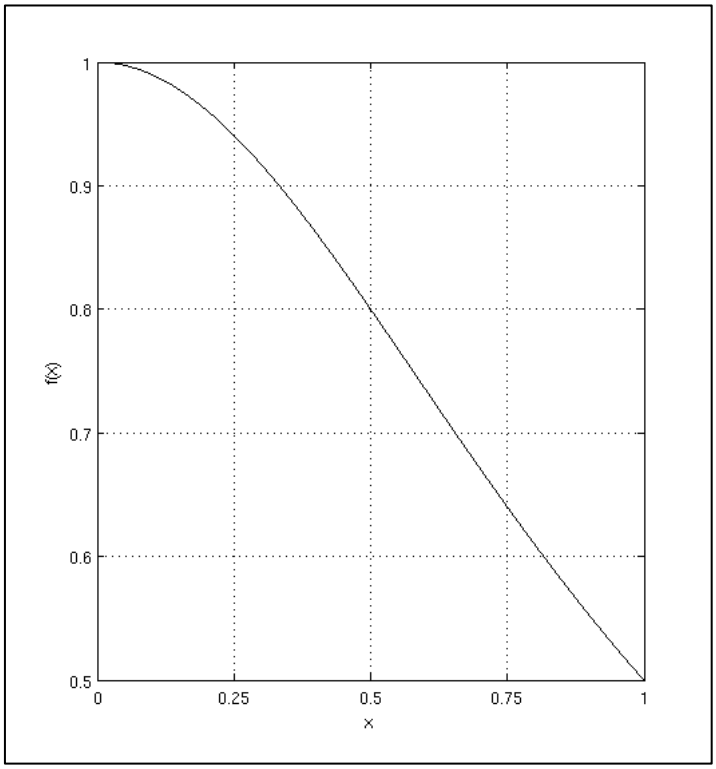
	Function	Optima Solution
2.1:	$f_{2.1}(x,y) = x^2 + y^2$	$f_{2.1}(1,1) = 2$
2.2:	$f_{2.2}(x,y) = x^2 + 2y^2$	$f_{2.2}(1,1) = 3$
2.3:	$f_{2.3}(x,y) = 100(y - x^2)^2 + (1 - x)^2$	$f_{2.3}(0,1) = 101$
2.4:	$f_{2.4}(x,y) = 20 + (x^2 - 10\cos(2\pi x)) + (y^2 - 10\cos(2\pi y))$	$f_{2.4}(0.5,0.5) = 41$

Appendix C – Plots of functions with one variable

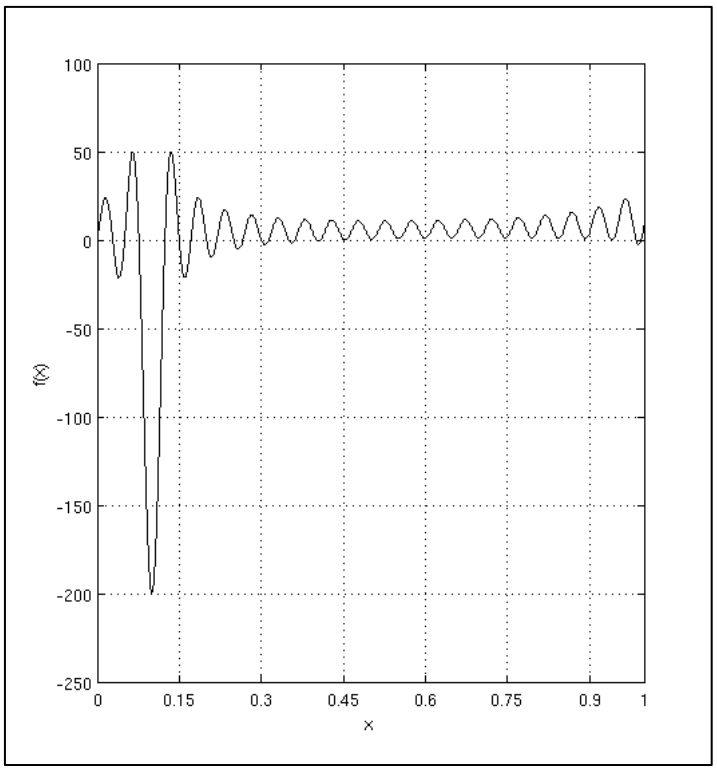
Function 1.1:



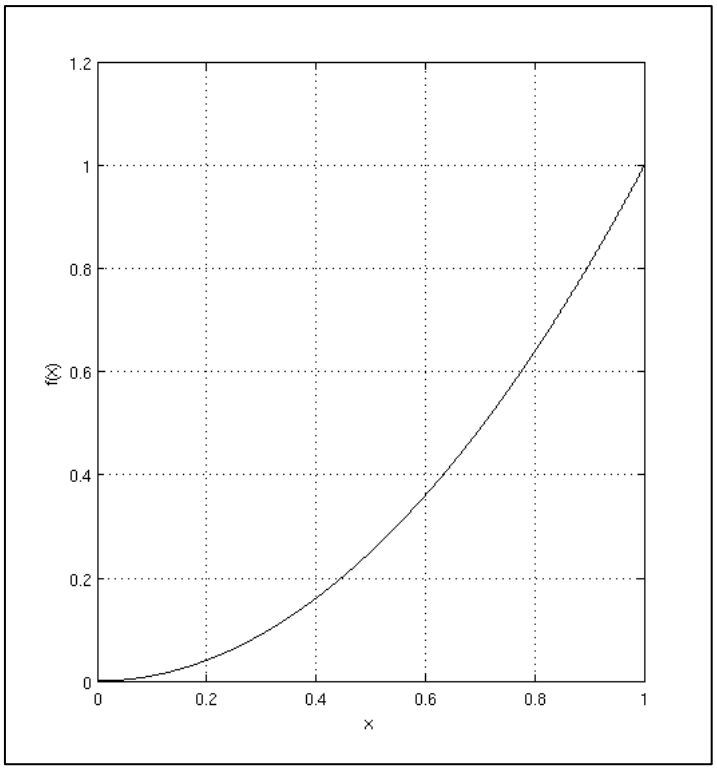
Function 1.2:



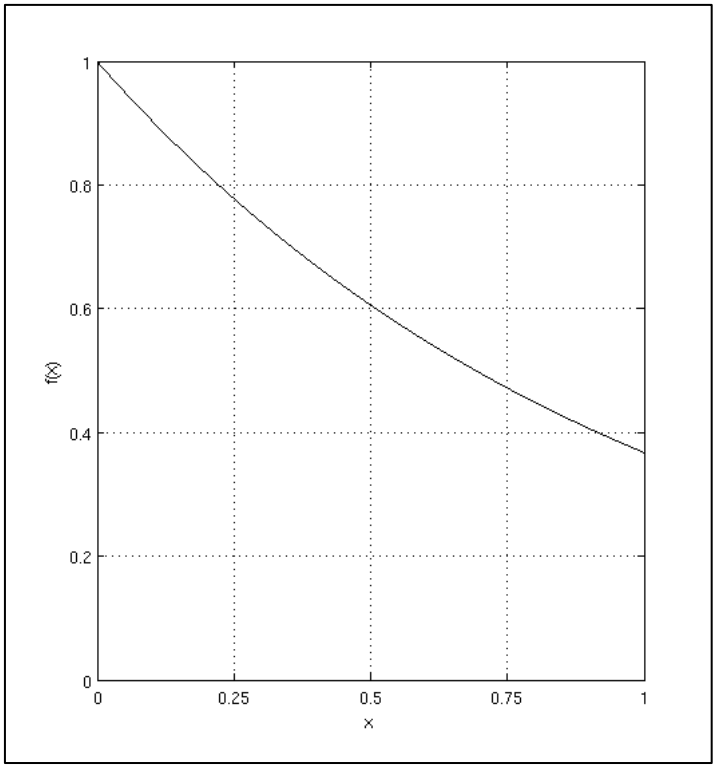
Function 1.3:



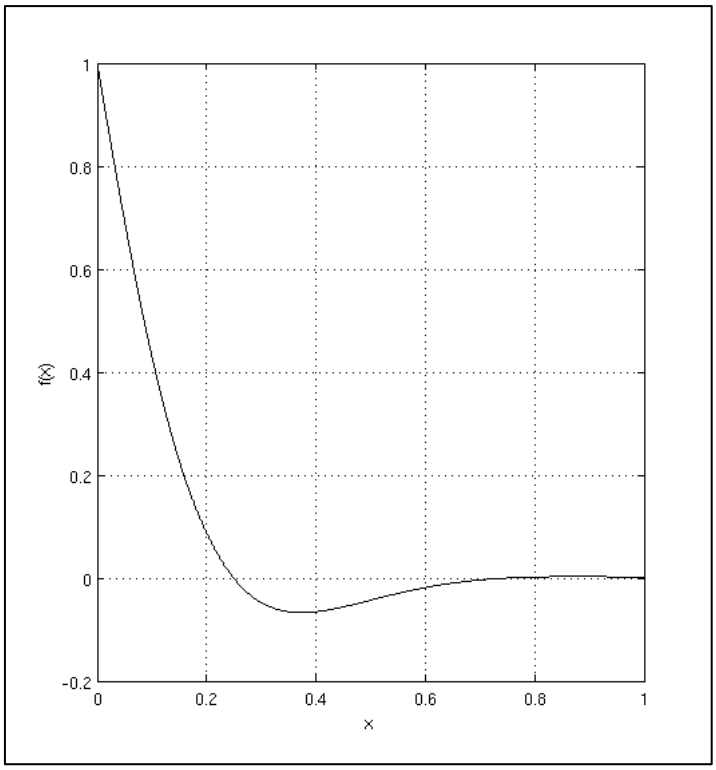
Function 1.4:



Function 1.5:



Function 1.6:



Appendix D – Results for Functions with One Variable

Function 1.1:

	Setup	h	Maximum	Minimum	Average	Variance
1.1_GA1	GA1	–	2,71799138	2,59694631	2,70089926	0,00043407
1.1_GA2	GA2	10^{-3}	2,71800693	2,56958862	2,69369835	0,00091469
1.1_GA2	GA2	10^{-5}	2,71772868	2,60796382	2,69491215	0,00060202
1.1_GA2	GA2	10^{-7}	2,71827548	2,60366527	2,69830477	0,00043711
1.1_GA2	GA2	10^{-9}	2,7182776	2,61245736	2,69601573	0,00043407
1.1_GA2	GA2	10^{-11}	2,71811462	2,62361244	2,69533418	0,00047023
1.1_GA2	GA2	10^{-13}	2,72004641	2,62234678	2,699018786	0,00047665
1.1_GA3	GA3	–	2,71828183	2,71819484	2,7182807	1,01109E-08
1.1_GA4	GA4	10^{-3}	2,71828228	2,71825619	2,71828179	1,40332E-08
1.1_GA4	GA4	10^{-5}	2,71828183	2,71808857	2,71827982	1,05287E-08
1.1_GA4	GA4	10^{-7}	2,71828183	2,71825437	2,71828096	1,19145E-08
1.1_GA4	GA4	10^{-9}	2,71828204	2,71798251	2,7182785	8,27804E-09
1.1_GA4	GA4	10^{-11}	2,71829226	2,71804801	2,71828937	1,59613E-08
1.1_GA4	GA4	10^{-13}	2,72004641	2,72004641	2,72004641	2,86859E-29
1.1_GA5	GA5	–	2,718281828	2,718194845	2,718280697	7,76222E-11
1.1_GA6	GA6	10^{-3}	2,718282282	2,718256186	2,718281787	9,47765E-12
1.1_GA6	GA6	10^{-5}	2,718281829	2,718088565	2,718279816	3,73658E-10
1.1_GA6	GA6	10^{-7}	2,718281829	2,718254373	2,718280961	1,67673E-11
1.1_GA6	GA6	10^{-9}	2,718282044	2,717982506	2,718278505	8,97824E-10
1.1_GA6	GA6	10^{-11}	2,718292258	2,718048009	2,718289371	6,04146E-10
1.1_GA6	GA6	10^{-13}	2,72004641	2,72004641	2,72004641	2,86859E-29
1.1_SA1	SA1	–	2,71828182	2,71827757	2,71828117	6,91991E-13
1.1_SA2	SA2	10^{-3}	2,71828228	2,71827339	2,71828144	1,77082E-12
1.1_SA2	SA2	10^{-5}	2,71828183	2,71827305	2,718281	1,83252E-12
1.1_SA2	SA2	10^{-7}	2,71828182	2,71827853	2,71828123	5,76652E-13
1.1_SA2	SA2	10^{-9}	2,71828182	2,71827605	2,71828121	7,56844E-13
1.1_SA2	SA2	10^{-11}	2,71829226	2,71827005	2,71829159	1,44923E-11
1.1_SA2	SA2	10^{-13}	2,72004641	2,72004641	2,72004641	2,86859E-29
1.1_SA3	SA3	–	2,71828183	2,71827087	2,71828058	2,71713E-12
1.1_SA4	SA4	10^{-3}	2,71828228	2,71827792	2,71828146	1,05605E-12
1.1_SA4	SA4	10^{-5}	2,71828183	2,71827613	2,71828079	1,42145E-12
1.1_SA4	SA4	10^{-7}	2,71828182	2,71827446	2,71828094	1,44208E-12
1.1_SA4	SA4	10^{-9}	2,71828182	2,71827738	2,71828082	1,20665E-12
1.1_SA4	SA4	10^{-11}	2,71829226	2,71827005	2,71829181	9,76116E-12
1.1_SA4	SA4	10^{-13}	2,72004641	2,72004641	2,72004641	2,86859E-29

Function 1.2:

	Setup	h	Maximum	Minimum	Average	Variance
1.2_GA1	GA1	–	1,0000000	0,99749277	0,99986237	9,70299E-08
1.2_GA2	GA2	10^{-3}	0,999999667	0,996339265	0,999827495	1,91615E-07
1.2_GA2	GA2	10^{-5}	0,999999999	0,997782703	0,999814398	1,52368E-07
1.2_GA2	GA2	10^{-7}	1	0,998132051	0,999876554	9,51364E-08
1.2_GA2	GA2	10^{-9}	1	0,998540584	0,999872431	6,72487E-08
1.2_GA2	GA2	10^{-11}	0,999999993	0,99814601	0,999841208	1,19797E-07
1.2_GA2	GA2	10^{-13}	1,00000086	0,998368055	0,99988152	8,46176E-08
1.2_GA3	GA3	–	1	1	1	–
1.2_GA4	GA4	10^{-3}	0,99999967	0,99999967	0,99999967	–
1.2_GA4	GA4	10^{-5}	1	1	1	–
1.2_GA4	GA4	10^{-7}	1	1	1	–
1.2_GA4	GA4	10^{-9}	1	1	1	–
1.2_GA4	GA4	10^{-11}	1	1	1	–
1.2_GA4	GA4	10^{-13}	1,000000863	1,00000005	1,000000387	1,62702E-13
1.2_GA5	GA5	–	1	1	1	–
1.2_GA6	GA6	10^{-3}	1	1	1	–
1.2_GA6	GA6	10^{-5}	1	1	1	–
1.2_GA6	GA6	10^{-7}	1	1	1	–
1.2_GA6	GA6	10^{-9}	1	1	1	–
1.2_GA6	GA6	10^{-11}	1	1	1	–
1.2_GA6	GA6	10^{-13}	1,000000863	1,00000005	1,000000237	1,18287E-13
1.2_SA1	SA1	–	1	0,99999996	1	6,42761E-17
1.2_SA2	SA2	10^{-3}	0,99999967	0,99999958	0,99999966	1,41184E-16
1.2_SA2	SA2	10^{-5}	1	0,99999997	1	2,99238E-17
1.2_SA2	SA2	10^{-7}	1	0,99999994	1	1,02398E-16
1.2_SA2	SA2	10^{-9}	1	0,99999992	1	1,09327E-16
1.2_SA2	SA2	10^{-11}	1	0,99999986	0,99999999	2,46445E-16
1.2_SA2	SA2	10^{-13}	1,00000086	0,99999978	1,00000015	8,52291E-14
1.2_SA3	SA3	–	1	0,99999988	0,99999999	2,93548E-16
1.2_SA4	SA4	10^{-3}	0,999999667	0,999999521	0,999999658	4,61357E-16
1.2_SA4	SA4	10^{-5}	1	0,999999937	0,999999992	1,86744E-16
1.2_SA4	SA4	10^{-7}	1	0,999999821	0,999999989	6,37882E-16
1.2_SA4	SA4	10^{-9}	1	0,999999865	0,999999993	3,73167E-16
1.2_SA4	SA4	10^{-11}	1	0,999999888	0,999999992	2,88404E-16
1.2_SA4	SA4	10^{-13}	1,000000863	0,999999779	0,999999779	4,12902E-14

Function 1.3:

	Setup	h	Maximum	Minimum	Average	Variance
1.3_GA1	GA1	–	47,4925022	23,0082964	45,0965458	18,375547
1.3_GA2	GA2	10^{-3}	47,3864439	22,9327126	44,6493007	23,9308349
1.3_GA2	GA2	10^{-5}	47,4925125	24,9423348	45,2609386	15,8670925
1.3_GA2	GA2	10^{-7}	47,4921337	22,6313857	45,1913394	20,5175772
1.3_GA2	GA2	10^{-9}	47,4925159	28,291709	44,9722638	20,5175772
1.3_GA2	GA2	10^{-11}	47,492521	22,4448016	45,732363	11,9377222
1.3_GA2	GA2	10^{-13}	47,5020023	17,9611881	45,2242466	26,98083
1.3_GA3	GA3	–	47,4925237	47,4166783	47,4754847	0,00065416
1.3_GA4	GA4	10^{-3}	47,3864591	47,3090105	47,3615172	0,00089281
1.3_GA4	GA4	10^{-5}	47,4925131	47,4168554	47,4776476	0,00052161
1.3_GA4	GA4	10^{-7}	47,4925237	47,416245	47,4759563	0,0005879
1.3_GA4	GA4	10^{-9}	47,492521	47,4145052	47,4710752	0,00080566
1.3_GA4	GA4	10^{-11}	47,4925654	47,4152939	47,4726189	0,00077442
1.3_GA4	GA4	10^{-13}	47,5042228	47,4176254	47,4746686	0,00083046
1.3_GA5	GA5	–	47,4925237	46,600904	47,4040929	0,02263649
1.3_GA6	GA6	10^{-3}	47,3864592	43,2536125	47,2587939	0,21077976
1.3_GA6	GA6	10^{-5}	47,4925131	46,610199	47,4271111	0,01333656
1.3_GA6	GA6	10^{-7}	47,4925237	46,5634367	47,4208171	0,02800796
1.3_GA6	GA6	10^{-9}	47,4925241	44,5168844	47,3967598	0,11057167
1.3_GA6	GA6	10^{-11}	47,4926098	45,4771332	47,4022419	0,0651185
1.3_GA6	GA6	10^{-13}	47,5064432	47,4282169	47,4282169	0,01976401
1.3_SA1	SA1	–	47,4925237	44,2286558	47,3894619	0,20726851
1.3_SA2	SA2	10^{-3}	47,3864592	44,0662594	47,1972446	0,46970846
1.3_SA2	SA2	10^{-5}	47,4925131	47,4190441	47,4599743	0,00130498
1.3_SA2	SA2	10^{-7}	47,4925237	44,1485577	47,2555599	0,59731443
1.3_SA2	SA2	10^{-9}	47,4925241	44,148541	47,3879539	0,21766181
1.3_SA2	SA2	10^{-11}	47,4926098	44,1483738	47,2539778	0,61780908
1.3_SA2	SA2	10^{-13}	47,5086637	44,1557901	47,2952122	0,50874632
1.3_SA3	SA3	–	47,4925237	44,1485092	47,4186748	0,1104435
1.3_SA4	SA4	10^{-3}	47,3864592	47,3129354	47,3445713	0,00133683
1.3_SA4	SA4	10^{-5}	44,1485332	44,1485332	47,3537421	0,31695153
1.3_SA4	SA4	10^{-7}	44,2286621	44,2286621	47,4224153	0,10541947
1.3_SA4	SA4	10^{-9}	47,4925244	44,148514	47,288725	0,51494527
1.3_SA4	SA4	10^{-11}	47,4926098	44,1485959	47,4192299	0,11053428
1.3_SA4	SA4	10^{-13}	47,5064432	47,4265072	47,4634249	0,00135318

Function 1.4:

	Setup	h	Maximum	Minimum	Average	Variance
1.4_GA1	GA1	–	0,99997994	0,93062203	0,98374923	0,00025181
1.4_GA2	GA2	10^{-3}	0,99993821	0,9028746	0,98418465	0,00033321
1.4_GA2	GA2	10^{-5}	0,99995374	0,89200495	0,98596544	0,00034352
1.4_GA2	GA2	10^{-7}	0,9998787	0,91898938	0,98480651	0,00036033
1.4_GA2	GA2	10^{-9}	0,99991093	0,93876623	0,98633729	0,00015787
1.4_GA2	GA2	10^{-11}	1,00000008	0,90004948	0,9854754	0,00030992
1.4_GA2	GA2	10^{-13}	1,00031095	0,87735375	0,98382136	0,00044604
1.4_GA3	GA3	–	0,999999925	0,99977958	0,999957814	2,72189E-09
1.4_GA4	GA4	10^{-3}	1,00000033	0,999466367	0,999955816	4,86032E-09
1.4_GA4	GA4	10^{-5}	0,99999998	0,999758579	0,999964032	2,12677E-09
1.4_GA4	GA4	10^{-7}	0,999999017	0,999485873	0,999946487	5,61779E-09
1.4_GA4	GA4	10^{-9}	0,999999888	0,99971062	0,999954236	3,97657E-09
1.4_GA4	GA4	10^{-11}	1,000000083	0,999716976	0,999951122	3,16265E-09
1.4_GA4	GA4	10^{-13}	1,000310945	1,000033389	1,000305394	1,52518E-09
1.4_GA5	GA5	–	1	0,999919881	0,99999864	7,00307E-11
1.4_GA6	GA6	10^{-3}	1,000000333	0,999860338	0,999998762	1,96985E-10
1.4_GA6	GA6	10^{-5}	1	0,99998782	0,999999834	1,53775E-12
1.4_GA6	GA6	10^{-7}	1	0,999986	0,999999608	4,38588E-12
1.4_GA6	GA6	10^{-9}	1,000000027	0,999980959	0,99999943	7,43237E-12
1.4_GA6	GA6	10^{-11}	1,000000083	0,999861305	0,999998445	1,9502E-10
1.4_GA6	GA6	10^{-13}	1,000310945	0,999755834	1,000305394	3,08149E-09
1.4_SA1	SA1	–	1	0,999995984	0,999999457	5,05552E-13
1.4_SA2	SA2	10^{-3}	1,000000333	0,999997416	0,99999978	4,98353E-13
1.4_SA2	SA2	10^{-5}	0,999999999	0,999995449	0,999999504	4,98353E-13
1.4_SA2	SA2	10^{-7}	0,999999997	0,999995525	0,999999425	6,84231E-13
1.4_SA2	SA2	10^{-9}	1,000000027	0,999993532	0,999999491	6,75625E-13
1.4_SA2	SA2	10^{-11}	1,000000083	0,999994532	0,99999983	7,92938E-13
1.4_SA2	SA2	10^{-13}	1,000310945	1,000310945	1,000310945	–
1.4_SA3	SA3	–	0,999999988	0,999992287	0,999999077	1,63535E-12
1.4_SA4	SA4	10^{-3}	1,000000332	0,999991708	0,999999598	6,94721E-13
1.4_SA4	SA4	10^{-5}	1	0,999996014	0,999999289	6,94721E-13
1.4_SA4	SA4	10^{-7}	0,999999998	0,99999739	0,999999482	3,44797E-13
1.4_SA4	SA4	10^{-9}	0,999999972	0,99999628	1,000000083	5,71964E-13
1.4_SA4	SA4	10^{-11}	1,000000083	0,999991756	0,999999472	2,11346E-12
1.4_SA4	SA4	10^{-13}	1,000310945	1,000310945	1,000310945	–

Function 1.5:

	Setup	h	Maximum	Minimum	Average	Variance
1.5_GA1	GA1	–	0,999985746	0,957860892	0,992056362	6,05661E-05
1.5_GA2	GA2	10^{-3}	0,999988982	0,961923026	0,992257382	6,83987E-05
1.5_GA2	GA2	10^{-5}	0,99991624	0,958724787	0,992486827	5,77233E-05
1.5_GA2	GA2	10^{-7}	0,999999457	0,901503979	0,991154178	0,000166539
1.5_GA2	GA2	10^{-9}	0,999997862	0,953125745	0,992091135	9,16537E-05
1.5_GA2	GA2	10^{-11}	1,000000083	0,969058167	0,992765481	5,4402E-05
1.5_GA2	GA2	10^{-13}	1,000310945	0,956457136	0,992050886	7,80288E-05
1.5_GA3	GA3	–	1	0,99999679	0,99999984	1,8091E-13
1.5_GA4	GA4	10^{-3}	1,00000017	0,99999754	0,99999998	1,51526E-13
1.5_GA4	GA4	10^{-5}	1	0,99999814	0,99999986	5,91369E-14
1.5_GA4	GA4	10^{-7}	0,999997229	1,000000001	0,999999817	1,52252E-13
1.5_GA4	GA4	10^{-9}	1,000000027	0,999996974	0,999999845	1,46485E-13
1.5_GA4	GA4	10^{-11}	1,000000083	0,999994532	0,999999916	9,05771E-13
1.5_GA4	GA4	10^{-13}	1,000310945	1,000310945	1,000310945	–
1.5_GA5	GA5	–	1	0,999999496	0,999999992	2,61929E-15
1.5_GA6	GA6	10^{-3}	1,00000017	0,99997947	1,000000167	4,31508E-12
1.5_GA6	GA6	10^{-5}	1	0,9999998	0,999999997	2,61929E-15
1.5_GA6	GA6	10^{-7}	1,000000001	0,9999993	0,999999991	4,9696E-15
1.5_GA6	GA6	10^{-9}	1,000000083	0,999999916	1,000000019	5,83626E-16
1.5_GA6	GA6	10^{-11}	1,000005634	0,999961225	0,999999805	1,6419E-11
1.5_GA6	GA6	10^{-13}	1,000310945	1,000310945	1,000310945	–
1.5_SA1	SA1	–	1	0,99999631	0,99999972	2,66574E-13
1.5_SA2	SA2	10^{-3}	1,00000017	0,99998989	0,99999981	1,1346E-12
1.5_SA2	SA2	10^{-5}	1	0,99999835	0,99999976	1,16352E-13
1.5_SA2	SA2	10^{-7}	1	0,99999799	0,99999972	1,51652E-13
1.5_SA2	SA2	10^{-9}	1,00000003	0,99999775	0,99999972	2,09879E-13
1.5_SA2	SA2	10^{-11}	1,00000008	0,99998898	0,99999997	1,2326E-12
1.5_SA2	SA2	10^{-13}	1,00000008	0,99998898	0,99999997	1,2326E-12
1.5_SA3	SA3	–	1	0,999997513	0,9999996	2,11607E-13
1.5_SA4	SA4	10^{-3}	1,00000016	0,999998	0,99999981	1,69848E-13
1.5_SA4	SA4	10^{-5}	0,999999996	0,999997122	0,999999679	2,6669E-13
1.5_SA4	SA4	10^{-7}	1	0,999997319	0,999999627	2,78645E-13
1.5_SA4	SA4	10^{-9}	1,000000027	0,999997418	0,99999961	2,78645E-13
1.5_SA4	SA4	10^{-11}	1,000000083	0,999994532	1,000000027	3,08149E-13
1.5_SA4	SA4	10^{-13}	1,000310945	1,000310945	1,000310945	–

Function 1.6:

	Setup	h	Maximum	Minimum	Average	Variance
1.6_GA1	GA1	–	0,999847129	0,71812194	0,950040437	0,002879437
1.6_GA2	GA2	10^{-3}	0,999730345	0,566565209	0,961214249	0,00433091
1.6_GA2	GA2	10^{-5}	0,99996827	0,673952089	0,951011582	0,003556578
1.6_GA2	GA2	10^{-7}	0,999922135	0,777183442	0,954599505	0,002214533
1.6_GA2	GA2	10^{-9}	0,999970801	0,77465407	0,953829332	0,002188316
1.6_GA2	GA2	10^{-11}	0,999950123	0,735805167	0,942218548	0,003635083
1.6_GA2	GA2	10^{-13}	0,999755834	0,735002337	0,950411707	0,003014992
1.6_GA3	GA3	–	1	0,999992648	0,999999228	2,22217E-12
1.6_GA4	GA4	10^{-3}	1	0,999993272	0,999999234	1,84745E-12
1.6_GA4	GA4	10^{-5}	1	0,99999277	0,999999259	1,78523E-12
1.6_GA4	GA4	10^{-7}	1	0,999992135	0,999999179	2,39796E-12
1.6_GA4	GA4	10^{-9}	1,000000006	0,999977136	0,999999031	7,0493E-12
1.6_GA4	GA4	10^{-11}	1,000000777	0,999993144	0,999999458	1,74598E-12
1.6_GA4	GA4	10^{-13}	1,000102778	0,999964	1,000048655	9,31839E-10
1.6_GA5	GA5	–	1	0,997486731	0,999974174	6,314E-08
1.6_GA6	GA6	10^{-3}	1	0,999947222	0,999998919	4,69314E-11
1.6_GA6	GA6	10^{-5}	1	0,999974867	0,999999531	8,17994E-12
1.6_GA6	GA6	10^{-7}	1	0,998738909	0,999987305	1,59017E-08
1.6_GA6	GA6	10^{-9}	1,000000013	0,9999956	0,999999835	4,26774E-13
1.6_GA6	GA6	10^{-11}	1,000001471	0,999957062	0,999998771	4,07353E-11
1.6_GA6	GA6	10^{-13}	1,000102778	0,974914593	0,999796773	6,31784E-06
1.6_SA1	SA1	–	0,999999998	0,99999355	0,99999944	6,6887E-13
1.6_SA2	SA2	10^{-3}	1	0,999985595	0,999999107	3,08235E-12
1.6_SA2	SA2	10^{-5}	0,999999999	0,999985564	0,999999019	3,26805E-12
1.6_SA2	SA2	10^{-7}	0,999999996	0,999992991	0,999999335	1,29834E-12
1.6_SA2	SA2	10^{-9}	1,000000006	0,999985664	0,999999062	2,97044E-12
1.6_SA2	SA2	10^{-11}	1,000000777	0,999993144	0,999999403	1,4783E-12
1.6_SA2	SA2	10^{-13}	1,000102778	1,000033389	1,000096533	3,98317E-10
1.6_SA3	SA3	–	0,99999998	0,99999175	0,999998785	2,54679E-12
1.6_SA4	SA4	10^{-3}	0,999999991	0,999990965	0,999998893	2,19644E-12
1.6_SA4	SA4	10^{-5}	0,999999991	0,999987231	0,999998731	4,00004E-12
1.6_SA4	SA4	10^{-7}	0,999999997	0,999991433	0,999999094	1,41886E-12
1.6_SA4	SA4	10^{-9}	0,999999999	0,999986337	0,999998923	3,78861E-12
1.6_SA4	SA4	10^{-11}	1,000000777	0,99999245	0,999999042	2,75758E-12
1.6_SA4	SA4	10^{-13}	1,000102778	1,000033389	1,000091676	6,53649E-10