1. 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'.

1.1. 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

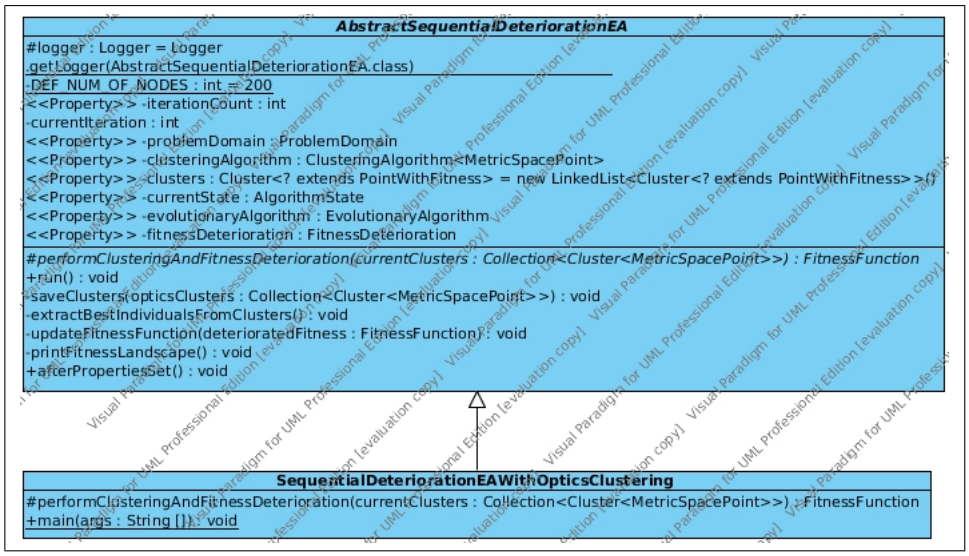

Figure 1.1: Main algorithm package

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

1.2. 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

1.2.1. Technologies

– Spring [**?**] - application framework

– Maven [**?**] - project management and build automation

– Mockito [**?**] - testing framework

– JAMA [**?**] - linear algebra package

1.2.2. 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.

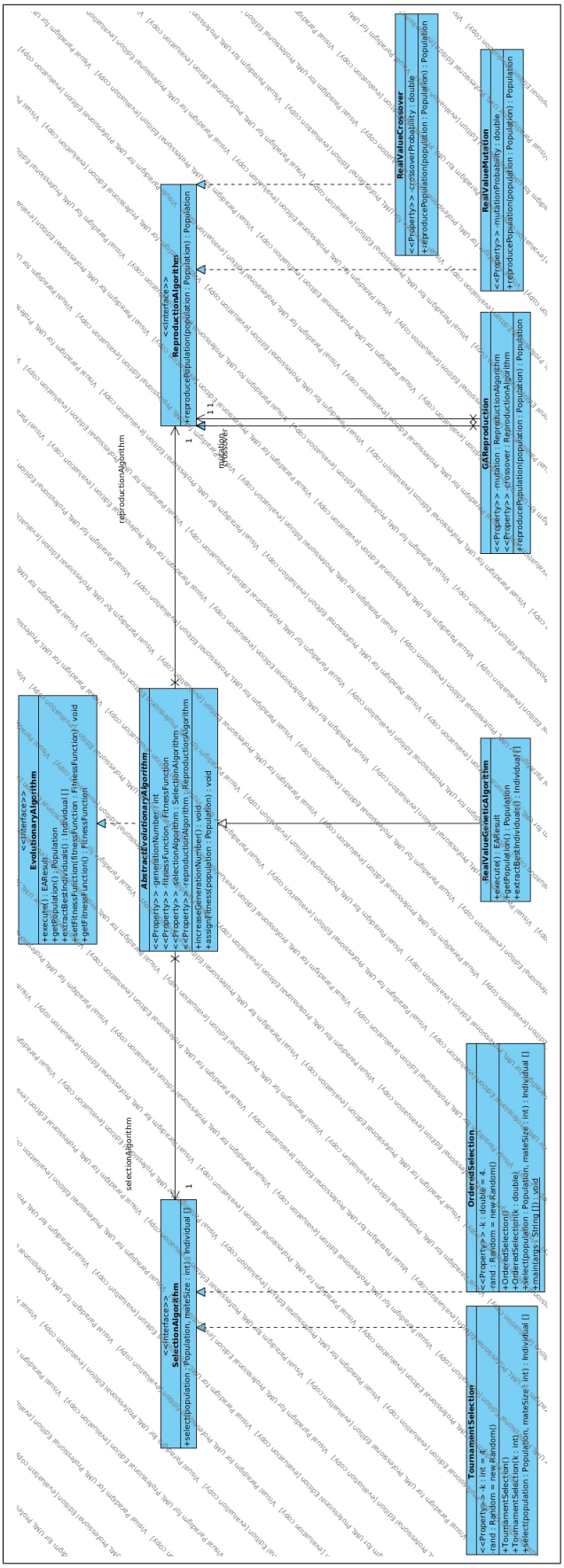

Figure 1.2: Evolutionary algorithms package

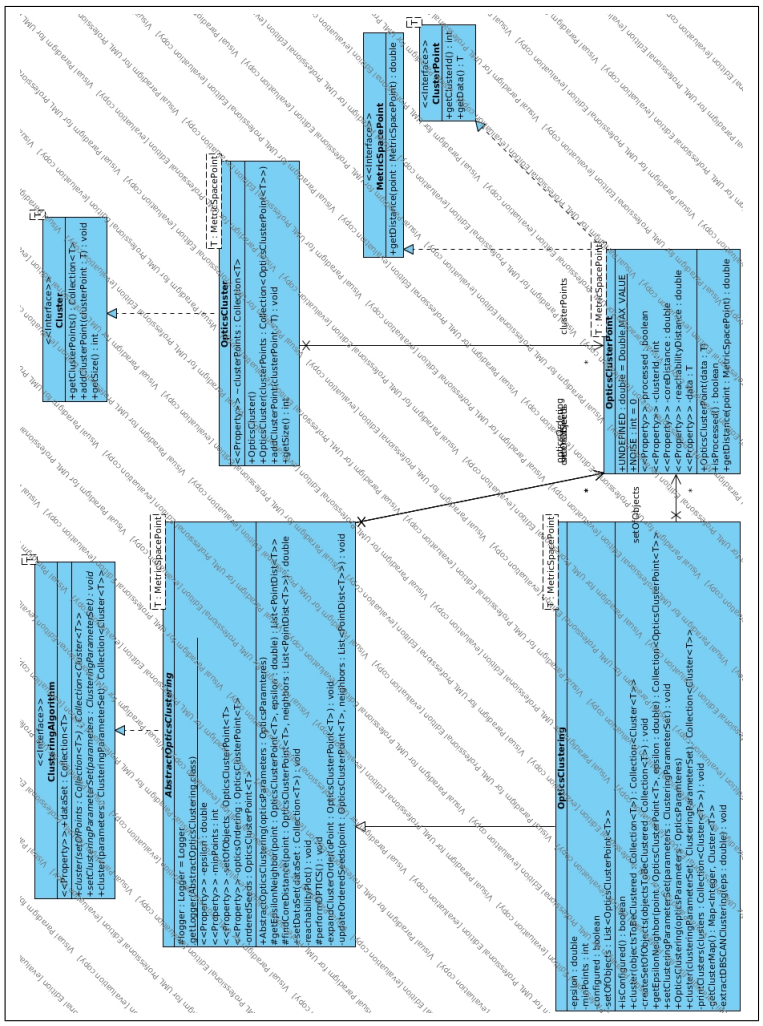

Figure 1.3: Clustering package

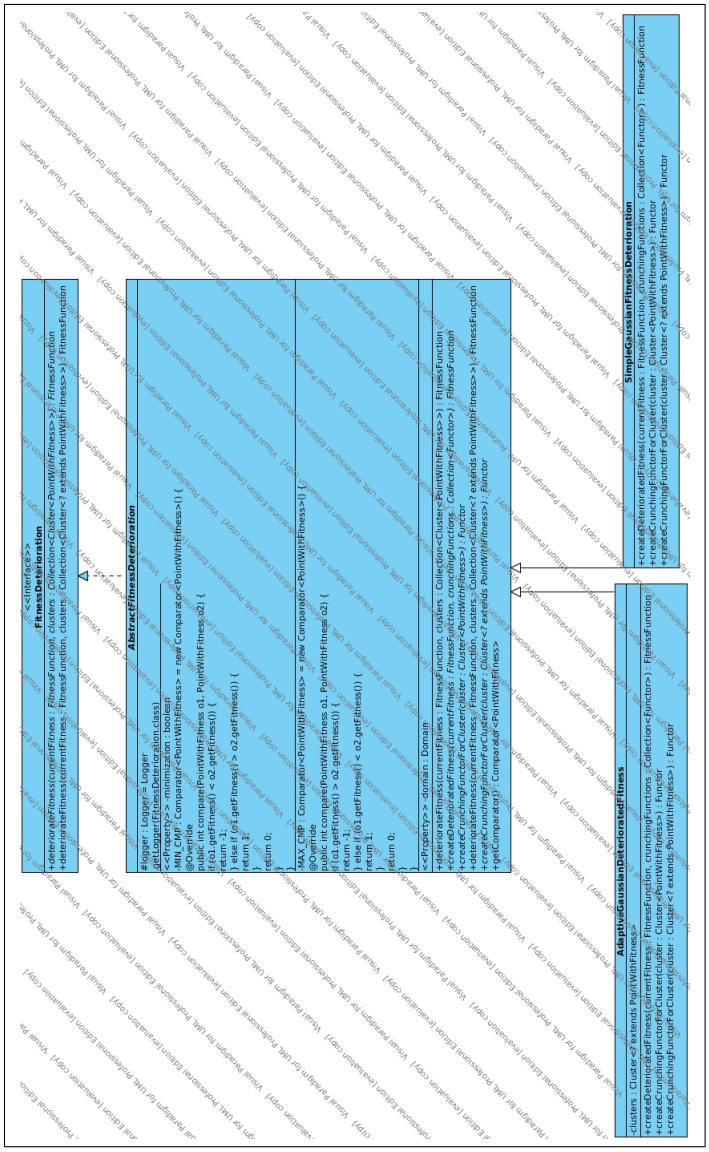

Figure 1.4: Fitness deterioration package

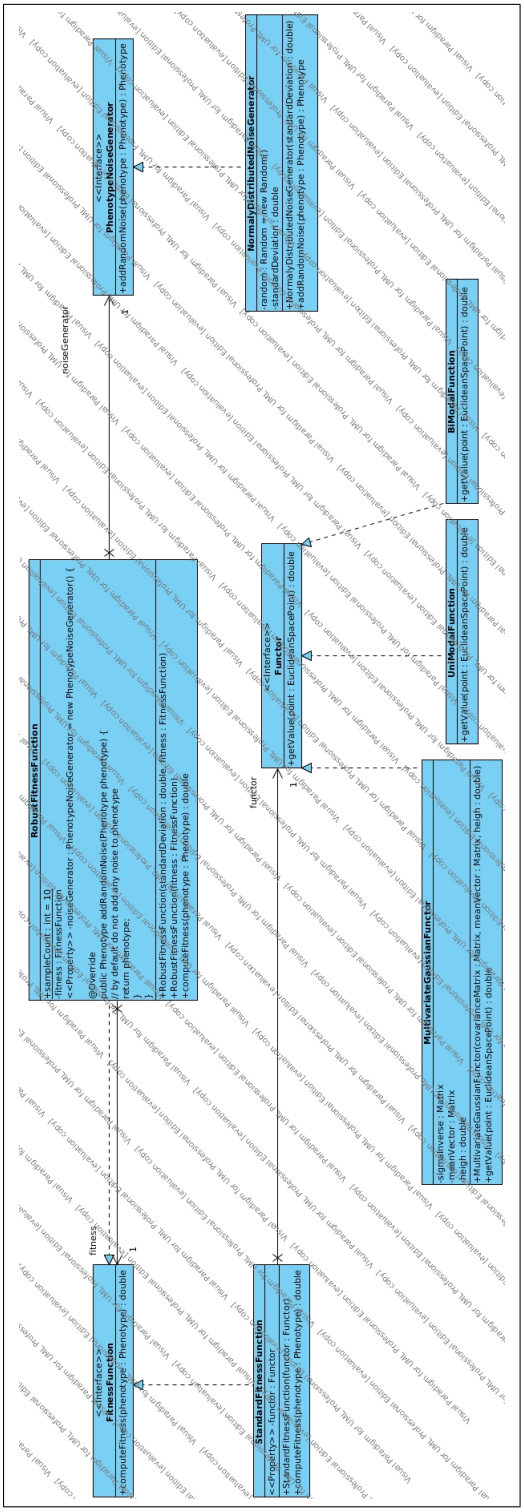

Figure 1.5: Fitness and functors package

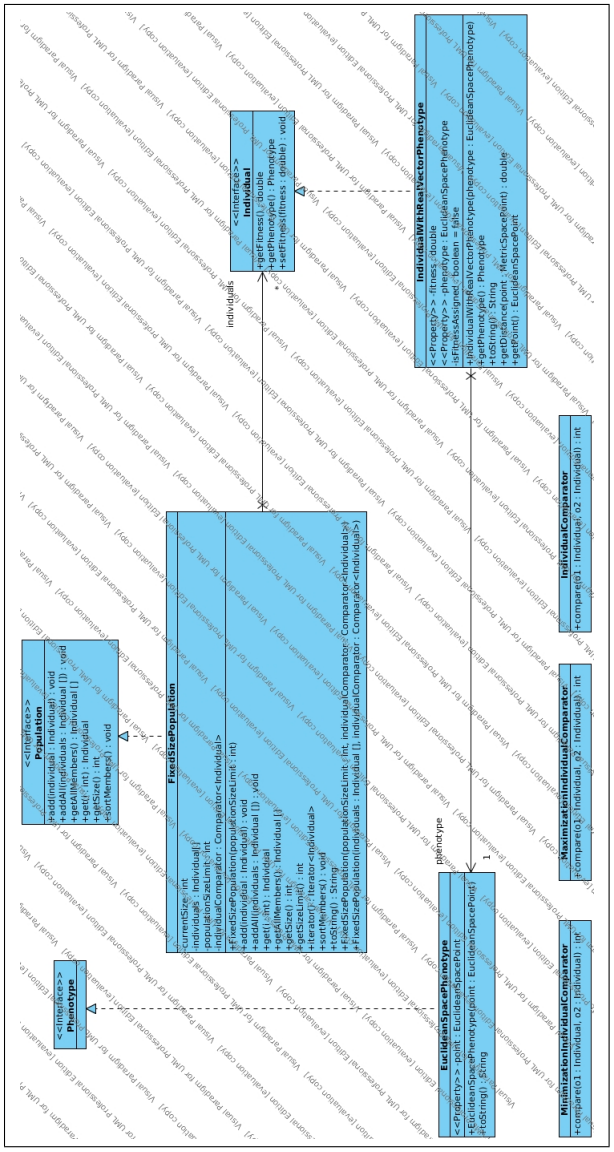

Figure 1.6: Population package