Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

BACHELOR PROJECT

Clustering in Evolutionary Algorithms with Dependent Parent Selection

Prague, 2010 **Author: Tomáš Juchelka**

I declare that this bachelor project is all my own work and I have cited all sources I have used in the bibliography.

In Prague 25.5.2010

[illegible]
signature

Acknowledgement

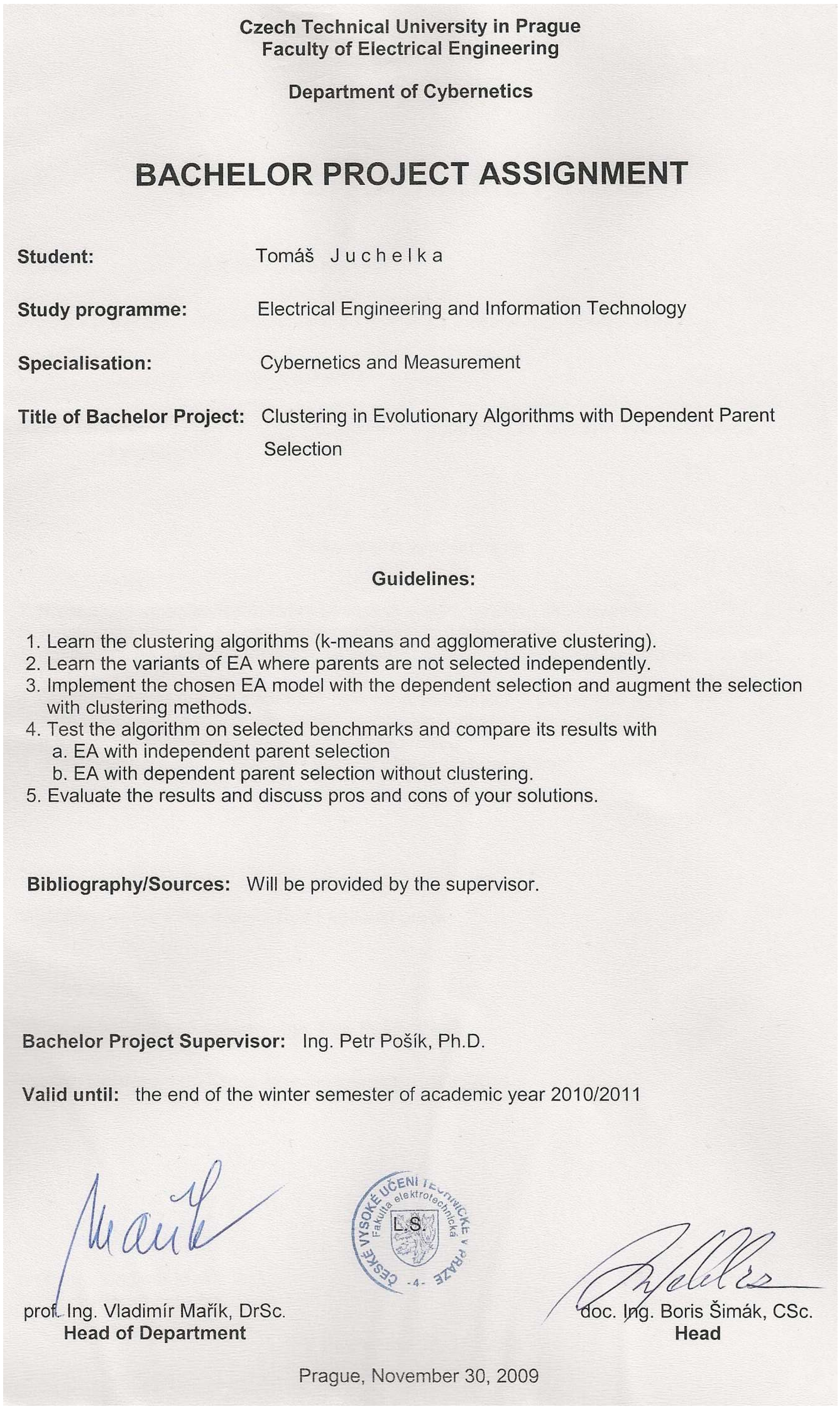
I would like to express my thanks to my supervisor Ing. Petr Pošík, Ph.D. for his help and guidance to complete this project.

Abstract

The goal of this work is to explore the impact of clustering on the performance of evolutionary algorithms. This work focuses on hierarchical clustering. The effect of these modifications is tested on unimodal and multimodal functions with real representation. The results show that clustering is very problem dependent and may be useful for some multimodal problems where the algorithm was able to find better solution, but on the other side its application on unimodal problems is unsuitable in light of its time consuption.

Abstrakt

Cílem této práce je prozkoumat efekt shlukování na výkon evolučních algoritmů. Tato práce se zaměřuje na hierarchické shlukování. Efekt těchto modifikací je testován na unimodálních a multimodálních funkcích s reálnou reprezentací. Výsledky ukazují, že shlukování je velmi závislé na daném problému a může být užitečné pro některé multimodální problémy, kde algoritmus byl schopen najít lepší řešení, ale na druhou stranu jeho použití na unimodálních problémech je nevhodné, vzhledem k jeho časové náročnosti.



Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Tomáš J u c h e l k a
Study programme:	Electrical Engineering and Information Technology
Specialisation:	Cybernetics and Measurement
Title of Bachelor Project:	Clustering in Evolutionary Algorithms with Dependent Parent Selection

Guidelines:

1. Learn the clustering algorithms (k-means and agglomerative clustering).
2. Learn the variants of EA where parents are not selected independently.
3. Implement the chosen EA model with the dependent selection and augment the selection with clustering methods.
4. Test the algorithm on selected benchmarks and compare its results with
   a. EA with independent parent selection
   b. EA with dependent parent selection without clustering.
5. Evaluate the results and discuss pros and cons of your solutions.

Bibliography/Sources: Will be provided by the supervisor.

Bachelor Project Supervisor: Ing. Petr Pošík, Ph.D.

Valid until: the end of the winter semester of academic year 2010/2011

[illegible]

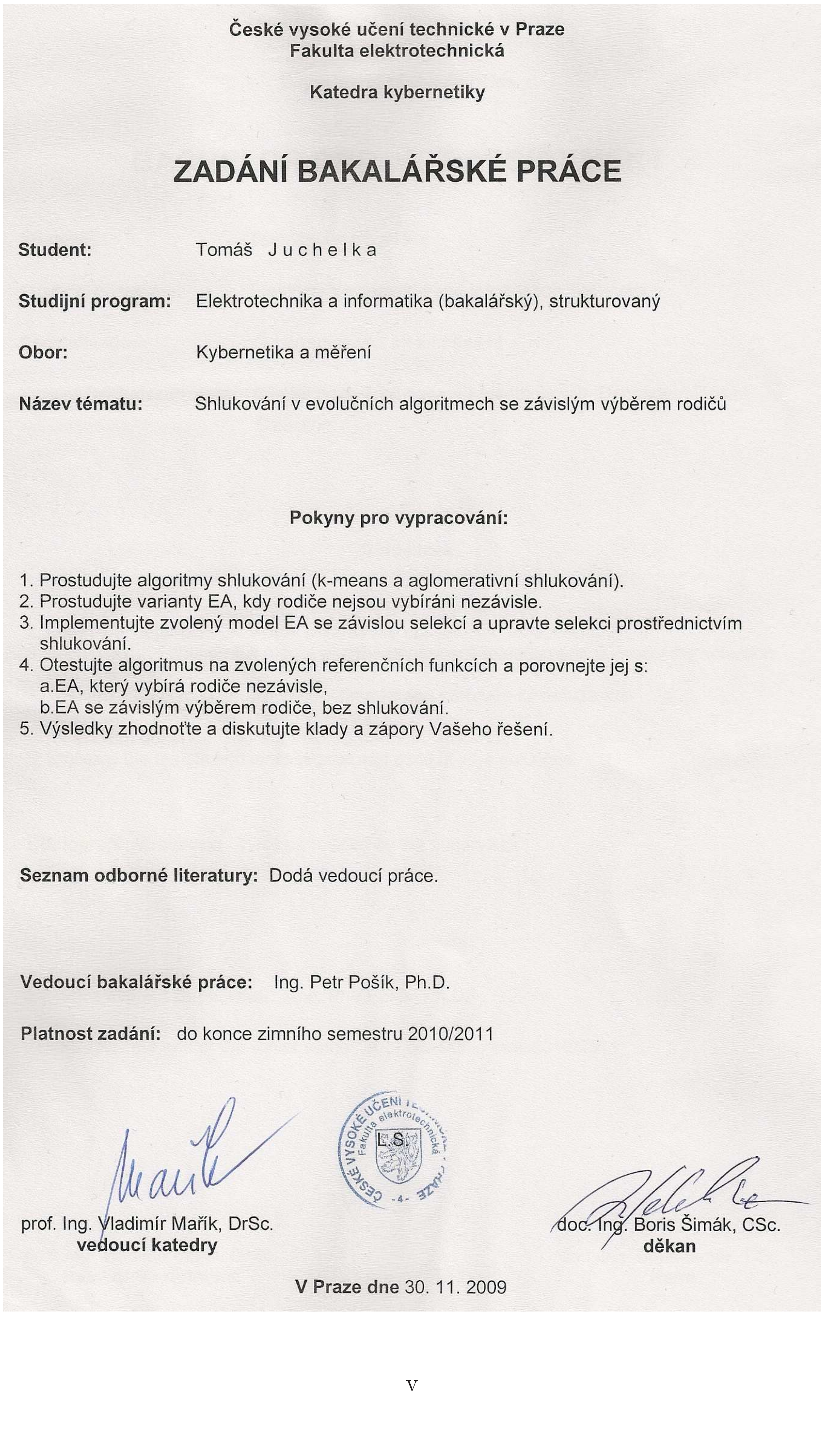
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

[illegible]

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

doc. Ing. Boris Šimák, CSc.
Head

Prague, November 30, 2009



České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Tomáš J u c h e l k a
Studijní program:	Elektrotechnika a informatika (bakalářský), strukturovaný
Obor:	Kybernetika a měření
Název tématu:	Shlukování v evolučních algoritmech se závislým výběrem rodičů

Pokyny pro vypracování:

1. Prostudujte algoritmy shlukování (k-means a aglomerativní shlukování).
2. Prostudujte varianty EA, kdy rodiče nejsou vybíráni nezávisle.
3. Implementujte zvolený model EA se závislou selekcí a upravte selekci prostřednictvím shlukování.
4. Otestujte algoritmus na zvolených referenčních funkcích a porovnejte jej s:
   a.EA, který vybírá rodiče nezávisle,
   b.EA se závislým výběrem rodiče, bez shlukování.
5. Výsledky zhodnoťte a diskutujte klady a zápory Vašeho řešení.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí bakalářské práce: Ing. Petr Pošík, Ph.D.

Platnost zadání: do konce zimního semestru 2010/2011

[illegible]

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

[illegible]

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 30. 11. 2009

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Evolutionary algorithms are one of the well known optimalization methods inspired in biology. They use principles that can be found in nature for finding the optimal solution to a problem. They are highly adjustable and can be applied on almost every problem. Although the evolutionary algorithms work fine, this work tries to improve them with clustering methods. In classic selections, better individuals have greater probability to become parents for next generation. This work extends Václav Havlín's analysis [8] and tries to explore the impact of clustering on each type of selection. There is an effort to consider similarities between individuals which are selected by methods, explained in that analysis and to achieve a similar effect as in Parallel EAs [3]. Parallel EAs are supposed to explore a problem's search space more evenly and they may avoid population stagnation because of better capability for maintaining overall diversity. The goal of this work is to consider influence of clustering on the results. There is a chance that clustering will have positive effect on multimodal problems, because of the higher diversity in population. It is possible that this modification, due to its time consumption, may be useless on some problems.

Chapter 2 describes basics and terminology of evolutionary algorithms. Chapter 3 explains clustering process and its variants. In chapter 4, there are listed functions used for testing the algorithm. Chapter 5 shows the test results. The chapter 6 contains an assessment.

Chapter 2

Evolutionary algorithms

2.1 Introduction to evolutionary algorithms

Evolutionary algorithms (EA) are stochastic search methods inspired in biology [1]. EA works on a population of potential solutions randomly generated at the beginning of the search process. Each individual represents a solution in search space of possible solutions and is evaluated by a function which is able to judge the quality of an individual [2]. This evaluation function is called fitness function. This function must be at least capable of differentiating between two individuals.

Those solutions, that are better, as determined by the evaluation function, are more likely to be chosen as parents for the next generation of offspring. In each generation, a new set of solutions is generated probabilistically in the neighborhood of old solutions by the process of selecting individuals according to their quality of fitness function. Each individual contains its own genotype. Genotype in biology contains a genetic information. In EA the genotype represents a solution and is mostly formed by a string of binary or real numbers, but also it may be any other representation. The principle of EA is still the same but may vary in natural processes which are included. A big advantage of EAs compared to classical methods is that complete knowledge of a problem isn't needed.

2.2 Genetic algorithms

Genetic algorithms (GA) are part of the evolutionary algorithms. GA use techniques inspired in biology and model natural processes, such as selection, recombination and mutation. Each individual is typically represented as a string called genome. Individuals are evaluated by a fitness function according to quality of genome thus solution.

Initial population is randomly generated at the beginning of search process, then it is evaluated by the fitness function. At this time we know which solutions are worse and which are better. This is important for using the selection. An offspring is formed with the individuals which are stochastically selected from the current population and then modified. This modification includes recombination and mutation.

We can describe GA this way [7]:

Step 0: (Initialization)
The initial population is randomly generated. Because it is a random process, it is good to start GA several times and try to eliminate effect of initial conditions. It's possible to generate the initial population in area where optimal solution can be found.

Step 1: (Selection)
Two parents are chosen from population according to some selection method.

Step 2: (Recombination)
Also called crossover, is the next step of generating new population. An offspring is produced as a combination of two chosen parents. It's obvious that the offspring has some similar characteristics as his parents.

Step 0: (Mutation)
The offspring is mutated with constant probability. It means that his genome is amended somehow.

2.3 Designing evolutionary algorithms

The essential idea of evolutionary problem solving is quite simple. A population of candidate solutions to the problem is evolved over successive iterations of variation and selection. Variation provides the mechanism for discovering new solutions. Selection determines which solutions to maintain for further exploration. The goal of the exploration is to find a solution. The time in which any solution can be discovered depends on the representation of solution, the evaluation function, the variation and selection operators, the size of population and on other options. The key to designing successful evolutionary algorithms lies in making the appropriate choices in light of the problem [2].

Evolutionary algorithm used in this work can be captured by the equation:

$$x[t+1] = v(s(x[t])) \tag{2.1}$$

Where *x[t]* is the population under a representation x at time t, v is the variation operator and s is the selection operator described in 2.3.4.

2.3.1 Representation

A representation is a mapping from the state space of possible solutions to the state space of encoded solutions in any data structure. The most used data represenation is a fixed-length vector of symbols. A typical representation is a list of bits 0,1 called binary string.

2.3.2 Evaluation function

The evaluation function is an instrument for judging the quality of individuals. This function itself may represent the solution, or simply compare two candidate solutions. An accuracy of the evaluation function has wide effect on quality of discovered solution and obviously on the duration of the search process.

2.3.3 Variation operators

In general, the choice of operators depends on the representation. These operators serve to create a new population from the old one in a recombination process. There exist operators that work with one or two or more parents. By this process, new possible solutions which contain genetic information from their parents, are created.

2.3.4 Selection

It would be appropriate to distinguish two different contexts of this term. Firstly, the process of selecting parents (selection) and secondly the process of selecting individuals for replacement (replacement strategy). The selection methods are more desribed in 2.4.

2.3.5 Initialization

The difference equation 2.1 indicates the evolution over each generation. Therefore the initial population is needed to be defined before starting. It's possible to generate population randomly or it's useful to focus the generation process where the solution is situated. For benchmarking evolutionary algorithms it is useful to generate population at random form.

2.4 Selection methods

As selections of parents for next generation this work uses Fitness-based and Fitness-independent selections, described in work of Václav Havlín [8].

2.4.1 Fitness-based selection

In Fitness-based selection, both parents are selected by value of fitness and no matter which individuals was chosen. This is classic selection method and may have different selection pressure which affects the convergence. An example is Tournament selection, Ranking selection, Proportionate selection (Roulette-wheel) etc.

2.4.2 Fitness-independent selection

V. Havlín implemented genetic algorithms [8] with fitness-independent selection of the second parent. First parent is always selected by the best fitness function from a tournament, formed by individuals randomly selected from the population. Second parent selection depends on one of these selection methods: Closest, Farthest, Random. In Closest selection, the second parent is selected from the tournament to be the closest to the first parent. It is measured by the Euclidian distance. In Farthest selection the farthest parent is chosen. In Random selection, the second parent is selected without the tournament, straight from the whole population.

Chapter 3

Selection with Clustering

3.1 Clustering

Clustering [5] is a common method used in many fields, including marketing, data and image analysis, data mining, pattern recognition and many others. It is the process of organizing objects into groups whose members are similar in some way. In this case the similarity means the distance between objects.

3.2 Distance Measure

An important part of a clustering is a way how to measure the distance between objects. An easy way is to use the euclidean distnace metric which is defined as:

$$d(x,y) = \left(\sum_{k=1}^{D} |x_k - y_k|^p \right)^{1/p} \tag{3.1}$$

For the euclidean distance $p = 2$. D is number of dimensions.

3.3 Hierarchical clustering

Hierarchical clustering doesn't create clusters in a single step, but the clusters are created successively by a process based on some linkage method. It maintains a distance matrix *N*N*, where distances between clusters are stored. This is quite time consuming, but without this, it would be necessary to compute all distances at each step. The hierarchical clustering is often represented as a dendrogram. There are two major methods of hierarchical clustering - agglomerative and divisive.

3.3.1 Linkage types

Linkage defines the way how the clusters will be constructed. In this work there are used three types of linkage. Single, average and complete linkage. Each type has wide effect on size and position of clusters.

3.3.1.1 Single linkage

Single linkage is also called nearest neighbour. The distance between two clusters is equal to the shortest distance from any member of the first cluster to any member of the second cluster. This method often leads to the creation of one big and few smaller clusters. The distance between clusters X and Y is defined as:

$$D(X, Y) = min(d(x, y)) \tag{3.2}$$

Where d(x,y) is distance between elements from clusters.

3.3.1.2 Average linkage

The distance is computed as an average pairwise distance between clusters. This type of linkage makes the clusters most balanced.

3.3.1.3 Complete linkage

The distance between two clusters is equal to the greatest distance from any member of the first cluster to any member of the second cluster. It works like single linkage but looks for the maximal distance between elements.

3.3.2 Dendogram

A dendogram is a tree that illustrates order of merging the clusters with hierarchical clustering. Each cluster is represented as a node, connected with other cluster. The edge, connecting clusters, shows the distance computed between clusters. By dendogram analyzing the number of clusters can be determined.

3.3.3 Agglomerative clustering

Agglomerative clustering (bottom-up) transforms each individual to a single cluster, and then successively merges clusters until all of them are merged into a single remaining cluster. The important step is defining optimal number of clusters. In this work there is used method based on measure distance between two merging clusters. The algorithm stops when the distance is higher than the previous one. The algoritm is described as follows [5]:

1) Make N clusters, each with one individual.

2) Find nearest two similar clusters and merge them together.

3) Compute distance between the new cluster and the others.

4) Repeat steps 2 and 3 till only one cluster remains.

3.3.4 Divisive clustering

Divisive (top-down) clustering works like agglomerative one but there is a difference. Divisive algorithm starts with one cluster which contains the whole population. Then the cluster is divided into several smaller clusters which are also being divided until it is appropriate.

3.4 Partitional clustering

Partitional clustering creates all clusters at once. Generally it is necessary to set the number of clusters at the beginning. These algorithms are fast and dependent on initial set. Because of this fact the K-means algorithm in this work isn't used for testing but it's only mentioned or somewhere it can be shown for comparison.

3.4.1 K-means clustering

K-means is fast algorithm and it can converge to different cluster arrangement everytime it's started on the same population. At the beginning k centroids are chosen and each individual is assigned to the nearest centroid. After assignement of all individuals, new centroids are computed. K-means minimizes mean deviation between population and centroids. The algorithm may look like this [4]:

1) Choose k centroids in current population.

2) Assign individuals to the nearest centroids.

3) Compute new position of centroids as an average of cluster members.

4) If the position of the centroids changes, go to step 2.

3.5 Application of clustering

The application of clustering in EA is trying to find similarities between individuals and to group them. These similarities can be based on many observations. The goal is to use the selections desribed in 2.4 not on the whole population, but on the separated parts (clusters) independently.

There are two possible cases. The first one is to generate as many individuals as the size of the actual cluster. The second one is to balance the offspring's number. Each cluster will produce the same ammount of new individuals while it doesn't matter on its size. There can be assigned numbers to the both cases that will represent them. Suppose that 0 stands for balancing and 1 means the same size. It looks like that the optimal case lies somewhere between these two extreme cases. So there are these possibilities: First one is that offspring size of each cluster is proportional to its size:

$$n'_i = \frac{n_i}{N} N_{\text{Offs}} \tag{3.3}$$

Second one is that each cluster has the same offspring size:

$$n'_i = \frac{1}{N_c} N_{\text{Offs}} \tag{3.4}$$

By using parameter α, there will be a combination described as:

$$n'_i = \alpha \frac{n_i}{N} N_{\text{Offs}} + (1-\alpha)\frac{1}{N_c} N_{\text{Offs}} \tag{3.5}$$

Where N is population size, N_{Offs} is offspring size, N_c is total number of clusters and n_i is cluster size. Minimum number of clusters is limited to prevent the creation of one big cluster, but it has side effect, that there must always be the minimum number of clusters, even if they contain only one individual.

Chapter 4

Test functions

For benchmarking the algorithm modificated by clustering, functions, which cover typical dificulties which can occur in continuous domain search, are selected. All the functions are scalable with the dimension. Some functions have no specific value of their optimal solution. Complete description of the functions can be found in [6]

4.1 Unimodal functions

4.1.1 Sphere function

The easiest continuous domain search problem. It's unimodal and symetric function.

$$f_{Sphere}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 \tag{4.1}$$

$\mathbf{x}^* = \mathbf{0},\ f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5.12; 5.12]^D$

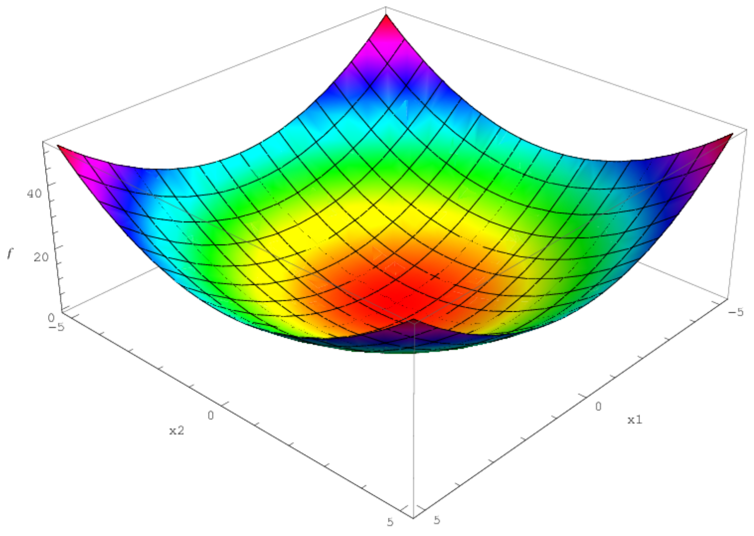


Figure 4.1: Sphere function

4.1.2 Ellipsoidal function

Globally quadratic unimodal function with smooth local irregularities.

$$f_{Ellipsoidal}(\mathbf{x}) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} x_i^2 \tag{4.2}$$

$\mathbf{x}^* = \mathbf{0},\ f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5; 5]^D$

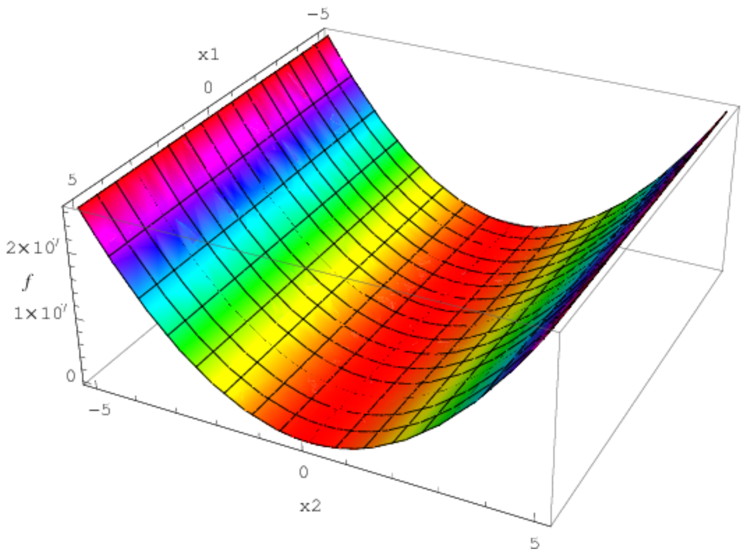


Figure 4.2: Ellipsoidal function

4.1.3 Sharp ridge function

In this function, the gradient remains constant, when the ridge is approached from a given point. Approaching the ridge is initially effective, but becomes ineffective close to the ridge where the rigde needs to be followed in z-direction to its optimum.

$$f_{SharpRidge}(\mathbf{x}) = x_1^2 + 100\sqrt{\sum_{i=2}^{D} x_i^2} \tag{4.3}$$

$\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5.12; 5.12]^D$

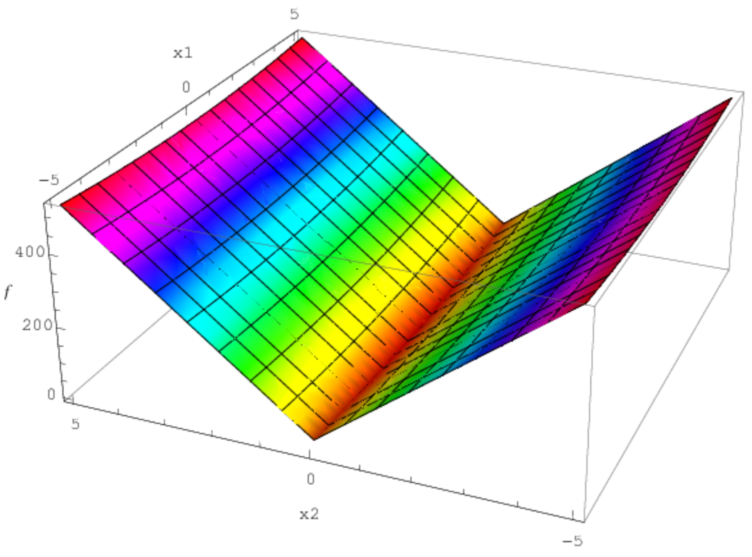


Figure 4.3: Ellipsoidal function

4.1.4 Linear function

Purely linear function.

$$f_{Linear}(\mathbf{x}) = \sum_{i=1}^{D} -x_i \tag{4.4}$$

$\mathbf{x}^* = \mathbf{10}$, $f(\mathbf{x}^*) = -10 \cdot D$ for $\mathbf{x} \in [-10; 10]^d$

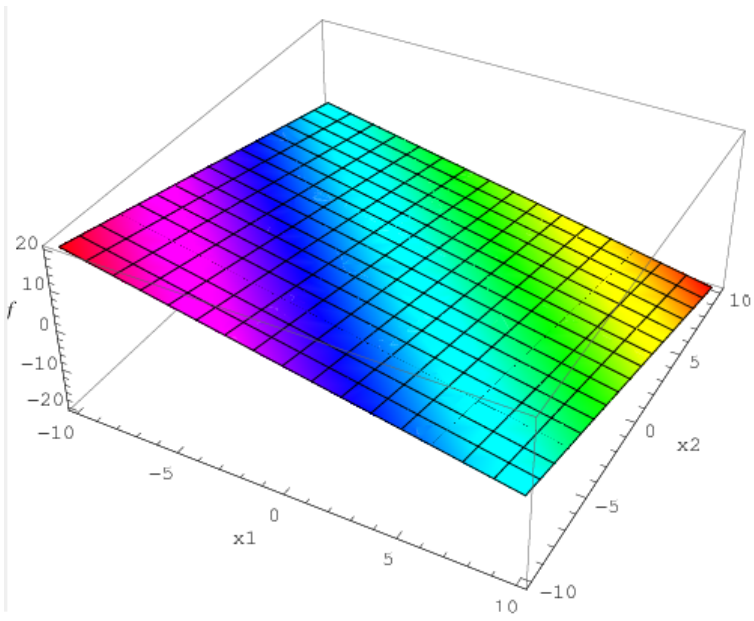


Figure 4.4: Linear function

4.2 Multimodal functions

4.2.1 Double funnel function

Double funnel function is a minimum from two sphere functions. Both create their own optimum. It is possible to change size of the barrier between them and simulate different relative size and depth by parameters *h,s*. Global optimum is placed in $\mu_1 = 2.5$ and local optimum is moved to $\mu_2 = -\sqrt{\frac{\mu_1^2 - h}{s}}$. This work will focus on the case where $s = 0.2$ and $h = 1$.

$$f_{DoubleFunnel}(\mathbf{x}) = \min\left(\sum_{i=1}^{D}(x_i - \mu_1)^2, h \cdot D + s\sum_{i=1}^{D}(x_i - \mu_2)^2\right) \tag{4.5}$$

$\mathbf{x}^* = \mathbf{2.5},\ f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5; 5]^D$

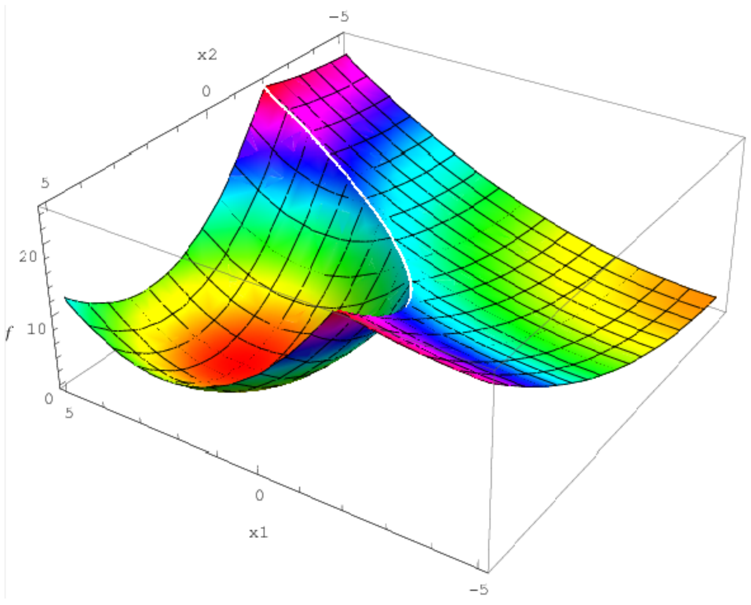


Figure 4.5: Double funnel function

4.2.2 Rastrigin function

Multiomodal function with a regular structure, many local optima and one global optima of zero at the origin. The function is highly multimodal. Rastrigin's function is often used to test the genetic algorithm.

$$f_{Rastrigin}(\mathbf{x}) = 10\left(D - \sum_{i=1}^{D} \cos(2\pi x_i)\right) + \sum_{i=1}^{D} x_i^2 \tag{4.6}$$

$\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5.12; 5.12]^D$

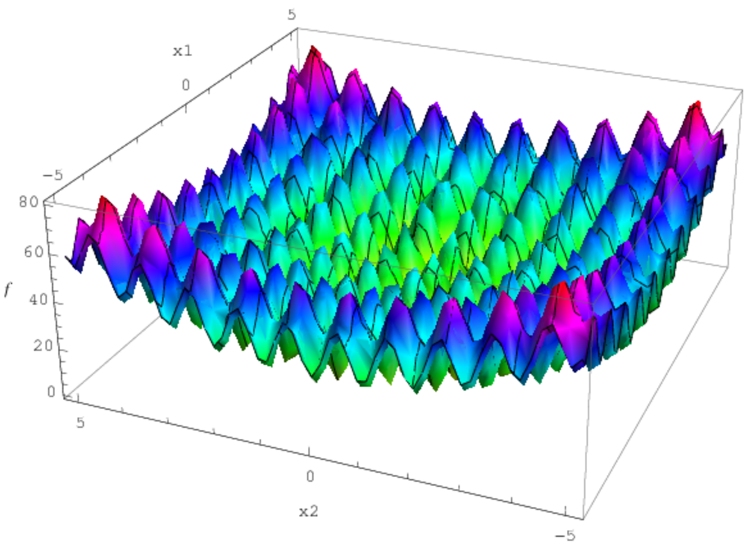


Figure 4.6: Rastrigin function

4.2.3 Rosenbrock function

So-called banana function. Global optimum lies in long narrow valley. It's easy to find the valley but it's hard to reach global optimum.

$$f_{Rosenbrock}(\mathbf{x}) = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right) \tag{4.7}$$

$\mathbf{x}^* = \mathbf{1},\ f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-2.048; 2.048]^D$

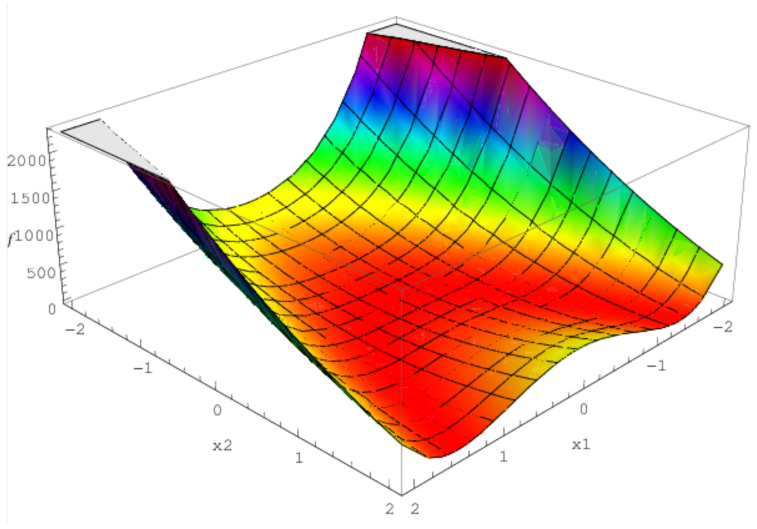


Figure 4.7: Rosenbrock function

4.2.4 Weierstrass function

Highly rugged and moderately repetitive landscape, where the global optimum is not unique.

$$f_{Weierstrass}(\mathbf{x}) = 10\left(\frac{1}{D}\sum_{i=1}^{D}\sum_{k=0}^{11}\frac{1}{2^k}\cos(2\pi 3^k(x_i+0.5)) - f_0\right)^3 \tag{4.8}$$

Where $f_0 = \sum_{k=0}^{11}\frac{1}{2^k}\cos(2\pi 3^k 1/2)$
$\mathbf{x}^* =$ not unique, $f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5;5]^D$

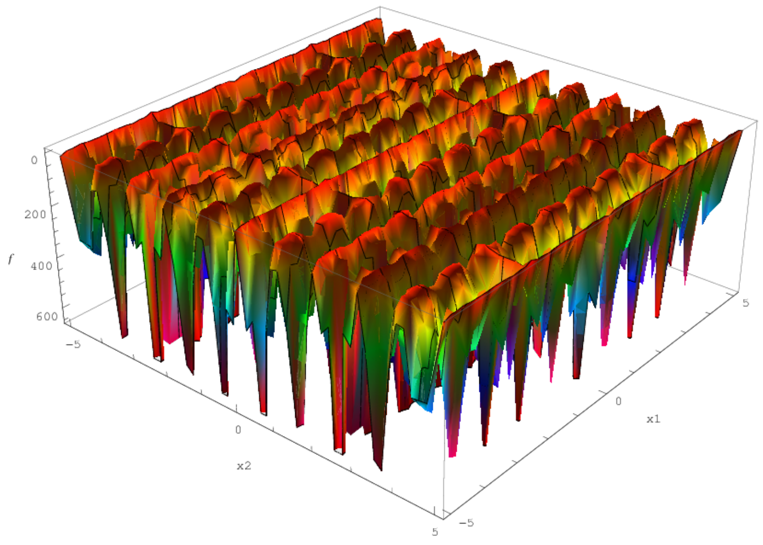


Figure 4.8: Weierstrass function

4.2.5 Schaffers F7 function

A highly multimodal function where frequency and amplitude of the modulation vary.

$$f_{SchaffersF7}(\mathbf{x}) = \sum_{i=1}^{D-1}(x_i^2 + x_{i+1}^2)^{0.25}(\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1) \tag{4.9}$$

$\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5;5]^D$

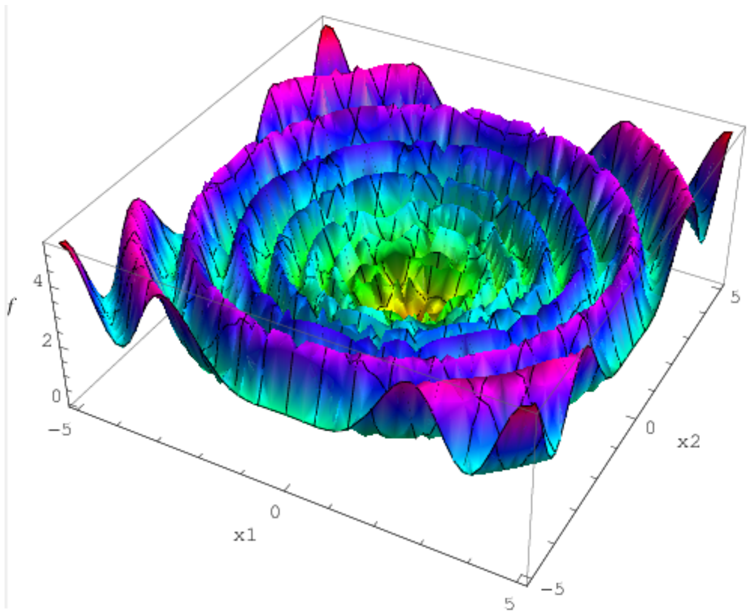


Figure 4.9: Schaffers F7 function

4.2.6 Katsuura function

Highly rugged and highly repetitive function with weak global structure.

$$f_{Katsuura}(\mathbf{x}) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{32} \frac{|2^j x_i - [2^j x_i]|}{2^j} \right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} \qquad (4.10)$$

$\mathbf{x}^* =$ not unique, $f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5; 5]^D$

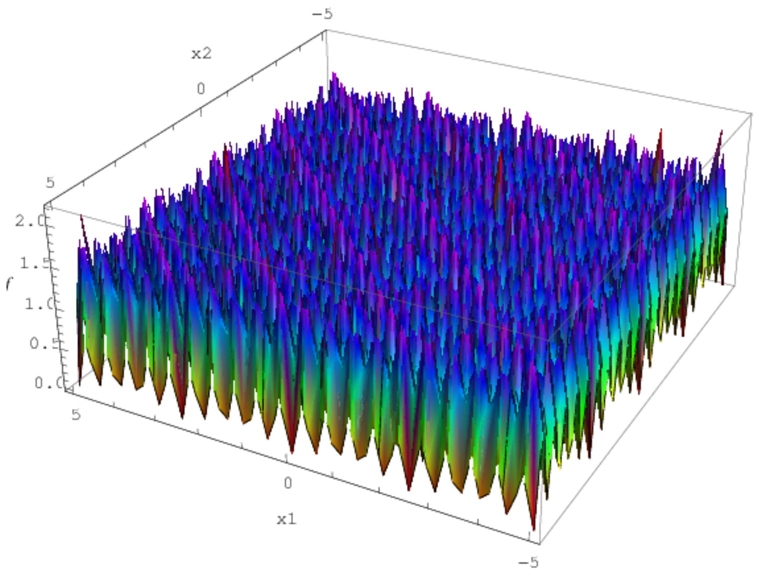


Figure 4.10: Katsuura function

4.2.7 Michalewicz function

Multimodal function with D! local optima. Larger exponent leads to more difficult search. The function values for points in the space outside the narrow peaks give very little information on the location of the global optimum.

$$f_{Michalewicz}(\mathbf{x}) = -\sum_{i=1}^{D} \sin(x_i) \left(\sin\left(\frac{i.x_i^2}{\pi} \right) \right)^{20} \tag{4.11}$$

$\mathbf{x}^* =$ not specified, $f(\mathbf{x}^*) =$ -4.687 for 5D and -9.66 for 10D for $\mathbf{x} \in [0;\pi]^D$

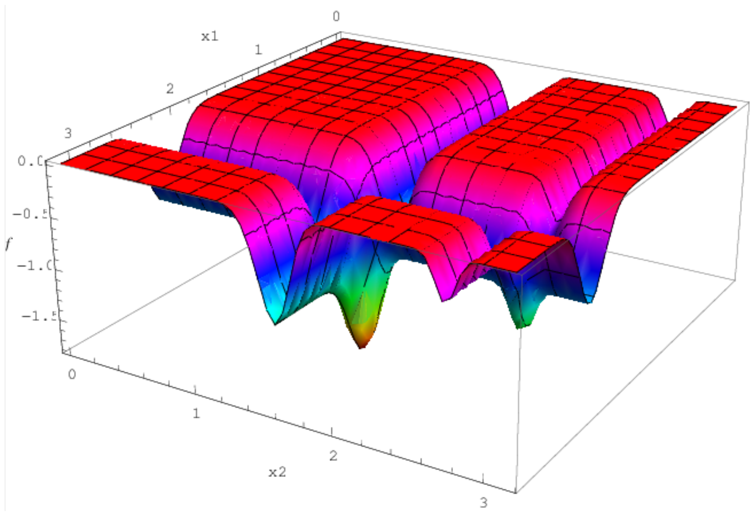


Figure 4.11: Michalewicz function

4.2.8 Griewangk function

Griewangk's function is similar to Rastrigin's function. It has many widespread local minima. However, the location of the minima are regularly distributed. We are using this function in a two versions differing by a size of search space.

$$f_{Griewangk}(\mathbf{x}) = -\sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{4.12}$$

Version 1: $\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-512; 512]^D$
Version 2: $\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [-5.12; 5.12]^D$

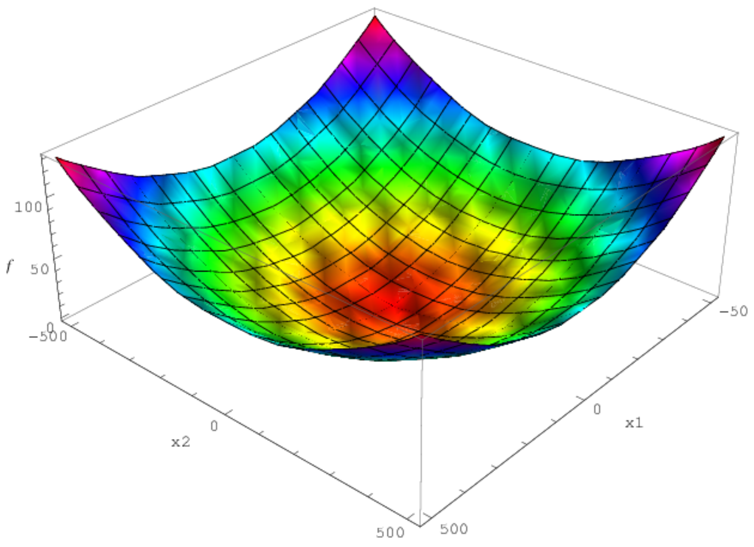


Figure 4.12: Griewangk function 1

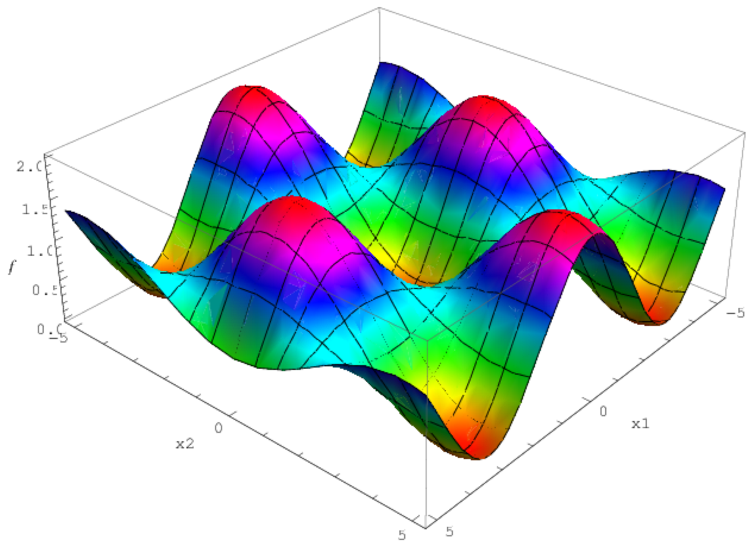


Figure 4.13: Griewangk function 2

4.2.9 EggHolder function

EggHolder is quite complicated function similar to an egg holder. Global optimum isn't defined.

$$f_{EggHolder}(\mathbf{x}) = \sum_{i=1}^{D-1} -(x_{i+1}+47)\sin(\sqrt{|x_{i+1}+\frac{x_i}{2}+47|}) + + \sin(\sqrt{|x_i-(x_{i+1}+47)|})(-x_i) \tag{4.13}$$

$\mathbf{x}^* = \text{unknown},\ f(\mathbf{x}^*) = \text{unknown for } \mathbf{x} \in [-512;512]^D$

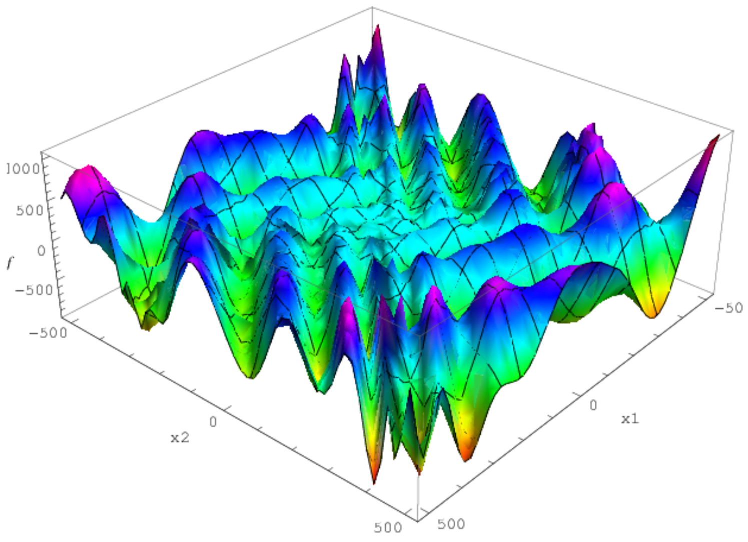


Figure 4.14: EggHolder function

4.2.10 TwoValleys function

x	fitness
0	10
0.1	0
0.2	10
0.6	1
1	10

Table 4.1: TwoValleys

$\mathbf{x}^* = \mathbf{0.1}$, $f(\mathbf{x}^*) = 0$ for $\mathbf{x} \in [0;1]^D$

In one dimension this W-shaped function is defined by the points in the table 4.1. In more dimensions it consists of a sum of particular dimensions. In the one-dimensional function there are two valleys, the first one is very narrow and contains the global optimum and the second one is wide and contains only local optimum. In more dimensions the structure is more complicated.

Chapter 5

Experiments

Objective of this chapter is to test, whether hierarchical clustering with single, average and complete linkage may improve the selections described in 2.4. For this purpose, we have selected some unimodal and multimodal functions described in chapter 4. Section 5.1 tries to answer the question, how many offspring should produce each cluster. Section 5.3 shows the results of experiments for each test function, selection method and linkage.

5.1 Offspring analysis

This analysis is trying to find out the optimal number of offspring according to the cluster size. So the goal is to find out the number between 0 and 1, thus parameter called α. Parameter α can be determined by the tests made on each function and then consider which value of α is best for each problem. It's needed to choose the value of α in which the algorithm returns sufficient solutions for each problem. This is a compromise, although the value is the best possible, there are still problems where it has negative effect.

For this task three functions with different structure and modality were selected: Sphere, Rastrigin and Weierstrass. These three functions are selected as representatives of our test functions. The parameter α was chosen on the basis of results of tests made on these functions. There is a convention, that fitness selection has blue, random cyan and farthest green color. The results are in fig. 5.1. Many values of α parameter was tested, but these two values seems to be the best ones. For better view, graphs contain only these two values for comparison. The graphs 5.1 show, that the best value of α is 0.9 and this value will be used for experiments.

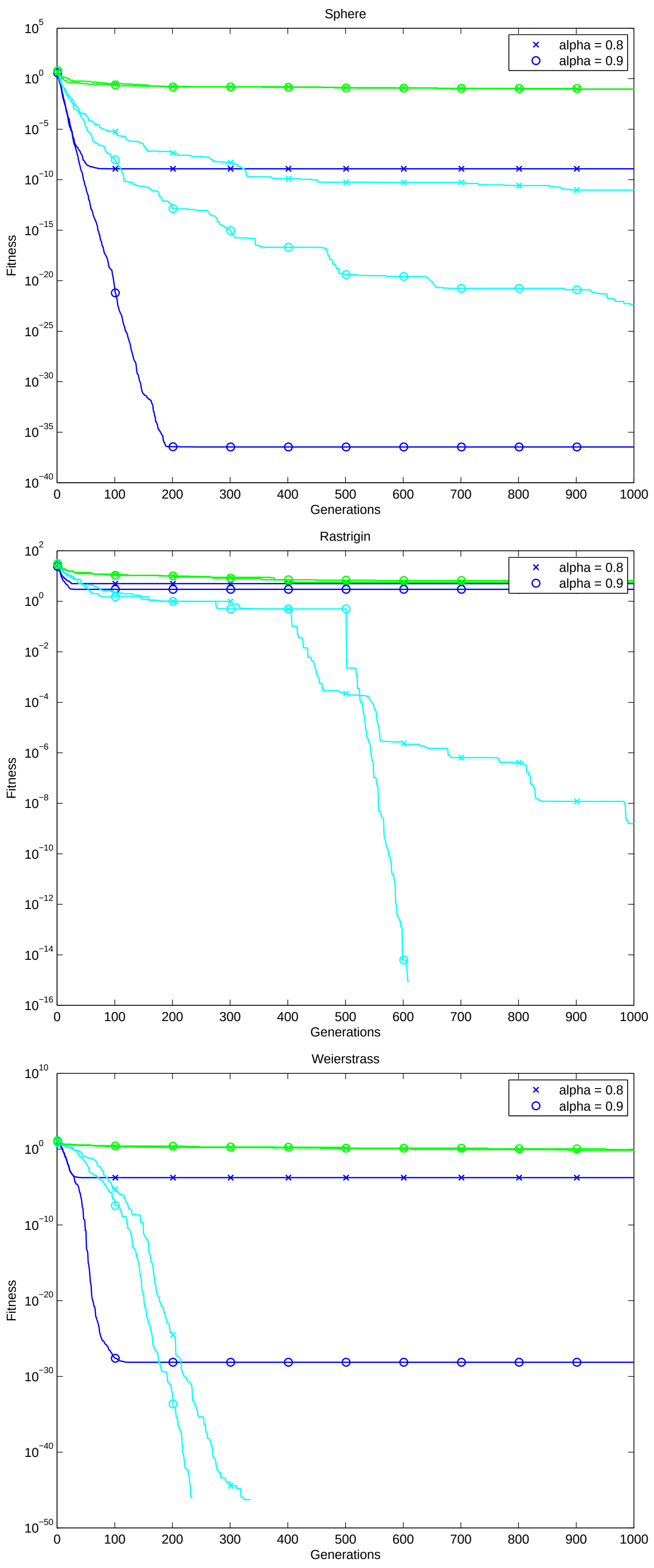


Figure 5.1: Parameter alpha exploration.

5.2 Settings

The main pointer of the algorithm's performance is the fitness function of the best individual in the actual generation. To avoid the effect of setting the initial conditions, each test is started 10 times and for each selection the same set of seeds for a random number generator is used. In graphs there is shown a median of these runs. The population size = 100, dimension = 5, number of generations = 1000. On the basis of 5.1, the parameter α is equal to 0.9 for each problem.

5.3 Results

The closest selection was omitted from the tests because the clustering can't have positive effect on this selection. Otherwise on farthest selection and on the others may occur any improvements. In the graphs below, there are shown statistics of the algorithm with applied clustering for some functions compared with the normal algorithm. In the title, there is written the name of the function and which selection is the algorithm using.

5.3.1 Unsuccessful results

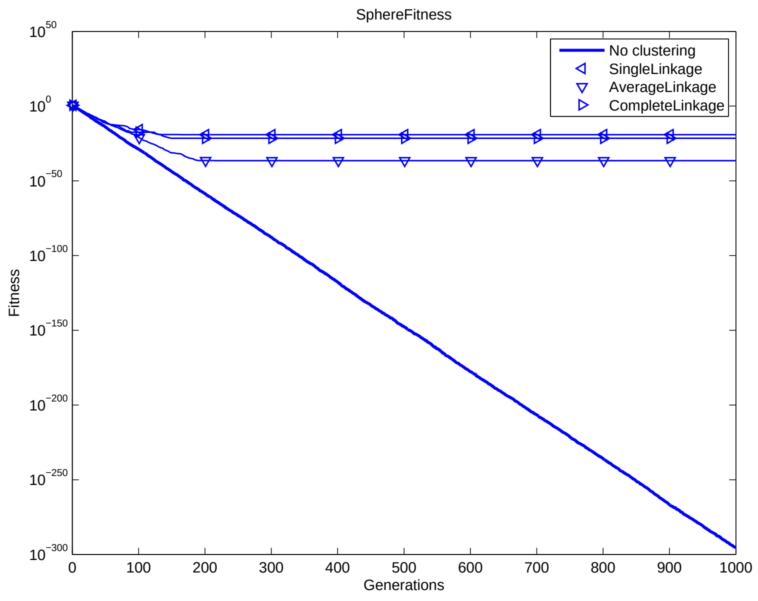


Figure 5.2: The effect of clustering for each linkage type on the Sphere function.

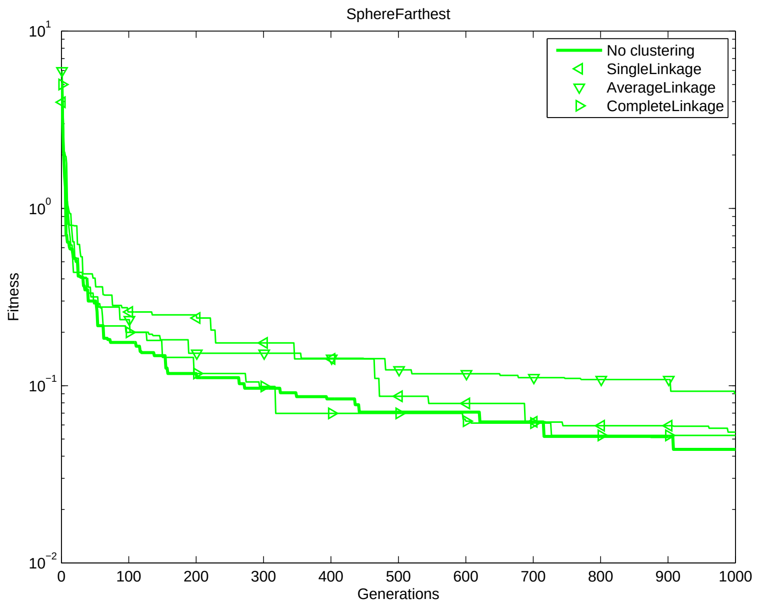


Figure 5.3: The effect of clustering for each linkage type on the Sphere function.

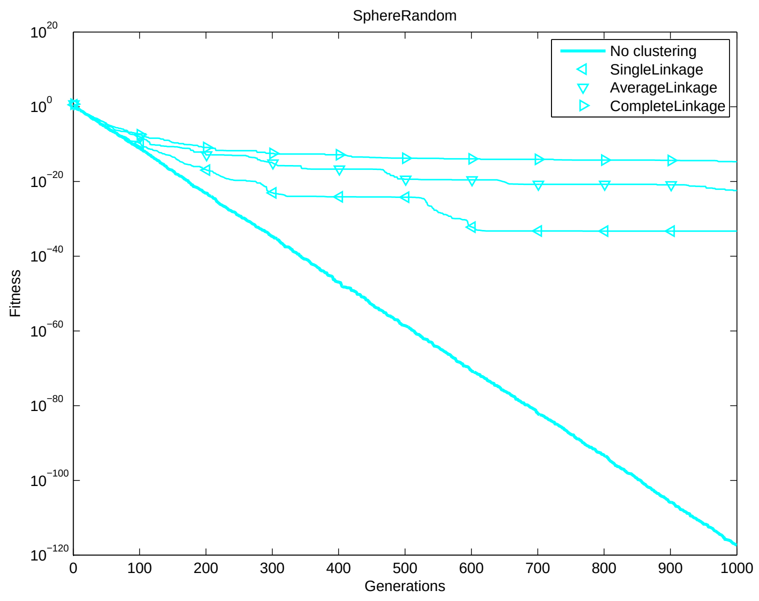


Figure 5.4: The effect of clustering for each linkage type on the Sphere function.

On the unimodal functions like Sphere, Ellipsoidal and Sharp ridge the situation is the same. So it's possible to review the effect collectively, while in graphs, there is shown only the Sphere function. These functions are unimodal and clustering has negative effect here. The creation of clusters out of the function optimum, which is unique, causes that individuals are creating offspring in an area where the solution does not lie. In fact, this means that the algorithm is focusing on the wrong areas. It slows down the search process and prevents the full convergence.

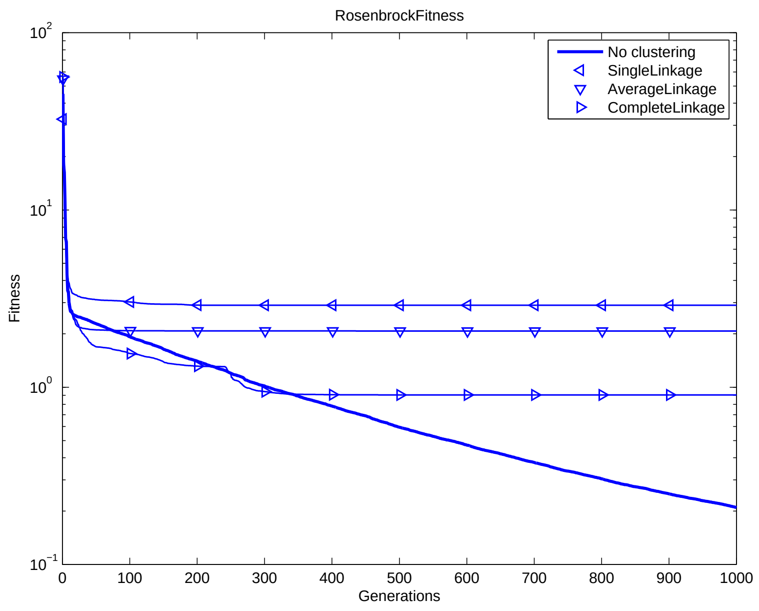


Figure 5.5: The effect of clustering for each linkage type on the Rosenbrock function.

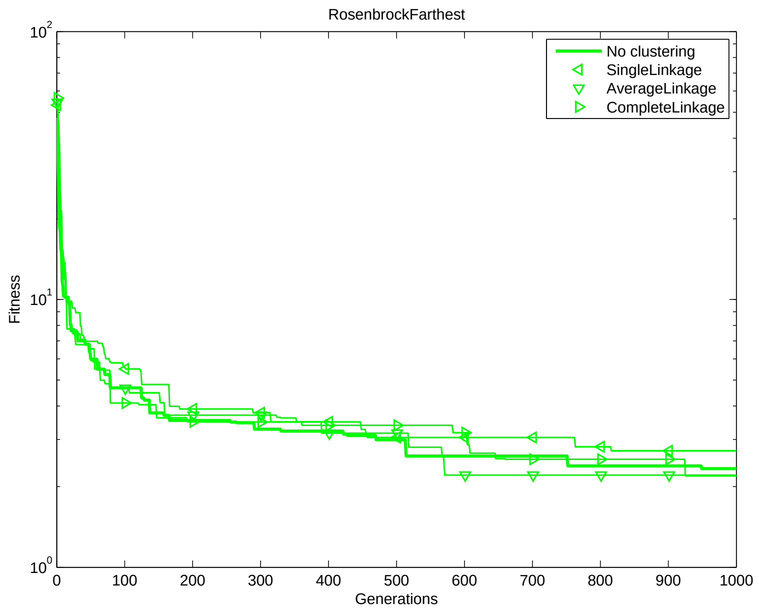


Figure 5.6: The effect of clustering for each linkage type on the Rosenbrock function.

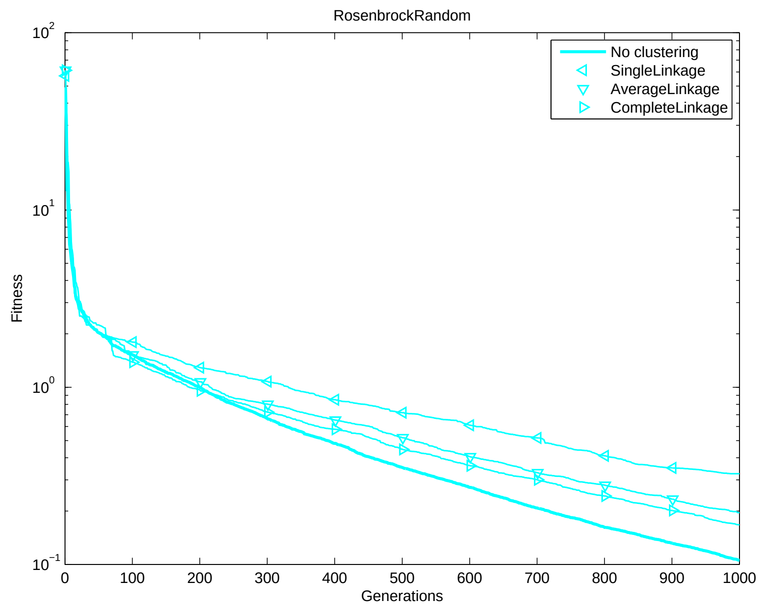


Figure 5.7: The effect of clustering for each linkage type on the Rosenbrock function.

Rosenbrock, SchaffersF7 and Griewangk functions indicate similar behavior. It's probablly caused by their global structure. The results aren't totally bad, in many cases the clustering copies the original GA. But on fitness selection, the algorithm gets stuck early.

5.3.2 Neutral results

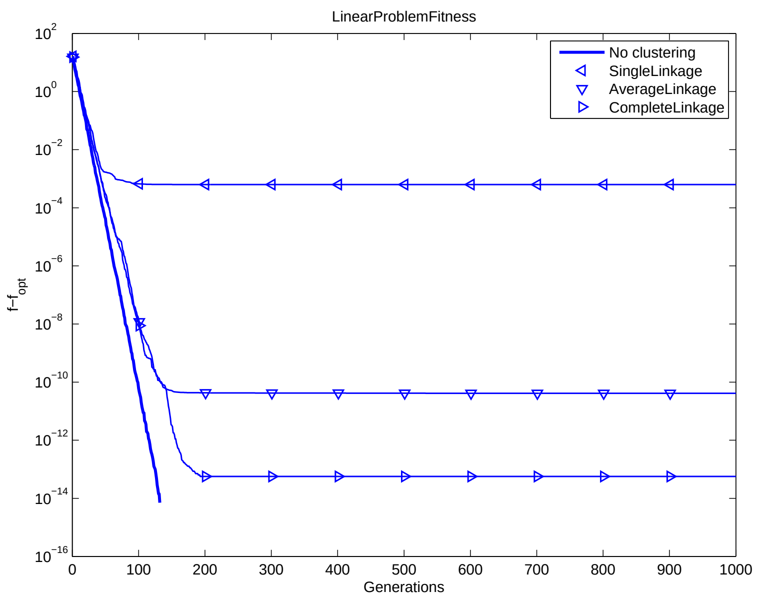


Figure 5.8: The effect of clustering for each linkage type on the Linear function.

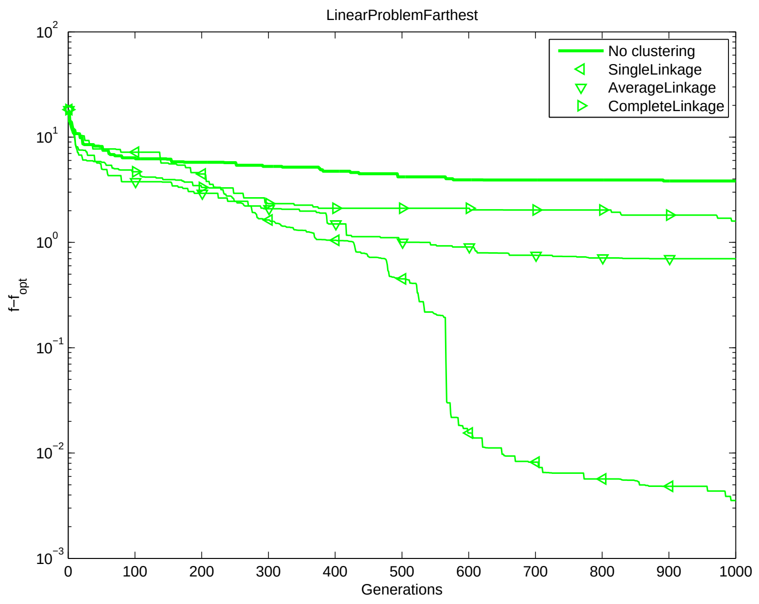


Figure 5.9: The effect of clustering for each linkage type on the Linear function.

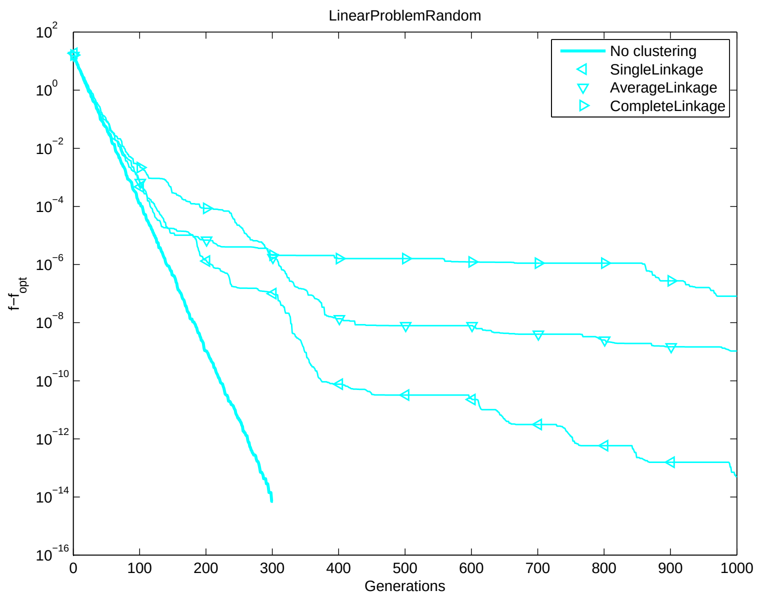


Figure 5.10: The effect of clustering for each linkage type on the Linear function.

Linear function is very simple and everywhere in the function's domain, there is the information about location of the global optima. In the initialization part are probablly generated candidate solutions near the optimum and the convergence is fast even with the clustering. It leads to creation of one big cluster in a short time. So the effect is neutral to random and fitness selections and a little improvement can be seen on farthest selection. In general on the unimodal functions, the effect on farthest selection should be better that on the others. This indicates also farthest selection on the Sphere function in fig. 5.3.

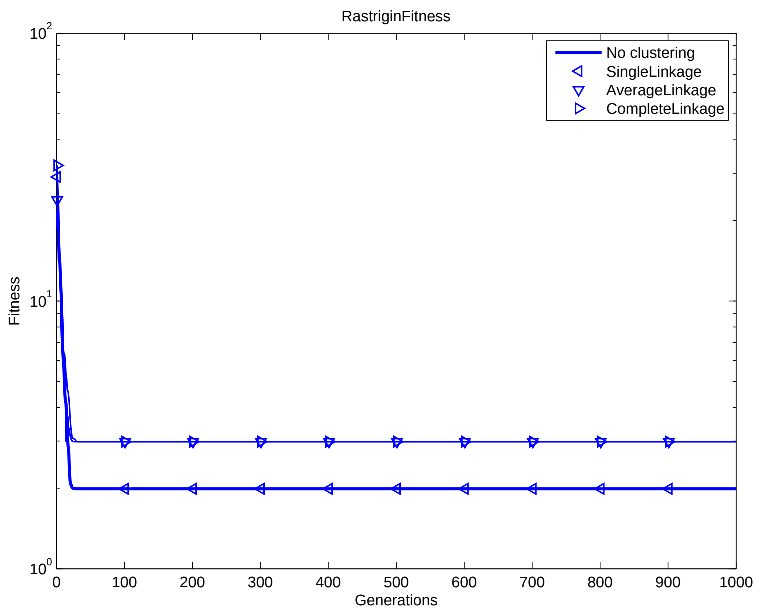


Figure 5.11: The effect of clustering for each linkage type on the Rastrigin function.

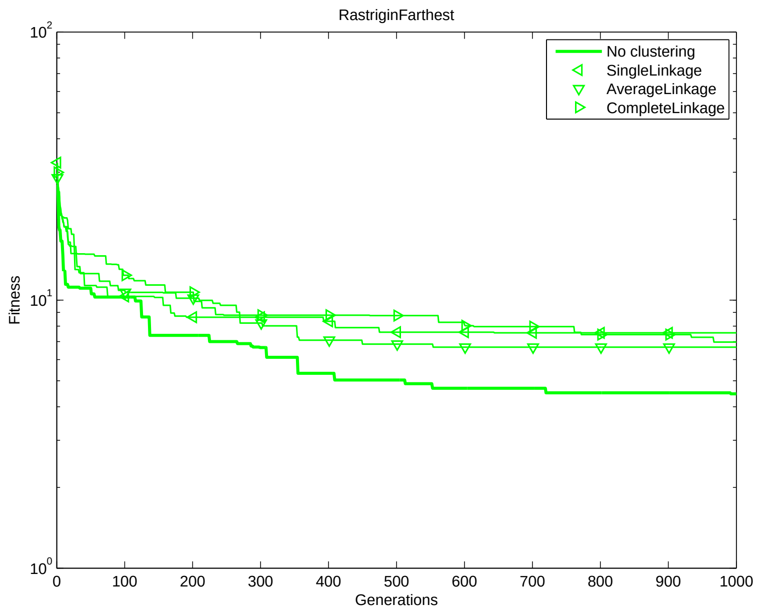


Figure 5.12: The effect of clustering for each linkage type on the Rastrigin function.

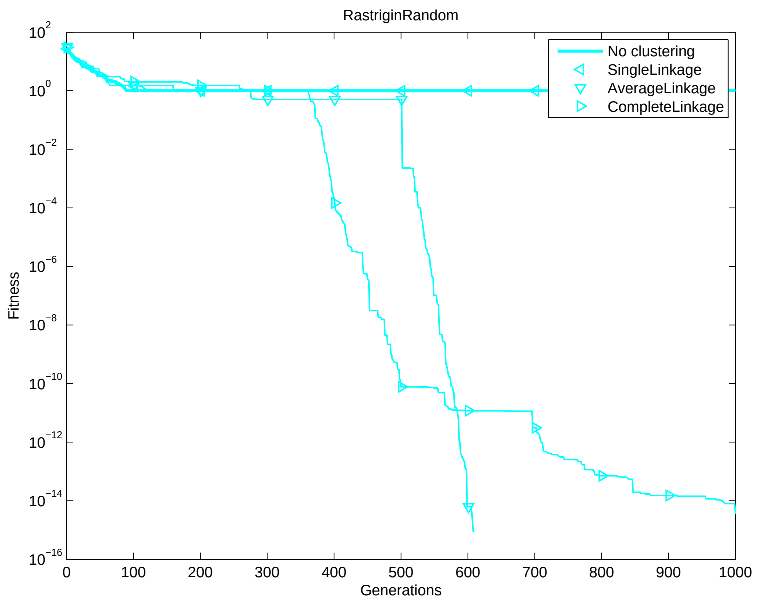


Figure 5.13: The effect of clustering for each linkage type on the Rastrigin function.

On the Rastrigin function, clustering has different effect on each selection. While on the fitness selection it doesn't help and on the farthest selection it rather deteriorates, it helps on the random selection fig. 5.13 especially average linkage allow the algorithm to converge.

5.3.3 Successful results

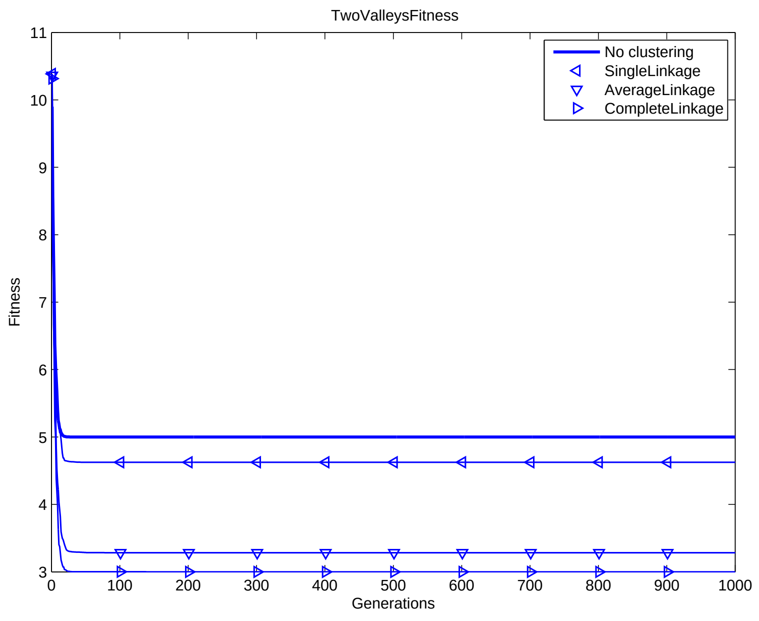


Figure 5.14: The effect of clustering for each linkage type on the Two valleys function.

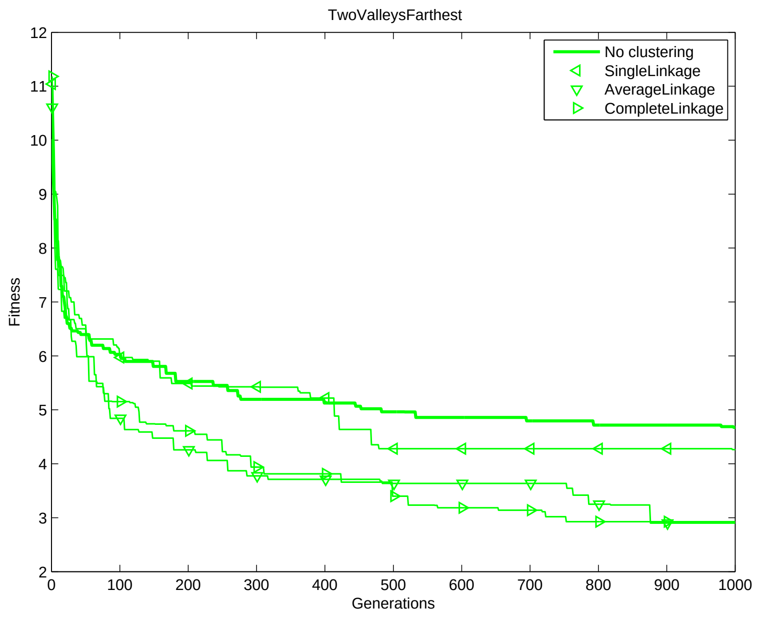


Figure 5.15: The effect of clustering for each linkage type on the Two valleys function.

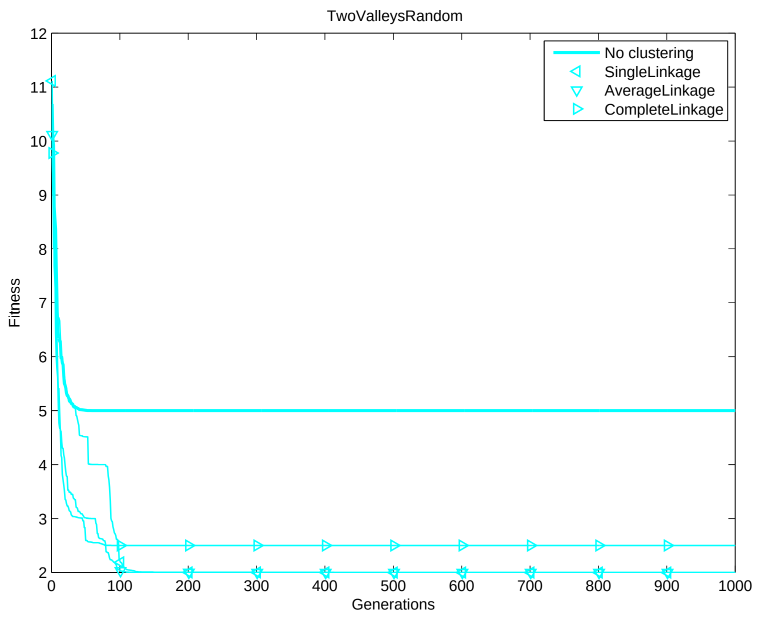


Figure 5.16: The effect of clustering for each linkage type on the Two valleys function.

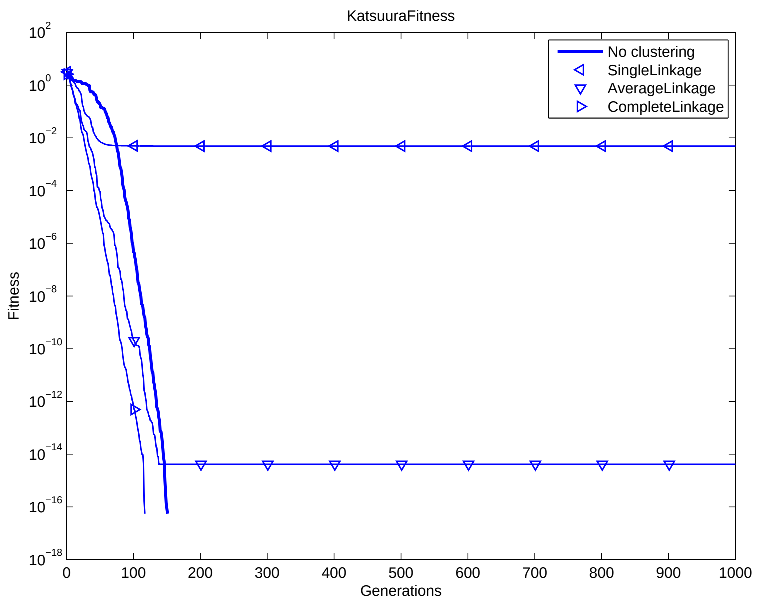


Figure 5.17: The effect of clustering for each linkage type on the Katsuura function.

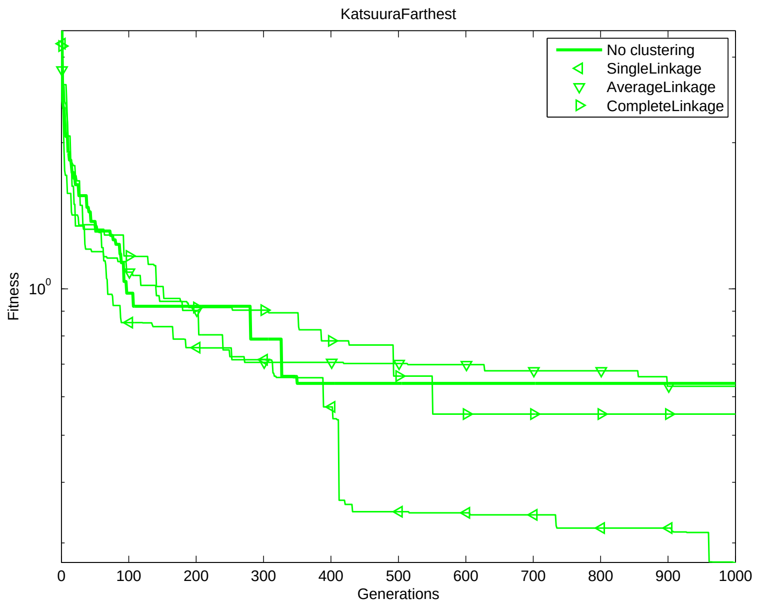


Figure 5.18: The effect of clustering for each linkage type on the Katsuura function.

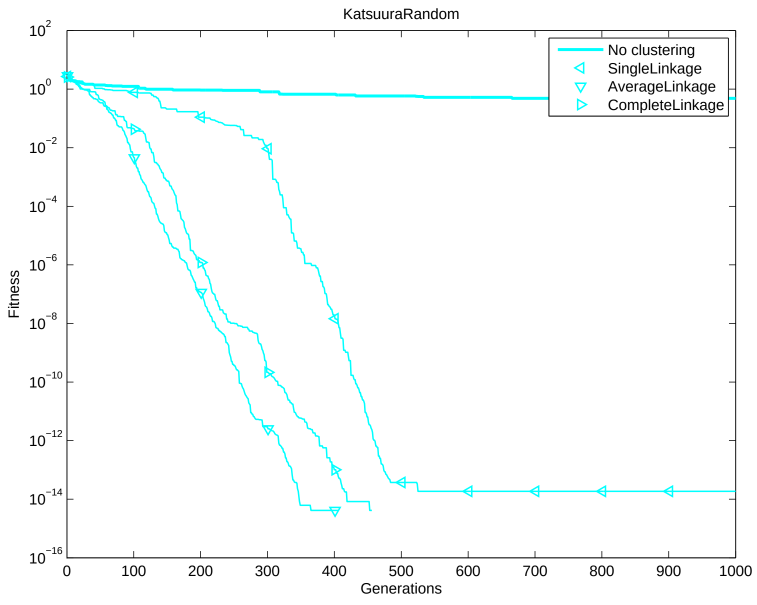


Figure 5.19: The effect of clustering for each linkage type on the Katsuura function.

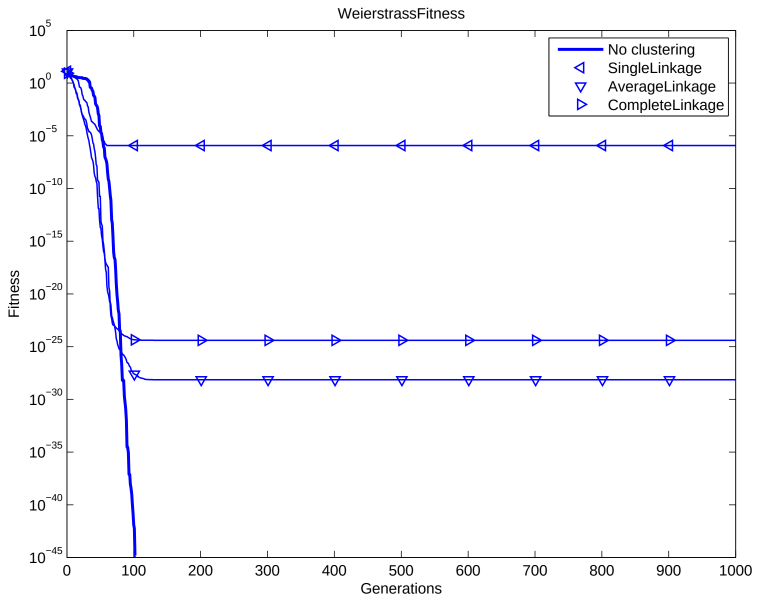


Figure 5.20: The effect of clustering for each linkage type on the Weierstrass function.

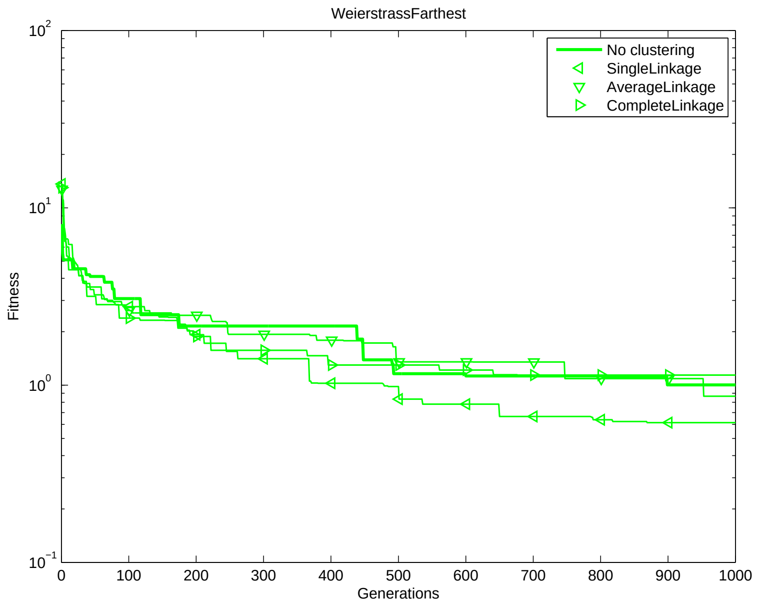


Figure 5.21: The effect of clustering for each linkage type on the Weierstrass function.

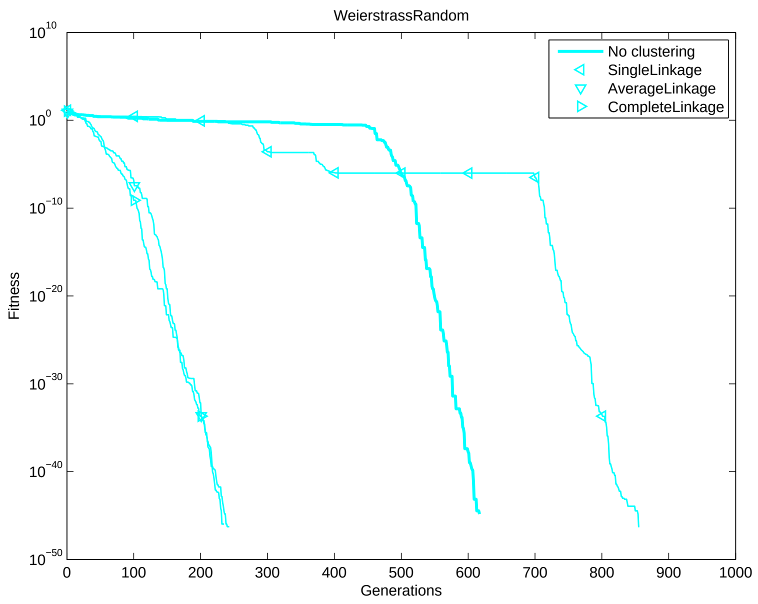


Figure 5.22: The effect of clustering for each linkage type on the Weierstrass function.

On these funtions: TwoValleys, Katsuura, Weierstrass and EggHolder the clustering succeeded. On TwoValleys function, the clustering works great and all the selections are improved. The shape of these functions tells on which functions the clustering may be useful.

5.4 Final comparison

To show all the results and to make any decision, it was necessary to evaluate them by some criterion. As the criterion, it was taken: Successful, Neutral, Unsuccessful. The comparison was made at the end of search process for different types of linkage and the runs was evaluated due to the clustering effect. So, if the run with clustering was better than the run without clustering, it was marked as 1, if the same or simiilar, it was assigned 0 or -1 where it did not work. Somewhere it is hard to decide, in which group it belongs. Although this assignement isn't precise, it summarizes the results. Three dimensional table was created for functions, linkage types and selection methods. In tables below, there are listed the scores for each dimension separately and for combination of all dimensions. It may tell, whether some combination is in general better choice.

Functions	Total score
Ellipsoidal	-9
SharpRidge	-9
Sphere	-7
Rosenbrock	-6
Michalewicz	-6
SchaffersF7	-5
DoubleFunnel	-4
Griewangk	-3
Griewangk2	-1
Rastrigin	-1
Linear	0
Weierstrass	0
EggHolder	1
Katsuura	5
TwoValleys	9

Table 5.1: Total score for test functions in range from -9 to 9.

Fitness selection	Random selection	Farthest selection
-23	-9	-4

Table 5.2: Total score for selections in range from -45 to 45.

Single linkage	Average linkage	Complete linkage
-15	-12	-9

Table 5.3: Total score for linkages in range from -45 to 45.

	Fitness selection	Random selection	Farthest selection
Single linkage	-10	-5	0
Average linkage	-6	-6	-3
Complete linkage	-7	-1	-1

Table 5.4: Score for combination of linkages and selections in range from -15 to 15.

	Fitness selection	Random selection	Farthest selection
Ellipsoidal	-3	-3	-3
SharpRidge	-3	-3	-3
Sphere	-3	-3	-1
Rosenbrock	-3	-3	0
Michalewicz	-1	-2	-3
SchaffersF7	-3	-2	0
DoubleFunnel	-1	0	-3
Griewangk	-3	0	0
Griewangk2	1	-2	0
Rastrigin	0	2	-3
Linear	-1	-2	3
Weierstrass	-3	2	1
EggHolder	-3	1	3
Katsuura	0	3	2
TwoValleys	3	3	3

Table 5.5: Score for combination of functions and selections in range from -3 to 3.

	Single linkage	Average linkage	Complete linkage
Ellipsoidal	-3	-3	-3
SharpRidge	-3	-3	-3
Sphere	-2	-3	-2
Rosenbrock	-2	-2	-2
Michalewicz	-3	-2	-1
SchaffersF7	-2	-2	-1
DoubleFunnel	-2	-1	-1
Griewangk	-1	-1	-1
Griewangk2	1	-1	-1
Rastrigin	-1	0	0
Linear	0	0	0
Weierstrass	0	0	0
EggHolder	-1	1	1
Katsuura	1	1	3
TwoValleys	3	3	3

Table 5.6: Score for combination of functions and linkages in range from -3 to 3.

Results in the tables confirm the assumptions. It appears that Complete linkage has balanced performance which may be related to fact that it creates balanced clusters. behind it there is placed Average linkage which is similar. As it was expected Single linkage improves Farthest selection, but it doesn't work on multimodal problems because this selection leads to creation of a one big cluster. As the best choice it seems to be combination of Complete linkage and Random selection which works great on multimodal problems.

Chapter 6

Conclusion

In general, the results show that clustering depend on given problem. There exist a lot of options which should be taken into account like linkge type, value of alpha, selection method, tournament size and so on. There must be done some simplifications and review the results in compact form. Clustering has better impact on random selection because when a size of a cluster is small, thus the cluster has small number of individuals, the random selection allows next generation to be generated by more than only two parents. Because if the cluster size is the same as a tournament size, in other selections the population is generated only by two parents. This causes the decrease of diversity in the population. Results in the tables are mostly negative. It is because there are used many similar functions, where the clustering doesn't work. There is no one optimal configuration which could be applied on every problem. Clustering had in many cases good performance in the first few iterations, so it could be the subject of further experiments for example stop the clustering after a predefined number of iterations.

On multimodal functions the clustering is quite effective. Somewhere the algorithm finds the solution faster or is able to find better solution. Genetic algorithm with clustering maintains a diversity in population what is essential in solving multimodal problems. For multimodal problems, the clustering may improve a performance and its usage is suitable. Otherwise if there isn't a knowledge about problem, it is better to use classic genetic algorithm. Weak point of clustering is its time consumption so it is important to consider a possible profit. Clustering may be very useful under certain conditions and provides additional space for improvement.

Bibliography

[1] Bäck, T., Fogel, D., Michalewicz, Z., 1997.
Handbook of Evolutionary Computation, Oxford Univ. Press..

[2] Michalewicz Z., Fogel D., 2000.
How to solve it: Modern Heuristics, Springer-Verlag Berlin Heidelberg.

[3] Tomassini M., 2005.
Spatially Structured Evolutionary Algorithms, Springer-Verlag Berlin Heidelberg.

[4] MacQueen, J. B., 1967.
Some Methods for classification and Analysis of Multivariate Observations, University of California Press.

[5] Johnson, S. C., 1967.
Hierarchical Clustering Schemes, Psychometrika.

[6] Finck S., Hanseny N., Ros R., Auger A., 2010.
Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noiseless Functions, Working Paper.

[7] Schwefel H. P., 1995.
Evolution and Optimum Seeking, New York: Wiley & Sons.

[8] Havlín V., 2009.
Bachelor project: Preferenční křížení v evolučních algoritmech, source code included, ČVUT.

Appendix A

Content of attached CD

Attached CD contains these files:

- TomasJuchelka-BP-2010.pdf: Text of bachelor project.
- BP-LaTeX-source: Text of bachelor project in source format.
- Clustering: Java project for NetBeans IDE.
- Matlab-graphs: Scripts for generating graphs in MATLAB.
- Readme.txt: Operating instructions.
- Clustering-source: Project source codes.
- Original bachelor project assignment and Declaration.