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

. . .

Contents

1. Introduction

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

1.1. A statement of a problem

A Global Optimization Algorithm is defined as optimization algorithm that employs measures that prevent convergence to local optima and increase the probability of finding a global optimum.

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 problems require the location and maintenance of multiple robust local solutions, i.e local solutions whose basins of attraction areproperly 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 like crowding or fitness sharing which promote diversity of population, which in turn delay premature convergence and likely enable the algorithm to find multiple optimal solutions in single population.

Standard niching methods are often inefective and hard to introduce in existing evolutionary algorithms. In this paper we adopt a different approach to multimodal function optimization. Instead of embedding a niching method in the evolutionary algorithm itself we use a hybrid approach in which we perform several runs of a evolutionary algorithm and alter the fitness function in every subsequent run in a way that prevents exploration of basins of attraction which were found in previous runs of the algorithm.

In each iteration we run EA, then cluster received population and based on the assumption that clusters of individuals obtained from the clustering algorithm are located in basins of attraction we interpolate each basin by multidimensional Gaussian function. By combining these functions with current objective function in a proper way we create deteriorated fitness function which will discourages future runs from revisiting the same area.

This work tries to find an effective fitness deterioration technique in high-dimensional domain spaces. We have implemented a general-purpose framework which can be used to test our fitness deterioration techniques in conjunction with various evolutionary algorithms. While our algorithm may be used with many types of EAs it would be the most efficient when used with algorithms which are capable of finding many local solutions in single run. This is why for tests we choose so called Hierarchical Genetic Strategy which performs efficient concurrent search in the optimization landscape by many small populations.

The quality of the deterioration process strongly depends on clustering results. We choose density-based algorithm called OPTICS as with this method we can extract clusters of different densities very efficiently and choose clusters which give the best accuracy of fitness deterioration process.

1.2. Related Work

As mentioned before this work focus on finding solutions for multi-modal optimization tasks. There are many publications which decsribes how to extend EAs to multi-modal optimization. Most of them focuses on *Niching methods* [8, 9, 10] which address this issue by maintaining a population of diverse solutions througout the time and this way they allow parallel convergence into multiple good solutions in multimodal domains.

Our solution works by iterating a simple GA and maintaining the best solution of each run off-line, by detection of basins of attraction and degeneration of fitness landscape. We may consider our algorihtm as a variant of *Sequential niching* approach (throughout this paper we use terms *Sequential niching* and *Fitness deterioration* interchangeably).

At this point it is worth mentioning some of the works of Prof. A. Obuchowicz especially the publication [3] which is the only one I found which use the term *fitness deterioration* explicitly. In [3] he describes ESSS-DOF algorithm (Evolutionary Search with Soft Selection with Deterioration of Objective Function) as an extension to the ESSS method which maintaing population diversity by the following schema:

> When the population converges to local optimum we degenerate the objective function which cause the rapid migration of individuals and enable the population to escape for the local optimum.

The algorithm degenerate the objective function by composing it with Gaussian function which approximate the local optimum. Our deterioration algorithm described in detail in chapter 4 uses Gaussian functions as well (Gaussian function has got many useful properties which makes it well-suited to the fitness deterioration. We describe these characteristics in chapter 4).

Mentined methods are incorporated directly into the basic cycle of evolutionary algorithm which differs from our *Sequential niching* technique. The sequential niching approach has several advantages:

– it is simple to incorporate in existing optimization methods

– it efficiently finds many local solutions

- it provides reasonable stop criterion which in this case is based on the quality of clusters returned by the clustering algorithm

2. Algorithm

2.1. A Hybrid Approach

Our deterioration algorithm may be easily incorporated in existing optimization methods. This can be expressed as a general hybrid approach to global optimization in which we do not change the implementation of the used EA but treat it as an integral element of our iterative process. This can be visualized by the following pseudo code:

1: **while** $i < getIterationCount()$ **do**
2: $\quad execute(evolutionaryAlgorithm)$
3: $\quad population = getPopulation(evolutionaryAlgorithm)$
4: $\quad clusters = cluster(population)$
5: $\quad$ **if** $clusters.isEmpty()$ **then**
6: $\quad\quad break$
7: $\quad$ **end if**
8: $\quad detFitness = performCrunching(clusters, currentFitness)$
9: $\quad saveClusters(clusters)$
10: $\quad updateFitness(detFitness)$
11: **end while**
12: $execute(evolutionaryAlgorithm)$
13: $extractBestClusters()$

The condition in while statement should be treated as control statement rather than the real termination criterion. It may be useful in cases where the clustering algorithm is misconfigured and always returns some clusters in which case the condition would prevent our algorithm to run endlessly. Our actual stop criterion is based on the condition inside the loop (line 5): if the clustering algorithm did not find any group of similar individuals or the group has low quality measures it jumps out of the loop and then we perform the last invocation of the EA in order to increase the probability of finding the local optimum which has not been explored during iterations.

Using this general scheme has one important advantage: we do not have to change existing implementation of a given EA in contrast to standard niching methods which must be incorporated directly in the evolutionary algorithm. But what is more important, it provides a reasonable stop criterion which in this case is based on the quality of clusters returned by the clustering algorithm.

In the subsequent chapters we will describe components of the general algorithm in more details. In chapter 3 we will discuss the clustering method used in our algorithm and how does it influence the deterioration process. Chapter 4 describes deterioration algorithm in more detail and chapter 5 shows the results of the tests.

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 can be used to determine stop criteria (see section 3.2) and the latter usecase is used in our deterioration schema (see chapter 4).

3.1. Cluster Extension

Basin of attraction is a term used in dynamic systems defined as the set of initial conditions leading to long-time behavior that approaches the particular attractor. In the field of global optimization, basin of attraction may be defined in the following way.

Definition. Basin of attraction is a subset A of the problem space which contains local solution and for each point which belongs to A the gradient of the objective function at that point will lead towards the solution.

Clusters may be seen as an approximation of the basin of attraction, because the distribution of individuals which flooded to the basins provides useful information about its size and shape. So the clustering algorithm may be used to detect the set of individuals which belongs to the same basin of attraction. Such a set may be later described by extracting some statistical information from that set, e.g the center point, the radius of the set, covariance matrix etc. This is what we called a **Cluster extension**.

3.2. Clustering as a Stop Criterion

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 founded. Some of the common termination criteria such as:

– maximum computation time

– total number of iterations

– no improvement for a specified number of iterations are only applicable to a specific problems and none of them can be used as a general stop criterion. The clustering, which is performed in every iteration, on the other hand may give 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. 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 EA is likely to converge to such solutions, so having found no clusters of individuals after performing the EA in a given iteration shows that the population does not converge to any robust solution. In such circumstances we terminate the main loop of the algorithm.

3.3. OPTICS

We have choosen density-base clustering algorihtm called *OPTICS: Ordering Points To Identify the Clustering Structure* [2]. In density clustering clusters are regarded as regions in the data space in which the objects are dense and which are separated by regions of low object density. These regions 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*. 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 $minPts$.

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:

Definition. core-distance - the core-distance of an object p is simply the smallest distance ϵ' between p and an object in its ϵ-neighborhood such that p would be a core object with respect to ϵ' if this neighbor is contained in $N_\epsilon(p)$. Otherwise, the core-distance is *UNDEFINED*

Definition. reachability-distance - the reachability-distance of an object p with respect to another object o is the smallest distance such that p is directly density-reachable from o if o is a core object

This information is sufficient to extract all density-based clusterings with respect to any distance ϵ' which is smaller that the generating distance ϵ

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 3.1 shows the result of *OPTICS* clustering for a sample set of points. Figure 3.2 shows reachablity plot for various *generating distances* - ϵ.

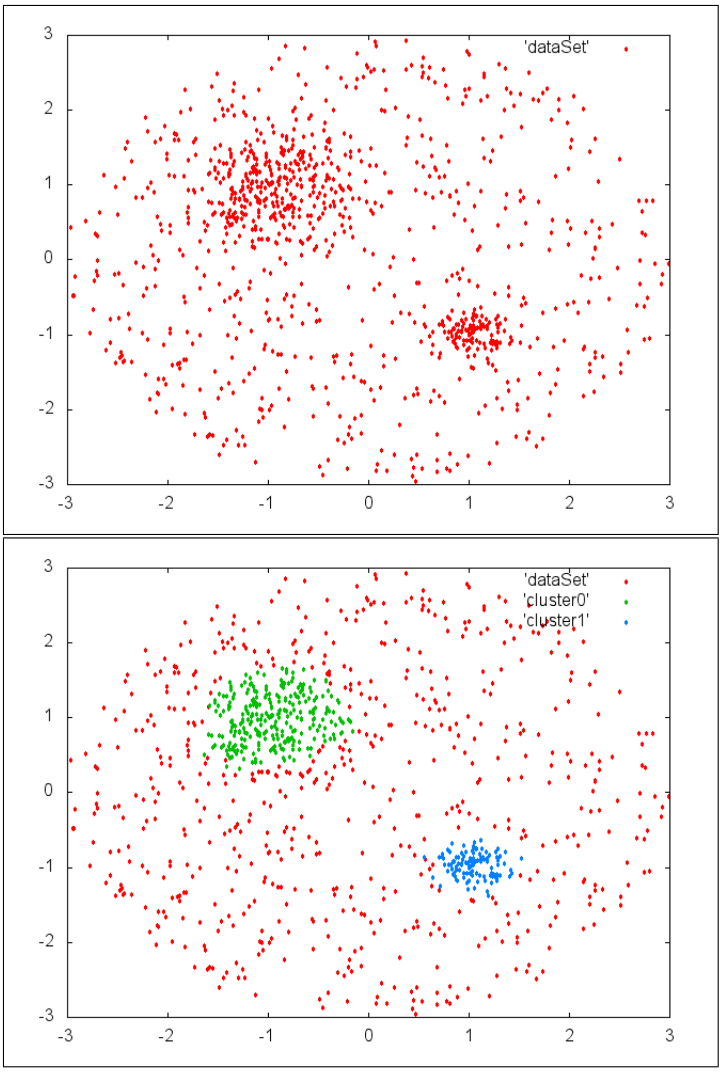

Figure 3.1: Visualization of the DBSCAN algorithm applied to Optics ordering of simple 2-dimensional data set which consists of 1000 points. Optics paramters: $minPts = 20, \epsilon = 1.2$, the two clusters was found using DBSCAN paramters: $\epsilon' = 0.2$

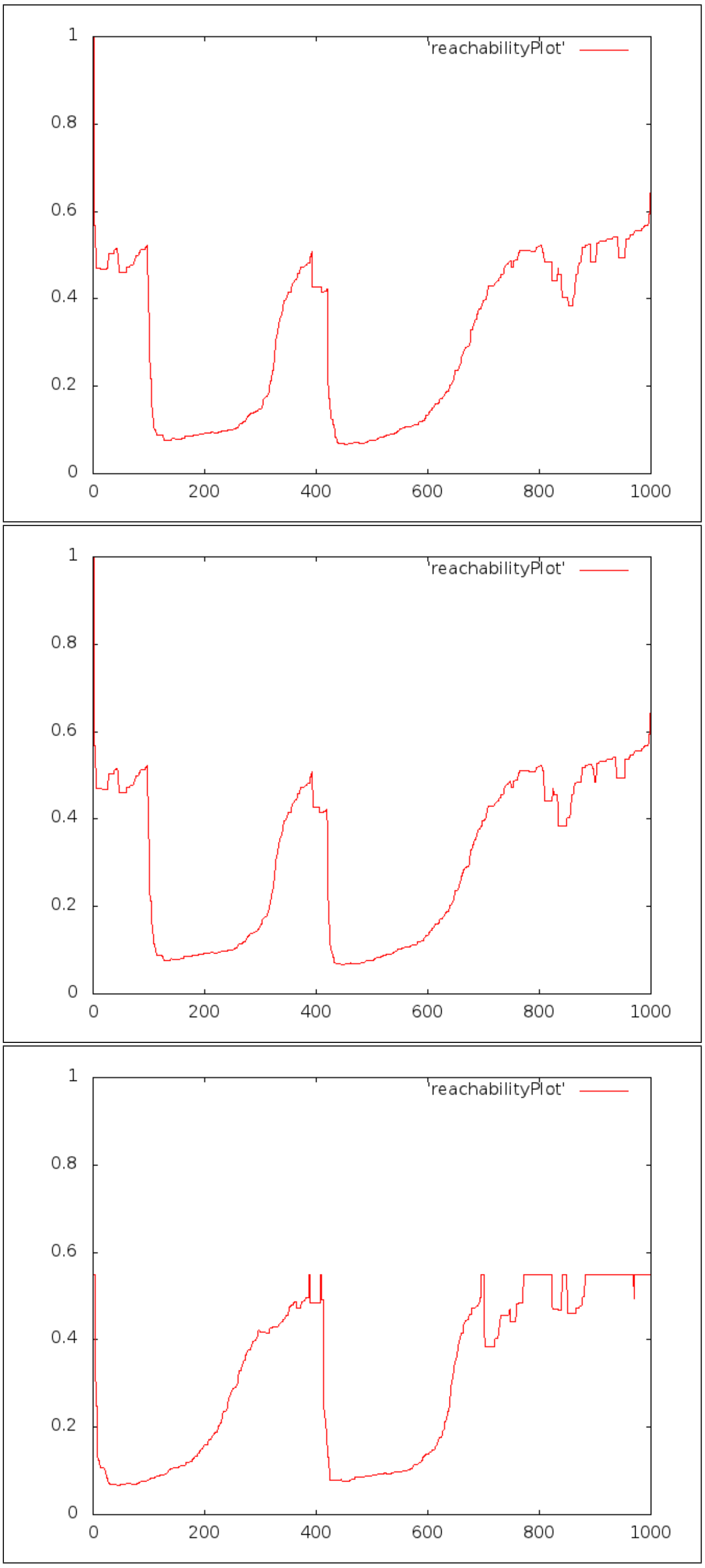

Figure 3.2: Reachability plot for data set presented on figure 3.1. Optics paramters (minPts, ϵ) are: (20, 1.5), (20, 1.0), (20, 0.5) respectively. The two cavities which are visible in each plot depict the two of the clusters on figure 3.1. This proves that there is a large range of values for ϵ for 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

TODO: description of the deterioration process, not very computationally intensive, easy to improve in subsequent runs, interpolation accuracy is not crucial, what is most important is to minimize the probability of finding the basins of attraction which was previously explored, use knowledge from clusteres;

4.1. Sequential niching

TODO: basic description of deterioration: when used, what approach, connection with clustering algorithms, advantages of OPTICS algorithm (improving deterioration by extracting clusters wiht different densities, cheapness)

4.1.1. Crunching functions

When degenerating a single basin of attraction represented by a cluster of individuals, we are looking for function with the following properties (TODO: why):

1. cheap (in multi-dimensional spaces)
2. easily adjustable to the shape of the basin of attraction
3. has low impact on the areas of fitness landscape which are distant from the cluster so that we do not introduce unnecesary noise to the fitness landscape in further iterations
4. symmetric

A class of functions which which are suitable for deterioration are so called *(*kernel functions) [16]. Examples of kernels are:

- Triangular $K(u) = (1 - |u|)\,\mathbf{1}_{\{|u| \leq 1\}}$
- Epanechnikov $K(u) = \frac{3}{4}(1 - u^2)\,\mathbf{1}_{\{|u| \leq 1\}}$
- Quartic $K(u) = \frac{15}{16}(1 - u^2)^2\,\mathbf{1}_{\{|u| \leq 1\}}$
- Gaussian $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$

From mentioned function only Gaussian kernel meets all requirements (remaining kernels are defined on some finite intervale, which makes them computationally inefficient in high-dimensional spaces).

4.2. Basic Scheme

The basic version of our fitness crunching algorithm is as follows:
For each cluster generate one or more multi-dimensional Gaussian function:

$$g(x) = -F_k(x_{max})exp(\frac{-1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)) \tag{4.1}$$

where F_k is a fitness function in kth iteration of the algorithm, Σ is an unbiased sample covariance matrix [15] estimated from the cluster population:

$$\Sigma = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)(x_i-\mu)^T \tag{4.2}$$

Fitness function in $k+1$th iteration is of the form:

$$F_{k+1} = F_k + \sum_{i=1}^{M} g_i \tag{4.3}$$

where M is the number of generated Gaussian functions.

Because of the fast convergence of the HGS subpopulation (and populations generated by other algorithms) to the local minimum, clusters sometimes becomes very dense in areas of local optimum, therefore Gaussians created for such clusters does not approximate a basin of attraction well, speaking informaly: Gaussian functions created for such clusters consist of high and thin peaks which deteriorate only the area inside the cluster, not the basin of attraction in which the cluster resides. To overcome this issue we developed so called *Covariance Matrix Adjustment (CMA)* algorithm described in section 4.4.

4.3. Adaptive Scheme

TODO: describe in detail
detailed description of adaptive scheme deterioration

4.4. Covariance Matrix Adjustment

We use sample covariance matrix as an estimator [15], which is extremely sensitive to outliers. However we may take this property as our advantage and incorporate it CMA algotithm. Having given a cluster of points the CMA algorithms works as follows:

4.5. Results

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

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

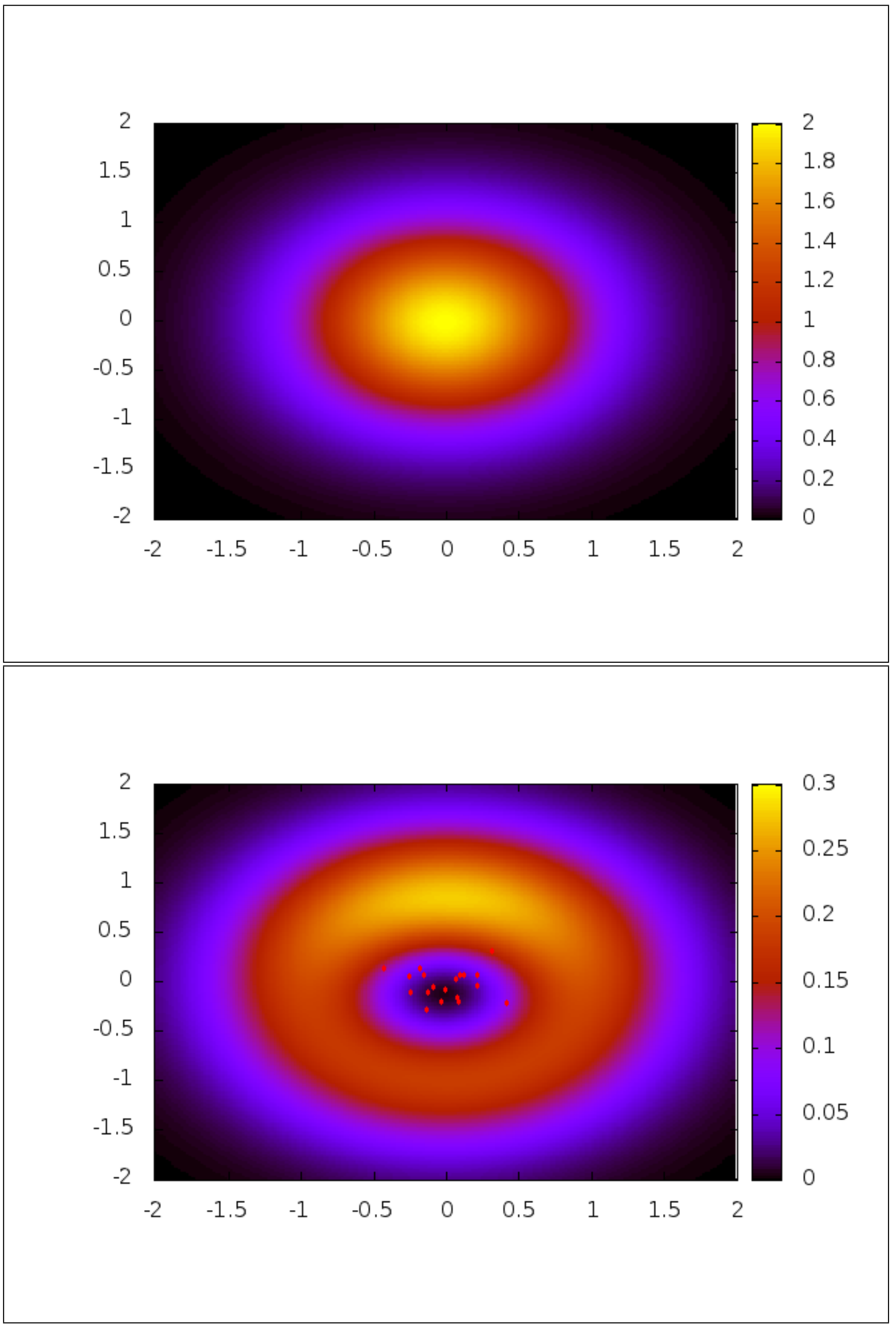

Figure 4.1: The result of basic deterioration scheme with CMA applied to unimodal function: $f(X) = 2e^{-(x^2+y^2)}$. Optics paramters: $minPts = 20, \epsilon = 0.4$, algorithm: SGA, iterationCount=1

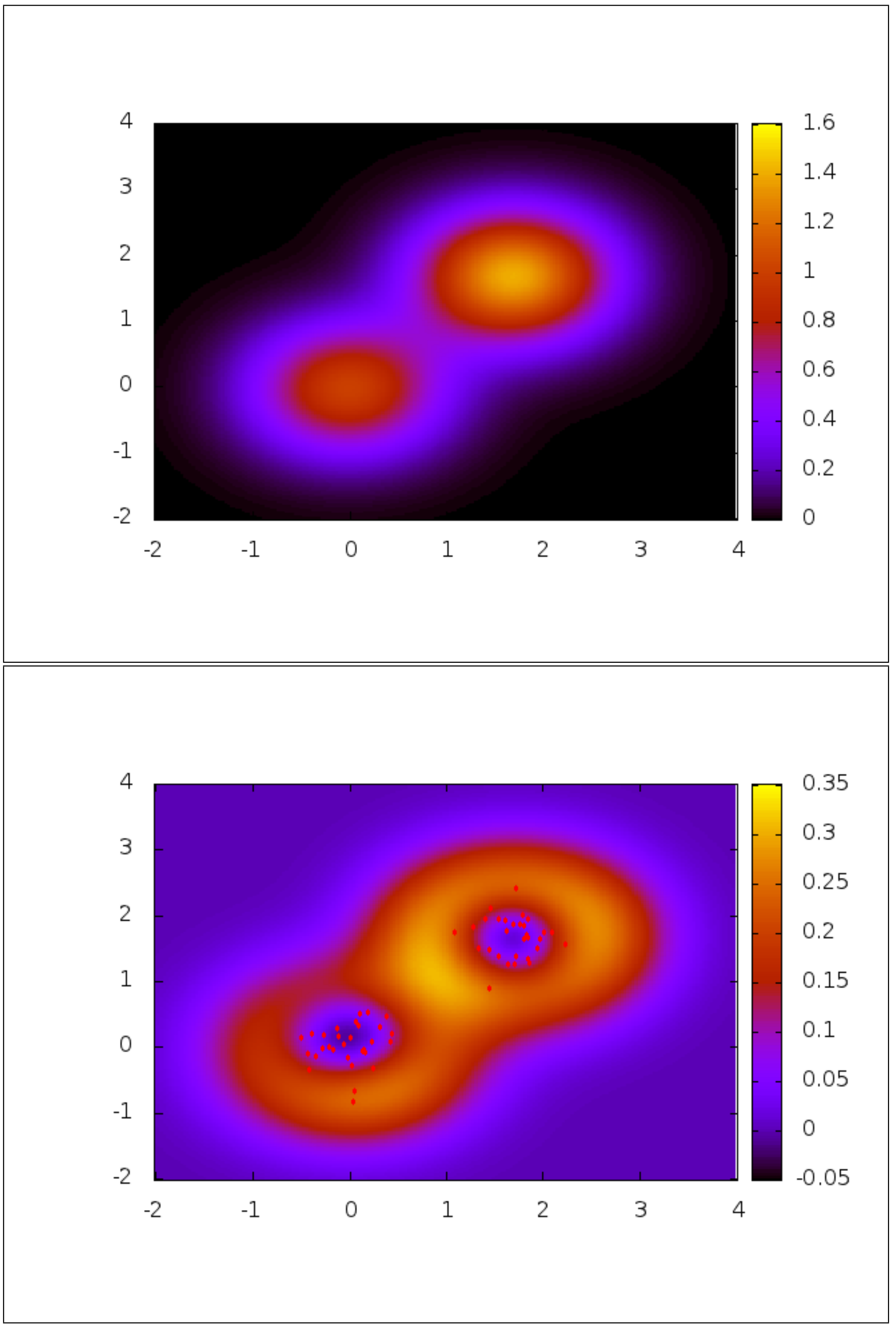

Figure 4.2: The result of basic deterioration scheme with CMA applied to bimodal function: $f(X) = e^{-(x^2+y^2)} + 1.4e^{-((x-1.7)^2+(y-1.7)^2)}$. Optics paramters: $minPts = 20, \epsilon = 0.4$, algorithm: SGA, iterationCount=2

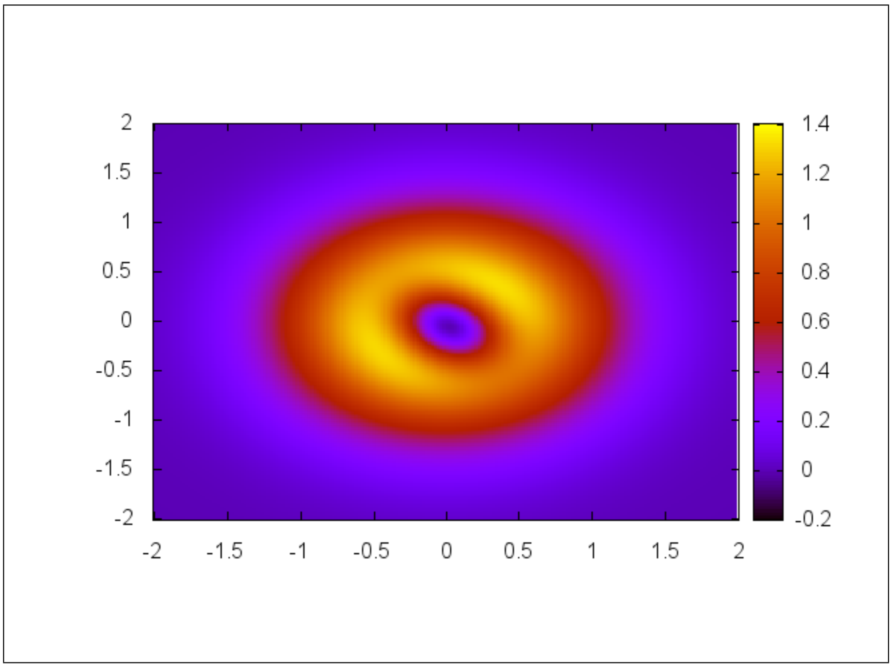

Figure 4.3: The result of basic deterioration scheme without CMA applied to unimodal function from 4.1. Optics paramters: $minPts = 20, \epsilon = 0.4$, algorithm: SGA, iterationCount=2. We may see that the overall landscape decreases only by 30 percent, while using CMA gives us 85 percent of decline.

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

Figures 4.3 and 4.4 shows how much the algorithm benefit from using the CMA algorithm. The plots show deterioration results applied to functions from 4.1 and 4.2 without the CMA algorithm.

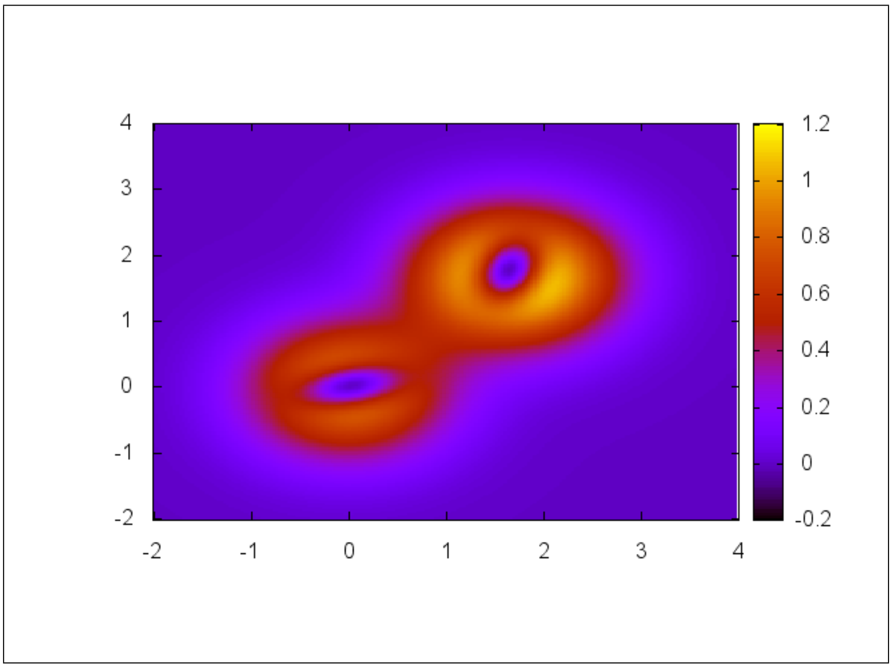

Figure 4.4: The result of basic deterioration scheme without CMA applied to bimodal function from 4.2. Optics paramters: $minPts = 20, \epsilon = 0.4$, algorithm: SGA, iterationCount=2. We see 25 percent of deterioration, while CMA gives us 78 percent when applied to the same case.

5. Tested Algorithms

5.1. HGS

why HGS is well suited to our algorithm (suitable for clustering, fast convergence in leaves)

5.2. Tests

5.2.1. Benchmark functions

uni, bi and multimodal functions

5.2.2. Accuracy measures

how many optimas have been found, diagrams

5.2.3. Efficiency measures

6. Implementation

6.1. Architecture

TODO: architecture details, sample of spring application context, description of a problem domain and fitness function

6.2. Implementation in Java

clean structure, good test coverage, modular architercture, extensible,

6.2.1. Technologies

– Spring [12] - application framework

– Maven [13] - project management and build automation

– Mockito [14] - testing framework

– JAMA [11] - linear algebra package

Spring, Maven, JUnit, Mockito, JAMA, TDD approach

6.2.2. Diagrams

class diagrams, sequence diagrams

Below you may find class diagrams for each of the module implemented in our framework. It shows a general overview of the structure of a system: used classes, their attributes, operations and the relationships between the classes.

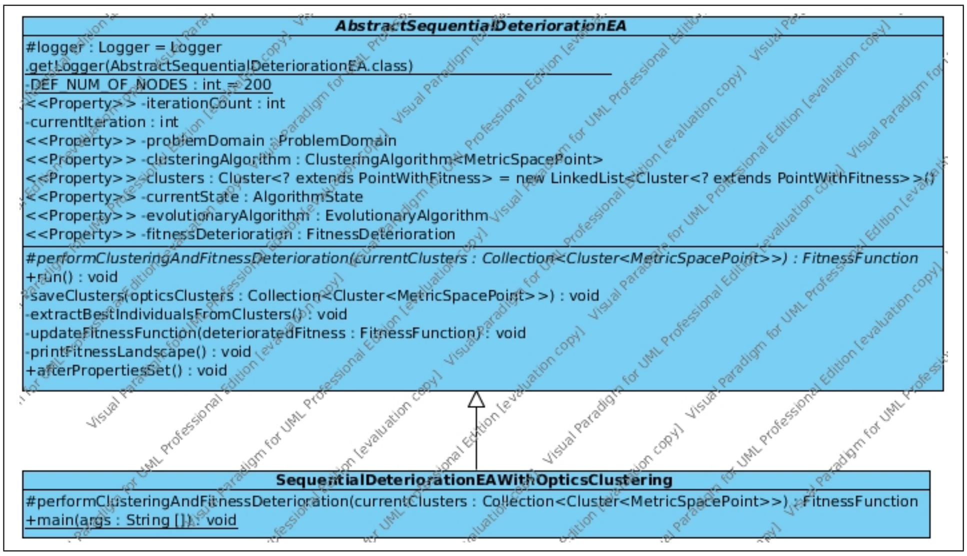

Figure 6.1: Main algorithm package

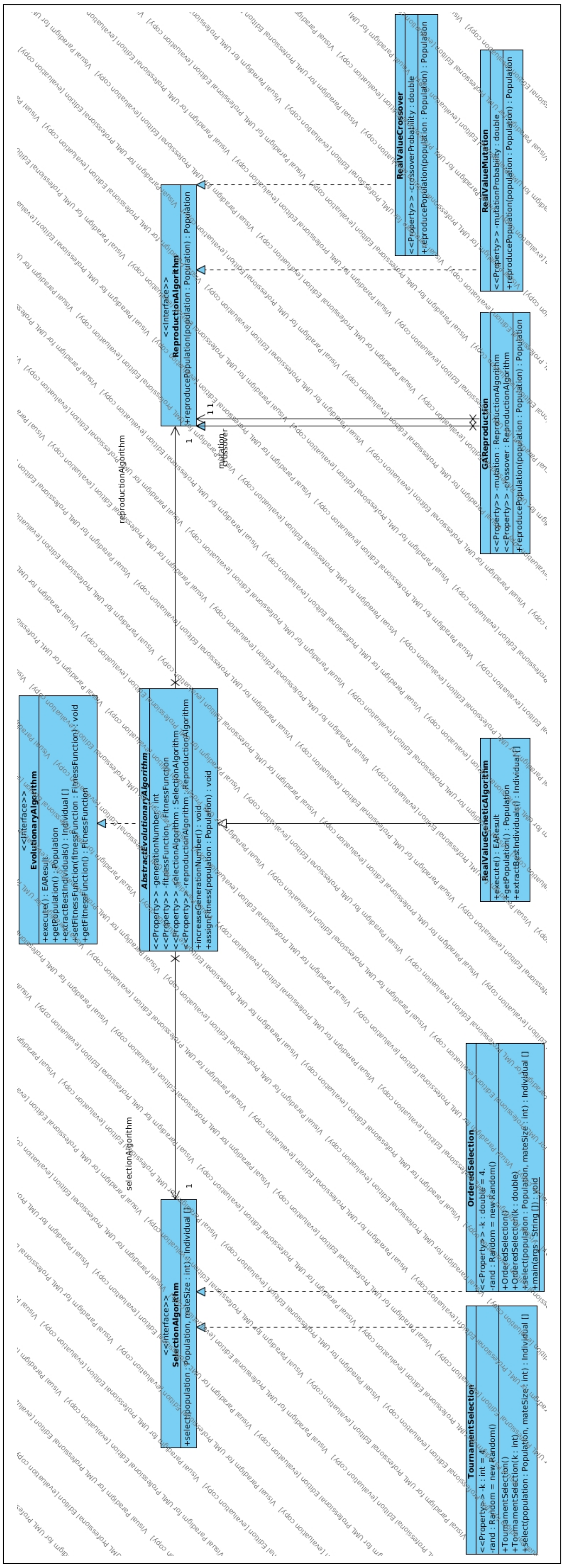

Figure 6.2: Evolutionary algorithms package

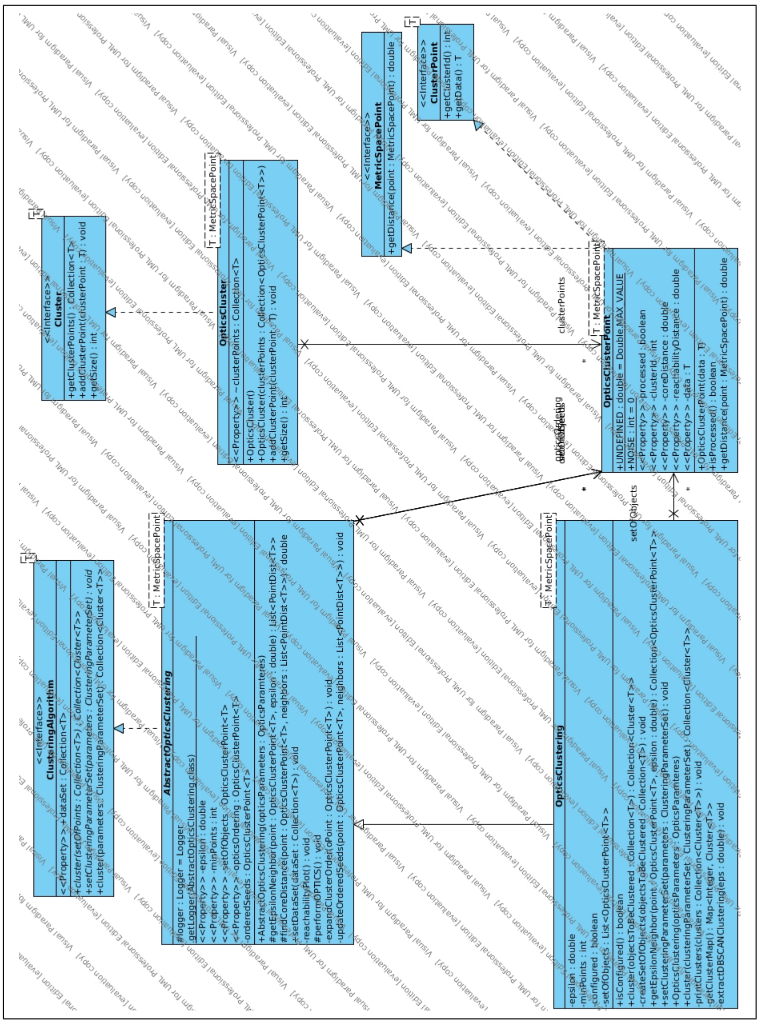

Figure 6.3: Clustering package

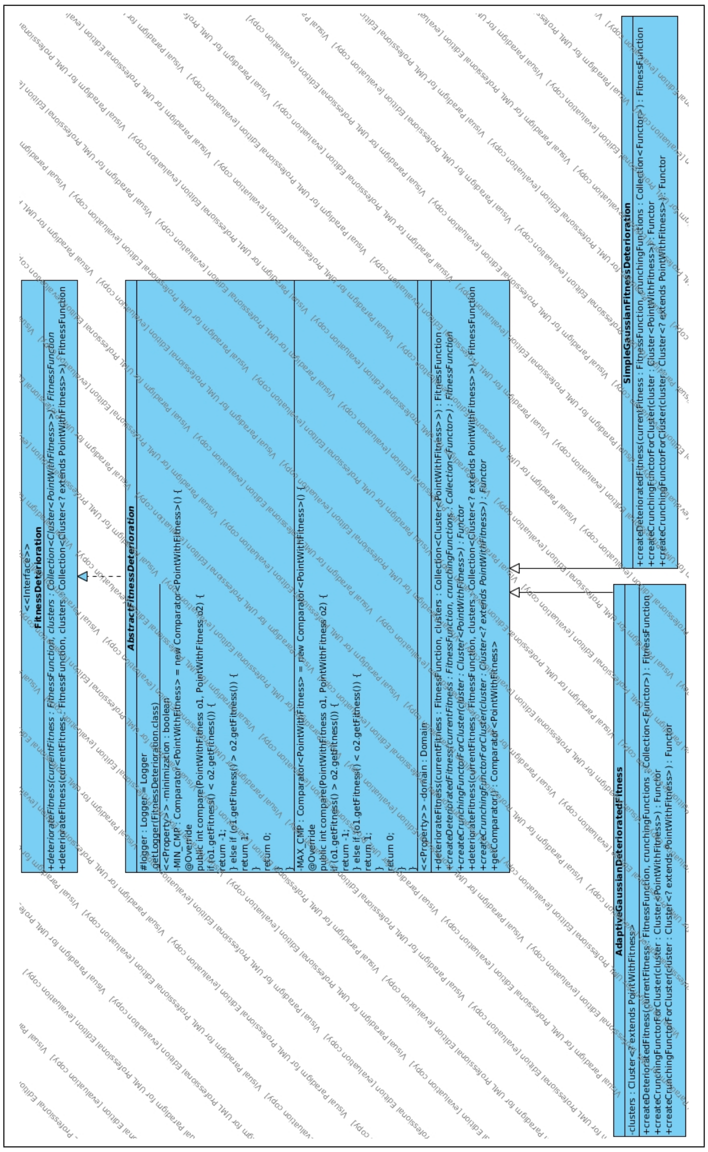

Figure 6.4: Fitness deterioration package

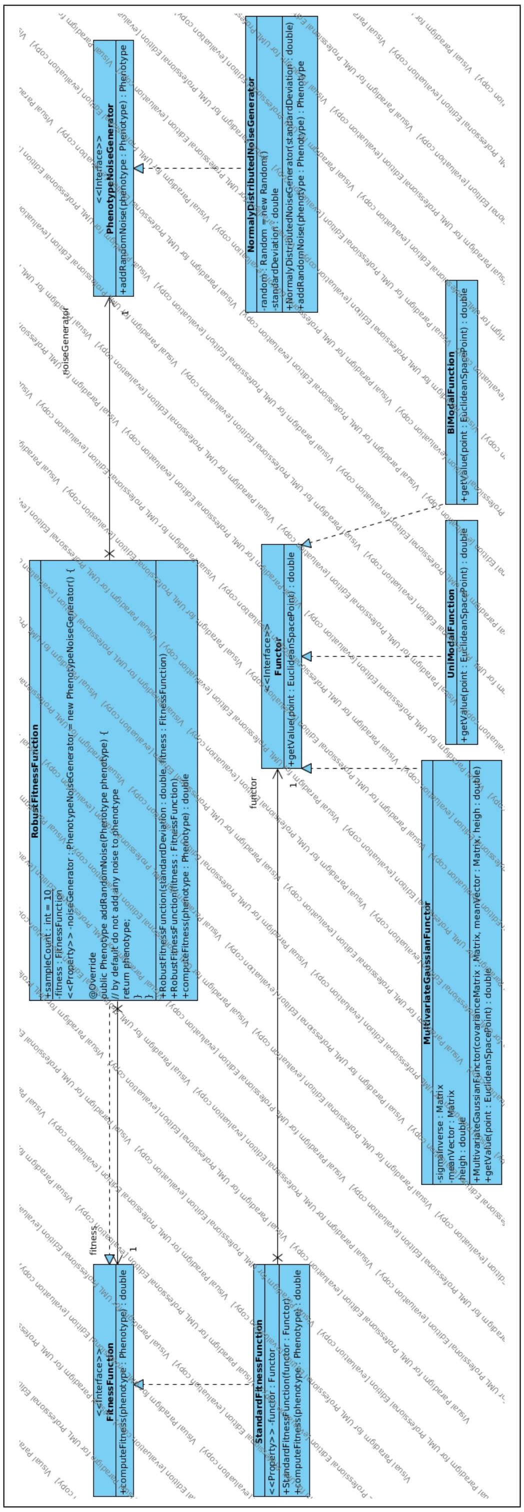

Figure 6.5: Fitness and functors package

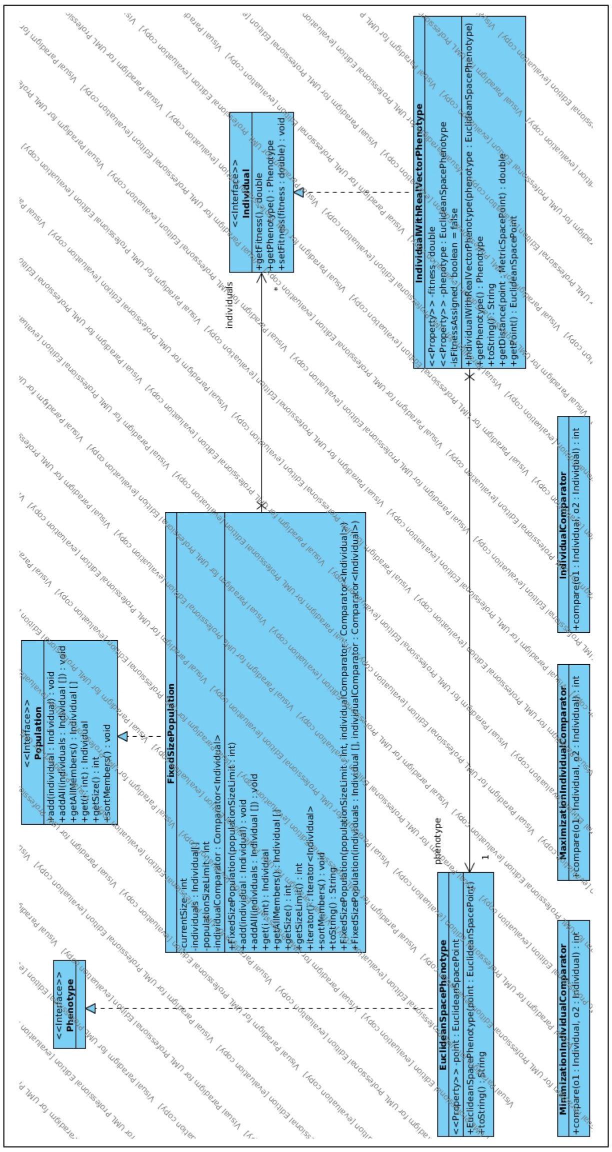

Figure 6.6: Population package

7. Conclusions

7.1. Summary

7.2. Future Research

Bibliography

[1] *Global Optimization Algorithms - Theory and Application* by Thomas Weise

[2] *OPTICS: Ordering Points To Identify the Clustering Structure* by Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jrg Sander

[3] *Evolutionary search with soft selection* by A. Obuchowicz

[4] *Foundations of global genetic optimization* by Robert Schaefer, Henryk Telega

[5] *Introduction to Data Mining* by Tan, Steinbach, Kumar

[6] *Foundations of modern probability* by Kallenberg

[7] *No Free Lunch Theorems for Optimization* by David Wolpert, William Macready

[8] *Niching Methods for Genetic Algorithms* by Samir W. Mahfound

[9] *Genetic algorithms with sharing for multimodal function optimization* by D. E. Goldberg and J. Richardson

[10] *Genetic algorithms with dynamic niche sharing for multimodal function optimization* by B. Miller and M. Shaw

[11] http://math.nist.gov/javanumerics/jama/

[12] http://www.springsource.org/

[13] http://maven.apache.org/

[14] http://mockito.googlecode.com/svn/tags/1.8.0/javadoc/org/mockito/Mockito.html

[15] http://en.wikipedia.org/wiki/Estimation_of_covariance_matrices

[16] http://en.wikipedia.org/wiki/Kernel_statistics

[17] *Hierarchical Genetic Strategy with real number encoding* by Robert Schaefer and Joanna Koodziej