Generating of realistic polycrystalline structure

Gdansk University of Technology

MathMods Erasmus Mundus Joint Master Degree

Rustam Akhunov

July 5, 2018

Contents

Abstract

The three-dimensional polycrystalline models with the log-normal grain size distribution are constructed by constrained Voronoi tessellation. For achieving needed distribution of grain sizes and grain orientation I used Genetic Algorithm with Least Square. GA is also used for achieving realistic distribution for grain orientation distribution. Molecular-dynamics method is used in LAMMPS software to obtain the relaxed polycrystal. For the relaxation process and for simulation multibody potential is used.

0.1 Introduction

There are known a lot of modern application of polycristalline materials. Polycristallines are used for tritium generation [1], automotive, healthcare and non-destructive testing [2], solar cells [3], semiconducting elements [4] etc. With increasing demand of new properties of new materials increases also need for the modern methods of polycristalline simulation [7]. And this work is one more attempt to deal with the simulation of polycryrstallines in molecular dynamics environment such as LAMMPS [5, 6]

There are numbers of experiments and theoretical research for the intrinsic structure, effective thermal conductivity, lattice dynamical, optical and thermodynamic properties of polycristalline [7]. But still many characteristics have not been researched sufficiently.

We know few methods for generating digital nanostructured materials for use in the simulation: PoissonVoronoi tessellation (PVT) [20], Monodispersive grain size (MGS) [10], LaguerreVoronoi tessellation (LVT) [21], JohnsonMehl (JM) model [22] etc. In this work I used Voronoi tesselation model for the reason of its good solution for the complex sctructural nanocrystals and also convinient interface for the genetic algorithm, that I used during our polycristalline generation. It is also generally accepted that many single-phase fully dense nanocrystals are described by a log-normal grain size distribution. And I took this distribution as a reference for polycristalline generation.

The goal of this work is implementation of general methodology of generation realistic polycrystalline structure. It should correspond grain sizes and orienation distribution, have minimal energy and be relevant to experimental density, lattice constant etc. The methodology also should be fast enough in order to do generation in one or several days on adequate resources. This all I am trying to do using existing results for different polycrystallines.

0.2 Overview of existing methods and results

Before my research I did an overview of existing results in several related papers. One of the most important article for our research was "Constructing three-dimensional (3D) nanocrystalline models of Li_4SiO_4 for numerical modeling and simulation" [7].

In this paper authors considers problem of generation Li_4SiO_4 polycrys-

talline that is narrowing the result for particular matter. But I rather try to implement more general approach. Of course by making more general method I face much more problems with efficiency and time consuming of the generation method. But even general method can be fitted to the particular element.

The authors use genetic algorithm to reach particular distribution of grain sizes and as I can see they use only mutation operator without crossover. But I in contrary use both mutation and crossover. Moreover I implemented 3 variants of crossover and 3 variants of mutation in order to find optimal operators for our task. I implemented numerous computing experiments and chose optimal combination of operators and other algorithm parameters.

But it is necessary to pay tribute to the good results obtained during implementation of the algorithm. Authors were able to reach very good permormance. In particular the computation time per step was 3.7s with 6 processors and it is much better compared to the result 3.9s with 32 processors [9].

Another interesting paper dedicated to the problem is "Lattice models of polycrystalline microstructures: A quantitative approach" [11].

Here authors refer to available methods for creating polycrystalline structure. Among several different approaches they mention Voronoi tesselation as a good method for obtaining the polycrystalline structure. It is noticed that this method is very popular and wide spread in fitting to the experimental data and therefore presented in special commercial software such as AMIRA, MIMICS, SURFDRIVER.

Of course Voronoi tesselation method has disadvantages like that it represents only convex polyhedra that is not true for real polycrystals. It is also known that real grains are not necessary polyhedra at all. Nonetheless, Voronoi tesselation remains very quick and reliable method of generation structures close to the real polycrystalls.

In the paper it is also noticed that instead of focusing on precise microstructure it is much better focus on major statistics like distribution of grain size and grain orientation. It is actually what I am doing in my work.

However, in the paper the main goal is achieving the mechanically equivalent polycrystalline. Although this approach is good for mechanical properties of material it loses any other aspects of real matter. This is why I cannot apply current results in our research.

Another significant result in realistic generation of polycrystalline structure was achieved in "Mechanical properties of polycrystalline graphene based

2

on a realistic atomistic model" [14].

Here authors did great work in order to obtain realistic atomic structure of graphene. They implemented algorithm for atomic structure generation based on the growth principle. It means that after random location of grain centers they tried to simulate growth of grains by considering borders of growing grain step by step. They also followed misorientation distribution that have one to one correspondence with orientation distrubution. It is also emphasized the importance of realistic simulation of grain boundaries because of their significant impact on mechanical and electrical properties of material.

The main disadvantages in the work is that the result is valid only for graphene and graphene like structures. I will try to bypass this limitation in my work. There is also problem with algorithm used in the paper. This approach needs more computing resources and have some difficulties in parallelism. Usually such structure construction need days for obtaining usable results if the system has usual for simulation sizes. The real distribution of grain sizes is absolutely neglected in the paper. It is considered only uniform distribution for the grains that does not follow real structures. But nonetheless, I borowed some ideas for misorientation distribution in my work.

In following "Studying the elastic properties of nanocrystalline copper using a model of randomly packed uniform grains" [15] paper I found interesting new approach in studying mechanincal properties of nanoscaled polycrystalline materials. Authors offer concentrate on finding dependency of elastic module on grain boundaries and therefore they make artificial ideal polycrystall where they obtain uniform grain sizes that is not true for realistic structures.

The last very interesting paper to talk about that was considered during literature overview was "Generation of 3D polycrystalline microstructures with a conditioned Laguerre-Voronoi tessellation technique" [19].

Here authors got really singnificant results in obtaining very close to real size distribution polycrystalline structure. In order to do this they used extended Voronoi tessellation method so called Laguerre-Voronoi tessellation. They implemented procedures that allows to associate particles weights with the grains sizes. This method helps to obtain very realistic distribution of grain sizes and use it in the following simulation that they actually did. During simulation using this kind of tessallation they compared simulation results with real experiments with annealed aluminum. Obtained results show high efficiency of this approach and can be used in our work too. Thank

3

to developer of Voro++ library this tessellation is already implemented in the library and I can use Laguerre-Voronoi tesselation as first approximation in our genetic algorithm. It can significantly improve time of calculation.

Now from the papers about theoretical statements I move to the practical results that currently exist and can help to generate polycrystalline structure.

Firstly, I would like to mention very useful modern software so called nanoSculpt [16]. It allows to construct atomic structure inside any shape and form. It also supports constructing of Voronoi tessellation for polycrystalls in any shape. It was written using C language and quite easy extensible. But unfortunatelly, currently this package does not support following particular distribution of grain size and orientation during costruction of structure.

Another interesting tool is AtomEye [17] developed by MIT staffs. It is very comprehensive and powerfull tool for dealing with atomic structure. One of the most interesting util for me in this package is so called voronoirize tool. It can be used for building polycrystalline structures. The software is written in C language. But again problem it does not follow user defined distributions of sizes and orientations.

And the last tool that I would like to mention is Atomsk [18]. According to the website title of the utils it is the Swiss-army knife of atomic simulations. It supports many and many methods for atomic simulation. You can obtain numereous output format for any simulation programs you need. But I faced the same problem as in mentioned above tools. To generate polycrystalline structure with particular distributions you need to define orientation and size for each grain by hand. It is not convinient in particular with big sized simulation boxes. Moreover this util is written in Fortran language that is not very convinient tool for wide support and extension. Particularly in industry field.

As you can see there are many researchs and tool dedicated to the problem of generating of polycrystalline structures. But there are several problems that you can face while using them. Firstly they do not support generation based on user defined distribution that makes these tools very hard to use in realistic conditions. Secondly, tools and methods which support realistic structures work on very slow algorithm that needs days for obtaining trustful results. Thirdly, sometimes tools are written in "not good" languages and are hard to maintain.

These all reasons inspired me to create useful tool for generating realistic polycrystalline structure written in C++ language and backed by LAMMPS simulation software.

0.3 Building the Voronoi tessellation

As it was mentioned above I use Voronoi tessellation for building polycrystalline structure. But for obtaining real stucture I have to fit some properties of natural crystalls during generation of tessellation. In this work I am especially interested in fitting tessellation to the two crucial distributions: grain size distribution and grain orientation distribution. All generation done in periodic condition of the boundary.

0.3.1 Grain size distribution

Grain size distribution depends on many characteristics of polycrystal growth (velocity of crystallization, extra magnetic field etc) and environmental factors (temperature, pressure etc). It is widly beleived that in general any grain size distribution in polycrystals can be fitted to the lognormal function [7, 27]:

$$f(x) = \frac{1}{x} \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(ln(x) - \mu)^2}{2\sigma^2}) \tag{1}$$

As one can see there are only 2 parameters that can influence distribution shape σ and μ. Environmental characteristics can change these 2 parameters and impact on grain size distribution. Of course, there are plenty other types of grain size distribution like bimodal lognormal distribution or uniform distribution. But in general, type of distribution does not affect to the research made in the current work and this is why just simple lognormal distribution is considered. Moreover 1 is used as a fitting function for the grain size distribution in overwhelming majority of scientific papers regarding polycrystalline structures [7, 27].

The reason why one can be interested in the grain size distribution is that this polycrystal matter description have direct effect to the mechanical and electromagnetic properties of the polycrystalline structure. And very often just knowledge of an average grain size cannot give us specific parameters of the material. In research about effect of grain size distribution on mechanical properties [8] one can find dependency of yielding on the grain size distribution. In the paper annealed aluminum is considered. Depending on annealing temperature and lubrication authors obtained different grain size distributions and therefore different stress-strain dependency hence different yielding, though average grain size is the same for all specimen.

Grain size distribution can be expressed as a dependency of appearing frequency of the grain size. But grain size can be interpreted in many different values. In paper one can find grain size as volume of grain, area of the surface of the grain, simple area in case of films (2D nano-objects) or grain diameter (sometimes radius). In this work diameter of a grain is chosen as a meause for the distribution. Moreover to make all evolutions dimensionless normalized grain size is used. Normalizing is made by dividing grain diameter with average grain diameter across all grains of the material.

As an aim for the given work is to build some structure close to the given distribution (unimodal lognormal in this case) one need to become familiar with popular methods of achieving some particular distributions.

Monodispersive grain size model **??** implies attempt to simulate polycrystall by identically-shaped rhombohedral grains. The main purpose of such simulation just research on some boundary effects. Of course such a model does not reflex the real structure.

PoissonVoronoi tessellation **??** and LaguerreVoronoi tessellation **??** models are improvment for the classical Voronoi tessellation. These methods allow to fit some given distribution directly to the tessellation generation process. The methods are extremely fast compared to genetic algorithm described below. There is a problem, one obtains distribution similar to the reference distribution but the distribution not close enough to be able to say about dependency of particular distribution and material properties. However these approaches can be used as an extremely good first approximation for genetic algorithm.

Classical Voronoi tessellation backed by genetic algorithm comprehensive and relatively modern method for polycrystalline structure generation. It is used here in work and will be discussed in detail below.

Because of simplicity of implementation, low resource demanding, relatively fast computation and a few other advantages of the genetic algorithm method it was chosen as the only method for the generation not only grain size distribution but also orientation distribution and first level of relaxation of the structure.

0.3.2 Genetic algorithm for the grain size distribution

There are several key elements and terms needed to define before implementation of genetic algorithm (GA). First of all when one refer to the GA it is needed to understand what is individual for the current task. In case of

current work one deals with the Voronoi tessellation. Voronoi tessellation, in its turn, deals with a set of 3D points in the space so called seeds or generators. Thus the object that holds these seeds will be considered as an individual for the task. Remember that individual represents a solution for the given problem and in case of current research the right individual will generate correct polycrystalline structure hence solve the given task.

The set of individuals is called population. The size of population is the number of individuals in the set. Population in some fixed time is called generation. The first generation influence strictly on the speed of GA's convergence. And not only speed but also problem of stucking in local minima also depends on initial population. Therefore strategy regarding forming initial generation can significantly impove GA. In this work uniform i.e. random generation of individuals is chosen. It helps to cover space of solutions independently and bypass local traps. Of course, the size of population is also important for good peformance. But this parameter will be chosen later based on calculation experiments.

Another crucial term of GA is genome. Genome consists of genes and genes reflects some properties of the individual. In this work each 3D seed is considered as gene and the whole set of seeds for given individual is a genome. So, all further manipulations in GA will be done on set of seeds.

In order to be able to manipulate genes one needs to encode them, convert from genotype to so called phenotype. It was chosen real number encoding for the seeds coordinates because it is natural and convinient representation of the 3D points. Each point is represented by set of 3 real numbers: x, y, z coordinate of corresponding 3D point. The whole genotype is kind of 2 dimensional array. First level of array has size of number of individuals. The inner level has size 3 - the number of coordinates. Thereby there is 3*N sized array representing an individual.

The fitness is characteristic of individual showing the approximation of indvidual's phenotype to the objective function. It is another important term that has to be defined. Instead of fitness calculation it might be calculated inverse value so called fitness penalty. It shows the same, how far or close the individual's phenotype to the defined function by taking some difference between expected and current values. In this work fitness penalty calculated by formula [7]:

$$W^2 = \frac{1}{N} \sum_{i=1}^{N} [P_r(d_i) - P(d_i)]^2 \tag{2}$$

where N - number of points where penalty is calculated, P_r is size-distribution of individual, P is expected size-distribution (fitting function), d_i is point where penalty is calculated.

The size-distribution of individual is constructed by dividing whole interval of possible grain sizes into equal-distant subintervals - bins. For each bin number of grains with size within this interval is calculated. This amount of grains is divided by total number of grains in individual to obtain normalized frequency. Hence, N is the number of bins on which the whole interval is divided.

The expected size-distribution in this study is lognormal distribution. Each point where value of distribution is calculated is the center of the bin.

The formula 2 was chosen for the reason of fast convergence of GA with such a penalty function compared to several other meausers [7]. Hence, convergence of GA in context of penalty function is tendency of penalty function's value to zero.

Before moving to the discussion about GA operators it is needed to mention about **breeding** and new population **selection** strategies that are used in this work.

Breeding is the process of generation of new population candidates. It can be done by many different ways but in this work crossover and mutation operators are used for this purpose. After obtaining initial population the cycle runs to obtain all further generations. In this cycle successively crossover and mutation are applied. This gives candidates for the new generation. New candidates are mixed in common pool with the current population. The new population is selected based on individuals' fitness penalty. Best individuals go to the next generation and the process repeats until **stop criteria** is reached.

Described breeding strategy was selected because of its simplicity and relatively ease of implementation. Furthermore it is classical scheme for the GA. Described selection strategy was chosen for the reason of guarantee of improvement of population from generation to generation.

As was mentioned, two classical operators are presented in this implementation of GA for size distribution. **Crossover** operator is an operator

8

allowing to mix up genomes of the individuals (parents) with the same analogy as crossing within species in nature. **Mutation** operator is an operator allowing to generate new genomes (bring new information to the population) by changing somehow some part of genomes in individual.

Crossover operator can be of different implementations but all implementations include two main elements: individual selection and crossover procedure. It worths to mention, that selection of individual for crossover and selection individuals for a new generation are absolutely different operations.

Classical selection for crossover roulette wheel selection is not good choice for GA in current situation. Roulette wheel selection is very noisy and the rate of evolution strictly depends on the variance of population fitnesses. Another selection is random selection it is even more disruptive than roulette wheel and therefore it is also not acceptable. Tournament selection is the best variant for the current task. Competetion between individuals and random selection of individuals for the competitions bring selection pressure, which drives GA to get better fitness for the next generation. Finally, tournament selection was chosen for the GA implementation.

Crossover procedure consists of two parts: cross site random selection and swapping genes between individuals following the cross site. Cross site selection might be one-, two- or multi-point. In this work it was used one- and two-points crossover because of it simplicity in implementation and good influence on the evolution rate. Uniform crossover is also implemented for diversity purpose and for ability to check how crossover operator impact on evolution rate and convergence of GA.

In case of Voronoi tessellation's seeds it is necessary to define how evalute cross site selection. The seeds are 3D points in the space and therefore one can determine cross site as a plane dividing individual's volume into two parts in case of one-point crossover. In case of two-point crossover one needs 2 cross section planes and so forth for multi-point crossover. One-point crossover is selected as an operator for the current study therefore it is needed to find only one cross section plane. But the problem that to find such a plane you need to take into account problem of dividing individual's volume to the parts with the same number of particles (seeds). When you divide 2 volumes with randomly located seeds there is not guarantee that your division will be successful and corresponding parts of 2 individuals will contain equal amount of seeds. Even when one will try to adjust the plane to make division useful (bring it to such a condition that respective parts of individuals have the same amount of particles) and change the angle and/or shift of the plane it
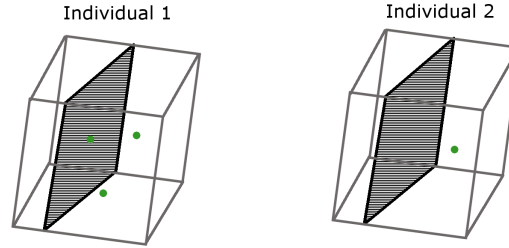
Figure 1: Cross section plane very rarely divide individuals' volume into parts with equal amount of seeds

is much more probable that one will fail in such refinement (see Figure 1).

To avoid problem with planes seed-to-seed mapping approach was invented. Seed-to-seed mapping is some assosiation between seeds of one individual with seeds of another individual. The association for this study is made based on distance between seeds of two individuals. The distance was calculated after overlaying two sets of seeds in one volume. It's calculated only between seeds from different individuals. Example of such association is on Figure 2.

After obtaining such a map between seeds it is easy to choose cross site for crossover. But now cross site is not a cross section plane but it is kind of free form surface that wrap some equal amount of seeds in two individuals. Equality keeps because of calculated mapping between seeds (see Figure 3). One needs just choose randomly initial point for crossover. Then it is necessary to choose some defined number neighboring seeds in one individual and interchange seeds between individuals according the mapping. Of course such a mapping is not unique because one can have cases with equal distances between one seeds from one individual and several seeds from another individual. This problem is solved just by selecting random neighboring seed with equal distance and if seed is already used in map it skiped and new nearest seed is searching (with exclusion of aready busy, paired seeds).

Having such a mapping between seeds it is possible to perform two-point crossover. It is needed just repeat procedure for one-point crossover two times on the same parents.

Uniform or random crossover is made by random choose of points fo the offspring either from one individual or from another one.

So far, the discussion about crossover operator was in condition that one
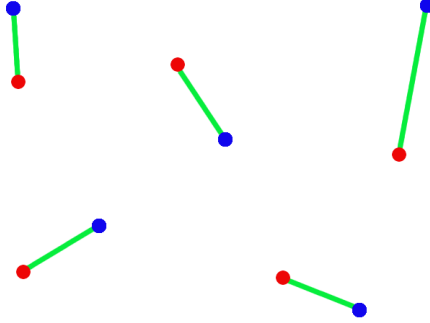
Figure 2: Mapping of two individuals' points. Blue colored dots are points of first individual and red ones are of the second individual. Pairs are made of points from first and second individual according to the shortest distance between them. Green lines show which points are chosen for pairing. Each case of pairs is made between closest points.

deals only with 2 individuals for each turn. But this operator can be not only for two individuals (parents) but also for three parents and even more parents. But this type of operators as many other variants like Precedence Preservative Crossover, Partially Matched Crossover etc., but all of them are more complicated in implementation and very hardly predictable. That managed choice for current implementation.

Mutation operator heps to bring new information to the generation. It works similarly to mutation in the nature. Individual has genes and mutation operator changes some of the genes and thereby help to bypass local minimas by extending search space. Number of genes that is changed due to mutation defined by mutation probability.

Mutation operator can flip bits if genome encoded by binary string. But in case of 3D seeds it is not applicable. Interganging of genes when one part of genome interganged with another part of the same genome is also not acceptable because it narrows diversity and can negatively affect on evolution rate. The same results come from reversing but it is valid only for bit encoding of genome. The best choice for current study is just slight change of coordinates of seeds. Character of coordinate's changing produce parameters that will help to adjust operator behavior to the given task. These param-
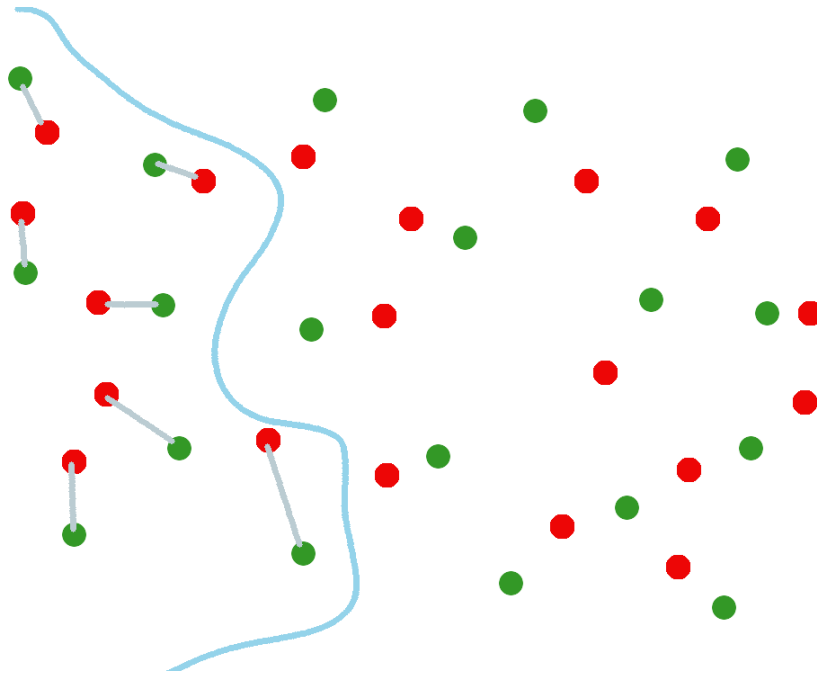
11

Figure 3: Free form volume used for crossover. One type colored dots are seeds from one individual, another type colored - from another individual. Light gray lines are pairing of seeds. Blue line is free form cross site.

eters will be discussed later. Another possibility to implement mutation is just reinitialization of seed. How many seeds will be reinitialized depends on mutation probability.

As was mentioned mutation operator was implemented as an operator that just shifts coordinates of seed with some given probability. It is important to define maximum shift value that can be performed during mutation. One can take some fixed value for the shifting. This approach has a disadvantage because one needs to change the fixed shifting any time when sizes of simulation volume change. Another approach to bind shifting to the dimensions of volume so it would be automaticaly adjusted to any volume. In this work adjustable approach is chosen. It is done by using average volume per seed. This value is pretty simple:

$$V_{average} = \frac{V}{N} \tag{3}$$

where V is volume, N is number of seeds.

Average volume can be used for calculating shifting by taking cubic root from its value, i.e.:

$$d = \sqrt[3]{V_{average}} \tag{4}$$

This value is auto-adjusting to the given volume and you have guarantee that you will not go far out of volume when you perform mutation. It is important because big amplitude of shifting can cause of not covergence of GA.

If one has all necessary elements for GA it is important what is **stop criteria** for GA. When it should stop its evoluations. There are several classical conditions for GA stop.

- **Maximum generations** - GA stops when specified number of generation was calculated. It is useful criteria but in this work modifed version of this criteria is used.

- **Maximum evoluation of fitness function** - GA stops when one calculated specified number of fitness function calculations. Fitness is calculated for each individual, so more individuals you have more fitness you calcualted and less runs (generation) you have.

- **Elapsed time** - GA ends when a specified time has elapsed. Note: usually If the maximum number of generation has been exceeded before the specified elapsed time limit, the process will end.

- **Penalty/Fitness threshold** - GA stops when penalty of fitness value is less or more than some specified threshold.

- And others criteria which were not considered here.

Maximum evoluation of fitness and Penalty threshold are two criteria which were used in current study to stop GA. The first criteria was chosen because of convinient obtaining results for comparing two or more configuration of GA. Comparing is easy when one can assess the same amount of calculations for different configuration of GA, other ways one will obtain biased results that do not reflect real situation. And the second parameter is more practical in sense of application of obtained structure to the simulation. There is some error in distribution that does not affect to the characteristic of material and there is no sense to continue to refine the result and run GA further. The penalty that is used for termination is taken for best individual.

The evolution process depends on many factors. The main parameters that influence convergence and convergence rate are **population size**, **crossover probability** and **mutation probability**.

Mutation probability parameter explained above, it is how ofter mutation will be applied to the offspring. Crossover probability is a parameter describing how often crossover will be executed. Because of offspring selection strategy in this study there is no mechanism to manage this parameter. Population size is amount of individual in the population. This parameter is very important and will be tested for the best value for algorithm evoluation. When you have big number of individuals you obtain best observe of solution space and therefore you have minimum probability to trap into local minima. But big amount means a lot of calculations, huge computational and time demands. Time depends on population size not lineary but in O(nlogn) manner and big size can leads to big time. In case of little size of population one can perform calculations very fast but easily trap into local minima **??**.

Besides these classical GA parameters each implementation can have many other adjustable parameters. Further discussion about parameters of current implementation will be presented.

As was mentioned in current study crossover probability has no direct entity and therefore it is not adjustable. But two rest parameters are separated and can be adjusted. Besides these two implemenation can highlight 10 more parameters. Some of them were tested and some of them were extracted and remained just for further study outside of the current work.

Crossover operator type - is different implementation of crossover operator. In this work there are one-, two-point operators and uniform op-

erator. It is needed to understand which of them is more aplicable for the current task.

Mutation operator type - is different implementation of mutation operator. In this work there are uniform distributed, normal distributed mutation and reinitialization. Uniform distributed mutation means that shift of coordinates selected in range (0, d), where d is from 4, uniformly. Normal distributed mutation means that shift selected from the same range but according to the normal distribution. And reinitialization mutation means that instead of shifting just total reinitialization of seed applies. It is also needed to understand which of them is more aplicable for the current task.

Factor for mutation amplitude - is a factor for 4. The purpose to understand how such a factor impact to GA. But in current study it is outside of scope and was remained just for further research.

Minimum and maximum number for crossover - is a maximum and minimum number of seeds that are used for crossover operation. The these numbers cannot exceed the total number of seeds in individual. And numbers cannot be negative.

Set of parameters for flexible GA - flexible GA is term intoduced by author to name process when mutation probability and population size can change during evoluation in order to improve convergence characteristics. This set includes so called **_thresholds_**. This is number of runs of GA after which mutation probability and population sizes are changed. For now algorithm has only 2 thresholds. After each threshold mutation probability changes by **_threshold mutation factor_** - multiplicative factor for probability. And population size changes by unique for each threshold **_threshold population factors_**. They are two because of two thresholds. The idea came from approach in paper "Constructing three-dimensional (3D) nanocrystalline models of Li4SiO4 for numerical modeling and simulation" **??**.

In order to find optimal set of parameters series of computation experiments are done. Configuration is a set of parameters for GA. So, aim is to find optimal configuration.

To obtain trustworthy results for each configuration 48 runs are done. 48 is not a random number. It was choosed due to specification of supercomputer that was used for all calculations. The supercomputer was kindly provided by Gdansk University of Technology.

Because there are 12 parameters in total it is difficult to optimize them all together. This is why decision of study parameters by pairs came to the

15

work.

The first pair of parameters is mutation operator type and crossover operator type. They were study in order to find best combination of operators. During study all other parameters are fixed. Because there are 3 types of mutation operator and 3 types of crossover operator to total number of configuration for this pair is 9. The population size is taken equal to 70, mutation probability - 0.01, shift factor - 1.0, from and to number for crossover - 10 and 100 respectively, thresholds are 500 and 1000, mutation factor - 10, threshold population factors 5 and 9. These are default parameters for all computations, i.e. if parameter is not in pair that is undergoing study then default value is used.

All computational experiments produce data for best, worst and average fitness penalty. Standard deviation for each fitness type is also calculated. Comparison for configuration is made based on best fitness penalty (lowest one). For the best combination in pair best, worst and average penalties are depicted.

The result for mutation operator type and crossover operator type pair is presented on Figure 4. First of all it is needed to explain meaning of the graphs' labels. Label encode the full parameter set i.e. configuration of GA. Each parameter separated by "_" sign. The order of set in label is fixed: crossover operator type, mutation operator type, mutation probability, population size, factor for mutation amplitude, "from" and "to" numbers for crossover, first and second threshold numbers, threshold factor for mutation, first and second threshold factors for population size. But labels on Figure Figure 4 are shortened to show only first 4 parameters. It is made because all other parameters are fixed and more convinient to show short labels.

As one can see on Figure 4 all lines with uniform crossover operator (last 3 lines in label list) have bad velocity of convergence in the very beginning and in the steady state they show worse results than other operators. It is a reason why uniform crossover operator has be declined for further consideration.

Before going further in rest operators consideration it is needed to mention that on Figure 5 one can see the same lines as on Figure 4 but drawn only as error bars. In the beginning of graphs one can see that all lines begin in with more or less the same point with considering error intervals. It proves that 48 runs is enough to get reliable statistic for current task. It is also worth to mention that error decreases with increase of iterations.

On Figure 6 it is possible to see behavior of lines from Figure 4 in detail in steady state. Now it is possible to notice that the best line is
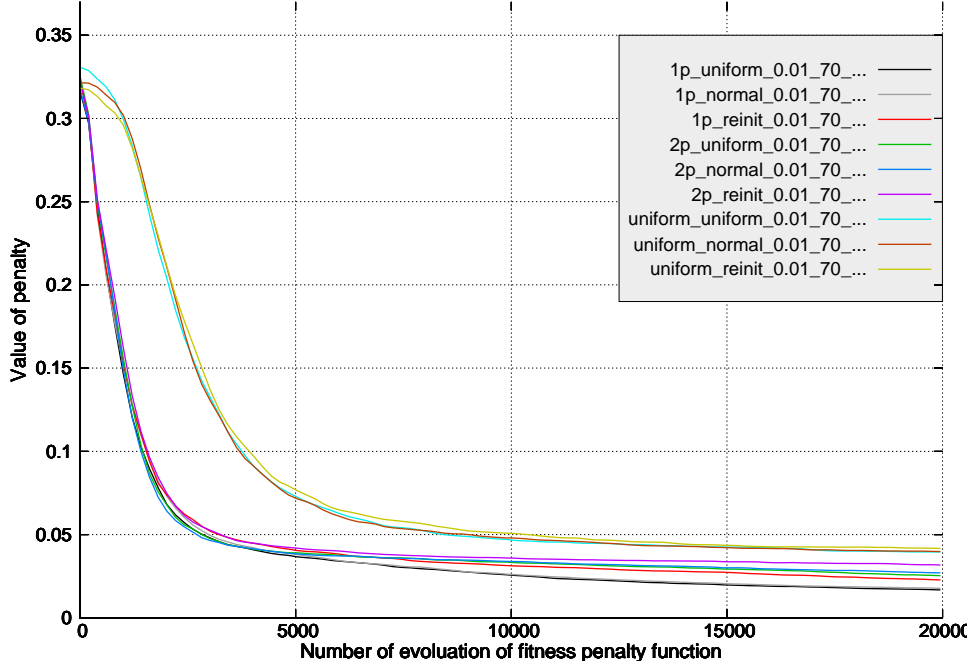
16

Figure 4: Comparison of crossover operators type with mutation operator type

"1p_uniform_0.01_70_..." though line "1p_normal_0.01_70_..." is comparable. It might be, one can adjust parameters of normal distribution to obtain better results but for our research it is enough to use uniform mutation operator for simplicity reason.

The next pair of parameters is mutation probability and population size. This pair is much more complicated because now there are no discrete values for parameters and one needs to find optimal value for continuous space. In frame of current study the search of optimal solution will be quite primitive and more complitated methods remained for further research. The search strategy will be done in following manner. For each parameter of pair some discrete values are chosen. For each value among chosen discrete parameters computations are performed. From obtained data fitting function is founded and all other comparison of parameters made based on fitted function.

Just for visual comparison Figure 7, Figure 8, Figure 9 show dependency of mutation probability on population size. Figure 7 show fixed mutation probability equal to 0.01 and different population size, Figure 8 - mutation
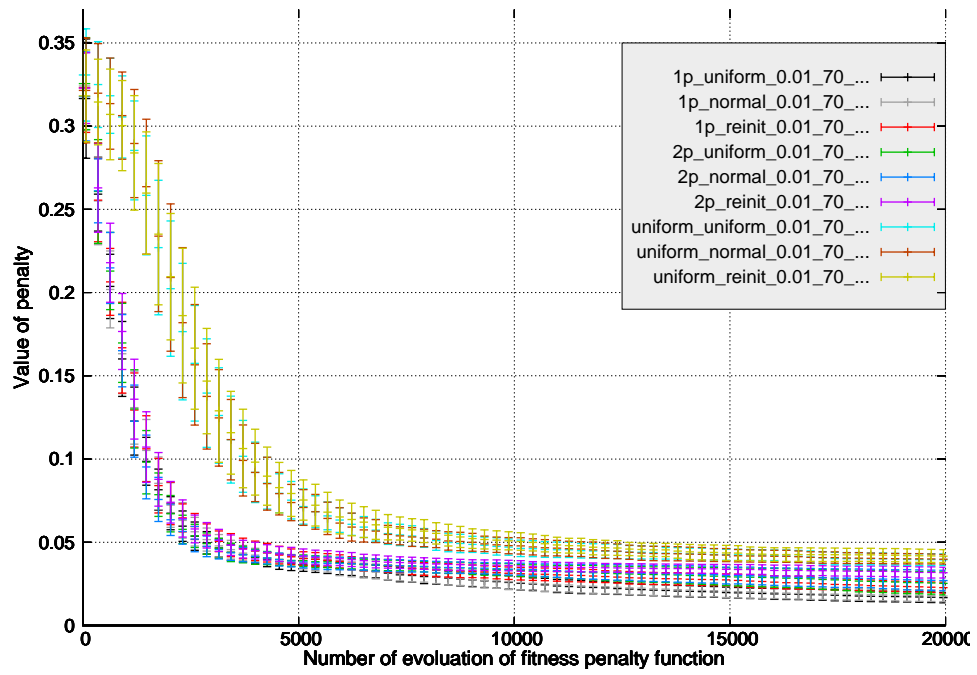
Figure 5: Comparison of crossover operators type with mutation operator type, standard deviation.
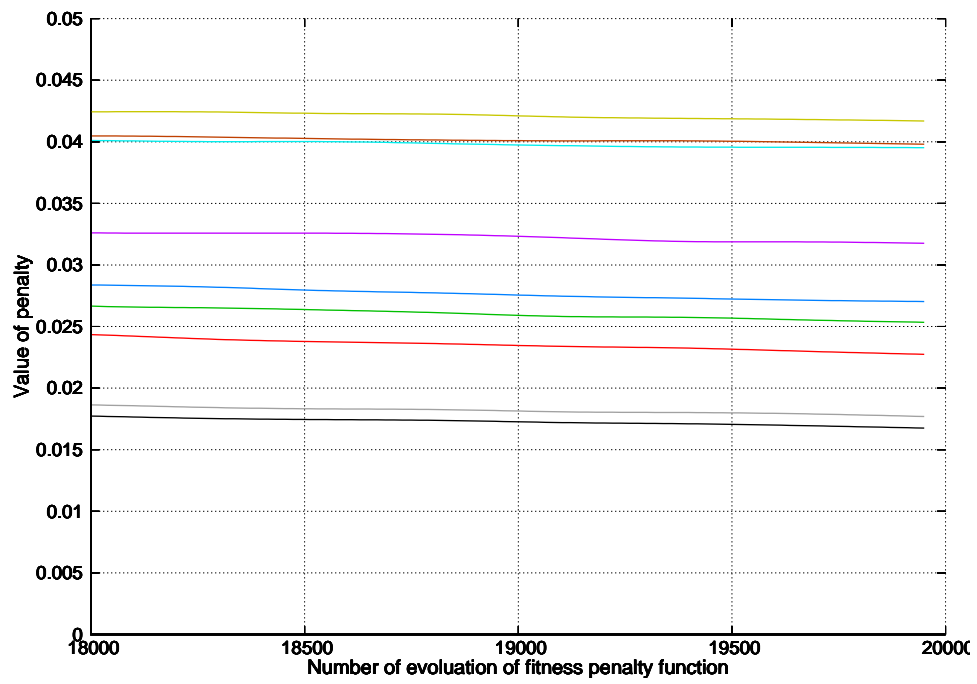
Figure 6: Comparison of crossover operators type with mutation operator type, zoomed to the end interval of graph.
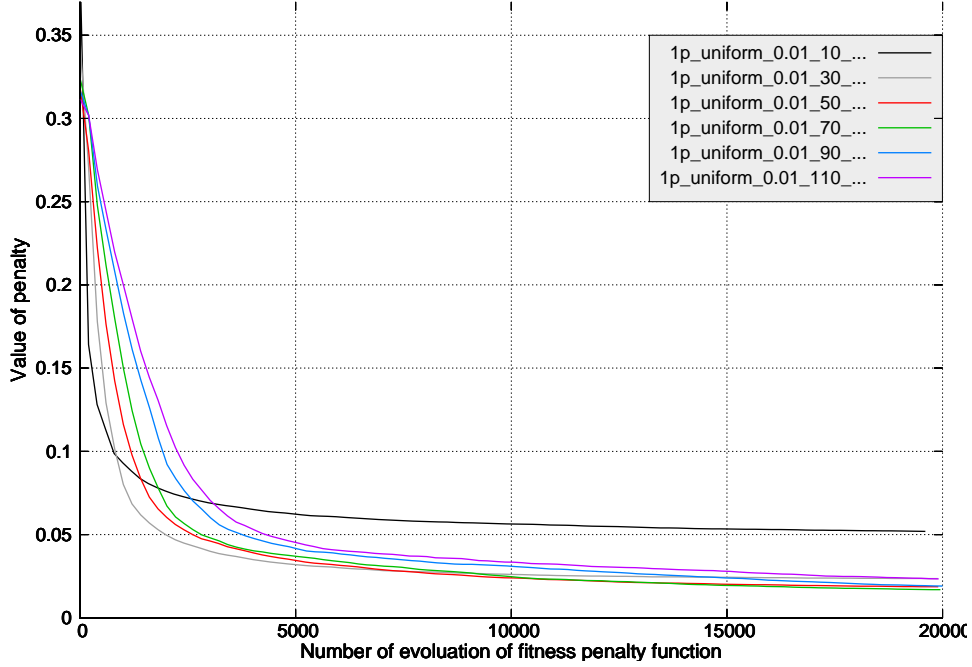
Figure 7: Comparison of fixed mutation probability 0.01 and different population size

probability 0.1, Figure 9 - mutation probability 0.8. It is only part of studied data and just for visualization of dependency.

For quantative comparison GA with different values of mutation probabilty and population size and finding of optimal combintation it is needed to fit data to some function. After series of experiments it was decided to divide data to 2 parts and fit these 2 parts with different functions. First part is in iterval [0, 3500] and the second part from 3500 till the end of data. For the first interval the fitting function is:

$$sig_1(x) = \frac{a}{b + x^d} + c \tag{5}$$

And for the second interval is:

$$sig_2(x) = \frac{a}{b + x} + c \tag{6}$$

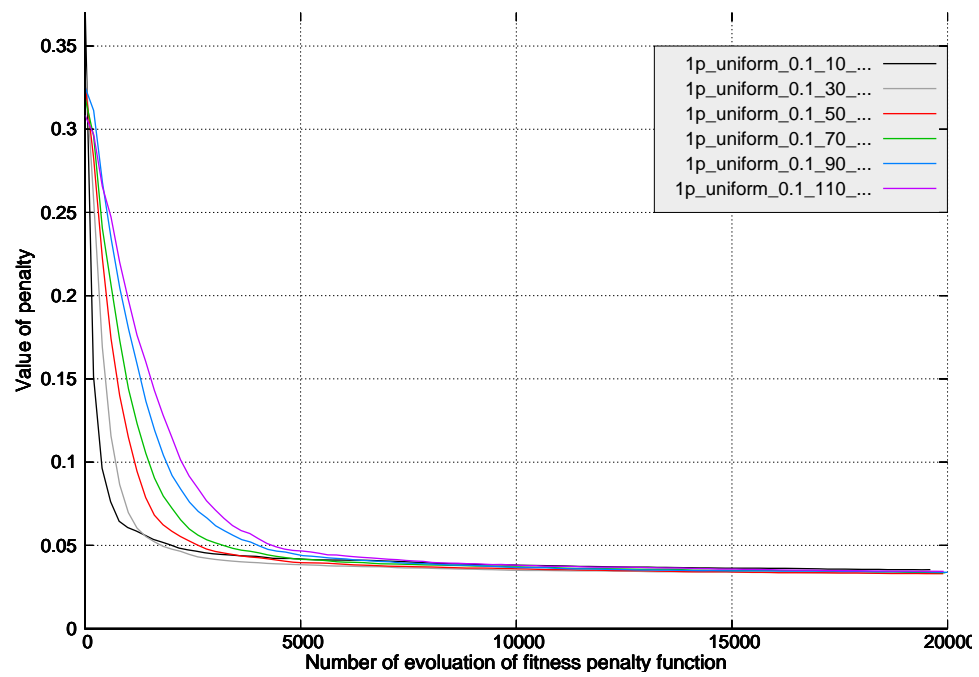So, the task is to find a, b, c, d parameters for the above functions. The

20

Figure 8: Comparison of fixed mutation probability 0.1 and different population size
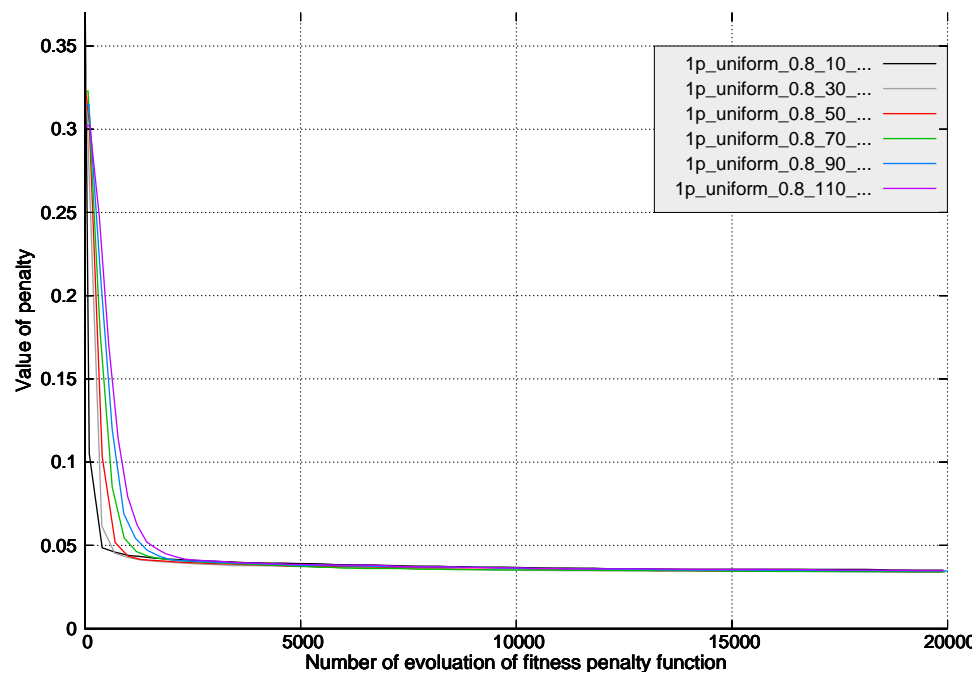
Figure 9: Comparison of fixed mutation probability 0.8 and different population size

results after fitting one can find in appendix and here just the best result is presented. But before presenting the result it is needed to introduce how one can compare different obtained functions and say that one is better then another for current task. The best solutions is present the measure for functions. The idea, that first interval shows how fast is convergence and the second interval shows if configuration converges, was accepted. For measuring speed of convergence the average velocity on interval was considered:

$$mes_1(sig_1(x)) = \frac{sig_1(x_{start}) - sig_1(x_{end})}{3500} \tag{7}$$

where x_{start} is first point of interval x_{end} is end point of interval, 3500 was chosen because it is length of the interval.

And for the second interval just parameter c was condered is the measure because it is limit point for Formula 6 when $x \to \infty$:

$$mes_2(sig_2(x)) = \lim_{x \to \infty} sig_2(x) = \lim_{x \to \infty} \frac{a}{b + x} + c = c \tag{8}$$

The surface showing dependency of first measure on mutation probability and population size is shown on Figure 10 and dependency of GA limit value on the same parameters is shown on Figure 11

As one can see evolution rate is higher on small populations. And limit is lower (better convergence) on big populations. The best result according to the first measure showed configuration with mutation probability - 0.8 and population size - 10. It means that with such a configuration GA converges faster, at least in the very beginning of algorithm run. But for the best convergence value the best configuration is mutation probability - 0.01 and population size - 90, i.e. the configuration has lower limit. Such different behavior in different configuration prompted to extend method used in one of mentioned papers ?? and use it for the current task.

The idea of thresholds was a decision how to combine results for the different configuration. The meaning of the approach is change mutation probablity and population size when algorithm reach number of runs equal to threshold. This idea can be extended to many number of thresholds but in this work it was limited by only two thresholds.

First step of optimization of thresholds was done by changing the first threshold and second thresholds but keeping difference between them the same and equal to 100 runs. The result is shown on Figure 12. As one can
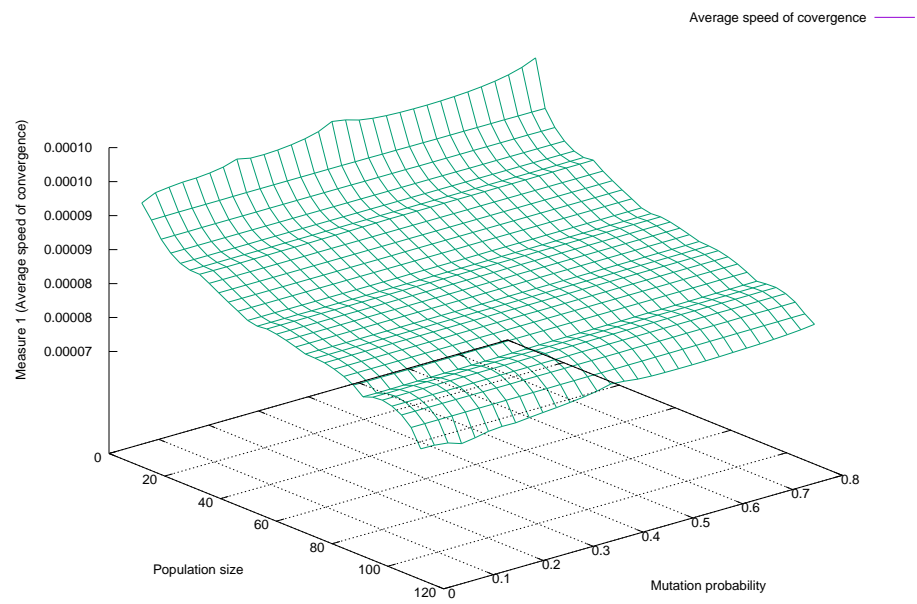
Figure 10: Average evolution rate dependency on mutation probability and population size
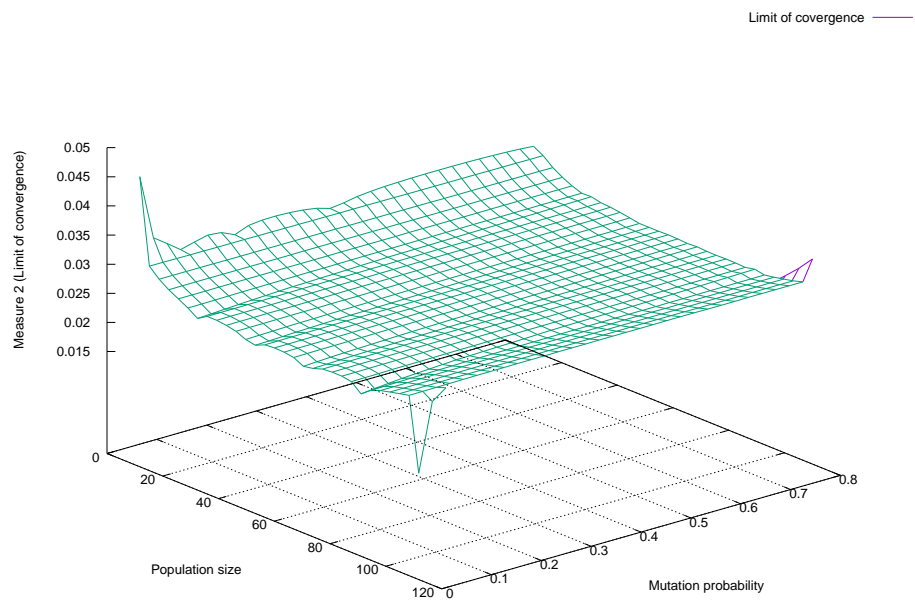
Figure 11: Limit of convergence dependency on mutation probability and population size
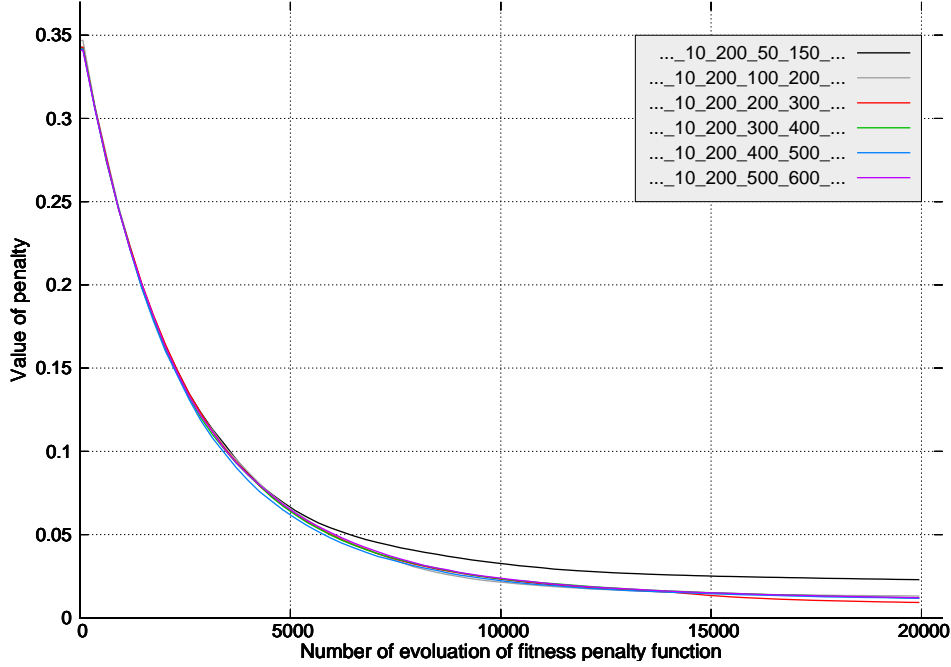
Figure 12: Comparison of different values of first threshold while second threshold is fixed

notice while first threshold is equal to 200 the GA has best result in penalty. Notice, labels again shortened. But now they are shortened from both sides start and end. The third shown parameter in the labels is first threshold while fourth - second threshold.

Second step of optimization of thresholds was done by fixing first threshold equal to 200 (see previous result) and changing difference between thresholds. The result is shown on Figure 13. As one can notice that results are very similar to each other, but in this work threshold equal to 500 was chosen.

Of course these threshold parameters could be optimized more seriously but it goes away from current work. The rest parameters remained as it was declared previously for default values.

After having the optimal configuration 1000 of session on such configuration were run in order to obtain the best fitted Voronoi structure and statistics for the configuration. The result of fitting is presented on Figure for convergence and on Figure for comparing size distribution before GA application and after. During run penalty reduced 1000 times compared to
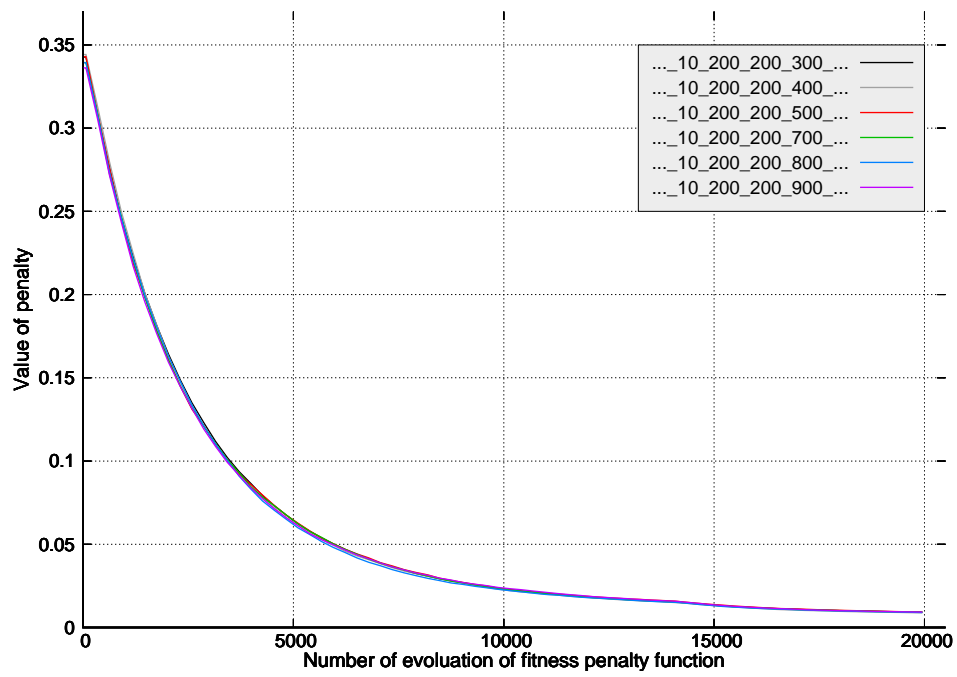
26

Figure 13: Comparison of different values of second threshold while first threshold is fixed and equal to 200

the initial distribution. On Figure 3 lines are presented: the best individual's penalty, the worst individual's penalty and the average penalty for the number of evaluating of penalty function.

0.3.3 Building the Voronoi tessellation fitting the given grain orientation distribution

Firsty it is needed to understand why grain orientation distribution is so important. Different orientation of neighboring grains leads to the grain boundary appearing. Boundary in its turn is very important because:
- Grain boundary vary significantly in their characteristics (mobility, energy, chemistry)
- Many properties of material depends on the nature of grain boundaries
- Material can have good or bad mechanical, corrosion properties depending on the type of grain boundaries

One of the popular method to represent the grain distribution is to calculate distribution of misorientation angels between neighboring grains. Given two orientations (grains, crystals), the misorientation is the transformation required to transform tensor quantities (vectors, stress, strain) from one set of crystal axes to the other set. For cubic lattice (in current study comparison will be for experimental data for FCC materials) there are 24 variants of rotation (transformation) that can transform one grain to another. But for misorientation distribution we need only lowest angle of misorientation. To calculate misorientation one need to know Euler angles of neighboring grains. For each grain with Euler angles $[\phi_1, \Phi, \phi_2]$ it is needed to calculate orientation matrix:

$$
g = \begin{bmatrix} c\phi_1 c\phi_2 - s\phi_1 s\phi_2 c\Phi & s\phi_1 c\phi_2 - c\phi_1 s\phi_2 c\Phi & s\phi_2 c\Phi \\ -c\phi_1 s\phi_2 - s\phi_1 c\phi_2 c\Phi & -s\phi_1 s\phi_2 - c\phi_1 c\phi_2 c\Phi & c\phi_2 s\Phi \\ s\phi_1 s\Phi & -c\phi_2 s\Phi & c\Phi \end{bmatrix} \quad (9)
$$

where c is cos, s is sin.
The misorientation matrix is then:

$$
\Delta g_{1,2} = g_1 A_{rot_i} g_2^{-1} \quad (10)
$$

where g_1 is orientation matrix for the first grain, g_2 is orientation matrix for the second grain and g_2^{-1} is invert matrix for g_2, A_{rot_i} is i-th rotation matrix (there are 24 of them, see appendix).

28

And from misorientation matrix it is possible to calculate misorientation angle:

$$cos\Theta = (trace(\Delta g_{1,2}) - 1)/2 \tag{11}$$

From 24 matrices one obtain 24 angles, but only smallest is valid for distribution.

There are many different misorientation angle distributions. The shape of distributions for different type of lattice can be find in Hans Grimmer's paper [12]. But currently in the work just cubic lattice is beeing considered. This is why Mackenzie distribution [13] is taken as a primary distribution for this study. It does not mean only this distribution is possible to reach in frame of current work. The framework is able to fit any user defined distribution. But Mackenzie distribution is taken as a reference for initial research.

Default random grain orientation distribution is described by Mackenzie distribution. This is why when one generate grain orientation randomly and even pseudo-randomly one obtain distribution very close the Mackenzie distribution. In order to check if implemented algorith works with any kind of initial grain orientation distributions, the initial orientations selection algorithm was changed. Initial orientation selection is based on generating Euler angles within $[0°, 9°]$ interval. This algorithm generates texture in simulated structure. The texture for polycrytalline is orientation distribution with some prefered direction.

[ToDo] show comparison of different pairs and configs

[ToDo] show result for best configuration

0.4 Filling tessellation with particles (atoms)

After getting correct container for polycrystalline and getting orientations of grains (in Euler angles) one can fill the container by atoms. Atoms should follow specified lattice and lattice has to be defined by base atoms. In general any of possible lattices can be defined in code but in this work only cubic structures are considered.

The process of lattice generation is pretty simple. The grain's seed point is considered as a starting point for lattice generation. Atoms located in oX, oY, oZ directions starting from this point. The distance between starting point and last atoms in each directions is of diameter of the grain. After

generating none-rotated lattice it is needed to rotate it according to the Euler angles obtained in previous step. It is done straightforward according to the defenition of Euler angles.

After rotation one need to remove excess atoms that stay outside of the grain volume. This procedure is done in this way: for all planes of grain coefficients of the plane equation are calculated $Ax + By + Cz = D$, for the seed point with coordinates (x_1, y_1, z_1) signs for each plane of grain are calculated $Ax_1 + By_1 + Cz_1 - D$, now the process repeated for all atoms of generated lattice and even if one sign differs from the sign of seed then this atom lie outside of grain volume and one needs to remove it.

It is important to notice that if the generated lattice will be moved in any axis by distance equal to lattice constant then we obtain absolutelly similar position of atoms (inside of grain because of periodicity of lattice) but we still get different position of we shift generatio by distance less then lattice constant. This feature will be used during first step of relaxation and this possible shifting is called **shifting vector** in current study. So, finally for each grain one got 9 parameters: 3 coordinates of seed, 3 coordinates of shifting vector, 3 Euler angles.

[ToDo] show one filled polycrystalline structure (colored grains)

0.5 Relaxation of atomic structure

[ToDo] say about energy minimization for the matter

[ToDo] say about potential used for total energy calculation

[ToDo] explain how works optimization of shifting vector of the grains

[ToDo] say that we use GA for shifting optimization. explain searching strategy for the optimal configuration

[ToDo] show results for GA for shifting

[ToDo] say how to optimize energy by removing excess atoms

[ToDo] say about measure for bad atoms (bad atom ranging)

[ToDo] show results for atom removal optimization

0.6 Conclusion

0.7 Acknowlegments

0.8 Appendix

[ToDo] show genotype to phenotype encoding

[ToDo] show block scheme of genetic algo for sizes and orientation

[ToDo] show block scheme for filling

[ToDo] show block scheme for relaxation

[ToDo] give link to the repo

Bibliography

[1] Johnson C. E. J. *Tritium behavior in lithium ceramics* Nucl. Mater 1999

[2] Pardo, Lorena, Ricote, Jess (Eds.) *Multifunctional Polycrystalline Ferro-electric Materials* Springer 2011

[3] Schropp, Ruud E. I. Zeman, Miro *Amorphous and microcrystalline silicon solar cells : modeling, materials, and device technology* Boston (Mass.) : Kluwer academic 1998.

[4] Harbeke, G. (Ed.) *Polycrystalline Semiconductors* Springer-Verlag Berlin Heidelberg 1985

[5] S. Plimpton *Fast Parallel Algorithms for Short-Range Molecular Dynamics* J Comp Phys, 117, 1-19 1995

[6] S. Plimpton *LAMMPS WWW Site* http://lammps.sandia.gov

[7] Shen, Y. et al. *Constructing three-dimensional (3D) nanocrystalline models of Li4SiO4 for numerical modeling and simulation.* Sci. Rep. 5, 10698; doi: 10.1038/srep10698 2015

[8] D Terada et al. *Effect of grain size distribution on mechanical properties of ultrafine grained Al severely deformed by ARB process and subsequently annealed* 2010 J. Phys.: Conf. Ser. 240 012111

[9] Suzudo T. Kaburaki H. *An evolutional approach to the numerical construction of polycrystalline structures using the Voronoi tessellation* Phys. Lett. A 373, 44844488 (2009).

[10] Wang J. et al. *Computer simulation of the structure and thermo-elastic properties of a model nanocrystalline material* Philos. Mag. A. 73, 517555 1996

[11] Antonio Rinaldi, Dusan Krajcinovic, Pedro Peralta, Ying-Cheng Lai *Lattice models of polycrystalline microstructures: A quantitative approach* Mechanics of Materials 40 (2008) 1736

[12] Hans Grimmer *The distribution of disorientation angles if all relative orientations of neighbouring grains are equally probable* Scripta Metallurgica Volume 13, Issue 2, February 1979, Pages 161-164

[13] Mackenzie, J.K. *Second Paper on the Statistics Associated with the Random Disorientation of Cubes* Biometrika 45,229. (1958)

[14] Jani Kotakoski and Jannik C. Meyer *Mechanical properties of polycrystalline graphene based on a realistic atomistic model* Phys. Rev. B 85, 195447 24 May 2012

[15] Guo-Jie J. Gao Yun-Jiang Wang Shigenobu Ogata *Studying the elastic properties of nanocrystalline copper using a model of randomly packed uniform grains* Computational Materials Science Volume 79, November 2013, Pages 56-62

[16] A.Prakash M.Hummel S.Schmauder E.Bitzeka *Nanosculpt: A methodology for generating complex realistic configurations for atomistic simulations* MethodsX Volume 3, 2016, Pages 219-230

[17] Ju Li *AtomEye: an efficient atomistic configuration viewer* Modelling Simul. Mater. Sci. Eng. 11 (2003) 173177

[18] Pierre Hirel *Atomsk: A tool for manipulating and converting atomic data files* Computer Physics Communications Volume 197, December 2015, Pages 212-219

[19] SimoneFalco JiaweiJian Francesco De Cola Nik Petrinic *Generation of 3D polycrystalline microstructures with a conditioned Laguerre-Voronoi tessellation technique* Computational Materials Science Volume 136, August 2017, Pages 20-28

[20] Weaire D. et al. *On the distribution of cell areas in a Voronoi network* Philos. Mag. B 53, 101105 1986

[21] Fan Z. G. et al. *Simulation of polycrystalline structure with Voronoi diagram in Laguerre geometry based on random closed packing of spheres* Comput. Mater. Sci. 29, 301308 2004

[22] Okabe A. et al. *Spatial Tessellations-Concepts and Applications of Voronoi Diagrams* Wiley, New York, 2000

[23] S. N. Sivanandam , S. N. Deepa *Introduction to Genetic Algorithms*, Springer Publishing Company, Incorporated 2007

[24] Melanie Mitchell *An Introduction to Genetic Algorithms* MIT Press, Cambridge, MA 1998

[25] David E. Goldberg *Genetic Algorithms* Pearson Education, 2006

[26] S. Suwas and R. K. Ray *Crystallographic Texture of Materials* Springer-Verlag London 2014

[27] Xuan Liu, Andrew P. Warren, *Comparison of crystal orientation mapping-based and image-based measurement of grain size and grain size distribution in a thin aluminum film* Acta Materialia 79, 138145 2014