AGH

University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

DEPARTMENT OF COMPUTER SCIENCE

MASTER OF SCIENCE THESIS

ADRIAN WOLNY

IMPROVING POPULATION-BASED ALGORITHMS USED IN GLOBAL OPTIMIZATION WITH FITNESS DETERIORATION TECHNIQUES

SUPERVISOR:
Robert Schaefer, Prof., PhD, DSc

Krakow 2011

I would like to thank Prof. Schaefer for his invaluable support and motivation throughout the duration of my work

Contents

1. Introduction 7

1.1. A statement of a problem 8

1.2. Related Work 8

1.2.1. Fitness deterioration techniques 8

1.2.2. Using clustering in fitness deteriorartion 9

1.3. Critical remarks 11

2. Cluster Supported Fitness Deterioration Algorithm 12

2.1. Cluster vs. cluster extension 12

2.2. CSFD algorithm description 13

2.2.1. CSFD pseudo-code 13

2.2.2. Termination conditions 14

3. Clustering 16

3.1. OPTICS 16

4. Fitness Deterioration 21

4.1. Fitness deterioration and clustering 23

4.2. Basic Scheme 24

4.3. Weighted Scheme 26

4.4. Crunching Function Adjustment 26

4.4.1. CFA Results 27

5. Tested Algorithms 29

5.1. Genetic Engine 29

5.2. Test functions 30

5.3. CSFD initial parameters 30

5.4. Efficiency measures and CSFD results 31

6. Implementation 34

6.1. Architecture 34

6.2. Implementation in Java 35

6.2.1. Technologies 35

7. Conclusions 36
7.1. Summary 36
7.2. Future Research 38
A. Installation and Program Results 39
A.1. Installation and Execution Instructions 39
A.2. Program Results 39
B. Spring Configuration 40
B.1. Sample Spring configuration 40

1. Introduction

> It is impossible for any optimization algorithm to outperform random walks on all possible problems.
>
> ... a conclusion from No Free Lunch Theorem

Evolutionary algorithms are known as a generic population-based metaheuristics which often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape; this generality is shown to be a great successes in many real-life problems. Evolutionary algorithms have the tendency to lose diversity within their population of feasible solutions and to converge into a single solution. However, there are domains where the global solution may not suffice, such as a multi-modal optimization tasks. Such problems require finding and maintenance of multiple robust local solutions, i.e. local solutions whose basins of attraction are properly wide and deep.

The most common technique in evolutionary algorithm which is used to achieve this goal is to incorporate some sort of niching method. In [Mah95] and [MM95], Mahfound suggests a classification of niching methods based on the way multiple niches are found in a GA:

- *Parallel Niching methods*: Niching methods belonging to this category finds and maintains multiple niches simultaneously in a single population. Examples of parallel niching methods are *Crowding function* and *Sharing*

- *Sequential Niching methods*: These niching methods find multiple niches iteratively

Sequential niching is one of the most promising evolutionary strategies for analyzing multi-modal global optimization problems in the continuous domains embedded in the vector metric spaces.

This work presents a new hybrid approach to sequential niching strategies called *Cluster Supported Fitness Deterioration* (CSFD). In each iteration CSFD performs the clustering of the random sample by OPTICS algorithm and then the deterioration of the fitness on areas occupied by clusters. The selection pressure pushes away the next-step sample (population) from the basins of attraction of minimizers already recognized, speeding up finding of new ones. The main advantages of CSFD are low memory an computational complexity even in case of large dimensional problems and high accuracy of deterioration obtained by the flexible cluster definition delivered by OPTICS.

1.1. A statement of a problem

We deal with the class of *global optimization problems* (GOP) which are leading to find all global, or even all local minimizers to the real-valued objective function defined on the set $\mathcal{D}$ (called admissible set) embedded in a finite dimensional normed vector space $V \supset \mathcal{D}$, $dim(V) = N < +\infty$, where the norm induces the complete metric (Banach space). The set $\mathcal{D}$ is assumed to be continuous with respect to the metric and regular in some way, usually having the Lipshitz boundary.

Typical difficulties appearing by solving GOPs are the huge volume of $\mathcal{D}$, multi-modality of the objective and its weak regularity (sometimes only continuity, or even discontinuity on the subset of the zero measure).

Stochastic, population-based heuristics (Evolutionary Algorithms, Simulated Annealing, Ant Colony, Particle Swarm, etc.) are best suited to solve GOPs and they frequently outperform deterministic techniques (see e.g. [PR02]). Because the remaininder of this paper utilizes only genetic techniques we will denote by $F : \mathcal{D}\,(\text{or}\,\mathcal{D}_r, U) \to \mathbb{R}$ the generic *fitness function* that expresses the GOP objective.

One of the well known disadvantages of some population-based, stochastic heuristics (especially Genetic Algorithms) is the *premature convergence* which consists in long-term stay of almost all individuals in the basin of attraction of the single local minimum. Premature convergence dramatically decreases an ability of finding many local/global minima with the acceptable computational cost assumed as the number of objective evaluations.

The basin of attraction $\mathcal{B}_{x^+} \subset \mathcal{D}$ of the local or global minimizer $x^+ \in \mathcal{D}$ may be roughly described as the connected part of maximum fitness level set that contains x^+ and does not intersect with basins of attraction of other local minimizers. The precise mathematical definition of basin of attraction was introduced by [KT87], [DS75] and may be found also in [Sch07].

As we mentioned before the *Cluster Supported Fitness Deterioration* algorithm is an extension of *sequential niching* technique which works by forcing the single population or the group of populations to move from the basins of attraction of minima that have been recognized during the course of the algorithm.

1.2. Related Work

1.2.1. Fitness deterioration techniques

The behavior of individuals suitable for niching may be obtained by the proper fitness modification that leads to its leveling on the central part of the basin of attraction of local minima already encountered. Selection removes individuals from such areas forcing them to find regions of larger fitness e.g. the basins of attraction of other minima not yet found. The fitness leveling mentioned above is sometimes called *fitness deterioration* [Obu97b] or *hill crunching* [SAT04].

Sequential niching by fitness deterioration was introduced by [BBM93]. Exhaustive research in this direction was performed by Obuchowicz, Patan and Korbicz. They introduced the class of evolutionary algorithms called *Evolutionary Search with Soft Selection - Deterioration of the Objective Function*

(ESSS-DOF). The draft of this strategy is presented in Algorithm 1. The fitness modification performed

Algorithm 1 Draft of the ESSS-DOF strategy

1: Create initial population P_0;
2: $t \leftarrow 0$;
3: **repeat**
4: Evaluate population P_t;
5: Distinguish the best fit individual x^* from P_t;
6: **if** $(trap\_test)$ **then**
7: Memorize x^*;
8: $F \leftarrow F - \hat{F}$;
9: **end if**
10: Perform selection with the fitness F;
11: Perform genetic operations;
12: $t \leftarrow t + 1$;
13: **until** $(stop\_condition)$

in the line 8 of Algorithm 1 is based on the formula

$$\hat{F}(x) = F_{min} \exp\left(-\sum_{i=1}^{N}\left(\frac{x_i - \bar{y}_i}{\sigma_i}\right)^2\right) \tag{1.1}$$

where $\bar{y}_i$ are coordinates of the expected centroid of phenotypes that correspond to the population P_t and σ_i^2 stands for the variance of the individuals location in the i^{th} direction, while $F_{min} = F(x^*)$ is the maximum individual fitness that appears in the population P_t. Finally N is the dimension of the admissible domain $\mathcal{D}$.

The logical variable $trap\_test$ is true if the mean fitness in the population increases less than $p\%$ during the last n_{trap} genetic epochs, or the standard deviation of the phenotypes displacement is less than the mutation range during the last n_{trap} genetic epochs. The logical variable $stop\_condition$ is true if the proper stopping rule for the whole strategy is satisfied. The simplest possible stopping rule may be limit the number of genetic epochs after the last fitness modification during which no trap was found.

Test results of this effective approach to the global search were presented in [Obu97a], [OP97] and [OK99]. [Ara01] delivered another formula for fitness modification that leads to population niching.

1.2.2. Using clustering in fitness deteriorartion

Telega, Schaefer and Adamska (see [Tel99], [SAT04]) introduced the clustering techniques applied to the random sample (population) obtained by the genetic algorithm (GA) in order to make the fitness deterioration more accurate and effective. The resulted strategy was called *Clustered Genetic Search* (CGS).

The basic idea of CGS is to recognize the clusters of individuals contained by multiset of individuals being the current population or the sum of current populations obtained from the multi-deme model

or being the cumulated population (e.g. the sum of all populations from the prescribed number of last genetic epochs). Clusters should be sufficiently dense and well separated one from each other.

Next the *cluster extensions* being the regular subsets of $\mathcal{D}$ with the positive measure containing the cluster points are constructed. Pseudocode of the proposed strategy is depicted in the Algorithm 2.

Algorithm 2 Draft of the CGS strategy

1: $CLE \leftarrow \emptyset$;
2: Create initial population P_0;
3: **repeat**
4: Evaluate fitness F outside CLE;
5: Modify fitness according to (1.2);
6: Perform GA until the local stooping criterion is satisfied;
7: Recognize new clusters;
8: Construct new cluster extensions and update CLE;
9: **until** (The whole domain $\mathcal{D}$ has been processed) **or** (A satisfactory set of cluster extension has been found)

The CGS strategy utilizes the following fitness modification

$$\hat{F}(x) = \begin{cases} F(x) & \text{if} \quad x \in \mathcal{D} \setminus CLE \\ F_{max} & \text{if} \quad x \in CLE \end{cases} \tag{1.2}$$

where F_{max} is the maximum fitness value already encountered and $CLE \subset \mathcal{D}$ stands for the union of cluster extensions already recognized.

The admissible domain $\mathcal{D}$ is divided into hypercubes that constitute a grid (raster) and the cluster extensions are unions of hypercubes. Every cluster extension can be recognized in one or many steps of the main loop. After the GA is stopped (line 6 in Algorithm 2), new parts of cluster extensions can be detected by the analysis of the density of individuals in the hypercubes. The hypercube that contains the best individual is selected as the seed. Neighboring hypercubes with the density of individuals greater than an arbitrary threshold are attached to the cluster extension. A rough local optimization method is started in each new part of the cluster extension, and the result of this optimization is retained. If this local method ends in the already recognized cluster extension then this part is attached to it.

The local stopping criterion distinguishes two kinds of behavior of the GA utilized by CGS. The first one is that it finds new parts of cluster extensions after few generations, and the second is the chaotic behavior (individuals are uniformly distributed over $\mathcal{D} \setminus CLE$). The latter corresponds to the recognition of a plateau (or areas where the fitness has small variability) outside of the already known cluster extensions. Other cases are treated as the situation when GA does not fit to the particular problem, and a refinement of SGA parameters is suggested.

The CGS stopping strategy is to check stagnation of a sequence of some estimator of distribution of population individuals. If it does not change, then check if an arbitrary number of hypercubes has the density of individuals below the threshold. If so, then begin the clustering procedure; otherwise, check if individuals are uniformly distributed.

Computations show that CGS constitutes a "filter" that eliminates local minima with small fitness variability and narrow basins of attraction. Such property can be useful in some cases. The CGS strategy should be especially convenient for functions with large areas of small variability (areas similar to plateaus) which can be difficult for other global and local optimization methods.

The Simple Genetic Algorithm (SGA) (see e.g. [Sch07]) was used in the CGS instance described above. Another implementation utilized the multi-deme Hierarchic Genetic Search (see [SK03]) and the FMM-EMM method for finding cluster extensions (see [SA04]).

The CGS works well if the cluster extensions fall into the basins of attraction of local and global minimizers and fill them sufficiently. The CGS correctness and the correctness of the proposed CGS stopping strategy can be partially verified for the case of SGA sampling (see [Tel99], [SJ01], [Sch07]).

Summing up, CGS may be classified as the sequential niching, even if the parallel HGS is applied as the sampling engine and if more than one cluster extension is encountered in its single step.

1.3. Critical remarks

Deep analysis of the deterioration techniques presented in Sections 1.2.1, 1.2.2 exhibits their several serious disadvantages:

– Huge memory complexity of memorizing the cluster extensions which appears especially in case of CGS with a raster representation of cluster extensions and a large dimension N of the admissible domain $\mathcal{D}$. In the computational practice, the GOP with N greater then 10 may bring the memory problems.

– Unsatisfactory accuracy of fitness approximation on the area of cluster extensions that leads to the incorrect deterioration and finally may lead to removing individuals from unchecked areas or the multiple check of non-promising areas already browsed.

– For the case of both strategies described in Sections 1.2.1, 1.2.2 the error has a different origin. In the case of ESS-DOF the approximation of fitness in area surrounding local minimum (see formula (1.1)) is very rough and might be unsatisfactory in case of elongated basins of attraction. No matter how CGS finds the area of fitness leveling quite good as the cluster extensions, the fitness modification by the general constant (see formula (1.2)) may result in artifacts (artificial minima) at the borders of cluster extensions.

– The strong dependency of deterioration technique from the evolutionary technique used which can be specially observed in ESS-DOF. This feature generally disables to use deterioration in the transparent way in case of complex, adaptive strategies with many genetic engines. Sometimes this dependency might be helpful by profiling deterioration according to the particular GA instance.

The above discussion clearly shows the way of necessary improvements. A deterioration technique that combines low memory complexity of the exponential fitness improvement $\hat{F}$ with the accuracy of clustering will be presented in following Sections.

2. Cluster Supported Fitness Deterioration Algorithm

We suggest another approach to sequential niching which exploits some of the ideas from [Obu97b], [SAT04], [SA04] and which tries to overcome problems specified in Section 1.3. This strategy is called *Cluster Supported Fitness Deterioration* (CSFD).

2.1. Cluster vs. cluster extension

Before we dive into the detailed description of the CSFD algorithm let us make a clear distinction between two important notions of the *cluster extension* and the *cluster* itself, which are necessary for further research.

Definition 1. *We will call by cluster C a selected subset of the population P (the mulitset of individuals) returned by GA. We assume that each cluster will be disjoint with any other cluster from P. (i.e. when C_i, C_k are clusters, then $i \neq k \Longrightarrow C_i \cap C_k = \emptyset$).*

The individuals are assigned to the clusters using the method described in Chapter 3. Generally, individuals that belong to a particular cluster C have to be densely distributed and well separated from individuals from other clusters.

Definition 2. *The extension CE of the cluster C is the connected and regular subset of the admissible domain $\mathcal{D}$ with the positive measure, containing the cluster points, i.e., each element from C belongs to CE. Regular set means here the set with the Lipshitz boundary (see e.g. [Zei85]).*

Cluster extensions may be obtained in many different ways. One of them is the raster procedure introduced by [Tel99] described in Section 1.2.2.

Here we will use the cluster extensions CE being the ellypsoid whose middlepoint is located at the cluster C centroid and the measures of diversity of individuals inside C (e.g. in the form of standard deviation) in order to determine the length and the direction of its axes (see Chapter 4).

The CSFD strategy runs the GA and then distinguishes clusters from the resulting population and constructs their extensions in each iteration. The information contained in each cluster allows for effective finding the good approximation of the local minimizer x^+ contained in its extension (e.g. by running the accurate local method starting from the best fitted individual in the cluster). Another advantage is the ability to approximate the shape and the volume of basins of attraction $\mathcal{B}_{x^+}$ by cluster extensions. This

information is utilized by the fitness deterioration technique (see Section **??**). It might be also helpful for the stability analysis of x^+.

It may happen that cluster extensions obtained from two or more clusters lie inside the same basin of attraction. This may happen when in one iteration we manage to deteriorate only a part of the basin of attraction and in the next iteration we obtain the cluster inside other part of the basin. We will use the procedure suggested by [Tel99] in order to detect such situation starting a single local, cheep method from each new cluster extension. If the method converges inside one of existing cluster extensions CE, then CE and CE' are joined (which indicates the fact that both CE' and CE lie in the same basin).

2.2. CSFD algorithm description

In principle the algorithm works like the CGS strategy presented in [SAT04]. It uses GA to obtain a random sample (population of individuals) from the domain $\mathcal{D}$. If the problem space contains some robust solution located inside broad basins of attraction, it is very likely that individuals returned from the GA are gathered inside one or more of such basins, so the next step of the algorithm is to apply the clustering algorithm to distinguish groups of individuals (clusters) from the population. Under the assumption that distribution of individuals inside a given cluster provides information about the topology of the basin in which the cluster resides, the algorithm approximate the basin using this information in order to deteriorate the fitness over the basins area and thus to prevent convergence to the same solutions multiple times.

2.2.1. CSFD pseudo-code

The CSFD strategy takes a hybrid approach which uses an arbitrary GA to obtain a random sample from which it tries to extract as much information as possible in order to to find approximations of basins of attraction. The only need for the GA is to be well tuned to the GOP to be solved, i.e., the population has to concentrate in a basin of attraction of at least one local robust minimizer (see [Sch07]). Then the fitness deterioration is applied in localized areas to prevent exploration of the same basins multiple times during the course of the search. Algorithm 3 shows the general idea behind the Cluster Supported Fitness Deterioration. The algorithm components will be described in a top-down manner in next Sections, here we provide the general overview only.

The condition in "while" statement (line 2, Algorithm 3) should be treated as a control statement rather than the real termination criterion. The CSFD stopping criterion is based on the conditions checked inside the main loop (line 6, Algorithm 3). If the clustering algorithm has found no group of similar individuals or the deterioration has a low quality, CSFD is stopped and returns all clusters found.

In each iteration we execute the GA (line 3, Algorithm 3) which is treated as a black-box algorithm, i.e. we may do not need to modify or know the implementation of the GA used. Then we take the population returned by the GA and extract so called *clustering structure* (described with details in Section 3.1) which contains the information about clusters and their internal mean densities. Note that so far we have not clustered the population returned by the GA, instead we extract the information about the

Algorithm 3 Draft of the CSFD strategy

1: $CL \leftarrow \emptyset$
2: **while** $i < maxGenerationNumber$ **do**
3: $execute(GA)$
4: $pop \leftarrow getPopulation(GA)$
5: $clusterStruct \leftarrow extractClusterStruct(pop)$
6: **if** $noClusters(clusterStruct)$ **then**
7: **return**
8: **end if**
9: $detFitness \leftarrow$
$performFitnessDet(clusterStruct, currentFitness)$
10: **if** $quality(detFitness, currentFitness, pop) < th$ **then**
11: **return**
12: **end if**
13: $CL \leftarrow CL \cup clusters$
14: $updateFitness(detFitness)$
15: $popSize = popSize + indNum$
16: **end while**

internal clustering structure for further processing. If the clustering structure contains promising clusters then we perform the *fitness deterioration* – the process of degradation of the fitness landscape in areas occupied by clusters of individuals which are assumed to agglomerate inside basins of attraction. In the line 9, Algorithm 3 we execute the complex procedure *performFitnessDet* (described in details in Section **??**) which actually performs the *clustering* and *fitness deterioration*. Next in the line 10, Algorithm 3 we check if the new fitness (returned by the procedure *performFitnessDet*) fulfills the quality requirements described in Section **??**. Finally, we save the clusters returned by the deterioration process and update the fitness function for further iterations (lines 13, 14, Algorithm 3). Exploratory capabilities can be increased during later iterations by increasing the population size of the GA (line 15, Algorithm 3). The CSFD strongly depends on the clustering algorithm used as a way to find parts of basins of attraction and as a global termination criterion for the CSFD algorithm (see Section 2.2.2).

2.2.2. Termination conditions

Two types of stopping and termination criterions are used by CSFD:

– *Local termination criterion* which is used as a stoppng criterion for the GA inside the main loop of the CSFD.

– *Global stopping criterion* which is based on the result of the clustering algorithm applied to the population returned by the GA and which finishes the whole CSFD stategy.

The termination criterion in classic evolutionary algorithms is hard to define and very often problem dependent, as we do not have any global information about the fitness landscape and therefore we can only compare one solution to another previously found. For a *local termination criterion* we may use some of the standard well-known stopping criteria for evolutionary algorithms like:

– *Expected first hitting time (FHT)* (see e.g. [PR02]) which tries to set meaningful upper bounds for the number of iterations required to reach the sufficiently small neighborhood of solution.

– *Efficiency measures* (see e.g. [Gol89]) e.g. *Running Mean*, the difference between the current best objective value found and the average of the best objective values of the last t generations is equal or less than a given threshold ϵ.

In terms of the *global stopping criterion* we focus on the distribution of individuals in the admissible domain $\mathcal{D}$. We follow the idea introduced by [Tel99] and proved by [SA04] to be successful for multi-modal problems with robust solutions.

The global stopping criterion for CSFD algorithm is based on the clustering analysis and defined as follows:

The CSFD algorithm is being terminated if the clustering process applied to the population of individuals returned by the genetic algorithm returns no clusters or the quality measure of the fitness deterioration performed on found clusters is too low.

The clustering, which is performed in every iteration of the CSFD algorithm, gives us some clues about the global characteristics of the fitness landscape i.e. when the clustering algorithm performed on the final population finds nothing it is very likely that in previous iterations we have deteriorated the fitness landscape in places where the most desirable solutions reside and there is no use in continuing the searching process. This is considered to be true because we are looking for robust solutions which are resistant to noise and lie in basins of attraction which are significantly wide and deep. The population of GA is likely to converge to such solutions, so having found no clusters of individuals after performing the sufficient number of GA epochs shows that the population would not converge to any robust solution.

Such kind of condition may be precisely formulated in terms of the convergence of sampling measures and partially verified in case if the genetic engine is SGA with the focusing heuristic (see [Tel99], [Sch07], [SJ01]).

3. Clustering

Clustering algorithms divide a dataset into several disjoint subsets. All elements in such a subset share common features like, for example, spatial proximity. Clustering is used as a stand-alone tool to get insight into the distribution of a data set or as a preprocessing step for other algorithms operating on the detected clusters. The former is used to determine stop criteria as described in Section 2.2.2 and the latter is used in our fitness deterioration algorithm to improve its accuracy.

A cluster extension (see Definition 2) may be seen as an approximation of the basin of attraction, moreover the distribution of individuals which were flooded to the basins provides additional useful information about its shape. The clustering algorithm may be used to detect the set of individuals which belongs to the same basin of attraction. The CSFD provides the information about detected sets (basins of attraction) in the form of a proper representation of cluster extensions which are just a convenient way to describe basins of attraction (e.g the center point, the radius of the set, covariance matrix etc.).

3.1. OPTICS

We have choosen density-base clustering algorihtm called *OPTICS: Ordering Points To Identify the Clustering Structure* [ABKS99]. Density-base clustering generally needs the data which is the subset or the multiset of objects (points) which belong to a finite dimensional vector metric space V. Clusters are regarded as subsets in which the objects are dense and which are separated by regions of low object density. These subsets may have an arbitrary shape and the points inside a region may be arbitrarily distributed.

OPTICS is an extendsion to a well-known density clustering algorithm called *DBSCAN* [EKSX96]. The basic idea for *DBSCAN* is that for each point of a cluster the neighborhood of a given radius ϵ has to contain at least a minimum number of points $minPts$. Figure 3.2 shows the result of *OPTICS* clustering for a sample set of points.

In particular each object q of the cluster C has to be surrounded by the neighborhood $N_\epsilon(q) \subset V$ of a given radius ϵ that contains at least $minPts$ other objects. The formal definition for this notion is as follows:

Definition 3. *An object $p \in P$ is directly density-reachable from an object $q \in P$ with respect to the parameters $\epsilon \in \mathbb{R}_+$, $minPts \in \mathbb{N}$ in a set or multiset of data P if:*

– $p \in N_\epsilon(q)$,

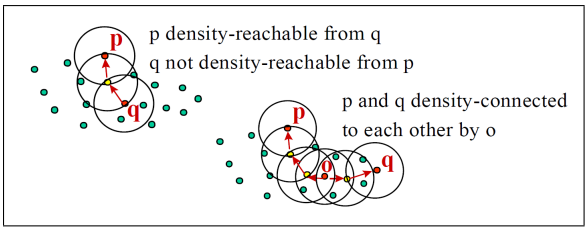

Figure 3.1: Density-reachability and connectivity

– $Card(N_\epsilon(q) \cap P) \geq minPts$.

The condition $Card(N_\epsilon(q) \cap P) \geq minPts$ is called the *core object condition*. If this condition holds for an object q, then we call q a *core object*. Only from core objects, other objects can be directly density-reachable.

Definition 4. *An object p is density-reachable from an object q with respect to the parameters $\epsilon \in \mathbb{R}_+$, in the set or multiset of objects P if there is a chain of objects $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that $p_i \in P$ and p_{i+1} is directly density-reachable from p_i for $i = 1, \ldots, n-1$ with respect to ϵ and $minPts$.*

The density-reachability relation is not symmetric in general in P. Only core objects can be mutually density-reachable.

Definition 5. *An object p is density-connected to an object q with respect to ϵ and $minPts$ in the set or multiset of objects P if there is an object $o \in P$ such that both p and q are density-reachable from o with respect to ϵ and $minPts$ in P.*

Both phenomena are illustrated by Figure 3.1. A density-based *cluster* is now defined as a multiset of density-connected objects which is maximal with respect to density-reachability and the *noise* is the set of objects not contained in any cluster.

Definition 6. *Let P be a set or multiset of objects. A cluster C with respect to ϵ and $minPts$ in P is a non-empty subset of P satisfying the following conditions:*

– *Maximality: $\forall p, q \in P$: if $p \in C$ and q is density-reachable from p wrt. ϵ and $minPts$, then also $q \in C$.*

– *Connectivity: $\forall p, q \in C$: p is density-connected to q wrt. ϵ and $minPts$ in P.*

Every object not contained in any cluster is noise.

OPTICS works like *DBSCAN* but for an infinite number of distance paramters ϵ_i which are smaller than a *generating distance* ϵ. The only difference is that we do not assing cluster memberships. Instead, we store the *order* in which the objects are processed (the main priniple is that we always have to select an object which is density-reachable with respect to the lowest ϵ value to guarantee that clusters with higher density are finished first) and the information which would be used by *DBSCAN* algorithm to

assing cluster memberships. This information consists of only two values for each object: *core-distance* and *reachability-distance*.

Definition 7. *Let $p \in P$, let ϵ be a distance value, let $N_\epsilon(p)$ be the ϵ-neighborhood of p, let $minPts$ be a natural number and let $minPts - distance(p)$ be the distance from p to its $minPts$ neighbor. Then, the core-distance of p is defined as:*

$$\textit{core-distance}_{\epsilon,minPts}(p) = \begin{cases} \textit{UNDEFINED, if } Card(N_\epsilon(p)) < minPts \\ \textit{minPts-distance(p), otherwise} \end{cases}$$

Definition 8. *Let $p, o \in P$ let $N_\epsilon(o)$ be the ϵ-neighborhood of o, and let $minPts$ be a natural number. Then, the reachability-distance of p with respect to o is defined as:*

$$\textit{reachability-distance}_{\epsilon,minPts}(p, o) = \begin{cases} \textit{UNDEFINED, if } |N_\epsilon(o)| < minPts \\ \textit{max(core-distance(o), distance(o,p)), otherwise} \end{cases}$$

An advantage of cluster-ordering a data set compared to other clustering methods is that the ordering (which might be visualized by *reachability-plot* of ordered points) is rather insensitive to the input parameters of the method i.e. the *generating distance* ϵ and the value for $minPts$. Roughly speaking, the values have just to be *large* enough to yield a good result. The concrete values are not crucial because there is a broad range of possible values for which we always can see the clustering structure of a data set when looking at the corresponding *reachability-plot*. Figure 3.3 shows reachablity plot for various *generating distances* (ϵ) which might be seen as an empirical verification of the last argument.

The next chapter describes how *OPTICS ordering* properties are used to prevent degradation of areas which have not been explored during the course of the algorithm.

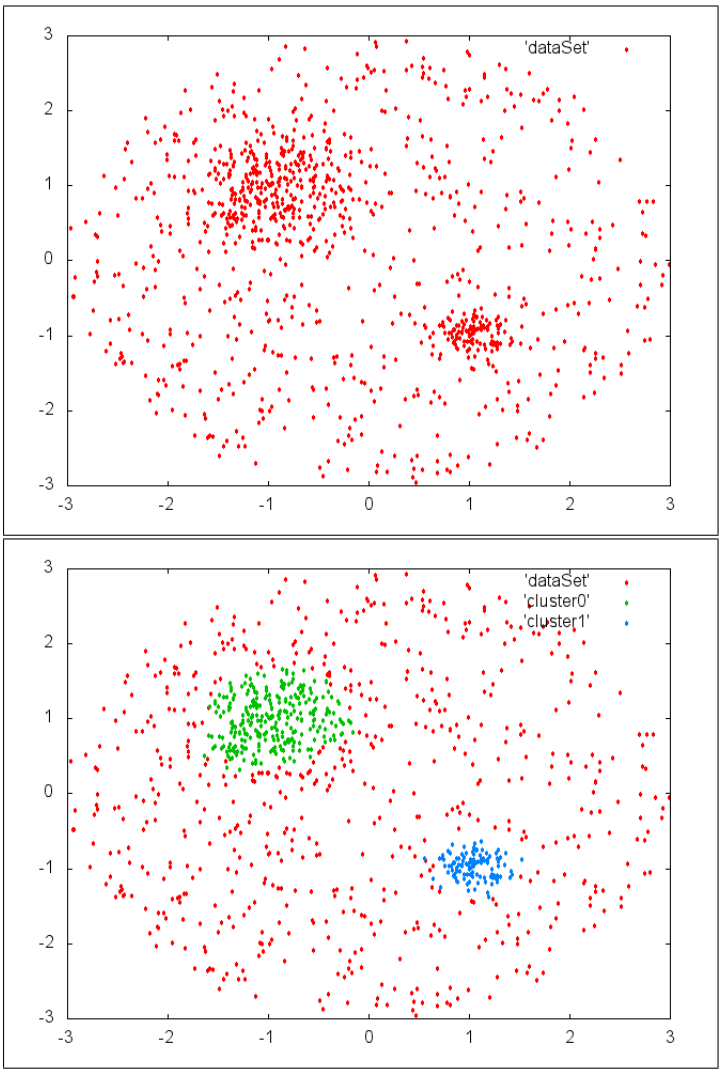

Figure 3.2: Visualization of the DBSCAN algorithm applied to OPTICS ordering of simple 2-dimensional data set which consists of 1000 points. OPTICS paramters: $minPts = 20, \epsilon = 1.2$, the two clusters was found using DBSCAN paramters: $\epsilon' = 0.2$ (see [ABKS99] to fully understand the meaning of the parameters)

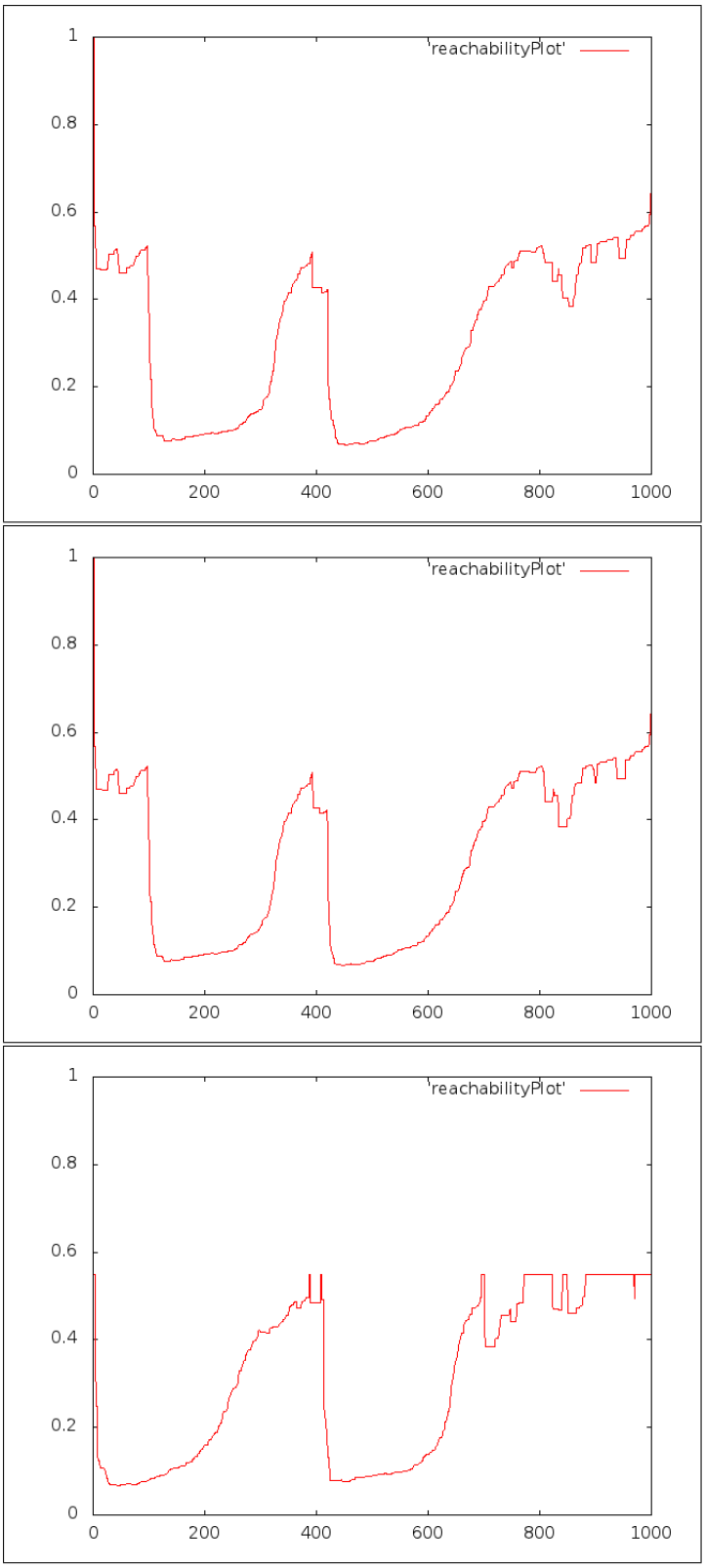

Figure 3.3: Reachability plot for data set presented on figure 3.1. Optics paramters (minPts, ϵ) are: (20, 1.5), (20, 1.0), (20, 0.5) respectively. The two cavities which are visible in each plot depict the two of the clusters on figure 3.1. This proves that there is a large range of values for ϵ (*generating distance*) for which the appearance of the reachability plot will not change significantly. (the flat shape of function from the last plot results from the fact that we truncate the reachability-distance to the generating distance ϵ, when the former is greater than the latter)

4. Fitness Deterioration

This chapter contains detailed description of the deterioration algorithm which is the central point of the *Cluster Supported Fitness Deterioration* algorithm described in chapter 2. Before diving into the details, here is the formal definition of the fitness deterioration process: *Fitness Deterioration* is a process of degrading the fitness function in areas occupied by groups of individuals obtained from clustering (see Section 1.2.1 and references inside). We suggest to achieve this goal in the CSFD strategy by creating a linear combination of the current fitness function and *crunching functions* which approximate the fitness in subsets of problem domain occupied by clusters. Let F_k be the fitness function in k-th iteration of CSFD (i.e. in the k-th execution of the main loop of the Algorithm 3) and $C_1, \ldots, C_{M_k}$ be M_k clusters found in k-th iteration of our sequential niching algorithm described in Section 2.2. For each cluster C_i we create the *crunching function* g_i and then we construct deteriorated fitness F_{k+1} (which will be used in the $(k+1)$-th iteration) as follows:

$$F_{k+1} = F_k + \sum_{i=1}^{M_k} \alpha_i g_i, \tag{4.1}$$

where the selection of the functional coefficients $\alpha_1, \ldots, \alpha_{M_k};\ \alpha_i : \mathcal{D} \to \mathbb{R}_+,\ i = 1, \ldots, M_k$ depends on the type of deterioration used and will be described later in Sections 4.2, 4.3 (see formulas (4.6), (4.7), (4.8)).

The *Crunching function* $g_i : \mathcal{D} \to \mathbb{R}_+$ is constructed for each newly recognized cluster of individuals $C_i \subset P$. Based on the assumption that clusters lie inside basins of attraction and that the distribution of individuals inside the cluster is a good approximation of the shape of the basin occupied by the cluster, the deterioration algorithm tries to exploit information provided by the clustering algorithm and based on that information it augments the fitness function in order to minimize the probability of finding already explored basins of attraction in further iterations.

We may ask ourselves why we do not prevent exploration of basins we found in previous iterations simply by remembering the regions occupied by clusters and ignoring individuals which fall in this regions. The answer is probably the most important reason why we have chosen fitness deterioration for this task. We can not prevent individuals to explore neighborhood of solutions found in previous iterations because such approach would cause our meta-heuristic to loose *completeness*.

Definition 9. *A set Op of search operations searchOp is complete if and only if every point g_1 in the search space G can be reached from every other point $g_2 \in \mathbb{G}$ by applying only operations $searchOp \in$*

Op.

$$\forall g_1, g_2 \in \mathbb{G} \Rightarrow \exists k \in \mathbb{N} : P(g_1 = Op^k(g_2)) > 0 \tag{4.2}$$

If the set of search operations is not complete, there are points in the search space which cannot be reached. Then, we are probably not able to explore the problem space adequately and possibly will not find satisfyingly good solution. That is why it is better to use fitness deterioration as a way to discourage rather then prevent individuals from sinking to the same basins of attraction twice.

Here we would like to emphasize the fact that the deterioration process does not try to accurately interpolate the fitness function in the neighborhood of the solution because it would be very expensive in a high dimensional spaces. Instead it tries to find simple crunching functions which would sufficiently degrade the fitness landscape in the areas occupied by the clusters and then remove the individuals from these regions in the next CSFD steps with the sufficiently high (but even less then 1) probability.

We now can define what are the characteristics of a good deterioration algorithm:

– *It has to be chep* in terms of space and time. We are looking for methods which can be successfully applied for solving high dimensional functions' optimization. After every iteration the resulting fitness function is the linear combination of crunching functions, therefor we must ensure that computation of a single crunching function is $O(1)$ and does not depend no the number of individuals inside a cluster or the number of clusters. Resulting fitness must be $O(k)$ where k is the number of solutions (clusters) found in previous iterations.

– *It has to discourage the population of EA from visiting the neighborhood of solutions found in previous iterations twice*. We do not want these regions to be inaccessible as this would break the **completeness** of our algorithm, we just want an individual to be given a low fitness when it finds itself in these regions.

– *It has to be easy to extend for subsequent solutions*. The way we construct fitness function in subsequent iterations already fulfills this requirement. To extend the deterioration for subsequent solutions we simply add new crunching functions multiplied by the scaling factors we define later in this chapter.

Definition 10. *Premature Convergence [PR02] An optimization process has prematurely converged to a local optimum if it is no longer able to explore other parts of the search space than the area currently being examined and there exists another region that contains a superior solution.*

The premature convergence is not a problem in our algorithm, because once the population finds itself in the local optima trap it is very likely that the population will be avoiding this region in the next iterations. Actually we encourage an optimization process performed by the used EA (e.g. SGA) to converge quickly to local solution in order to find new solutions faster in the next iterations.

An ideal GA engine to be used in our hybrid algorithm defined in Section 2.2 would be the one which produces many small populations in problem space and each population would use strong selection pressure with small mutation rate with high recombination rate to increase exploitation. This is why Hierarchical Genetical Search (HGS) [WSKS03] algorithm would be perfect for our needs as it possess

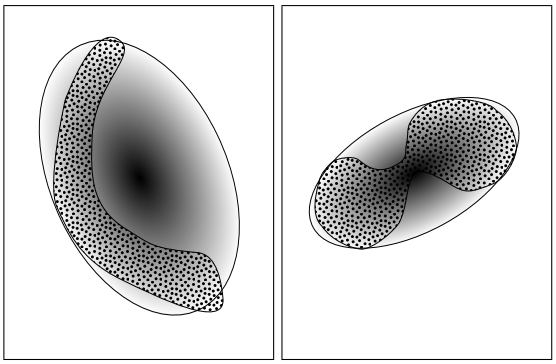

Figure 4.1: Example of two clusters returned by the clustering algorithm and corresponding Gaussian *crunching functions* which degrade areas distant from clusters.

all of the listed characteristics. However we may also choose some simple SGA or Evolution Strategy with small population size and strong exploitation characteristics.

4.1. Fitness deterioration and clustering

To increase the accuracy of the fitness deterioration process we want to use the maximum amount of information provided by the clustering algorithm. As we mentioned at the begining of this chapter we create one crunching function per cluster which, depending on the shape of the cluster, may not be very accurate or may even degrade areas of the fitness landscape which have not been explored yet and potentially contain valuable solutions. Figure 4.1 shows cases in which crunching functions created for extracted clusters strongly affect regions outside the clusters.

To increase the accuracy of our algorithm we use the following property of the OPTICS *ordering*:

> While creating the ordering, OPTICS constructs density-based clusters with respect to different densities simultaneously. OPTICS ordering actually contains the information about the intrinsic clustering structure of the input data set (up to the generating distance ϵ) (see [ABKS99]).

This might be shown for sample data (see Figure 4.3) by using its *reachability plot*. Once we create OPTICS ordering we may easily extract clusters with higher densities by decreasing ϵ and choose clusters for which MSE (Mean–Square Error) between the actual fitness function and the crunching function in the areas of clusters is minimal.

The algorithm works as follows (see Algorithm 4): having the OPTICS ordering of the population returned by the EA our method iteratively extracts clusters with higher densities by decreasing the *neighborhood radius* ϵ (see Figure 4.4), and then by constructing crunching function for extracted clusters and checking if the resulting crunching functions is more accurate than the best found in previous iterations (MSE comparison).

Algorithm 4 effectively prevents the deterioration process from destroying the fitness landscape in regions not yet explored by the CSFD algorithm. Figure 4.2 shows how the ϵ adjustment can improve the

Algorithm 4 Improving fitness deterioration accuracy

1: $\epsilon' = \epsilon$
2: **while** $\epsilon' > treshold$ **do**
3: $cs \leftarrow extractDBSCANClustering(\epsilon')$
4: $crunchFs \leftarrow createCrunchingFunctions(cs)$
5: $mse \leftarrow getMSE(crunchFs, currentFitness, cs)$
6: **if** $mse < minMSE$ **then**
7: $saveBestCrunching(mse, crunchFs)$
8: **end if**
9: $\epsilon' \leftarrow \epsilon' * 0.8$
10: **end while**

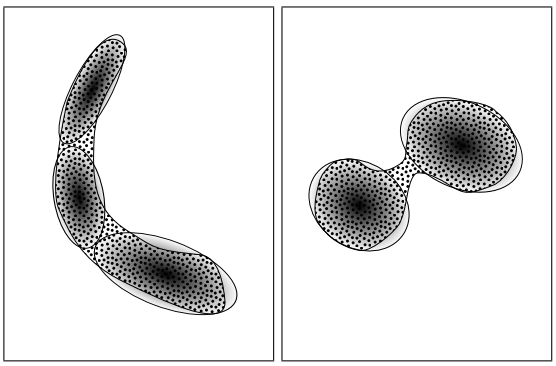

Figure 4.2: With *OPTICS ordering* we may extract cluster of higher densities and minimize the impact of the fitness deterioration on regions outside the clusters (instead of creating one crunching function per cluster like in figure 4.1 we extract denser clusters from the *ordering* and create crunching function which degrade only the region occupied by the cluster)

shape of the cluster extension with respect to the initial one (see Figure 4.1). To better understand how do we use OPTICS ordering to extract cluster of higher density see Figures 4.3 and 4.4.

4.2. Basic Scheme

The basic version of our deterioration algorithm is as follows. For each cluster C generate a multi-dimensional Gaussian function $g : V \to \mathbb{R}_+$

$$g(x) = F_k(x_{max})\, exp\left(-\frac{1}{2}(x-\overline{m})^T \Sigma^{-1} (x-\overline{m})\right) \tag{4.3}$$

where F_k is the fitness function in k-th iteration of the CSFD algorithm, x_{max} is the fittest individual from the cluster, the $\overline{m}$ is the cluster's centroid (mean phenotype of invividuals belonging to C) and Σ is unbiased sample covariance matrix estimated from the cluster population by the formula (4.4) (see e.g. [HL96])

$$\Sigma = \frac{1}{Card(C)-1} \sum_{x \in C} (x-\overline{m}) \otimes (x-\overline{m}) \tag{4.4}$$

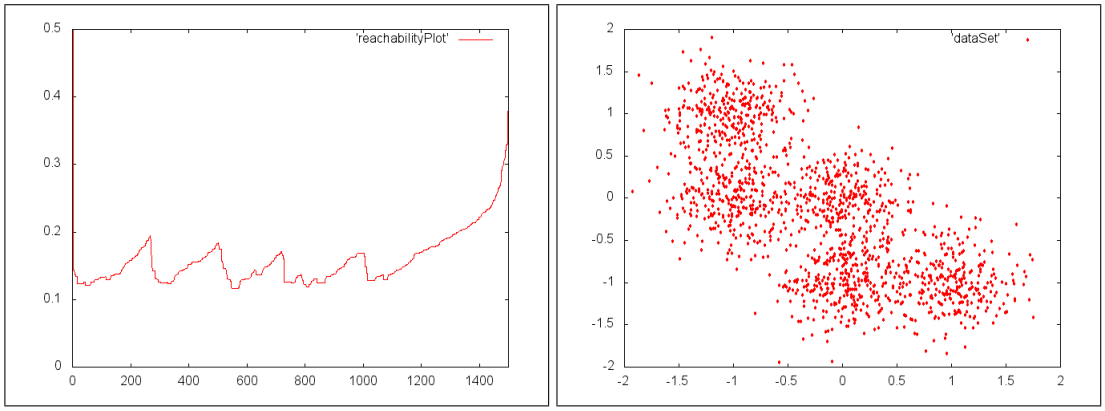

Figure 4.3: Sample data set of 1500 points and the corresponding reachability plot of the ordered points (shows the reachability distance of each individual in the data set - horizontal axis corresponds to the individuals in the data set and the vertical axis shows the reachability distance of a given individual) OPITCS ordering parameters: $\epsilon = 1.0$ $minPts = 30$. Cavities in the plot depict 5 clusters which might be extracted from the data set by the DBSCAN algorithm using proper value of ϵ' values

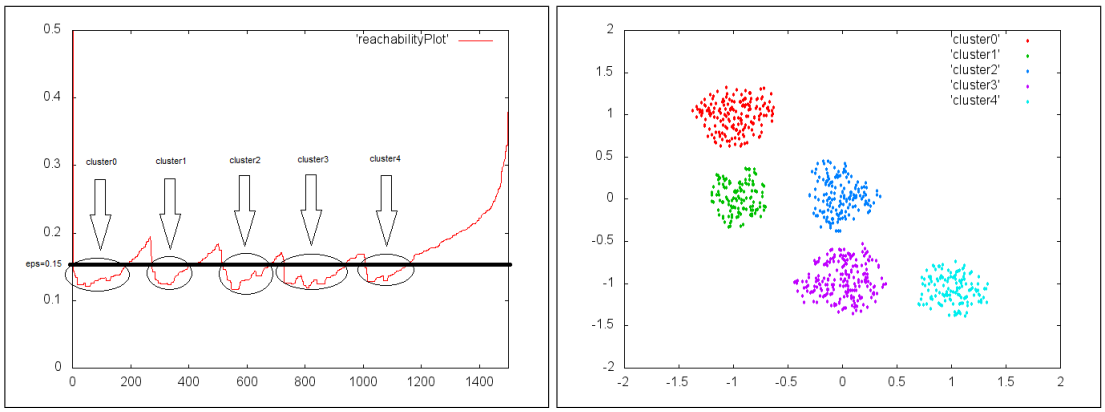

Figure 4.4: Extraction of clusters from the data set presented in Figure 4.3 using DBSCAN algorithm with $\epsilon' = 0.15$. The value of ϵ' is marked on the first plot which shows reachability distances and the "cut off" clusters

where $\otimes$ stands for the tensor product symbol. Please, notice, that the sum in (4.4) is spanned over all individuals belonging to the cluster C i.e. the phenotype x might be counted more than once, if is repeatedly represented in C.

Fitness function in the $(k+1)$-th iteration is obtained from the equation (4.1) by seting $\alpha_i \equiv 0$, $i = 1, \ldots, M_k$.

Big advantage of this algorithm are simplicity and speed. However, this version may cause strong deformation of the fitness landscape in areas which are distant from the already found clusters, which is unacceptable. To overcome this issue we developed so called *weighted scheme* described in the next Section 4.3.

4.3. Weighted Scheme

This type of fitness deterioration is more accurate and is likely to produce more stable fitness that basic scheme, cause the latter may produce sharp peaks in the fitness landscape, because of the very aggressive fitness degeneration. Weighted Scheme is also more complex cause it generates more computationally intensive fitness functions.

Initial steps are the same as in basic scheme, we create multi-dimensional Gaussian function for each cluster. What is different is how we compute the new (deteriorated) fitness is the following way

$$F_{k+1}(x) = F_k(x) + \sum_{i=1}^{M_k} \alpha_i(x) g_i(x), \tag{4.5}$$

where the α-coefficients are given by the following equations

$$\alpha_1(x) +, \ldots, + \alpha_{M_k}(x) = 1, \tag{4.6}$$

$$\alpha_i(x) = \xi(x) \left(\frac{1}{r_i(x)} \right), \tag{4.7}$$

$$\frac{1}{\xi(x)} = \frac{1}{r_1(x)} +, \ldots, + \frac{1}{r_{M_k}(x)}, \tag{4.8}$$

where $r_i(x) = \|x - \overline{m}_i\|$ is the distance between x and the C_i centeroid $\overline{m}_i$ (see Figure 4.5). So in order to compute fitness for a given individual x (more correctly for $ph(x)$) we have firstly compute distances $r_i(x)$. Then we compute $\xi(x)$ from the equation (4.8), next the α-coefficients from the system (4.6),(4.7) and finally the fitness value from (4.5). α-coefficients are computed separately for each new individual and this is why this method is more costly than the previous one. From the equations (4.5)-(4.8) it is clear that regions of domain which are distant from clusters found in previous iterations of the algorithm are very little affected by the crunching functions which is a big advantage over the basic scheme. However, experiments shows that the basic scheme yields very good results and is preferable over the weighted scheme due to the lower computational cost of the former.

4.4. Crunching Function Adjustment

Because of the fast convergence of populations generated by the GA algorithm to the local solutions and the features of the Algorithm 4 used to increase accuracy of the deterioration process, clusters

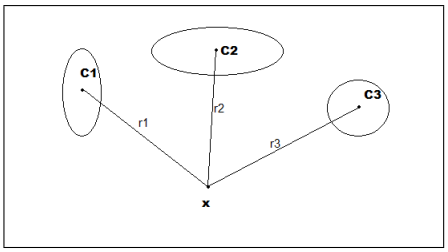

Figure 4.5: The coefficient α_i is inversely proportional to the distance from the center of cluster C_i to the x. α_i may be seen as the impact C_i has on x

sometimes become very dense in areas close to the local minimizers. Gaussians created for such clusters does not approximate a basin of attraction well, speaking informally: Gaussian functions created for such clusters consist of high and thin peaks which deteriorate only the area inside the cluster, only the narrow basin of attraction in which the cluster resides. To overcome this issue we developed so called *Crunching Function Adjustment* (CFA) algorithm described below.

We use sample covariance matrix as an estimator (see [HL96]), which is extremely sensitive to outliers. However we may take this property as our advantage and incorporate it CFA algorithm. Having given a cluster of points the CFA algorithms works as follows:

– We estimate the covariance matrix Σ and then compute its N eigenvectors ($N = dim(V)$ is the dimension of the problem space). They are ortogonal one to each other and define the orientation of the Gaussian "bell"

– for each eigenvector v_i we generate two points $\underline{p_i}, \overline{p_i}$ called *leading marks*

$$\begin{aligned} \overline{p_i} &= \overline{m} + \sqrt{\lambda_i}\, v_i \\ \underline{p_i} &= \overline{m} - \sqrt{\lambda_i}\, v_i \end{aligned} \tag{4.9}$$

where $\overline{m} \in V$ is the cluster's centroid and λ_i is an eigenvalue of the eigenvector v_i.

– Then we add these $2N$ generated leading marks to the initial multiset which constitute a cluster, and compute new covariance matrix.

– Because the sample covariance matrix is very sensitive to leading marks, the resulting covariance matrix produces a Gaussian function whose "bell hypersurface" is more stretched in directions of eigenvectors.

Please notice, that the improvement introduced above is purely heuristic, having no precise mathematical motivation. It was designed only for deterioration performed by Gauss functions and positively verified for 2D benchmarks (see Section 5.2) so its usefulness in other cases in unknown.

4.4.1. CFA Results

The figures below shows the result of our CSFD algorithm with CFA for two simple functions from $f : \mathbb{R}^2 \to \mathbb{R}$, specifically:

– $f(X) = 2e^{-(x^2+y^2)}$, where $X \in \mathbb{R}^2$

– $f(X) = e^{-(x^2+y^2)} + 1.4e^{-((x-1.7)^2+(y-1.7)^2)}$, where $X \in \mathbb{R}^2$

Table 4.1: The results of basic deterioration scheme applied to functions $f(X) = 2e^{-(x^2+y^2)}$ (up-left and down-left) and $f(X) = e^{-(x^2+y^2)} + 1.4e^{-((x-1.7)^2+(y-1.7)^2)}$ (up-right and down-right) with the populations generated with normal distribution around the solutions. For the fist function we may see that the overall landscape decreases only by 30% (down-left), while using CFA gives us 85% of decline (up-left). For the second function we see 25% of deterioration (down-right), while CFA gives us 78% (up-right)

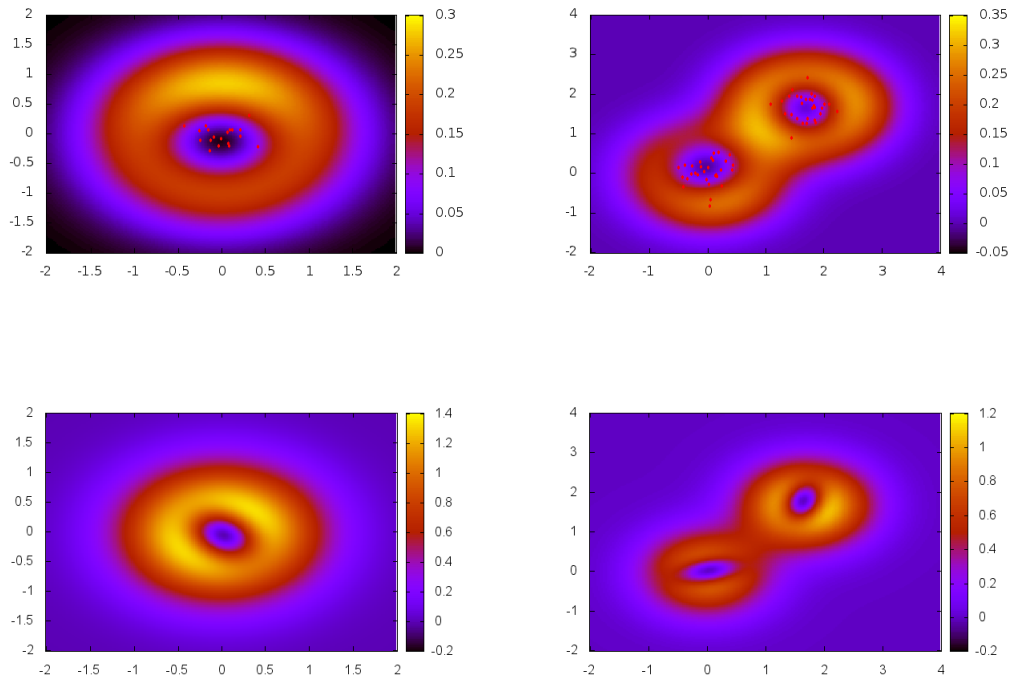

Table 4.1 shows how much the algorithm benefit from using the CFA algorithm.

5. Tested Algorithms

The whole consideration leading to define CSFD was performed for GOPs of finding local/global minimizers. It can be easily reformulated for GOPs of finding local/global maximizers for which the equivalent minimization problem can be established. In this case the landscape deterioration consists in "leveling hills" instead of "filling valleys". The particular class of such maximization GOPs is associated with continuous objectives (fitnesses) F defined on compacts in $\mathbb{R}^N$. In such cases we can set the new fitness as $-F$ plus the maximum value of F over the search domain in order to obtain the equivalent minimization problem.

The aim of our experiments is to show the efficiency of the *Cluster Supported Fitness Deterioration* CSFD in finding basins of attraction of local solutions.

5.1. Genetic Engine

The CSFD strategy uses the *Simple Evolutionary Algorithm* (SEA). In contrast to the *Simple Genetic Algorithm* (SGA), SEA uses a real values as a parameters of the chromosome in populations without performing coding and encoding process before calculates the fitness values of individuals (see e.g. [Sch07]). Namely, SEA is more straightforward, faster and more efficient than SGA.

We use the standard genetic operators for real-valued representation:

- Crossover: $Y = x_1 + N(mean, \sigma_c)(x_2 - x_1)$, where x_1, x_2 are individuals, $N(mean, \sigma_c)$ stands for the multivariate normal distribution, $mean = \frac{x_1+x_2}{2}$ is a $[x_1, x_2]$ centroid, σ_c is the parameter used to control exploration and exploitation of the SEA.

- Mutation: $y = x+N(0, \sigma_m)$, where x is the individual to be mutated, σ_m the configurable mutation parameter.

We want the genetic engine to be cheap and converge very quickly to local solutions, so we may efficiently deteriorate found basins of attraction in subsequent iterations. To improve the convergence of the population maintained by SEA we use proportional selection and we increase the exploitation capabilities of the SEA using proper parameters, usually: $\sigma_c \in [0.1, 0.3]$, $\sigma_m \in [0.4, 1.0]$, depending on the problem.

5.2. Test functions

In order to visualize the deterioration process we use three 2D test functions:

– Rastrigin:

$$F(x) = 20 + \sum_{i=1}^{2}(x_i^2 - 10cos(2\pi x_i)) \tag{5.1}$$

for $x_1, x_2 \in [-4, 4]$.

– Langermann:

$$F(x) = \sum_{j=1}^{5} c_j \, exp(-\frac{1}{\pi}((x_1 - a_j)^2 + (x_2 - b_j)^2))$$

$$cos(\pi ((x_1 - a_j)^2 + (x_2 - b_j)^2)) + 5, \tag{5.2}$$

$$a = (3, 5, 2, 1, 7),\ b = (5, 2, 1, 4, 9),\ c = (1, 2, 5, 2, 3)$$

for $x_1 \in [0, 4],\ x_2 \in [-1, 3]$.

– Griewangk:

$$F(x) = \frac{1}{4000} \sum_{i=1}^{2} x_i^2 \ - \ \Pi_{i=1}^{2} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{5.3}$$

for $x_1, x_2 \in [-4, 4]$.

W decide to handle maximization GOPs associated with benchmarks (5.1), (5.2), (5.3) because their results can be more expressively presented then the equivalent minimization ones. The CSFD instance dedicated to maximization problems were utilized (see Remark 5).

5.3. CSFD initial parameters

In each of the tests we use the same set of initial values for algorithm's parameters. The *Cluster Supported Fitness Deterioration* needs the following parameters to be configured:

– $popSize$ – population size of the genetic engine used in CSFD; it is advisable to use small population to minimize the fitness computation costs; usually: $40 \leq popSize \leq 100$

– ϵ and $minPts$ – OPTICS generating distance and minimum number of points in ϵ-neighborhood (see Section 3.1); the values for the ϵ should be 'large enough' to allow the the creation of proper *ordering*, usually $\epsilon \in [0.5, 1.0]$ depending on the selection pressure. $minPts$ should be chosen as follows:

$$minPts = \begin{cases} \frac{popSize}{4} & \text{if} \quad 10 \leq \frac{popSize}{4} \leq 20 \\ 10 & \text{if} \quad \frac{popSize}{4} < 10 \\ 20 & \text{if} \quad 20 < \frac{popSize}{4} \end{cases} \tag{5.4}$$

– σ_c and σ_m – standard deviations used for the *crossover* and *mutation* respectively (see Section 5.1)

– $mutationProb$ – mutation probability (crossover is always performed)

Table 5.1: Results of experiments (CSFD algorithm)

Function	*NBA*	*DoD*
Rastrigin	64/64	0.64
Langermann	16/18	0.59
Griewangk	4/4	0.84

5.4. Efficiency measures and CSFD results

We will use two simple measures in order to evaluate the deterioration quality. The first one was the number of recognized basins of attraction NBA. The basin of attraction of the local maximizer is considered as recognized by CSFD if the cluster of individuals was established and the cluster extension is wholly included in the basin.

The second measure, called the degree of deterioration DoD, is defined by the following formula:

$$DoD = \frac{F_0^{max} - F_M^{max}}{F_0^{max} - F_0^{min}} \tag{5.5}$$

where M is the number of iterations performed by the CSFD strategy, F_0^{max} denotes the maximum value of the fitness at the beginning of the algorithm (0-iteration), similarly F_0^{min} is the minimum value of the fitness function at the beginning of the algorithm and F_M^{max} is the maximum value of the fitness in the last iteration of the algorithm. DoD might be also expressed in percentage.

CSFD was run several times for each of the objective (5.1), (5.2), (5.3). The most typical behavior of this strategy in case of each objective are depicted in Figures 5.1, 5.2, 5.3. Moreover Table 5.1 gathers the metrics values for these computations.

The best performance was obtained for the Griewangk benchmark for which all local maxima were encountered and for which the maximum degree of fitness leveling 84% was obtained. All local maxima were also recognized in the case of the Rastrigin function, but the degree of deterioration was only 64%. The Langermann benchmark became most difficult for CSFD, since the degree of deterioration was only 59% and two basins of attraction (out of eighteen ones) remained unknown.

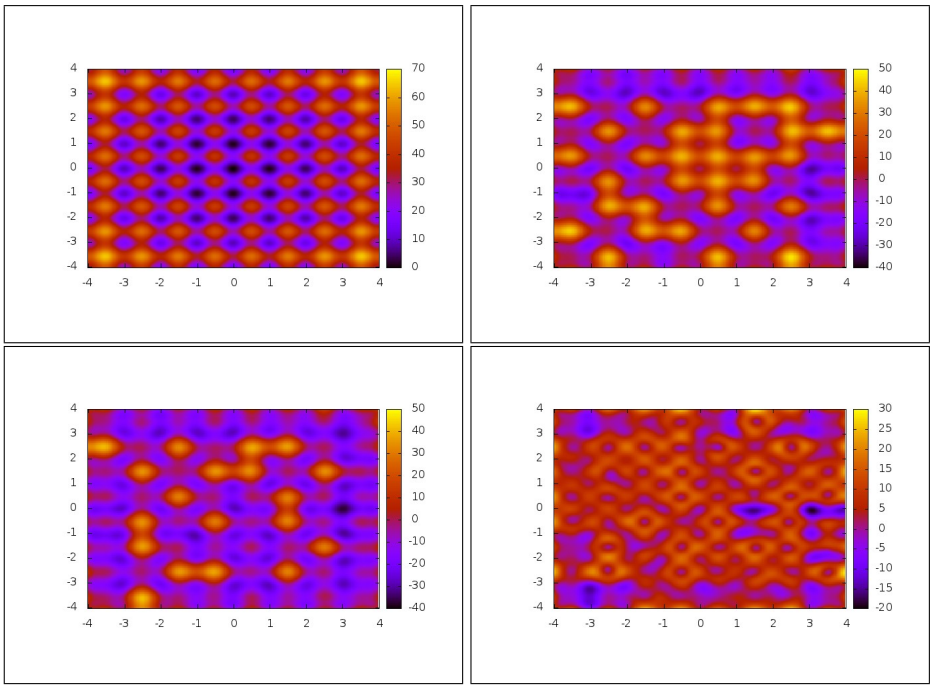

Figure 5.1: Original (the first one) and deteriorated fitness landscape of the Rastrigin function seen in the 33th, 45th and 64th interation of the CSFD strategy respectively. Parameters: $\epsilon = 0.7$, $minPts = 12$, $popSize = 50$, $\sigma_c = 0.1$, $\sigma_m = 0.5$

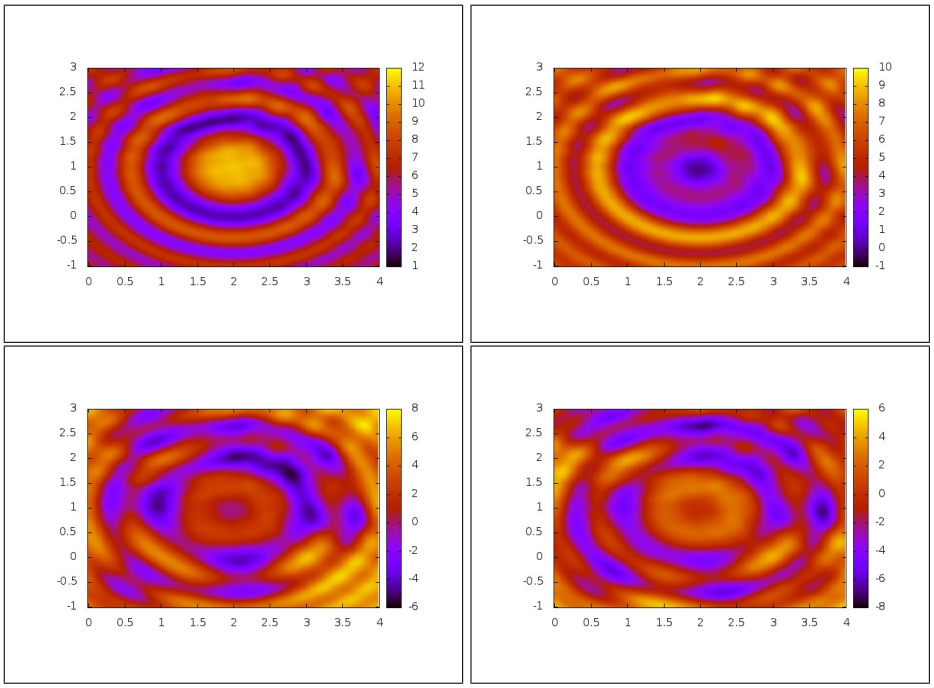

Figure 5.2: Original (the first one) and deteriorated fitness landscape of the Langermann function seen in the 1th, 8th and 16th interation of the CSFD strategy respectively. Parameters: $\epsilon = 0.7$, $minPts = 12$, $popSize = 80$, $\sigma_c = 0.1$, $\sigma_m = 0.5$

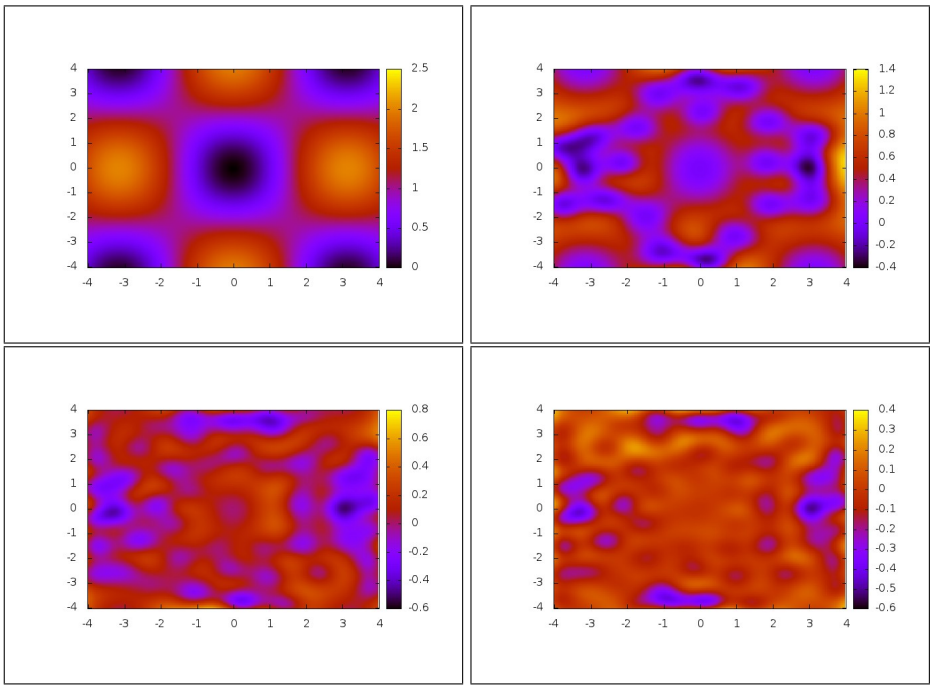

Figure 5.3: Original (the first one) and deteriorated fitness landscape of the Griewangk function seen in the 21th, 51th and 69th iteration of the CSFD strategy respectively. Parameters: $\epsilon = 0.7$, $minPts = 12$, $popSize = 50$, $\sigma_c = 0.1$, $\sigma_m = 0.5$

6. Implementation

To verify our algorithm we have created a simple, global optimization framework which enabled us to execute and test all of the algorithms described throughout this article.

We decided to use Java as a programming language and runtime environment for this project.

Using the framework we can find optima of real, mulimodal, multidimensional and continuous functions. We just need to provide the framework with the concrete fitness function implementation, and by specifying the problem domain.

The framework principle is to separate the interfaces from implementation. Our API provides interfaces for evolutionary algorithms, fitness assignment and fitness deterioration, genetic operators like selection and reproduction, phenotype, clustering algorithms and many more. This makes it compact, extensible and easy to introduce new implementations. For the detailed description of the classes and interfaces see subsection 'Implementation in Java'.

6.1. Architecture

The project consists of six main packages, which communicates with the others through clearly specified interfaces and together constitute a complete, lightweight framework for testing and execution of the global optimization algorithms especially the evolutionary algorithms.

- *Algorithm module* - central module of the framework. Contains implementation of the *Sequential Niching* algorithm as described in chapter 2. The main class in the module *ki.edu.agh.algorithm.SequentialNichingWithOpticsClustering* is responsible for creation of the Spring context and execution of the algorithm.

- *Clustering module* - contains interfaces for clustering algorithms and implementation of OPTICS.

- *Fitness deterioration module* - contains implementation of *FitnessDeterioration* interface. Most important are: *ki.edu.agh.deterioration.SimpleGaussianFitnessDeterioration* and *ki.edu.agh.deterioration.WeightedGaussianFitnessDeterioration*.

- *Evolutionary Algorithms module* - contains all the necessary interfaces for EA, selection and reproduction operators, individuals and phenotypes. Implementations of SGA and Evolution Strategy.

- *Printing Modul* - used to print populations and fitness landscape during the program execution

– *Statistics Utils* - based on JAMA. This package contains many useful statistical functions, e.g. CFA algorithm, covariance matrix estimation, multidimensional Gaussian function, fitness deterioration untilities.

6.2. Implementation in Java

The framework was written to be elegant and extensible. The system's modules are loosely coupled and each of system's components has little knowledge of the definitions of other separate components (see previous section). This allows for extensibility in further design. To implement the system we used Spring as an application framework, Maven for project management and build automation, JUnit and Mockito for testing and Git as revision control system. The source code, configuration files and sample results may be found at GitHub: https://github.com/wolny/Fitness-Deterioration

6.2.1. Technologies

– Spring - application framework

– Maven - project management and build automation

– Mockito - testing framework

– JAMA - linear algebra package

7. Conclusions

7.1. Summary

The aim of my Master Thesis was to find an effective fitness deterioration algorithm as defined in chapter 4, which minimizes the chance of multiple exploration of the same solution during the course of the *Sequential niching* algorithm. Taking into account the results of tests described in chapter 5 we may come to the conclusion that the goal was reached. However, tests was performed only for simple multimodal problems in which the obtained clusters were convex and provides a lot of information about underlying basins of attraction. Applying the algorithm for more demanding functions should be the next step in the further development of the algorithm.

The assumption that distribution of individuals inside a cluster provides information about its shape oversimplifies the real nature of the evolutionary algorithms, where the population distribution inside basins of attraction is very hard to predict and depends on many factors, including recombination versus mutation rate, selection algorithm, definition of genetic operators such as mutation and crossover. We can expect satisfactory results when choosing proportionate selection and genetic operators which are based on normal distribution. *Fitness Proportionate Selection* usually causes faster convergence to local solutions and normal distribution based reproduction operators tends to produce populations with useful information about fitness landscape.

Based on the results from chapter 5 we recommend to use *Basic scheme* (section 4.2) of the fitness deterioration, cause it is a great deal cheaper than *Weighted scheme* (section 4.3) and yields practically the same results as Weighted crunching functions. The latter are more accurate only for problems with many local solutions but even for such problems it is reasonable to use *Basic scheme* because it outperforms *Weighted scheme* in terms of efficiency.

The algorithm (see section 4.1) we used to prevent degradation of regions unexplored by the algorithm and to increase the fitness deterioration accuracy should perform well for cases when the cluster is not convex, provided that individuals inside the cluster are non-uniformly distributed, otherwise the extraction of denser clusters would not prevent the deterioration process from loosing information contained in regions not visited by the *sequential niching* yet.

The most valuable outcomes of this work are:

– Sequential niching with fitness deterioration is one of the most promising stochastic strategies for analyzing multi-modal global optimization problems in the continuous domains embedded in vector metric spaces.

– Existing instances of the strategy mentioned above exhibits several disadvantages: huge memory complexity of memorizing deteriorated regions (e.g. CGS with raster clustering); unsatisfactory accuracy of the fitness approximation that lead to the incorrect deterioration and finally may lead to removing individuals from unchecked areas or the multiple check of non-promising areas already browsed; dependency on the evolutionary technique used.

– The discussion presented in Sections 1.2.1, 1.2.2, 1.3 clearly shows the way of necessary improvements. The proposed deterioration strategy CSFD combines the low memory complexity of the exponential fitness improvement with the accuracy of clustering based techniques.

– The CSFD performs very well for 2D complex multi-modal functions like the ones used for testing (see Section 5.2) being also well suited to detect the basins of attraction of the local and global extrema. Exhaustive testing for higher dimensional problems will be the subject of future research.

– Performed experiments show that we can expect satisfactory results when the GA utilizes genetic operators based on the normal distribution. Fitness proportionate selection causes satisfactory convergence to local solutions and the normal distribution based reproduction operators tends to produce populations with useful information about fitness landscape.

– The Hierarchical Genetic Strategy (HGS) (see [SK03], [SAT04], WierzbaSemczukKolodziejSchaefer2003) would be very efficient from the standpoint of the deterioration process. This strategy performs an efficient concurrent search in the optimization landscape by many small populations. Creation of these populations is governed by dependent genetic processes with low complexity. Moreover, HGS is likely to find many solutions in a single run of the algorithm and that the hierarchy of populations generated by the algorithm are rapidly convergent.

– High accuracy of deterioration offered by the CSFD results from the positive synergy of two mechanisms: clusters obtained from the modified OPTICS (see Algorithm 4) and improved by leading marks approximate well basins of attraction of local/global extrema; the form of the weighted deteriorartion function (4.5) and form of weights (see formulas (4.6), (4.7), (4.8)) maximizes the effect of deterioration over the area of cluster extensions i.e. the area of basins of extrema already recognizes and prevent degradation of unexplored regions.

– Algorithm 4 which increases fitness deterioration accuracy performs well also in cases when the clusters are not convex e.g. for Langermann function (see Figure 5.2).

– Sequential niching obtained by CSFD preserves the asymptotic guarantee of success. The probability of sampling is significantly decreased over the cluster extensions but still greater then zero, so this regions are not excluded from future sampling, even in case of inaccurate deterioration. It seems to be the advantage over the sequential niching based on "tabu" techniques.

7.2. Future Research

This work focuses mainly on the *Fitness Deterioration* algorithm and shows the result of Basic and Weighted variants of the algorithm applied to simple multimodal functions (see chapter 4). Given the *Sequential niching* algorithm described at the beginning of this paper (chapter 2), the main direction of the future research should be to test this algorithm using many different evolutionary algorithms and evolution strategy. The most promising algorithm to choose would be *Hierarchical Genetic Strategy* (HGS) [WSKS03].

The Hierarchical Genetic Strategy (HGS) performs efficient concurrent search in the optimization landscape by many small populations. The creation of these populations is governed by dependent genetic processes with low complexity [WSKS03]. HGS is an example of parallel genetic algorithms and it performs very well especially in case of problems with many local extrema. Taking into account the fact that HGS is likely to find many solutions in a single run of the algorithm and that the hierarchy of populations generated by the algorithm are rapidly convergent it would be very efficient from the standpoint of the deterioration process.

There are many aspects of the implementation of *CSFD* algorithm, described in chapter 6, which can be improved as well. The framework uses very simple concurrency model which may be further developed to improve the overall performance of the algorithm. The most costly stages of the algorithm include: fitness function computations during EA algorithm and creation of *OPTICS ordering* [ABKS99]. The former problem might be solved by introducing some of the known parallel genetic algorithms and the latter might be tackled by proper domain decomposition. If we want to introduce a highly concurrent EA model again it would be best to use HGS because of its natural concurrent character.

A. Installation and Program Results

A.1. Installation and Execution Instructions

– Configuration files: *application-context.xml* - here you specify used algorithms, problem domain, fitness function and other aspects of the system (see sample configuration in appendix B); some of the most important properties are taken from *algorithm.properties* file using property placeholders

– To build the project invoke: *mvn clean install*.

– Program execution: *mvn exec:java* By default the main class of the project is: *ki.edu.agh.algorithm.SequentialDeteriorationEAWithOpticsClustering* (see *pom.xml* for more details)

A.2. Program Results

Program saves the results in 'diagrams' catalog:

– cluster's members are stored in files of the name 'clusterN', where N is the number of cluster

– populations for each run are stored in files 'populationN', where N is the number of EA run

– fitness landscape is stored in file 'originalFitnessLand'

– reachablity plot for OPTICS algorithm is stored in "reachabilityPlot'

– fitness landscape after each run is stored in 'fintessLand_iterN' where N is the number of EA run

B. Spring Configuration

B.1. Sample Spring configuration

Below is the sample configuration file of system modules which are defined as Spring beans. All placeholders which are present in the Spring configuratin file should be defined in the *algorithm.properties* file.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

  <bean
    class=
    "org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations" value="classpath:algorithm.properties" />
  </bean>

  <bean id="algorithm"
    class="ki.edu.agh.algorithm.SequentialNichingWithOpticsClustering">
    <property name="problemDomain">
      <bean class="ki.edu.agh.problem.MultimodalRealSpaceProblem">
        <!-- set domain and functor -->
        <property name="domain">
          <bean class="ki.edu.agh.problem.Domain">
            <property name="multidimensionalCube">
              <list>
                <!-- first dimension interval -->
                <bean class="ki.edu.agh.problem.Interval">
                  <constructor-arg value="-2." />
                  <constructor-arg value="4." />
                </bean>
```

```
            <!-- second dimension interval -->
            <bean class="ki.edu.agh.problem.Interval">
              <constructor-arg value="-2." />
              <constructor-arg value="4." />
            </bean>
          </list>
        </property>
      </bean>
    </property>
    <property name="fitnessFunction">
      <bean class="ki.edu.agh.fintess.StandardFitnessFunction">
        <constructor-arg>
          <bean class="ki.edu.agh.functors.BiModalFunction" />
        </constructor-arg>
      </bean>
    </property>
    <!-- specify if the the problem is minimization or maximization -->
    <property name="minimization" value="false" />
  </bean>
</property>

<property name="iterationCount" value="${algorithm.iterationCount}" />

<!-- clustering algorithm -->
<property name="clusteringAlgorithm">
  <bean class="ki.edu.agh.clustering.optics.OpticsClustering">
    <constructor-arg>
      <bean
        class="ki.edu.agh.clustering.optics.OpticsParamteres">
        <property name="minPoints" value="${optics.min_pts}" />
        <!-- problem's domain dependent -->
        <property name="epsilon" value="${optics.eps}" />
      </bean>
    </constructor-arg>
  </bean>
</property>

<!-- specifies evolutionary algorithm used -->
<property name="evolutionaryAlgorithm" ref="sga" />
```

```
    <!-- fitness deterioration algorithm -->
    <property name="fitnessDeterioration">
      <bean
        class="ki.edu.agh.deterioration.SimpleGaussianFitnessDeterioration" />
    </property>
  </bean>

  <bean id="sga" class="ki.edu.agh.evolutionary.algorithm.SGA">
    <!-- SGA configuration -->
  </bean>
</beans>
```

Bibliography

[ABKS99] M. Ankerst, M.M. Breunig, H.P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proc. ACM SIGMOD 1999 Int. Conf. on Management of Data*, 1999.

[Ara01] J. Arabas. *Lectures in Evolutionary Algorithms*. WNT, Warsaw, Poland, 2001.

[BBM93] D. Beasley, D.R. Bull, and R.R. Martin. A sequential niche for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125, 1993.

[DS75] L.C.W. Dixon and G.P. Szegö, editors. *Toward Global Optimization*. North Holland, 1975.

[EKSX96] M. Ester, H.P. Kriegel, J. Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.

[Gol89] D. Goldberg. *Genetic Algorithms and their Applications*. Addison–Wesley, 1989.

[HL96] J.P. Hoffbeck and D.A. Landgrebe. Covariance matrix estimation and classification with limited training data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

[KT87] A.H.G. Rinnoy Kan and G.T. Timmer. Stochastic global optimization methods. *Mathematical Programming*, 39, 1987.

[Mah95] S.W. Mahfound. Niching methods for genetic algorithms. *IlliGAL Report*, 1995.

[MM95] Samir W. Mahfoud and Samir W. Mahfoud. A comparison of parallel and sequential niching methods. In *In Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 136–143. Morgan Kaufmann, 1995.

[Obu97a] A. Obuchowicz. Adoption of the time–varying landscape using an evolutionary search with soft selection algorithm. In *Proc. of the 3-rd KAGiOG Conf.*, pages 245–251, Potok Złoty, Poland, 1997.

[Obu97b] A. Obuchowicz. The evolutionary search with soft selection and deterioration of the objective function. In *Proc. of 6-th Symp. Intelligent Information Systems and Artificial Intelligence IIS'97*, pages 288–295, Zakopane, Poland, 1997.

[OK99] A. Obuchowicz and J. Korbicz. Evolutionary search with soft selection algorithms in parameter optimization. In *Proc. of the PPAM'99 Conf.*, pages 578–586, Kazimierz Dolny, Poland, 1999.

[OP97] A. Obuchowicz and K. Patan. About some evolutionary algorithm cases. In *Proc. of the 2-nd KAGiOG Conf.*, pages 193–200, Rytro, Poland, 1997.

[PR02] M.P. Pardalos and H.E. Romeijn. *Handbook Global Optimization*, volume 2. Kluwer, 2002.

[SA04] R. Schaefer and K. Adamska. Well-tuned genetic algorithm and its advantage in detecting basins of attraction. In *Proc. of the 7-th KAEiOG Conf.*, pages 149–154, Kazimierz, Poland, 2004.

[SAT04] R. Schaefer, K. Adamska, and H. Telega. Genetic clustering in continuous landscape exploration. *Engineering Applications of Artificial Intelligence EAAI*, 17:407–416, 2004.

[Sch07] R. Schaefer. *Foundation of Glabal Genetic Optimization*. Springer, 2007.

[SJ01] R. Schaefer and Z.J. Jabłoński. On the convergence of sampling measures in global genetic search. *LNCS*, 2328:593–600, 2001.

[SK03] R. Schaefer and J. Kołodziej. Genetic search reinforced by the population hierarchy. In K.A. De Jong, R. Poli, and J.E. Rowe, editors, *Foundations of Genetic Algorithms 7*, pages 383–399. Morgan Kaufman, 2003.

[Tel99] H. Telega. *Parallel Algorithms for Solving Selected Inverse Problems*. PhD Thesis, AGH University of Science and Technology, Kraków, Poland, 1999.

[WSKS03] B. Wierzba, A. Semczuk, J. Kołodziej, and R. Schaefer. Hierarchical genetic strategy with real number encoding. In *Proc. of 6-th KAEiOG Conf.*, pages 231–237, Łagów Lubuski, Poland, 2003.

[Zei85] E. Zeidler. *Nonlinear Functional Analysis and its Application*. Springer, 1985.