DIPLOMARBEIT

Development of a Phenotype Algorithm for Particle Geometry Optimization

Ausgeführt am Institut für
Theoretische Physik
der Technischen Universität Wien
unter der Anleitung von
Ao. Univ. Prof. Dipl.-Ing. Dr. Gerhard Kahl

durch

Günther Doppelbauer
Obere Bahngasse 20/6, 1030 Wien

23. März 2009

Abstract

We implemented a search algorithm capable of finding the energetically most favorable and thus stable spatial configurations of particles interacting pairwise via potential functions, considering two different types of phenomena, namely cluster and lattice formation. This problem was successfully solved using concepts of genetic algorithms, in particular so-called phenotype algorithms introduced in the late 1990s. With the help of our algorithm, we gained results on well-known benchmark systems (including Lennard-Jones Clusters) as well as on a newly proposed two-dimensional system.

Contents

1 Introduction

In soft matter physics, the search for ordered equilibrium structures of systems interacting via potentials of different functional shape is an important task. We illustrate this for a particular class of systems, the colloidal dispersions - systems consisting of mesoscopic particles (colloids) in a microscopic solvent. The behavior of such systems can be described via effective interactions between the colloids, where the influence of the microscopic solvent particles on the colloids is incorporated through suitable coarse graining methods in an effective potential [1]. By varying parameters like temperature, salt concentration in the solvent or the chemical composition of the mesoscopic particles, the effective potential can be influenced in different ways, leading to the possibility of custom-made potentials. On solidification, these systems show a surprisingly wide variety of stable ordered structures. In contrast to hard matter (i.e. atomic) systems, these structures can be very complex and difficult to predict.

According to statistical mechanics, an equilibrium state of a system corresponds to an extremum of the appropriate thermodynamic potential. At constant volume, this potential is the free energy, which reduces at zero temperature to the internal energy of the system. On a microscopic level, the internal energy is a function of the coordinates of the interacting particles. Therefore, the search for equilibrium structures becomes an *optimization problem*: the global minimum of a *cost function* (in our case the internal energy) depending on a set of parameters (in our case the spatial coordinates of the particles) is to be found. All possible combinations of parameter values $\mathbf{x}$ (called candidate solutions) make up the *search space* and every point in search space corresponds to a cost value $f(\mathbf{x})$, for which the minimum $f(\mathbf{x}^*) = \min_{\mathbf{x}} f(\mathbf{x})$ (or maximum) should be found.

In practice, the solution of such optimization problems turns out to be extremely difficult for sufficiently large systems, even when one is searching for ordered structures. This due to the high dimensionality of the search space and the complicated functional dependence of the cost values on the parameters.

There is a wide variety of techniques to cope with optimization problems. *Local* optimization methods start from a certain point in search space and iteratively move to better neighboring points. These techniques usually find the local optimum of the cost function next to the starting point. Newtonian methods, utilizing the gradient of the cost function at hand, $\nabla_{\mathbf{x}} f(\mathbf{x})$, are a well known example for local optimization. *Global* optimization methods, on the other hand, are devoted to the generally much more difficult task of finding the overall optimum of the cost function, i.e. the candidate solution with the single best cost value possible.

Global optimization techniques can be divided in the following categories:

- **Deterministic approaches** explore the search space in a systematic way. Here, previously tested candidates determine in an unambiguous way the next candidate solutions.

- **Stochastic approaches** explore the search space in a random way. A very important example for these approaches are Monte Carlo methods [2].

- **Heuristic approaches** incorporate aspects of both aforementioned strategies: A heuristic search algorithm tries to explore the search space in a randomized, but (hopefully) intelligent way. This is achieved by taking both the information currently available (by previously evaluated candidate solutions) as well as random decisions into account when determining which point in search space should be tested next.

The search strategy this work is based on, namely genetic algorithms, belongs to the third category. Genetic algorithms explore multi-dimensional parameter spaces using several concepts inspired by natural (Darwinian) evolution processes [3], such as survival of the fittest, selection, recombination, inheritance and mutation.

Evolution itself can also be viewed as an optimization strategy: Individuals that are well adjusted to their environment have a high probability to survive and pass their genetic information (and thus their characteristic features) on to subsequent, probably even better adapted generations. However, it is important to note that this analogy is limited, since the cost function of problems we are investigating is generally fixed, whereas in natural evolution, there is an interaction between individuals and their environment, i.e. the “cost function” (viability of an individual or species) is influenced by the other “candidate solutions” (other individuals or species) that are present.

This thesis is organized in the following way:

- **Chapter 2** presents the concepts of the search strategy we use. In the first part, the standard approach of genetic algorithms is presented alongside some remarks on the theory behind them. The second part of this chapter introduces phenotype algorithms and tries to point out what makes them more suitable than standard genetic algorithms for problems like the ones we investigated.
- **Chapter 3** provides an overview of the characteristic features of the different model systems we studied.
- **Chapter 4** describes the technical details of our implementation of a phenotype algorithm as a FORTRAN 90 program.
- **Chapter 5** presents the results we obtained. Energy values of the most favorable configurations we found are provided and compared to previously published data if possible. Visualizations of some example structures are given at the end of this chapter.
- **Chapter 6** consists of a short overview of the work we did and points out possible future improvements for our algorithm.

2 Genetic Algorithms

Genetic algorithms (GAs) as computer simulations were probably first introduced by J.H. Holland [4] in 1975 and have ever since been further developed [5, 6, 7] and used on a wide variety of fields. These include, but are not limited to, physics, chemistry and material science (e.g. crystal structure prediction [8, 9], molecular structure optimization and cluster geometry optimization [10]), biology (e.g. protein and DNA structure prediction) [11], automated design (e.g. electronic circuits [12]), and economics (e.g. [13], [14]).

In our group, genetic algorithms have successfully been used for research on condensed matter theory [15, 16], 3D soft matter systems [17, 18, 19], 2D soft matter systems [20, 21, 22], and 2D soft matter layered systems [23].

Although Holland's schema theorem [4] and the building block hypothesis (see below) are widely considered an explanation for the power of genetic algorithms, there is still a debate over the mechanisms of the *adaptive capacity* (i.e. the average convergence towards better results) of GAs.

2.1 Standard Genetic Algorithms

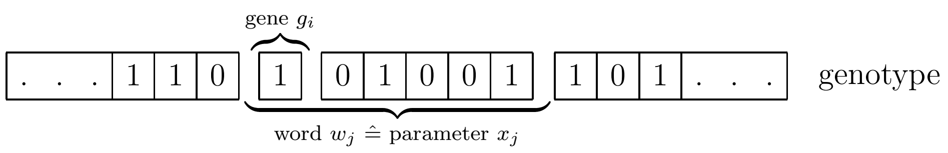
2.1.1 Basic concepts

The basic idea of GAs is to have a *population* consisting of candidate solutions (i.e. *individuals* $\mathcal{I}_j$), which evolves in time. *Genetic operations*, like selection, crossover and mutation, are applied on these individuals. The attributes of the individuals (i.e. *phenotype*) are coded into *genes*. A cost function determines the *fitness* of each individual. The chance of an individual to survive and to inherit its genetic information to subsequent generations depends on its fitness.

Here is a more detailed explanation of these concepts.

- **Genotype:** In GAs all information about an individual is encoded into and decoded from a string of genes. Every single gene can assume any value of a certain alphabet, $\mathcal{A} = \{a_1, \ldots, a_n\}$. The string of genes, in which all parameters of one candidate solution are encoded is called the individual's genotype (synonyms are chromosome and genome). All parameters correspond to substrings of the genotype. Such a substring, which can be decoded into one parameter x_j, could be called a *word* w_j or *genetic division*.

  Schematic representation of a genotype encoded in a binary alphabet:

  In nature, the genotype is encoded as a DNA sequence, where the bases adenine, cytosine, guanine, and thymine are the letters of the corresponding alphabet $\mathcal{A}^{\mathrm{DNA}} = \{A, C, G, T\}$.

  Note that the use of the term "gene" in GAs differs from the meaning it has in (classical) biological genetics, where it rather stands for "unit of inheritance", which has more in common with the substrings termed "words" above. A more consistent denomination for what we call gene here would be "allele".

- **Phenotype:** The phenotype stands for all properties (or parameters) of one individual and therefore represents one point in search space.

- **Coding:** For encoding and decoding a chromosome, one has to choose a genotype-phenotype map which of course has to be invertible. If it is not possible to encode variables with arbitrary numerical precision (e.g. real numbers as a finite string of binary numbers), the genetic search space represents a grid in real parameter space.

- **Fitness function:** The fitness of an individual is a function of its quality. While the quality (or rating) of a candidate solution is a well-defined value, the way in which its fitness depends on this value can be chosen. For example, if we search for configurations of particles with low energy, we will define fitness as some function of the energy. By modulating the fitness function, the chances for survival and mating of candidate solutions of different quality can be adjusted.

- **Representation:** The chosen genotype-phenotype map combined with the fitness function is called the representation of the optimization problem. By choosing a certain representation, the distribution of fitness values over all possible genotypes and therefore the way the search space of the problem at hand is represented in the algorithm[1] are implicitly determined.

- **Population and generation:** The set of individuals present at a certain time is termed the (current) population $P = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$. The number of individuals is in general constant, although it is possible to use GAs with variable population size. As the population propagates in time, new individuals are introduced into the population and old individuals are removed from it. The population at step i is called i th generation G_i.
- **Selection:** Before constructing new individuals by recombination, "parent"-individuals have to be selected from the population. In most cases an individual's chance of being selected increases with its fitness.
- **Crossover:** The mechanism of creating new individuals based on the genetic information of a number of parent individuals (usually two like in the natural analogon) is called crossover process.
- **Mutation:** Another way of creating new candidate solutions is by mutation of previously created individuals. The mutation-operator works directly on the genes. In typical implementations with a binary alphabet, one or several arbitrarily chosen genes are flipped by the mutation operator.

2.1.2 Algorithm

A pseudo-code of a generic genetic algorithm can be represented as follows (variables are typed in in `roman`, operations in *`italic`* letters):

```
program GA
  initialize(population)
  evaluate(population)
  while (not(termination_condition)) do
    parents=select_parents(population)
    new_generation=recombine(parents)
    new_generation=mutate(new_generation)
```

[1]This is often called *representation space*. The search space consists of all possible parameter configurations and their rating values, while the representation space consists of all possible genotypes and their fitness values.

```
    evaluate(new_generation)
    population=build_population(population,new_generation)
  end do
end program
```

Now I will describe the most important subroutines of a standard genetic algorithm in detail:

Initialization

In a first step, the starting population has to be initialized, i.e. all parameters of all individuals have to be assigned a value. Usually random values are chosen for those parameters, but it is also possible to use biased starting values, this is the case when some prior knowledge about the optimization problem is employed. When using uniformly distributed random starting values, the individuals can be directly initialized via their genes, i.e. every allele is assigned a random letter from the chosen alphabet. On the other hand, biased starting values are easier to incorporate via a phenotype-based initialization.

Encoding

As mentioned above, a bijective (i.e. revertible) genotype-phenotype mapping for the particular problem has to be found. Usually a binary alphabet $\mathcal{A}^{\text{binary}} = \{0, 1\}$ is used, because this allows efficient use of computational time and memory. The process of encoding a decimal integer number i into its binary representation $b_n b_{n-1} \ldots b_1 b_0$ works as follows

$$\begin{array}{ll}
b_0 = (i \mod 2) & i_1 = \text{int } (i/2) \\
b_1 = (i_1 \mod 2) & i_2 = \text{int } (i_1/2) \\
b_2 = (i_2 \mod 2) & i_3 = \text{int } (i_2/2) \\
\vdots & \vdots \\
b_n = (i_n \mod 2) & 0
\end{array}$$

The reverse operation is

$$i = b_n * 2^n + b_{n-1} * 2^{n-1} + \ldots + b_1 * 2 + b_0$$

A disadvantage of using the binary alphabet is that "neighboring values" in parameter space are not necessarily "neighbors" in their encoded representations. As an

Table 2.1: Decimal numbers 0-8 with their binary and Gray code representation

decimal	binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100

example, consider the integer numbers 3, and 4 with their binary representations 011 and 100. Another problem related to this encoding technique is that the "importance" of bit b_n increases exponentially with n. Thus the impact of a bit-flip on the phenotype differs strongly with the gene position. The first drawback mentioned above can be overcome by using Gray code [25] (also known as reflected binary code), which uses the binary alphabet $\mathcal{A}^{\text{binary}} = \{0, 1\}$, but different encoding rules. Two neighboring decimal numbers only differ in one bit in their Gray code representation. See table 2.1 for examples.

The schemata theorem (see below) suggest that "small" alphabets (i.e. alphabets with a small number of letters) are more suitable for GAs, since they maximize the number of schemata available, but there are also good arguments for using larger alphabets or even real numbers [28].

Evaluation

In the evaluation step the "quality" of each individual is determined. In a typical GA, two quantities are computed at this step: The first one is the *rating* $g(\mathcal{I}_j) = g_j$, an observable for which the global optimum should be found. An example for a rating value is the energy of a configuration of particles. Secondly, the *fitness* $f(g_j) = f_j$ of each candidate solution is assessed. As mentioned above, by choosing a fitness function, the "penalty" that is attributed to an individual in the "survival-of-the-fittest-game" for having a worse rating than others can be adjusted. There is no general rule for selecting an appropriate fitness function for an optimization problem.

Some common choices are listed below.

- **Proportional fitness:** $f_{\text{prop}}(g_i) = ag_i / \sum_{j=1}^{n} g_j$, where a is a parameter and n is the number of individuals in the population.
- **Linear fitness:** $f_{\text{lin}}(g_i) = L(g_i)$, where L is a linear function on $\mathbb{R}$. An example is $f_{\text{lin1}}(g_i) = 1/(\sum_{j=1}^{n} g_j - g_i)$
- **Exponential fitness:** $f_{\text{exp}}(g_i) = \exp\left(\frac{-a(g_i - g_{\min})}{g_{\max} - g_{\min}}\right)$, where a is a positive parameter and $g_{\max}$, $g_{\min}$ are the highest and lowest rating values in the current population. With this dynamically scaled fitness function, the individual with the lowest rating value is alway assigned fitness 1, while the individual with the highest rating value always has a fitness close to 0 (depending on a). This choice is suitable when positive as well as negative rating values are expected.

  In the p-dimensional space spanned by the p parameters of the optimization problem, the fitness values of all possible parameter configurations are called the *fitness-hypersurface.*

Selection

In this subroutine, "parent"-individuals, which are allowed to recombine and create offspring are selected. Since individuals with high fitness values have better chances to be selected, the average fitness of subsequent generations should increase. There are different schemes for calculating an individual's mating probability:

- **Linear ranking:** All candidate solutions are sorted in terms of their fitness values. A constant reproduction probability, which decreases with lower position, is assigned to each individual. As this method does not depend on the magnitude of the difference of fitness values, it is suitable to prevent "domination" of individuals with very high fitness values.
- **Tournament:** A number n of candidate solutions are randomly chosen from the population and the individual with the highest fitness value within this subset is allowed for reproduction. With higher n values (*selection pressure*), candidates with low fitness have a smaller chance to be selected. This procedure is repeated until the required number of parents is found.
- **Roulette wheel:** In this scheme, the reproduction probability p_i of an in-

dividual $\mathcal{I}_i$ is proportional to its fitness value. It is one of the most popular selection methods. The following formula is used:

$$p_i = \frac{f_i}{\sum\limits_{j=1}^{n_{ind}} f_j},$$

where n_{ind} is the number of individuals in the population. One could imagine this as a roulette wheel with variable slot width (see figure 2.1).

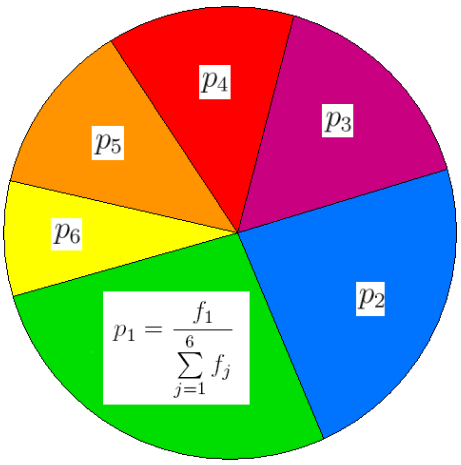


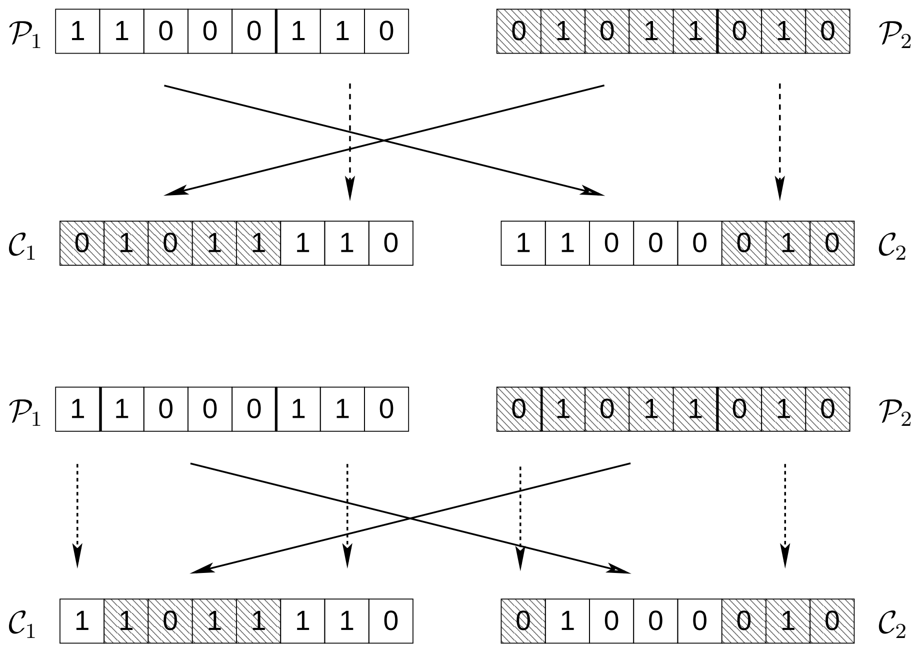
Figure 2.1: Schematic visualization of roulette wheel selection

Crossover

After determining which individuals are allowed to reproduce, a method for constructing the genes of the children based on the parental genetic information has to be defined. Although it would be possible to create a child from an arbitrary number of parents, we limit our considerations to two parents, similar to the natural analogon.

- **One-point crossover:** The gene-strings of each of the parents ($\mathcal{P}_1$, $\mathcal{P}_2$) are cut at the same, randomly chosen position. Then, two child chromosomes ($\mathcal{C}_1$, $\mathcal{C}_2$) are created, $\mathcal{C}_1$ consisting of the genes of $\mathcal{P}_1$ located left of the cutting position and the genes of $\mathcal{P}_2$ located right of the cutting position. $\mathcal{C}_2$ is created from the remaining segments. For a schematic illustration, see figure 2.2.

- **Two-point crossover:** Similar, but with two cutting positions - $\mathcal{C}_1$ is composed of the genes of $\mathcal{P}_1$ located left and right of the cutting positions and the



$\mathcal{P}_1$ 1 1 0 0 0 1 1 0
0 1 0 1 1 0 1 0 $\mathcal{P}_2$
$\mathcal{C}_1$ 0 1 0 1 1 1 1 0
1 1 0 0 0 0 1 0 $\mathcal{C}_2$
$\mathcal{P}_1$ 1 1 0 0 0 1 1 0
0 1 0 1 1 0 1 0 $\mathcal{P}_2$
$\mathcal{C}_1$ 1 1 0 1 1 1 1 0
0 1 0 0 0 0 1 0 $\mathcal{C}_2$

Figure 2.2: Schematic representations of one-point (top) and two-point (bottom) crossover operations with parent individuals at the upper and children at the lower part of each illustration. The crossover positions are marked by thick vertical lines. Images used by courtesy of Julia Fornleitner [20].

genes of $\mathcal{P}_2$ located in between these positions. For $\mathcal{C}_2$, the remaining segments are used. An illustration can be found in figure 2.2.

- **Random crossover:** This generalized method employs the so-called assembly vector $\mathcal{X}$, which is of the same length as the chromosomes. At first, each entry of $\mathcal{X}$ is chosen as a random bit. Then $\mathcal{C}_1$ is created bit-by-bit by copying allele i from $\mathcal{P}_1$ if the ith entry of X is 0 and copying allele i from $\mathcal{P}_2$ if the ith entry of X is 1. $\mathcal{C}_2$ is created in the opposite way, or alternatively by a bit inversion of $\mathcal{C}_1$. Sometimes this technique is called *uniform* crossover. This scheme is illustrated in figure 2.3.

Mutation

Mutation techniques are used to introduce new genetic sequences and to avoid "in-breeding" in analogy to mutation in evolutionary biology. Thus, mutation should

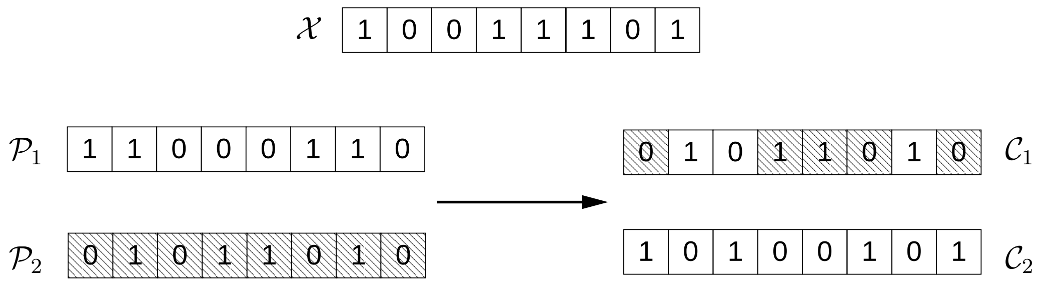


Figure 2.3: Schematic representation of a random crossover operation with inversion. Assembly vector $\mathcal{X}$ on top, parents on the left and children on the right side. $\mathcal{C}_2$ is the inversion of $\mathcal{C}_1$. Image used by courtesy of Julia Fornleitner [20].

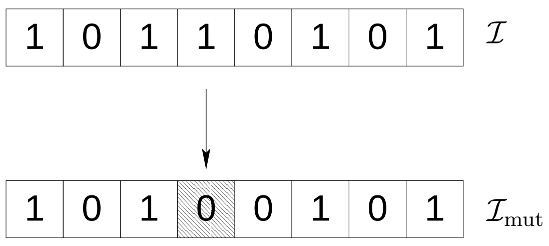


Figure 2.4: Schematic view of a single gene mutation. Image used by courtesy of Julia Fornleitner [20].

avoid that the GA becomes trapped in local minima.

In a typical mutation step, each gene of an individual has a chance p_{mut} to be changed to a random letter from the chosen encoding alphabet $\mathcal{A}$. Usually p_{mut} is rather small, ranging from 0.001 to 0.1. It can either be set as an external, fixed parameter or to increase with the number of generations.

Construction of new generations (Replacement)

The replacement scheme determines which individuals are used for building up the new generation G_{i+1}. Possible candidates are the individuals of the previous generation G_i and their offspring, either mutated or not.

- **Generational replacement:** A widely used replacement scheme is to discard the whole previous generation and construct the new population only out of (mutated) offspring. The advantage of this technique is that it can avoid premature convergence to a local minimum as it prevents a small number of individuals with very high fitness values from dominating the entire population. On the other hand, very good candidates are often lost from one generation to the next, thus the maximum and average rating within the population can decrease, leading to a slowdown in convergence.

- **Elitism:** Using this scheme, a number of "elite" individuals from the previous generation G_i survive and are included in the next generation. For example, G_{i+1} can be built up by a fixed number k of the highest rated individuals from G_i plus the $(n_{\text{ind}} - k)$ highest rated children. This makes sure that the maximum rating increases monotonically from generation to generation. Alternatively, all individuals from G_i and their offspring may be sorted with respect to their rating and the n_{ind} highest rated individuals are taken into the next generation, irrespective if they are parent or child individuals. With this approach, the average rating also increases monotonically, however, domination by candidate solutions in local minima can be a severe problem.

- **Niches:** Another concept that can improve convergence to the global minimum are niching methods, which allow the GA to maintain a population of diverse individuals (i.e. individuals with different features). A simple implementation of this concept is to permit only one individual within a certain interval of rating values in each generation. More sophisticated characteristics of the individuals could also be assessed and the number of individuals with certain features might be restricted in order to avoid getting stuck in local minima. When we consider crystal structure predictions as an example, the number of individuals belonging to the same symmetry class could be limited. This, of course, is highly problem specific and one has to have good knowledge of the system they are investigating before implementing such a scheme. For a deep investigation of this idea, see [29].

There are no general guidelines of how to choose the most appropriate replacement scheme for a given problem.

Termination condition and final refinement

The GA typically runs for a fixed number of generations, which has to be set as an external parameter. The experimenter has to set this value, usually relying on experience values of how many generations are needed until convergence. GAs are also sometimes set to terminate after a certain (acceptable) rating value is found. A convergence time limit can be used as a termination condition as well.

As stated earlier, the procedure to encode parameters as finite strings of a certain alphabet (i.e. with limited numerical precision) causes the GA to work on a finite grid in parameter space. Hence, the parameters of the fittest candidate solution found by the GA have to be further refined, typically using a steepest descent or conjugate gradient search.

Of course, one has to make sure that encoding precision is sufficient, so that the global minimum on the grid corresponds to the global minimum in real parameter space.

2.1.3 Schemata and Building Block Hypothesis

Although GAs are easy to implement and perform well on a large number of problems, the exact mechanism of their convergence is unknown. An attempt to explain this fact is the so-called building block hypothesis (BBH) [4, 5].

To formulate this hypothesis, we first need the concept of *schemata*, introduced by Holland [4]: A schema is class of strings with equal values at certain positions. For example, consider all strings of length six, consisting of letters of the binary alphabet. Then, both strings 100100 and 101110 would be part of the schema 10*1*0 (i.e. the subset of all strings with 1 in the first, 0 in the second, 1 in the fourth and 0 in the sixth position), where the symbol * represents the "wild-card" character and stands for an arbitrary letter. Schemata have certain characteristic features: A character, which is not a wild-card is called *defining position*, the *order* of a schema is the number of its defining positions and the *defining length* is the difference between the indices of the last and the first defining position of a schema. The schema in the example above has defining length six and order four. A schema with low defining length (and therefore low order) is considered "short". When the average fitness of all strings which belong to one schema is higher than the average fitness of all possible strings, the schema is said to be of "high fitness".

Holland argues that a GA implicitly tests large numbers of schemata while testing a relatively small number of points in parameter space. He calls that *implicit parallelism* of GAs.

In short, the BBH states that a GA recombines short, low-order and highly-fit schemata (which are called building blocks) to construct even fitter schemata of higher order and therefore converges to minima on the fitness-hypersurface by implicitly recognizing these schemata and propagating them to future generations.

Criticism

The BBH has been subject to sometimes heavy criticism for different reasons. On one hand, there have been empirical examinations, which suggest that GAs do not perform as well as expected on certain problems (see [26]). In some cases, a GA converged slower than a simple hill climbing algorithm on problems which were thought to be ideal for GAs. On the other hand the theoretical foundation of the BBH has been attacked (several such arguments are summed up in [27], section 3.3). A rather furious assault on the building block hypothesis can be found in [24], where the main argument is that the BBH makes far too strong *commonplace fitness structure assumptions*, namely

- **Abundant Basic Building Blocks:** A large number of basic building blocks[2] exists in the initial population.
- **Hierarchical Synergism:** *Antagonistic*[3] intersections between the building blocks of any level are rare, whereas *synergistic*[4] intersections between small collections of lower level building blocks are common.

The author of the aforementioned paper claims that the number of cases where these assumptions can be fulfilled is vastly outnumbered by the number of cases where they cannot be satisfied. Thus, in his opinion, the building block hypothesis cannot be an explanation for the convergence of GAs.

[2]Short schemata with high fitness are termed basic building blocks.

[3]An antagonistic intersection occurs when two schemata are recombined and the resulting schema (of higher order) has lower fitness than the initial schemata.

[4]Similarly, in a synergistic intersection the resulting schema has higher fitness than the constituent recombined schemata.

2.1.4 GAs and the No Free Lunch Theorems

The *No Free Lunch (NFL) Theorems for Search* were derived by Wolpert and Macready in 1995 [30] and expanded to optimization problems in 1997 [31]. Broadly speaking, these theorems state that if an algorithm A outperforms (i.e. finds the desired result faster than) an algorithm B on a certain number of problems (i.e. cost functions), B *has to* outperform A in exactly as many other problems (as long as certain conditions hold). This implies that, on average, no algorithm can outperform a random search when a sufficiently large number of problems is considered.

The Theorem

NFL can be stated in the following way (cf. [27], section 4.2): Let $\mathcal{X}$ be a discrete search space with points $x \in \mathcal{X}$ and f be a *cost function* (or rating function) with

$$f : \mathcal{X} \mapsto \mathcal{Y} \subset \mathbb{R}$$

The general objective is to find an optimal point (called solution) $x^* \in \mathcal{X}$, for which $f(x^*)$ is the global extremum. A search algorithm A generates a set of distinct points $\{d_m^x(i)\}$ with cost values $\{d_m^y(i)\}$, where $y = f(x)$ and $i = 1, \ldots, m$ is an index. The ensemble of points and cost values that have already been visited is noted as d_m. A starts from some point $d_1^x(1) = x_1$ with cost $d_1^y(1) = y_1$ and creates subsequent points depending on the previously visited points d_m in the following way

$$A : d \mapsto x, \text{with } x \in \mathcal{X} \setminus \{d_m^x(i)\};$$

i.e. A does not revisit points. The information contained in the sequence of points generated by A can be represented by a frequency table (histogram) $\mathbf{c}$ of the cost values $\{d_m^y(i)\}$. Some characteristics of $\mathbf{c}$, like its minimum, maximum, or mean value, can be used as measure of the quality of the algorithm's performance. For a given rating function, f, number of algorithm steps, m, and algorithm, A, the conditional probability $P[\mathbf{c}|f, m, A]$ for obtaining a certain frequency table $\mathbf{c}$ is the quantity measuring how well the algorithm works on this particular problem. Wolpert and Macready prove the following [31]:

For any pair of algorithms A_1 and A_2,

$$\sum_f P[\mathbf{c}|f, m, A_1] = \sum_f P[\mathbf{c}|f, m, A_2],$$

where the sum is over all possible functions f.

In this original formulation it is assumed that A is deterministic, which is not true for heuristic search algorithms like GAs. Nevertheless, the theorem can be restated so that it also holds for stochastic algorithms.

Implications

Since the NFL theorems are universal, they also apply to GAs, especially as Wolpert and Macready have extended their results to algorithms, which, like GAs, revisit points[5]. Although some researchers claim that the NFL theorems have virtually no significance for practical problems[6], hopes that one could construct something like a "universal GA", which performs extraordinarily well on a large number of problems without sacrificing performance on other problems cannot be fulfilled.

A more detailed introduction of these concepts and a discussion of their significance for practical applications can for example be found in [27], chapter 4 and [32].

2.2 Phenotype Algorithms

There are problems, (e.g. cluster geometry optimization, which will be the main topic in the following section), to which standard GAs are a suitable approach, but lead to unsatisfying results. It was suggested that for these problems string representations of the parameters are not ideal [33].

An important step forward was accomplished, when a GA, that operated directly on real-valued Cartesian coordinates, without any encoding and decoding, was introduced [34]. Although algorithms using this technique are still called *genetic* algorithms in most scientific contributions, this term is somewhat misleading, since the information processed by the algorithm is no longer represented in the form of gene strings. The algorithm's operators are rather applied on the phenotype-variables (see below). Therefore, Hartke introduced the denomination *phenotype algorithm* [35], which is more appropriate in my opinion.

[5]The authors do this by merely redefining a point-revisiting algorithm $A \rightarrow A'$, so that A' skips (i.e. does not evaluate) points previously visited.

[6]It is argued that, while NFL holds on average, cost functions encountered in real life have *structure* (which, itself, is a rather vague concept) and therefore sophisticated search algorithms are more suitable to find solutions than random search.

The next and probably most crucial step in applying modified GAs on cluster geometry optimization was first implemented by Deaven and Ho [7]. They executed a gradient-driven local optimization (in parameter space) on each individual after it was created (either initially, by recombination or by mutation). The fitness of the individuals was calculated as a function of their rating value after the optimization step. If the locally optimized individuals were encoded and (standard) crossover and mutation operators worked on them, the local optimization of the phenotype variables could be ruined due to these operations. Phenotypical crossover and mutation operators can quite easily be constructed in such a way that a large part of the optimized variables are preserved. These operators perturb comparatively small regions of the phenotype and leave large parts of it intact. Hence, the local optimization remains valid for the unchanged regions.

Phenotype algorithms with local optimization can be seen as corresponding to *Lamarckian*[7], rather than Darwinian evolution. In this picture, the local optimization can be seen as a "learning process" of individuals.

Since local optimization is quite "expensive" concerning computer time, Deaven and Ho [7] decided to radically reduce the number of individuals within a generation, from around 1000 in standard GA implementations to values ranging between 4 and 20.

Note that the concept of local optimization steps between the GA steps has a lot in common with the very successful *basin hopping* method introduced by Wales and Doye [36]. This was pointed out, for example, by Wales in [37]. Basin Hopping performs local optimization in between Monte Carlo steps (typically constant temperature Metropolis), where the acceptance criterion for a Monte Carlo step depends on the energy of the new configuration *after* optimization. More on this similarity - both approaches effectively transform the potential energy surface - will follow below.

2.2.1 Transformation of the rating hypersurface through local optimization

Some global optimization techniques use the concept of *hypersurface deformation* (for example, see [38]). These approaches attempt to simplify optimization problems by

[7]In this concept stated by J.B. Lamarck, individuals can inherit characteristics their parents *acquired during their lifetime*. This is often called *soft inheritance*.

transforming the rating hypersurface (or, more specifically, the potential energy surface, PES). The according transformations should smoothen the PES and reduce its number of local minima. The global minimum of the simplified PES is then mapped back to the original problem, hoping that it corresponds to its global minimum. The problem related to these techniques is that one can never be sure that the global minimum of the deformed hypersurface will map back onto the correct minimum of the original hypersurface.

The local optimization used in the phenotype algorithm described here (and, in a similar way, in the basin hopping technique) also simplifies the potential energy hypersurface, but in such a way, that a minimum on the transformed PES *always* corresponds to the according minimum on the original PES. As pointed out by Wales and Doye [36], the original hypersurface is effectively transformed on a step shaped surface, where each step corresponds to a basin of attraction (hence basin hopping) of a local minimum of the original hypersurface (for a schematic illustration, see figure 2.5). As neighboring parameter configurations lead to the same rating after the optimization step, values on the transformed hypersurface are visited more frequently. This effect can easily be avoided by including an energy-niche mechanism, i.e. only one candidate solution is allowed in a certain energy interval. Thus, the search-space the GA is working on is substantially reduced by this approach.

2.2.2 Phenotypical operators

As mentioned above, the operators of a phenotype algorithm work directly in the parameter space of the investigated problem. In our problem, they work on the Cartesian coordinates. Therefore, new mechanisms of crossover and mutation have to be introduced alongside a local optimization method. All other operators (e.g. evaluation and selection) can be defined as described in the section on standard genetic algorithms.

In early work by Zeiri [34], operators that averaged over coordinates of two parents or simply cut and pasted coordinates in array representation were used. Deaven and Ho [7] were the first to propose operators for a cluster geometry optimizing GA, that could preserve information gained by local optimization. Their approach, alongside possible modifications and refinements, will be presented here, after a short explanation of the notation used.

An individual $\mathcal{I}$ (i.e. an N-particle candidate cluster) is represented by a list of the

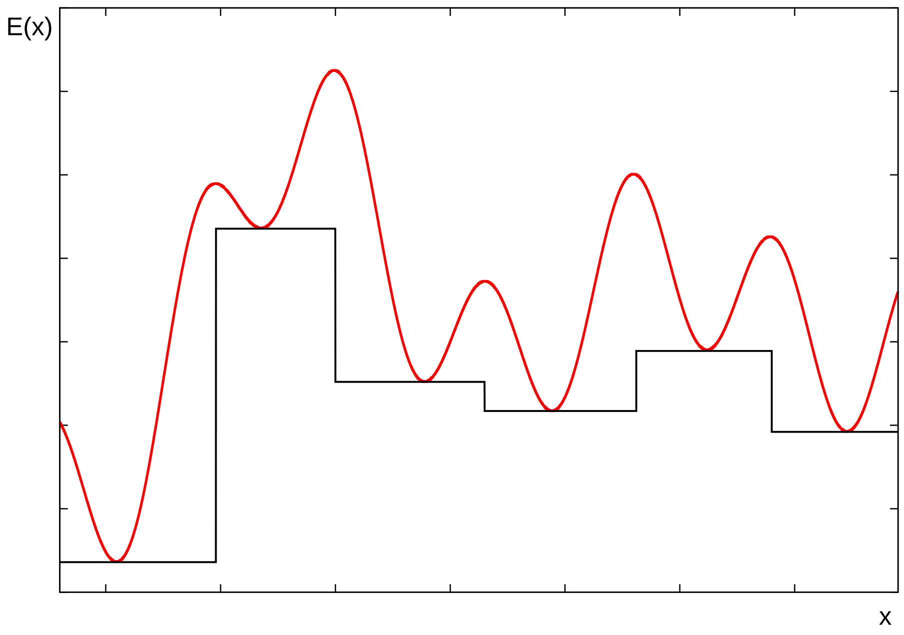


Figure 2.5: Schematic illustration of a potential energy surface (red) and its effective transformation to a step function (black).

Cartesian coordinates $\mathbf{x}_i$ of the particles the cluster consists of:

$$\mathcal{I} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$$

Crossover

Deaven and Ho's crossover operator $C : C(\mathcal{I}_{p1}, \mathcal{I}_{p2}) \to \mathcal{I}_c$ works on two parent individuals $\mathcal{I}_{p1}, \mathcal{I}_{p2}$ and produces a child individual $\mathcal{I}_c$ the following way: First, a random plane passing through the center of mass of both parent clusters is chosen. Then, the parent clusters are cut along this plane and the child $\mathcal{I}_c$ is assembled from those particles of $\mathcal{I}_{p1}$, which lie above the plane and those particles of $\mathcal{I}_{p2}$, which lie below the plane. It is possible that $\mathcal{I}_c$ does not contain exactly N particles after this procedure. In this case, the parent clusters are shifted to an equal amount in opposite directions perpendicular to the cutting plane and the operation is repeated until the child contains precisely N particles. Deaven and Ho call this mechanism *cut and splice* crossover. For an illustration see figure 2.6. With this approach, a second child consisting of the complementary particles could in principle be created (in accordance with the crossover operations described in the section on standard genetic algorithms), however these authors did not incorporate this possibility.

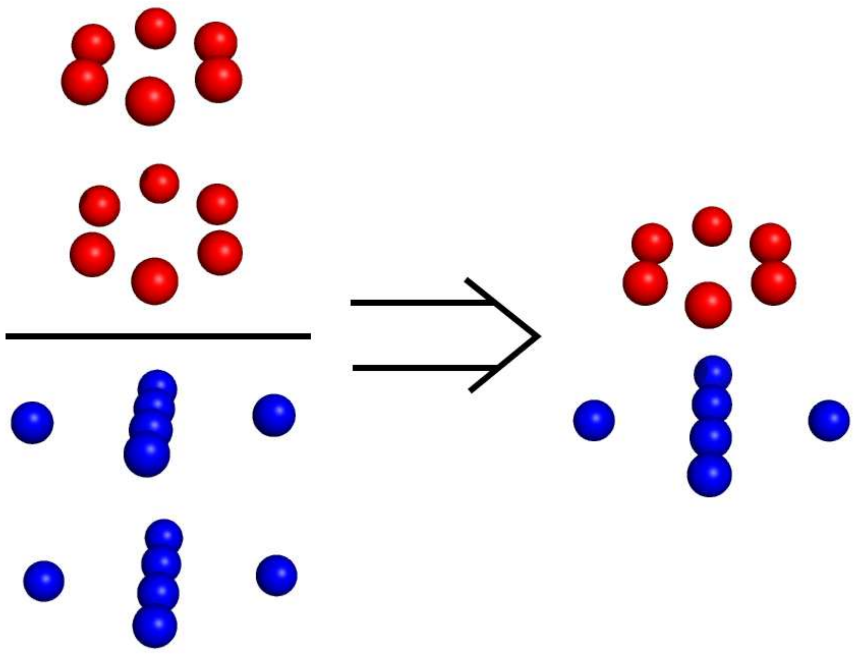


Figure 2.6: Cut and splice crossover. The parent individuals (left) are cut by a plane and the fragments are newly assembled to form the offspring (right).

As previously mentioned, the cut and splice-operator only disturbs the local optimization of a rather small part of a configuration, namely the region around the cut.

A variation of the cut and splice method has been proposed by Johnston in [39]. At first, random rotations around two perpendicular axes are applied on the parent clusters. Then, both parents are cut parallel to the xy-plane (this is an arbitrary choice, in principle any cutting plane could be used) and complementary fragments are spliced together. The cut plane can either be located at a random z-value, pass through the center of the clusters or can be chosen, depending on the fitness ratio of the parents (i.e. a large part of the parent cluster with higher fitness and a small part of the parent cluster with lower fitness are combined). A version of this mechanism which introduces two cutting planes (corresponding to a two-point crossover operator) is possible as well. This method can also be utilized to overcome the problem of producing children with a number of particles different from N: The cut plane is implicitly chosen by assembling the child of the j particles with the highest z-values from $\mathcal{I}_{p1}$ and the $(N-j)$ particles with the lowest z-values from $\mathcal{I}_{p2}$. j can again be either $\frac{N}{2}$, a random number, or biased by the fitness ratio of the parents.

For systems with periodic boundary conditions, a periodic cutting technique has been introduced [41].

Mutation

The mutation operator $M : M(\mathcal{I}) \to \mathcal{I}_{\mathrm{mut}}$ of Deaven and Ho performs one of two actions with equal probability. The first changes the coordinates $\mathbf{x}_i$ of $\mathcal{I}$ by a random distance a random number of times (unphysically close particles are separated between the steps). The second mutation mechanism performs a search for an adjacent watershed in the potential energy surface. After such a step, $\mathcal{I}_{\mathrm{mut}}$ usually has a higher energy, but should lie in a neighboring watershed of the PES.

There is a large number of possible other mutation operators and only a few should be addressed here. A straight-forward mutation is to assign a random number of particles new, random coordinates. These coordinates should of course be located within the expected extension of the cluster and very close particles have to be detached from each other. Alternatively, individuals can be cut in parts and these parts can be rotated against each other about an axis perpendicular to the cutting plane. Wolf and Landmann [40] introduced "twinning" and "etching" mutations. The twinning operation pastes half a cluster to its rotated image, while the etching operation adds some k particles to the candidate cluster, then performs energy minimization and finally removes the k most weakly bound particles, so that the cluster contains the desired number of particles again. The latter technique is similar to "directed mutation" operations (proposed by Hartke [35]), where the particle with the lowest binding energy is moved to the most favorable alternative position. Wolf and Landmann [40] also linked the occurrence of mutation to a possible stagnation of the population: Whenever the standard deviation of the parental cluster energies drops below a certain threshold value or there is no change in the energy distribution of the population for a given number of generations, individuals are mutated.

Local Optimization

In principle, any local search algorithm (i.e. an algorithm that starts from a candidate solution and iteratively moves to better neighbor solutions) could be applied on the individuals the GA produces. The most common choices will be described below, using the following notation: An optimal value of a rating function $f(\mathbf{x})$ depending on p parameters $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ is to be found.

- **Hill climbing:** The concept of a hill climbing search is rather simple: As a first step, parameter x_1 of the given candidate solution is increased by a certain value δ ($x_1 \to x_1 + \delta$). Next, the rating function f is evaluated. If the resulting value is better than the rating of the original individual, this step and its rating value are stored as a "best solution", otherwise the original individual remains as the best solution. For the second step, x_1 is decreased by δ ($x_1 \to x_1 - \delta$) and if the resulting rating is better than the one of the current best solution, the step and the corresponding rating are stored. Then steps for the second parameter x_2 in positive and negative direction are executed and evaluated. This is iterated for all possible $2p$ steps.

  After all these operations, the original individual is replaced by the one that has been stored as the best solution and the process starts anew. This procedure is iterated until the rating cannot be enhanced anymore. Then, δ is decreased, typically by $\delta \to \delta/3$, and the whole iteration is repeated.

  The termination condition of a hill climbing search is usually met when δ reaches a (very small) given value δ_{term} (i.e. the search converges). If the algorithm does not converge after a given number of steps, it is also terminated.

  If we combine all parameters x_i of the original solution into one vector $\mathbf{x}$, so that $x_i = \mathbf{x} \cdot \mathbf{e}_i$, the whole method can be written in a very compact way: Compute all values $f(\mathbf{x}_j)$ with $\mathbf{x}_j \in \{\mathbf{x} \pm \delta\mathbf{e}_i\}$. If $f(\mathbf{x}^*) = \min_{\mathbf{x}_j}\{f(\mathbf{x}_j)\}$ is lower than $f(\mathbf{x})$, replace $\mathbf{x}$ with $\mathbf{x}^*$ and repeat, otherwise decrease δ and repeat.

  The hill climbing approach is a very basic method that can be implemented rather easily. It works quite well on some problems, but its convergence speed is usually not very satisfactory, especially since the routine for local optimization is called extremely often in a phenotype algorithm like ours.

- **Gradient descent search:** This method, also known as "steepest descent" is quite similar to hill climbing, but it utilizes the gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ of the rating function. In this approach, the steps in parameter space are not constrained to go in the direction $\mathbf{e}_i$ of a particular parameter x_i, as each step goes in the direction of the steepest descent or ascent on the rating hypersurface, depending on whether one wants to reach a minimum or maximum. Additionally, a "line search" is applied, to determine the ideal step length, rather than having a fixed step length like in the hill climbing method.

In short, this algorithm works as follows: Compute the gradient $\Delta\mathbf{x} = -\nabla_{\mathbf{x}} f(\mathbf{x})$. Determine the ideal step length α^* as

$$f(\mathbf{x} + \alpha^* \Delta\mathbf{x}) = \min_{\alpha} \{f(\mathbf{x} + \alpha\Delta\mathbf{x})\}.$$

Replace $\mathbf{x} \to \mathbf{x} + \alpha^* \Delta\mathbf{x}$. Repeat until convergence (i.e. $|\vec{\nabla}_{\mathbf{x}} f(\mathbf{x})| \leq \delta$) or a certain number of steps is reached.

- **Conjugate gradient search:** The weakness of the gradient descent search is related to the fact that it converges slowly on a number of problems, because the directions of consecutive steps often differ considerably, so that the algorithm follows a zig-zag pattern. The idea behind conjugate gradient search is to improve this by incorporating previous search directions when calculating the direction of the subsequent step.

  The first search step is computed just like in a steepest descent algorithm

  $$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0^* \Delta\mathbf{x}_0.$$

  The following steps are determined as follows: Compute the gradient $\Delta\mathbf{x}_n = -\nabla_{\mathbf{x}} f(\mathbf{x}_n)$, then calculate the step direction as $\Lambda\mathbf{x}_n = \Delta\mathbf{x}_n + \beta_n \Lambda\mathbf{x}_{n-1}$. β_n can be obtained employing one of the the following expressions:

  - Fletcher-Reeves: $\beta_n^{\mathrm{FR}} = \frac{\Delta\mathbf{x}_n \cdot \Delta\mathbf{x}_n}{\Delta\mathbf{x}_{n-1} \cdot \Delta\mathbf{x}_{n-1}}$ [43]
  - Polak-Ribière: $\beta_n^{\mathrm{PR}} = \frac{\Delta\mathbf{x}_n \cdot (\Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-1})}{\Delta\mathbf{x}_{n-1} \cdot \Delta\mathbf{x}_{n-1}}$ [44]
  - Hestenes-Stiefel: $\beta_n^{\mathrm{HS}} = \frac{\Delta\mathbf{x}_n \cdot (\Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-1})}{\Lambda\mathbf{x}_n \cdot (\Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-1})}$ [45]

  Determine the ideal steplength α_n^* by a line search

  $$f(\mathbf{x}_n + \alpha_n^* \Lambda\mathbf{x}_n) = \min_{\alpha_n} \{f(\mathbf{x}_n + \alpha_n \Lambda\mathbf{x}_n)\}.$$

  Determine the new solution vector $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n^* \Lambda\mathbf{x}_n$, increment n ($n \to n+1$) and repeat until convergence or a certain number of steps is reached.

- **(L-)BFGS method:** The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [46, 47, 48, 49] can be considered as a quasi-Newton method for solving unconstrained non-linear optimization problems. The notation "quasi-Newton" refers to the fact that the Hessian (i.e. matrix of second derivatives of the rating function with respect to the parameters) does not have to be explicitly computed, as it is approximated by analyzing successive gradient vectors. The BFGS method can be seen as an extension of the conjugate gradient method, which uses not only the last, but more of the previous steps to compute the subsequent one. Therefore it expends more memory than a conjugate gradient optimization, but usually converges faster.

The L-BFGS algorithm is an implementation of the BFGS method, that requires a fixed amount of memory, which can be externally assigned [50, 51]. This improvement is achieved by storing a certain number of previous steps and when all available memory is used up, the oldest steps are overwritten by newer ones. L-BFGS-B is a further development of this software, that allows bounds (i.e. minimum and maximum values, but no general constraints) on the variables [52, 53]. The user is required to provide function values $f(\mathbf{x})$, first derivatives $\nabla_{\mathbf{x}} f(\mathbf{x})$, possible bounds on the variables $\mathbf{x}_{\mathrm{min}}, \mathbf{x}_{\mathrm{max}}$ and a number of parameters like desired convergence criteria and allowed memory (i.e. how many previous steps should be stored).

Deaven and Ho used a conjugate gradient search as local optimization method in their investigations [7]. Wales and Doye employed this technique in their first implementations of the basin-hopping algorithm as well [36], but later changed to L-BFGS [42], which is the standard method for cluster-geometry optimizing phenotype algorithms today [35, 39].

3 Problems Investigated

In this work, two different problems were examined. The first, namely 3D cluster geometry optimization, lead us to phenotype methods. For the second one, a two-dimensional system for which we tried to find quasicrystal-properties, we initially used a standard GA approach (i.e. with binary encoding). Later, with the insight gained from investigating the former system, we changed to a phenotype algorithm as well.

3.1 Cluster geometry optimization

Ensembles of N particles, where N ranges from a few tens to a few millions, are called clusters. Such entities are often seen as intermediate structures between molecules, consisting of a small number of atoms, and bulk solids, consisting of an extremely large number of atoms. A very popular example for clusters are fullerenes, which are made of a certain number of carbon atoms, where the "buckyball" structure C_{60} was the first to be discovered [54]. This was also the first test problem Deaven and Ho used their phenotype algorithm on [7].

Given the properties of the constituent particles in terms of their (pair-)interactions and their number N, their geometric arrangement is *a priori* unknown. In a $3N$-dimensional space (spanned by the coordinates of the N particles) the energy values of all possible particle configurations form the potential energy (hyper)surface (PES). The global minimum of this surface is called the ground state. Although the corresponding particle arrangement is not necessarily the structure that will be formed at finite temperature, it represents a good candidate for the most favorable configuration and thus is of central interest when investigating clusters. The number of local minima on the PES rises exponentially with the number of particles N (for example, see [55]), which renders the search for global minima very difficult as N increases. In fact, it has been shown that the problem is NP-hard[1] [56].

Because of the vast number of local minima on the PES and the difficulty of predicting the structural features of global minima for this particular problem, *ab-initio* (i.e. unbiased) search strategies are a valuable tool for such investigations. During the last 15 years, there has been considerable interest in this field of science and methods capable of efficiently locating global minima on the PES have been introduced. Among the most successful ones are basin hopping and genetic (phenotype) algorithms, which are both *ab-initio* approaches. A large number of results for clusters formed by different types of particles have been made available online at the "Cambridge cluster database" [58].

3.1.1 Lennard-Jones Clusters

Here we introduce clusters that are built up by particles interacting pairwise via the *Lennard-Jones (LJ) potential*:

$$V_{\mathrm{LJ}}(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right].$$

This interaction (see figure 3.1) is a mathematical model potential, which represents short-range repulsive and long-range attractive forces. r_{ij} is the interparticle distance, at $r_{ij} = \sigma$ the potential vanishes and ϵ is the depth of the potential well attained at $r = \sqrt[6]{2}\sigma$. The term $(\sigma/r)^{12}$ is responsible for the repulsive part and the term $-(\sigma/r)^{6}$ for the attractive part of the interaction.

The model has been suggested by John Lennard-Jones [59] as an attempt to image the behavior of neutral atoms or molecules, which repel each other at very small distances due to the Pauli exclusion principle[2] and attract each other at larger distances due

[1]In theoretical computer science, NP-hard means that the problem is "at least as hard as the hardest problems in NP", where NP represents the complexity class "nondeterministic polynomial-time". This nomenclature refers to the fact that the computation time of such problems is not greater than a polynomial function of the problem size on a nondeterministic Turing machine. Loosely speaking, problems in NP are yes-or-no-questions, whose answers can be verified (but not necessarily computed) in polynomial time. Popular examples for NP-hard problems are the Traveling Salesman problem and the halting problem [57].

In our case, the problem size corresponds to the number of particles N. Since our problem is at least as hard as the hardest problems in NP, it is implied that the time for finding the (exact) solution - on a deterministic computer - scales exponentially with N.

[2]This principle states that two identical fermions cannot occupy the same quantum state at the

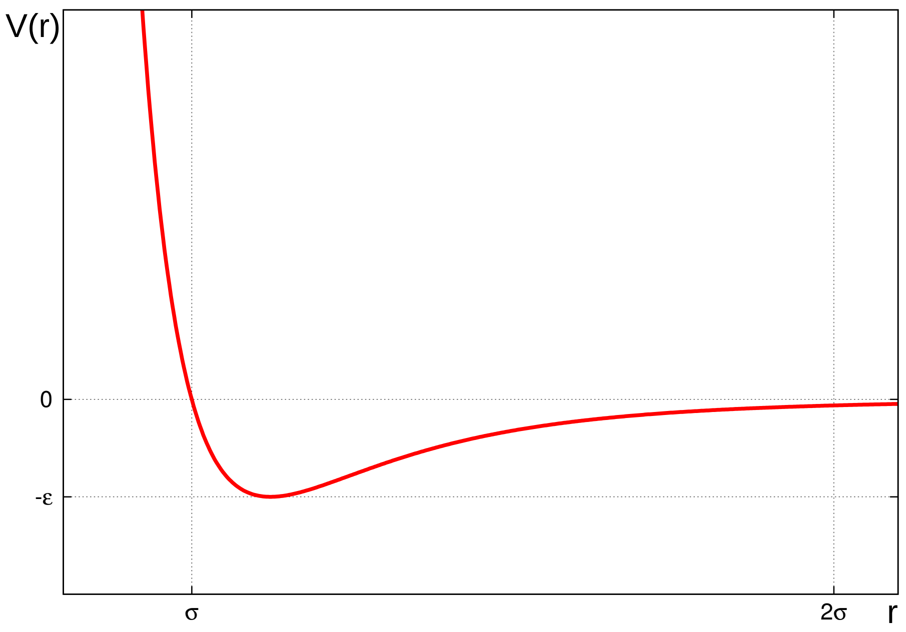


Figure 3.1: Lennard-Jones potential

to van der Waals forces[3]. The $\frac{1}{r^{12}}$ dependence of the first term is not justified by theory and is chosen for ease of computation, while the $\frac{1}{r^6}$ behavior of the attractive term is derived from models describing interactions between induced dipoles.

Clusters of Lennard-Jones particles have been investigated for many years (for example, see [55, 61, 62, 36]). Initially they were studied out of an interest for nucleation rates of noble gases, which are modeled particularly well by the LJ potential. Therefore, a large amount of data is available [58] and LJ clusters represent an ideal benchmark test for optimization algorithms.

For a majority of particle numbers, the energetically most favorable structure known today is based on so-called Mackay icosahedra (see figure 3.2 and [63]). These arrangements can be divided into 20 tetrahedral face-centered-cubic (fcc) units and therefore are sometimes called polytetrahedral structures. Complete icosahedral clusters are possible at 13, 55, 147,... particles. For other cluster sizes, a growth pattern can be formulated, where a growing layer of particles is successively added to a core

same time. Loosely speaking, if two atoms get so close that their electronic orbitals overlap, electrons (which are fermions) with equal spin effectively repel each other. This is due to the fact that they cannot occupy the same orbital (i.e. quantum state) and hence cannot be located at the same spatial position.

[3]Interactions between permanent or, as in this case, induced dipoles.

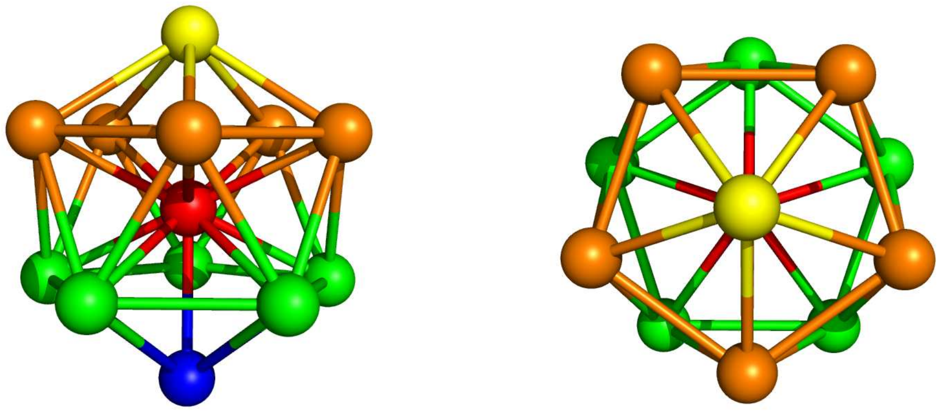


Figure 3.2: Mackay icosahedron consisting of 13 particles from different perspectives.

(i.e. a perfect icosahedral packing) until this process is completed at the next possible perfect Mackay icosahedron. Therefore, one can speak of icosahedral *shells*. These structures are believed to be dominant for particle numbers up to $N \sim 1600$ [64].

However, there are a few notable exceptions from this pattern in the range $2 \leq N \leq 150$. For $N = 38$, an fcc truncated octahedron is energetically more favorable than any known icosahedral structure [65]. For $75 \leq N \leq 77$ the optimal (known) configurations are based on the Marks decahedron [66]. For $N = 98$, the energetically most favorable structure published is a tetrahedral structure [67], and finally for $102 \leq N \leq 104$, there are again minimum energy clusters of decahedral shape [68]. Investigations [69] of the potential energy surface for $N = 38$ and $N = 75$ have shown that for these numbers of particles, the PES is also dominated by icosahedral local minima, despite having non-icosahedral global minima. This property makes it difficult for optimization methods to locate the global minima.

To model attractions of variable range, the LJ potential can be generalized to the so-called 2α-α *potential*:

$$V_{2\alpha-\alpha}(r_{ij}) = 4\epsilon\Big[\Big(\frac{\sigma}{r_{ij}}\Big)^{2\alpha} - \Big(\frac{\sigma}{r_{ij}}\Big)^{\alpha}\Big],$$

as proposed in [60].

Clusters of particles interacting via this potential (with $\alpha = 18$ - shown in figure 3.3 - and $\alpha = 100$) have been investigated in [72]. It was found that cluster geometries differ from the Lennard-Jones case for $\alpha = 18$ for particle numbers larger than ten. However, a further increase in α does not introduce additional cluster shapes.

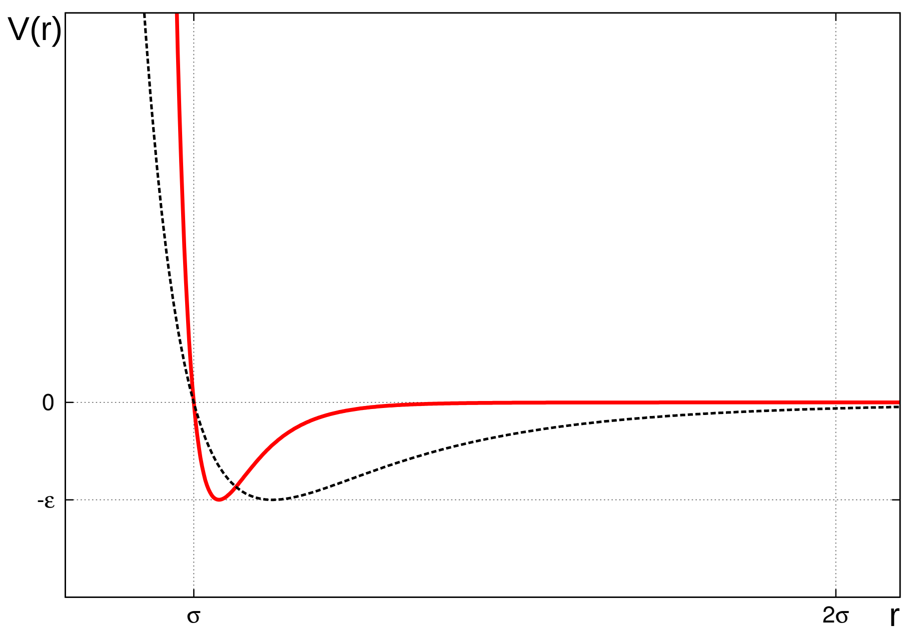


Figure 3.3: 2α-α potential with $\alpha = 18$ (red) compared to the standard Lennard-Jones potential (black, dashed)

3.1.2 Dzugutov Clusters

The *Dzugutov potential* (see figure 3.4)

$$V_{\mathrm{Dz}}(r_{ij}) = A(r_{ij}^{-m} - B)\exp\left(\frac{c}{r_{ij} - a}\right)\Theta(a - r_{ij}) + B\exp\left(\frac{d}{r_{ij} - b}\right)\Theta(b - r_{ij}),$$

where $\Theta(x)$ is the Heaviside step function and A, B, a, b, c, d and m are numerical parameters, was proposed by Mikhail Dzugutov [70], originally in order to study glass transitions in monoatomic systems. It has a local maximum approximately at $\sqrt{2}$ times the equilibrium pair distance $r_{\min}$, which should suppress the formation of close-packed configurations. This makes the Dzugutov potential a typical example for interaction potentials with short-ranged attractive and long-ranged repulsive forces.

Clusters of Dzugutov particles have been examined in [71]. As the authors point out, the curvature of the potential at its minimum, which is slightly different from the Lennard-Jones potential's curvature, and the disfavoring of distances near $\sqrt{2}r_{\min}$ influence the shape of the energetically most favorable structures of such clusters. From theoretical considerations and results based on a basin-hopping investigation, the authors conclude that non-compact polytetrahedral configurations should be the global minimum for most cluster sizes up to $N \sim 10000$.

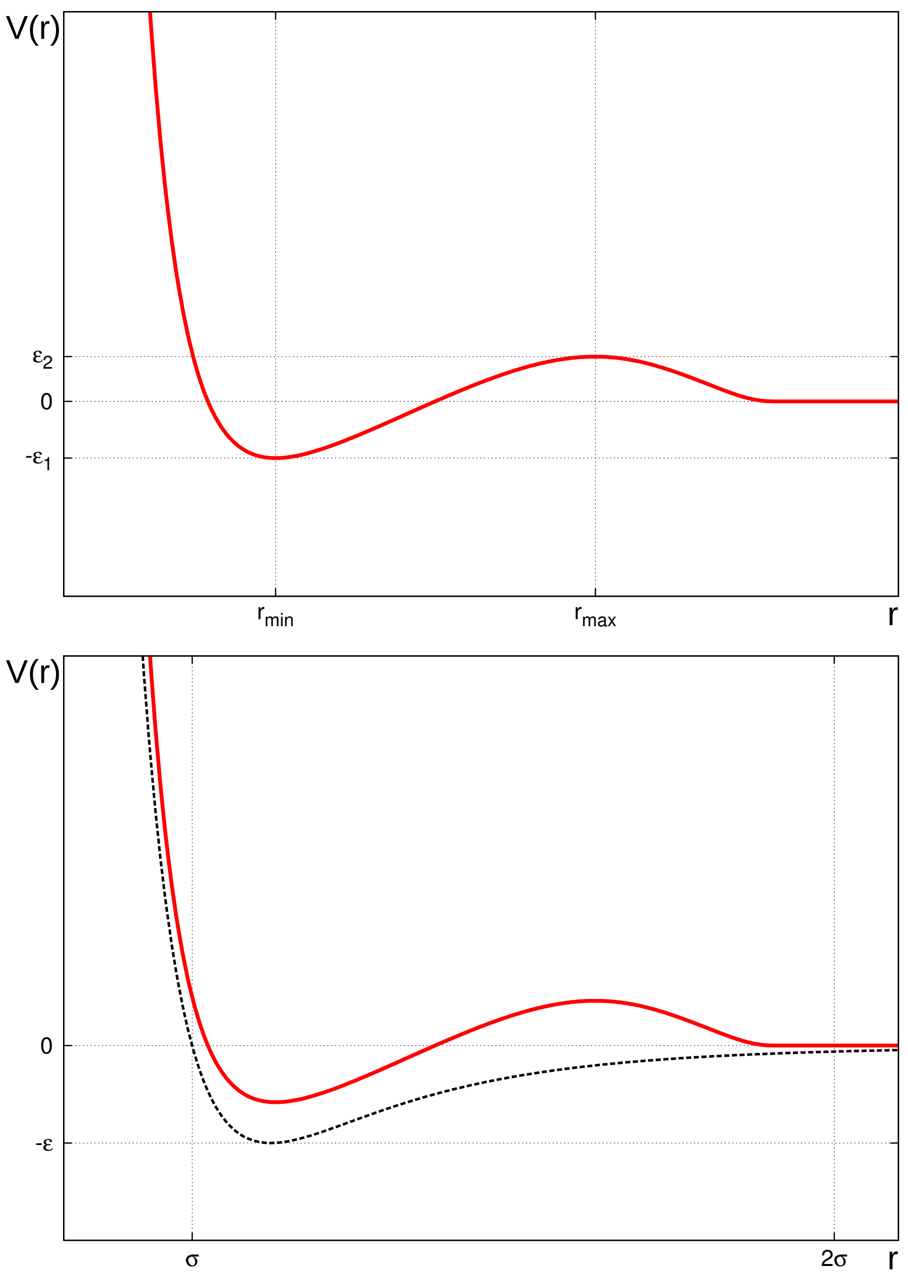


Figure 3.4: Top: Dzugutov potential with parameters $A = 5.82$, $B = 1.28$, $a = 1.87$, $b = 1.94$, $c = 1.10$, $d = 0.27$, $m = 16$; $r_{\text{min}} = 1.13$, $r_{\text{max}} = 1.44 r_{\text{min}}$, $\epsilon_1 = 0.581$, $\epsilon_2 = 0.791\epsilon_1$. Bottom: Dzugutov potential (red) compared to the standard Lennard-Jones potential (black, dashed).

3.2 2D-System

The second system we investigated is inspired by quasicrystals [73, 74]. These structures have quasiperiodic order, which means they show perfect long-range order, but (unlike crystals) no periodicity. Long-range order corresponds to a finite number of sharp peaks in the Fourier transform of a structure and therefore to sharp peaks in diffraction experiments.

A one-dimensional example for a non-periodic, long-range ordered structure is the so-called Fibonacci chain. It is a sequence of long (L) and short (S) intervals. The Fibonacci chain can be constructed by starting with one interval and iteratively executing the rules $S \rightarrow L$ and $L \rightarrow LS$, thereby building strings with increasing length. If we choose L as starting sequence, the first few strings are

$$\begin{array}{l} L \\ LS \\ LSL \\ LSLLS \\ LSLLSLSL \\ \vdots \end{array}$$

When the construction rule is iterated to infinity, the resulting sequence has no repetition distance (i.e. no periodicity) if the ratio of L and S is an irrational number. For the *canonical* Fibonacci chain, the ratio of the interval lengths is the so-called golden mean $\tau = (1 + \sqrt{5})/2$.

We have constructed a two-dimensional system confined in a quadratic box, where particles are forced to occupy a limited number of infinitely deep potential wells in horizontal direction. These wells are located at positions given by the first w segments of the canonical Fibonacci chain (see figure 3.6, top). In vertical direction, the particles can freely move within the bounds given by the box. The system has periodic boundary conditions in both x- and y-direction (for an illustration of this concept, see figure 3.6, bottom).

For the particle interaction, we use the *Gaussian core model* (GCM), introduced by Stillinger [75]. The GCM represents a good approximation for the effective interaction of certain mesoscopic macromolecules (e.g. interpenetrating polymer chains [76]). The functional form of the GCM-interaction is given by

$$V_{\mathrm{G}}(r) = \epsilon \exp\left(-\frac{r^2}{\sigma^2}\right),$$

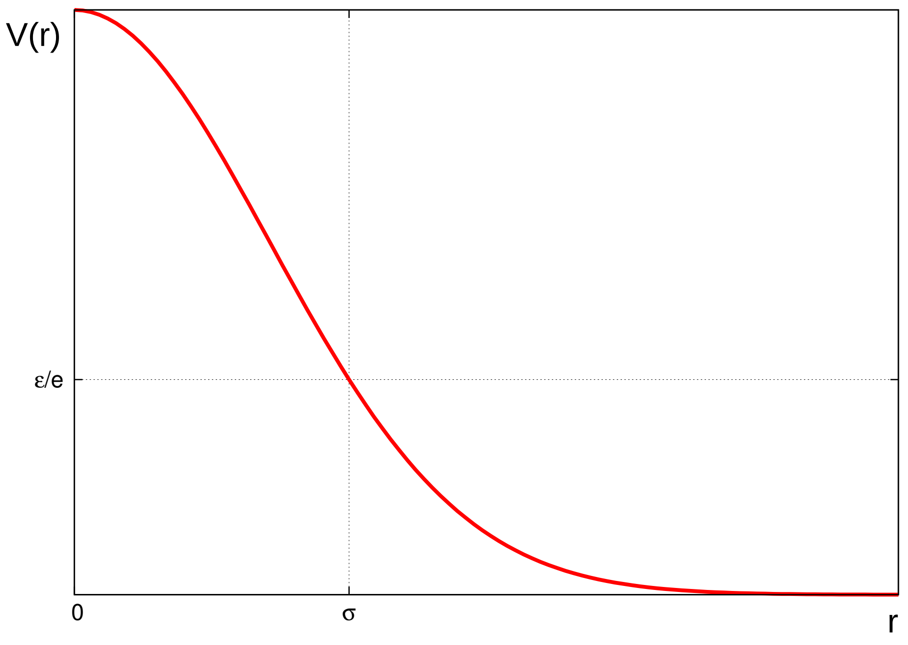
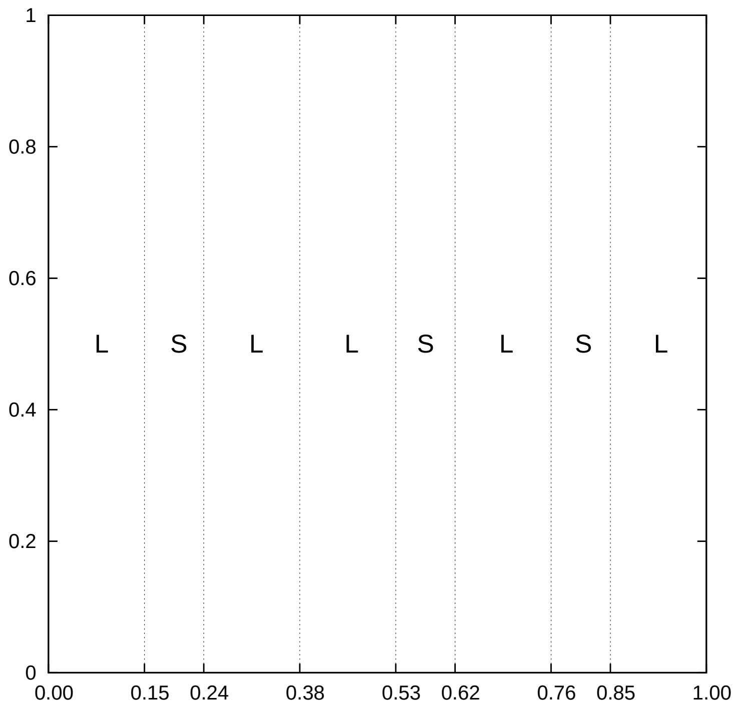


Figure 3.5: Gauss potential $V_{\mathrm{G}}(r) = \epsilon \exp\left(-\frac{r^2}{\sigma^2}\right)$.

with ϵ as a parameter defining the energy scale and σ as a parameter defining the length scale. This potential is purely repulsive and bounded (i.e. remains finite even at full particle overlap). A visualization of the Gauss function can be found in figure 3.5.

Of course, this system does not have true quasi-periodic structural properties, because the wells within the box represent only a finite portion of the Fibonacci chain, which is periodically continued by the boundary conditions.



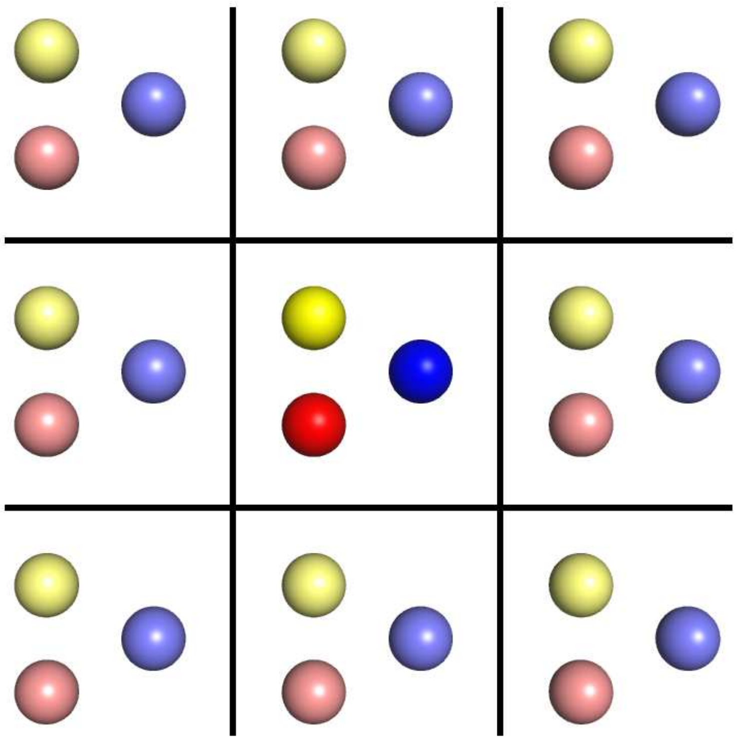


Figure 3.6: Top: Our 2D-system with eight wells, appearing as vertical dashed lines (arbitrary units). Bottom: Periodic boundary conditions; particles in the primary cell appear in dark, their periodic images in light colors.

4 Implementation of the GA

In this chapter, I describe our implementation of the modules required in a GA-program; for details I refer to chapter 2. Our GA was programmed in FORTRAN 90 and uses two external packages, namely *RANLUX*, a pseudo-random number generator proposed by M. Lüscher [77, 78] and *TOMS778* [79], a FORTRAN 90 implementation of L-BFGS-B for local optimization purposes (see end of subsection 2.2.2).

We used *ifort* (Intel Fortran Compiler) to compile our program and all simulations were run on our local "liquid" computer cluster.

The (standard) GA source code written by Dieter Gottwald [15] and improved by Julia Fornleitner and Gernot Pauschenwein [20, 17] was of great help for developing our own code.

The program is controlled via an input file. Here, the user can define the desired parameters (i.e. number of particles, number of individuals per generation, number of generations, interaction potential, etc.) at the beginning of each run. The program reads all entries from *inputfile* as part of the initialization routine. The output of the program consists of four files: the interaction potential is written to the file *potential.dat*; the energies of all individuals of all generations and details about the mating process are written to *analysis.dat*; whenever the up to that time most favorable candidate solution is found, its generation number and energy are written to *log.dat*; finally, at the end of a run, the Cartesian coordinates of the particles representing the best individual found are written to *plot.dat.*

4.1 Cluster geometry optimization

The following pseudo-code should provide an overview of our program. An in-depth explanation of these proceedings is given in the next sections.

```
program MY_GA
### INITIALIZATION ###
  define_variables
  initialize_ranlux
  read_parameters_from_inputfile
  allocate_memory
  check_and_plot_potential
### FIRST GENERATION ###
  set_coordinates(generation)
  optimize(generation)
  evaluate(generation)
  write_output
### REPRODUCTION ###
  do i=1,number_of_generations
    do j=1,(number_of_individuals − elitism_parameter)
      parents=select_parents(generation)
      child_generation(j)=recombine(parents)
    end do
    mutate(child_generation)
    optimize(child_generation)
    apply_niche_condition(child_generation)
    create_new_population(generation,child_generation)
    evaluate(generation)
    write_output
  end do
### TERMINATION ###
  plot_best_individual
  deallocate_memory
end program
```

4.1.1 Repeatedly used routines

Some routines are used on several occasions within the algorithm. Therefore I want to describe them at first, later I will refer to this section whenever these routines are employed.

Evaluation

- **Energy**: The energy E of a finite number of particles interacting via a pair potential $V(r)$ is calculated as

$$E(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} V(r_{ij}),$$

  with the number of particles N, particle coordinates $\mathbf{x}_i = (x_i, y_i, z_i)$ and interparticle distances

$$r_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}.$$

- **Forces**: The forces $\mathbf{f}$ acting on each particle due to the interparticle interaction (i.e. the gradient of the potential energy of every particle) have to be calculated for the local optimization procedure. The force acting on particle i is the sum over all forces originating from the interaction of this particle with all other particles j.

$$\mathbf{f}_i(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \sum_{j=1}^{N} \frac{\partial V(r_{ij})}{\partial r} \frac{\mathbf{x}_j - \mathbf{x}_i}{r_{ij}}, \; j \neq i$$

  Since the potentials we have used have an analytic shape, $\frac{\partial V}{\partial r}$ can be directly defined as a function.

- **Fitness**: Throughout we use an exponential fitness function

$$f_{\text{exp}}(\mathcal{I}_i) = \exp\left[\frac{-a(E(\mathcal{I}_i) - E_{\text{min}})}{E_{\text{max}} - E_{\text{min}}}\right],$$

  where the parameter a is read from the inputfile and E_{min}, E_{max} are the minimum and maximum energy values within the population. Since we employ the roulette wheel selection in all our calculations, a quantity called *modified fitness*

$$f_{\text{mod}}(\mathcal{I}_i) = \frac{\sum\limits_{j=1}^{i} f_{\text{exp}}(\mathcal{I}_j)}{\sum\limits_{j=1}^{n_{\text{ind}}} f_{\text{exp}}(\mathcal{I}_i)},$$

  where n_{ind} is the number of individuals, also needs to be calculated for all individuals $\mathcal{I}_i$ every time the fitness-subroutine is called. f_{mod} lies in the interval $[0, 1]$ for all candidate solutions. We define the space d_i one individual $\mathcal{I}_i$ occupies within the interval $[0, 1]$ as $d_i = f_{\text{mod}}(\mathcal{I}_i) - f_{\text{mod}}(\mathcal{I}_{i-1})$. The larger the absolute fitness value $f_{\text{exp}}(\mathcal{I}_i)$ of an individual, the larger is d_i. This quantity will be useful in the selection subroutine.

- **L-BFGS-B**: As already mentioned, we use a FORTRAN 90 version of L-BFGS-B available online [79]. We have to provide the following data to the subroutine:

parameter	description
n	number of variables; when we are working in d dimensions, this parameter is given by the number of particles times d
nbd(i)	we have to specify if there are bounds on any variables; we set nbd=2 for all variables (upper and lower bounds)
u(i), l(i)	values of upper/lower bounds for all variables; we confine the particles in a cubic box, hence the values for u(i) and l(i) are $\pm b$ (see below)
factr	value for termination condition on function values E: when the change of E from one optimization step to the subsequent step drops below (*factr* times the machine precision), the iteration is stopped; we chose factr=10^7 (machine precision=10^{-16})
pgtol	value for termination condition on the gradients $-\mathbf{f}_i$: when the maximum gradient component drops below *pgtol*, the iteration is stopped; we chose pgtol=10^{-5}
m	the number of previous steps stored in memory ($3 \leq m \leq 7$ is recommended)

Once the optimization is started, the routine will update the coordinates and ask for function values (i.e. energy) and gradients (i.e. negative forces) repeatedly, so that the corresponding evaluation subroutines described above have to be called each time L-BFGS-B asks for them.

- **Hill climbing**: For testing purposes, we also included a hill climbing search, which became obsolete in later runs because of a much slower convergence than the L-BFGS optimization.

4.1.2 Initialization

- **Define variables**: First we define a class[1] *t_generation*, in which all information on a generation of individuals can be stored. This includes coordinates of all individuals, forces acting on each individual, energy, fitness and modified fitness of all individuals as well as minimum, maximum and average energy within the generation. We also define a class *t_global*, where all parameters from the input file are stored.

- **Initialize *RANLUX***: At this step, the random number generator *RANLUX* is initialized with seed values depending on the system time in order to return different pseudo-random numbers at each run.

- **Read parameters from *inputfile***: As mentioned above, the user sets the parameters for a GA run in an input file. The variables described in the following table are imported at this step.

  Then, the size of the box b in which the particles are confined is computed. The maximum and minimum values of all Cartesian coordinates x, y, z are set as a function of the number of particles N ($b \sim N^{\frac{1}{3}}$).

[1] In object oriented programming languages, patterns (or blueprints) for objects (which are in our context variables) are called classes. All objects created from such a blueprint share the same structure. In our case, the classes are used to unify all parameters of the population in one variable for more convenient access. Actual variables created from a certain class will later be called *instances* of this class.

parameter	typ. values	description
PARTICLES (N)	10 - 100	number of particles the system should contain
INDIVIDUALS (n_{ind})	8 - 12	number of individuals making up a generation
GENERATIONS (n_{gen})	10^2 - 10^5	maximum number of generations until program termination
MUTATIONRATE (p_{mut1})	0.01 - 0.1	chance of an individual to be mutated
MUTATIONRATE2 (p_{mut2})	0.1 - 0.33	chance of a particle to be shifted if an individual is mutated
FITNESSPARAMETER (a)	3	coefficient of the exponent of the fitness function
ELITISM (e)	2 - 4	determines how many individuals from the prev. gen. survive
POTENTIAL_TYPE		specifies interaction potential (Lennard-Jones, Dzugutov, etc.)
POTENTIAL_x		different parameters of the potential function
COORDS_FROM_FILE	-3 - 3	determines if coordinates of initial individuals are seeded

- **Allocate memory**: Based on the values from *inputfile* (number of particles, individuals and generations), memory is allocated to instances of the class *t_generation*.

- **Check and plot potential**: The potential functions and their first derivatives are included in the source code as analytic functions. We are working in natural units of the Lennard-Jones potential, i.e. ϵ and σ are set to one. Since Lennard-Jones and Dzugutov potentials attain extremely high values for low interparticle distances r, we apply a cutoff at low distance to avoid numerical problems (particularly concerning the local optimization routine). This is done by checking the first derivative of the potential for small r-values and as soon as its value exceeds a certain limit at $r = r_c$, the first derivative is set to a constant value, so that the potential itself increases only in a linear way for $r < r_c$.

  Then, the interaction potential $V(r)$ is written to a file in tabulated form.

4.1.3 First Generation

- **Set coordinates**: If the user chooses unbiased starting values (COORDS_ FROM_ FILE=0), all particles of all individuals are appointed random values within the bounds determined earlier. If COORDS_FROM_FILE is set to a negative value $-m$, the initial values will be determined from a cluster configuration with $(N - m)$ particles. The user has to provide a file *startcoords.dat* which contains Cartesian coordinates of $(N - m)$ particles in tabulated form (i.e. a table with x, y and z-values in rows). The particle arrangement of each individual will then be composed of these coordinates plus m randomly located particles. The random particles are set to be situated outside the original $(N - m)$-cluster. The coordinates of these particles are separately determined for each starting individual. If COORDS_FROM_FILE is a positive value m, the coordinates are seeded from a larger cluster. The user has to provide a file containing coordinates of $N + m$ particles. The algorithm will randomly choose N particles from this file for every starting individual.

- **Optimize**: Directly after creation of the first generation, a local optimization is performed on each individual as described in section 4.1.1.

- **Evaluate**: The individuals are sorted by their energy (which is already known after the optimization step) in ascending order. Then, their fitness and modified fitness are computed. The whole generation is copied to another instance of the t_generation-class called *best_generation*.

- **Write output**: The energy of the best candidate solution is written to the logfile and the energies and fitness values of all solutions are stored in the analytics-file.

4.1.4 Reproduction

Now, the algorithm enters the reproduction-loop, where it remains until the desired number of generations n_{gen} has been created. The first task is represented by a further loop, which selects a number $2c$ of parent individuals and recombines them to form c children. $c = n_{\text{ind}} - e$ is the number of individuals within a generation n_{ind} less the number of individuals which survive from the previous generation (specified by the elitism parameter e).

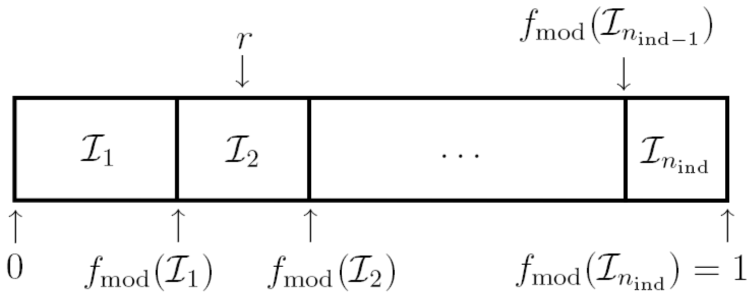

r
$f_{\text{mod}}(\mathcal{I}_{n_{\text{ind}-1}})$
$\mathcal{I}_1$
$\mathcal{I}_2$
$\cdots$
$\mathcal{I}_{n_{\text{ind}}}$
0
$f_{\text{mod}}(\mathcal{I}_1)$
$f_{\text{mod}}(\mathcal{I}_2)$
$f_{\text{mod}}(\mathcal{I}_{n_{\text{ind}}}) = 1$

Figure 4.1: Selection: $f_{\text{mod}}(\mathcal{I}_1) < r < f_{\text{mod}}(\mathcal{I}_2)$, therefore $\mathcal{I}_2$ is selected for recombination.

- **Select parents**: The roulette wheel selection is implemented as follows: A random real number r between zero and one is determined. The modified fitness values of all candidate solutions f_{mod} are compared to this random number. If the random number lies between $f_{\text{mod}}(\mathcal{I}_{i-1})$ and $f_{\text{mod}}(\mathcal{I}_i)$, $\mathcal{I}_i$ is chosen as a parent $\mathcal{P}_1$. An illustration of this process is depicted in figure 4.1. Then the process is repeated to select a second parent $\mathcal{P}_2$ under the constraint $\mathcal{P}_1 \neq \mathcal{P}_2$.

- **Recombine parents**: In a first step, both parent individuals are rotated via randomly chosen angles about two arbitrary axes passing through the origin. This is achieved by applying multiple matrix multiplications on the coordinate vectors of the particles (the matrices represent rotations about the coordinate axes). Then, the parent individuals are shifted so that their centers of mass lie in the origin of the coordinate system and all particles within each cluster are sorted with respect to their z-coordinates (this makes a simple implementation of the cut-and-splice crossover possible; any coordinate could have been chosen since the clusters have been rotated about random axes). Next, a random integer number n_z between zero and N (number of particles) is determined and the child individual is composed of the first n_z particles of $\mathcal{P}_1$ and the last $(N - n_z)$ particles of $\mathcal{P}_2$.

After the selection-and-recombination-loop, the following steps are performed.

- **Mutate**: For all children a random real number between zero and one is chosen. If this number is smaller than p_{mut1}, the individual enters the mutation subroutine, where another random real number within the same interval is determined for every particle. If this number is smaller than p_{mut2}, the particle is shifted to a random position within the allowed bounds.
- **Optimize**: All children are locally optimized.
- **Apply niche condition**: We want to have only one candidate solution within a certain energy interval $2|\delta E|$ (we chose an interval of length $|\delta E| = 5 \cdot 10^{-5}$ in natural units). Therefore, the algorithm checks for all individuals $\mathcal{I}_i$ if there is another candidate solution $\mathcal{I}_j$ with $E(\mathcal{I}_i) - \delta E < E(\mathcal{I}_j) < E(\mathcal{I}_i) + \delta E$. If this is the case, $\mathcal{I}_j$ is discarded. Two new parents are selected from the previous generation and recombined to form a new child, which is then optimized and replaces $\mathcal{I}_j$.
- **Create new population**: The first (i.e. energetically best) e individuals of the old generation are kept, while the remaining $n_{\text{ind}} - e$ individuals are overwritten by the children created in the steps described above.
- **Evaluate**: The candidate solutions are again sorted by ascending energy and the corresponding fitness values are calculated. If the energy of the best individual is lower than the lowest energy stored in *best_generation*, the current generation replaces *best_generation*.
- **Write output**: The index numbers of the individuals selected for recombination are written as pairs to the analytics file. In addition, energies and fitness values of all individuals are stored there as well. If a new best solution was found, the generation number and energy value are written to the logfile.

Then, the reproduction-loop returns to its start.

4.1.5 Termination

After n_{gen} generations, the following steps are taken:

- **Plot best individual**: The coordinates of particles of the energetically most favorable individual found in the whole process are written to the file *plot.dat* in the form

$$\begin{array}{ccc} x_{\text{particle 1}} & y_{\text{particle 1}} & z_{\text{particle 1}} \\ x_{\text{particle 2}} & y_{\text{particle 2}} & z_{\text{particle 2}} \\ \vdots & \vdots & \vdots \\ x_{\text{particle } N} & y_{\text{particle } N} & z_{\text{particle } N} \end{array}$$

for later visualization or seeding new runs.

- **Deallocate memory**: The memory reserved for the instances *generation* and *best_generation* is deallocated and the program is closed.

4.2 2D-System

For our 2D system introduced in chapter 3.2 we had to modify a few routines. These changes are listed below.

Evaluation

Since our system has periodic boundary conditions, the calculation of the energy has to incorporate the periodic images of the unit cell.

- **Energy**: Using periodic boundary conditions with the *minimum-image convention*, the total energy of the particles in the unit cell is the sum over all potential energies originating from the interactions of every particle with the closest image of each other particle.

$$E(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} V(r_{ij}),$$

where r_{ij} is given by

$$r_{ij} = \min_k [r_{ij_k}] = \min_k \left[\sqrt{(x_{j_k} - x_i)^2 + (y_{j_k} - y_i)^2} \right],$$

with $x_{j_k} \in \{x_j - b, x_j, x_j + b\}$, $y_{j_k} \in \{y_j - b, y_j, y_j + b\}$. b is the sidelength of the (quadratic) unit cell.

- **Forces**: The force $\mathbf{f}_i$ on particle i is calculated similarly:

$$\mathbf{f}_i(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \sum_{j=1}^{N} \frac{\partial V(r_{ij})}{\partial r} \frac{\mathbf{x}_j - \mathbf{x}_i}{r_{ij}}, \; r_{ij} = \min_k [r_{ij_k}], \; j \neq i.$$

All these calculations can be speeded up by first checking if particles are within cutoff-distance before calling the potential function.

- **Fitness**: With the purely repulsive Gauss potential

$$V_{\mathrm{G}} = \epsilon \exp\Big(-\frac{r^2}{\sigma^2}\Big),$$

only positive contributions to the potential energy are possible. Therefore, we could use an exponential fitness function without normalization:

$$f_{\mathrm{exp}}(\mathcal{I}_i) = \exp\left(-aE(\mathcal{I}_i)\right).$$

Optimization

For this system, the local optimization can only be applied to the y-coordinates, since the possible x-values are fixed (see section 3.2). Only the genetic operations have to find the most favorable number of particles for each well.

Unfortunately, the convergence to local minima of L-BFGS-B was as not as good as the convergence of our self-implemented hill climbing algorithm in the 2D system. Therefore, a GA run employing L-BFGS-B as local optimization method needed a distinctly larger number of generations to converge to the global minimum than a run with hill climbing. Nevertheless it was still favorable to make use of the L-BFGS method, because of its higher speed (i.e. an L-BFGS run with more generations was still much faster than a hill-climbing run with less generations).

To minimize the risk of missing global minima due to a not very reliable local minimization algorithm, we double-checked a few of our results with the slower hill-climbing runs.

Initialization

The following parameters are read from the input file:

parameter	typ. values	description
PARTICLES (N)	24 - 40	number of particles the system should contain
INDIVIDUALS (n_{ind})	8 - 20	number of individuals making up a generation
GENERATIONS (n_{gen})	10^2 - 10^5	maximum number of generations until program termination
MUTATIONRATE (p_{mut})	0.01 - 0.1	chance of a particle to be shifted to a random position
FITNESSPARAMETER (a)	1	coefficient of the exponent of the fitness function
ELITISM (e)	2 - 6	determines how many individuals from the previous generation survive
X_WELLS (w)	8	specifies the number of possible particle locations on the x-axis
SPACEFILLING (ρ^*)	0.1 - 2.0	specifies the desired covering density
POT_EPSILON (ϵ)	$10/\sqrt{2\pi}$	parameter of the potential energy function
POT_SIGMA_SQ (σ^2)	0.01	parameter of the potential energy function

- **Boxsize**: For the 2D system, we use a variable particle-size/boxsize ratio r_{part}/b. For the Gauss potential $V_{\text{G}} = \epsilon \exp\big(-\frac{r^2}{\sigma^2}\big)$, r_{part} can be defined as

$$V_{\text{G}}(r_{\text{part}}) = \frac{1}{e} V_{\text{G}}(r = 0) \to r_{\text{part}} = \sigma.$$

The covering density $\rho^* = N\sigma^2/A$, where A is the area of the unit cell, determines the edgelength $b = \sqrt{A}$ of the quadratic box the particles are confined in. All particles can occupy w predetermined positions in x-direction and any possible position between zero and b in y-direction.

- **X-wells**: The position of the wells p_w along the x-axis is calculated as follows:

$$\text{fib}(i) = i + \frac{1}{\tau}\,\text{int}\Big[\frac{i+1}{\tau}\Big]$$
$$p_w(i) = b\frac{\text{fib}(i)}{\text{fib}(w)},$$

where w is the number of wells, $0 \leq i < w$, $\tau = (1+\sqrt{5})/2$ is the golden ratio and fib(i) produces a Fibonacci chain.

- **Potential cutoff**: The cutoff radius r_{cut} is calculated with respect to the following condition:
$$V_{\mathrm{G}}(r_{\mathrm{cut}}) = 10^{-4} V_{\mathrm{G}}(r = 0).$$
For $r > r_{\mathrm{cut}}$, $V_{\mathrm{G}}(r)$ is set zero.

Crossover

Our system obviously has no (continuous) rotational symmetry and hence we could not employ a 2D version of the recombination process described in the prior section. Instead, we implemented a crossover mechanism more reminiscent of Deaven and Ho's [7] original cut and splice operator.

First, the centers of mass $\mathbf{c_1}$, $\mathbf{c_2}$ of the parent individuals p_1, p_2 are computed. A straight line $y = kx + d$ passing through both of these points cuts each individual in two parts. The child individual is composed of the particles of p_1 lying above this line ($y_{1,i} \geq kx_{1,i} + d$) and the particles of p_2 that lie below it ($y_{2,i} < kx_{2,i} + d$). Next, it has to be checked if the child contains the correct number of particles. If not, the straight line is shifted by a small distance ($d \to d + \delta$, usually $\delta = 0.1 \times b$, with boxsize b) and the process is repeated until a child with the correct number of particles is created.

Mutation

Here we used a mutation mechanism that differs slightly from the one used for cluster geometry optimization. There is only one mutation rate p_{mut}. For all particles of all individuals, a random number c between zero and one is chosen. If $c < p_{\mathrm{mut}}$, the particle is moved to a randomly determined x-well.

5 Results

In this chapter I will present the results of our simulations, along with information on the performance of the algorithm and, in the last subsection, visualizations of selected geometries of the systems discussed in this thesis.

5.1 Lennard-Jones Clusters

As pointed out earlier, small clusters of particles interacting via a standard Lennard-Jones potential are well studied ([36, 58]) and therefore act as a benchmark for groundstate predictions. We also investigated the case of a generalized L-J potential (2α-α potential) with $\alpha = 18$, which we compared to results published in [72].

5.1.1 Standard Lennard-Jones ($\alpha = 6$)

We did simulations for cluster particle numbers ranging from $N = 2$ to $N = 100$. For very low particle numbers, the energetic minimum was found after a simple local optimization, without any reproduction steps. For most of the higher particle numbers, our algorithm could reproduce the previously published ground states without any further problems (i.e., in a large percentage of runs and within a few hundred generations). Particle numbers around those of possible perfect Mackay icosahedra ($N = 13, 55$) needed comparatively small computation time, while clusters with $N = 38, 75, 76, 77, 98$ are particularly difficult because of anomalous structures (see section 3.1.1 and [37]). The cluster with $N = 38$ still did not pose any particular problems to our algorithm, the truncated octahedron ground state was found in two out of three runs within less than 1000 generations. However, the cases of $N = 75-77$ were much more difficult to treat. We found that for these clusters, our algorithm became "trapped" in icosahedral structures very easily; we point out that this also happened for basin hopping simulations [36]. Hence we tried an approach with a large number of simulations with relatively low generation numbers, which finally yielded the minimum energy structures (based on the Marks decahedron) for $N = 75, 76$ in a low percentage ($\sim 1\%$) of these runs. The ground state of the $N = 77$ cluster could only be found in a seeded run, where all starting geometries consisted of the coordinates of the previously found $N = 76$ minimum plus an additional, randomly positioned particle. For $N = 98$ an approach similar to $N = 75, 76$ was used. Here, it was easier to find the tetrahedral ground state than in the decahedral cases (success rate $\sim 4\%$).

Note that for the results shown in the following table, the parameters of the GA have not been optimized for each particular particle number. Putting more emphasis on an appropriate choice of these parameters, the number of successful runs and probably also the number of generations needed for successful convergence could be improved.

Table 5.1: Ground state energies of Lennard-Jones clusters

parameters					results		
particles	n_{ind}	e	$p_{\text{mut1}}/p_{\text{mut2}}$	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
2	8	3	0.01/0.33	10	-1.00000	0	3 out of 3
3	8	3	0.01/0.33	10	-3.00000	0	3 out of 3
4	8	3	0.01/0.33	10	-6.00000	0	3 out of 3
5	8	3	0.01/0.33	10	-9.10385	0	3 out of 3
6	8	3	0.01/0.33	10	-12.71206	0	3 out of 3
7	8	3	0.01/0.33	10	-16.50538	0	3 out of 3
8	8	3	0.01/0.33	10	-19.82149	0	3 out of 3
9	8	3	0.01/0.33	10	-24.11336	0	3 out of 3
10	8	3	0.01/0.33	10	-28.42253	0	3 out of 3
11	8	3	0.01/0.33	10	-32.75597	0	3 out of 3
12	8	3	0.01/0.33	10	-37.96760	0	3 out of 3
13	8	3	0.01/0.33	10	-44.32680	0	3 out of 3
14	8	3	0.01/0.33	10	-47.84516	0	3 out of 3
15	8	3	0.01/0.33	10	-52.32263	0	3 out of 3
16	8	3	0.01/0.33	10	-56.81574	0	3 out of 3
17	8	3	0.01/0.33	10	-61.31799	0	3 out of 3
18	8	3	0.01/0.33	10	-66.53095	9	3 out of 3
19	8	3	0.01/0.33	10	-72.65978	8	3 out of 3

Continued on next page

Table 5.1 – continued from previous page

parameters					results		
particles	n_{ind}	e	$p_{\text{mut1}}/p_{\text{mut2}}$	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
20	8	3	0.01/0.33	10	-77.17704	0	3 out of 3
21	8	3	0.01/0.33	50	-81.68457	1	3 out of 3
22	8	3	0.01/0.33	50	-86.80978	3	3 out of 3
23	8	3	0.01/0.33	50	-92.84447	8	3 out of 3
24	8	3	0.01/0.33	50	-97.34881	7	3 out of 3
25	8	3	0.01/0.33	50	-102.37266	5	3 out of 3
26	8	3	0.01/0.33	100	-108.31562	8	3 out of 3
27	8	3	0.01/0.33	100	-112.87358	47	3 out of 3
28	8	3	0.01/0.33	100	-117.82240	1	3 out of 3
29	8	3	0.01/0.33	100	-123.58737	6	3 out of 3
30	8	3	0.01/0.33	100	-128.28657	2	3 out of 3
31	8	3	0.01/0.33	200	-133.58642	47	3 out of 3
32	8	3	0.01/0.33	200	-139.63552	26	3 out of 3
33	8	3	0.01/0.33	200	-144.84272	33	3 out of 3
34	8	3	0.01/0.33	200	-150.04452	57	3 out of 3
35	8	3	0.01/0.33	200	-155.75664	33	3 out of 3
36	8	3	0.01/0.33	200	-161.82536	21	3 out of 3
37	8	3	0.01/0.33	200	-167.03367	18	3 out of 3
38	8	3	0.01/0.33	1000	-173.92842	262	2 out of 3
39	8	3	0.01/0.33	500	-180.03319	21	3 out of 3
40	8	3	0.01/0.33	500	-185.24984	17	3 out of 3
41	8	3	0.01/0.33	500	-190.53628	16	3 out of 3
42	8	3	0.01/0.33	500	-196.27753	18	3 out of 3
43	8	3	0.01/0.33	500	-202.36466	52	3 out of 3
44	8	3	0.01/0.33	500	-207.68873	118	3 out of 3
45	8	3	0.01/0.33	500	-213.78486	31	3 out of 3
46	8	3	0.01/0.33	500	-220.68033	32	3 out of 3
47	8	3	0.01/0.33	500	-226.01226	25	3 out of 3
48	8	3	0.01/0.33	500	-232.19953	10	3 out of 3
49	8	3	0.01/0.33	500	-239.09186	35	3 out of 3
50	8	3	0.01/0.33	500	-244.54993	31	3 out of 3
51	8	3	0.01/0.33	500	-251.25396	15	3 out of 3

Continued on next page

Table 5.1 – continued from previous page

parameters					results		
particles	n_{ind}	e	$p_{\text{mut1}}/p_{\text{mut2}}$	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
52	8	3	0.01/0.33	500	-258.22999	19	3 out of 3
53	8	3	0.01/0.33	500	-265.20302	17	3 out of 3
54	8	3	0.01/0.33	500	-272.20863	20	3 out of 3
55	8	3	0.01/0.33	500	-279.24847	8	3 out of 3
56	8	3	0.01/0.33	500	-283.64311	20	3 out of 3
57	8	3	0.01/0.33	500	-288.34262	15	3 out of 3
58	8	3	0.01/0.33	500	-294.37815	56	3 out of 3
59	8	3	0.01/0.33	500	-299.73807	113	3 out of 3
60	8	3	0.01/0.33	500	-305.87548	38	3 out of 3
61	8	3	0.01/0.33	500	-312.00890	44	1 out of 3
62	8	3	0.01/0.33	500	-317.35390	205	3 out of 3
63	8	3	0.01/0.33	500	-323.48973	126	2 out of 3
64	8	3	0.01/0.33	500	-329.62015	151	3 out of 3
65	8	3	0.01/0.33	500	-334.97153	65	1 out of 3
66	8	3	0.01/0.33	500	-341.11060	438	1 out of 3
67	8	3	0.01/0.33	500	-347.25201	311	1 out of 3
68	8	3	0.01/0.33	500	-353.39454	384	2 out of 3
69	8	3	0.01/0.33	1500	-359.88257	189	3 out of 3
70	8	3	0.01/0.33	1500	-366.89225	58	3 out of 3
71	8	3	0.01/0.33	1500	-373.34966	227	3 out of 3
72	8	3	0.01/0.33	1500	-378.63725	373	3 out of 3
73	8	3	0.01/0.33	1500	-384.78938	64	3 out of 3
74	8	3	0.01/0.33	1500	-390.90850	94	3 out of 3
75	8	3	0.15/0.33	600	-397.49233	192	2 out of 200
76	8	3	0.15/0.33	600	-402.89487	428	1 out of 200
*77	12	5	0.02/0.33	10	-409.08352	0	2 out of 10
78	8	3	0.01/0.33	2000	-414.79440	255	3 out of 3
79	8	3	0.01/0.33	2000	-421.81090	420	3 out of 3
80	8	3	0.01/0.33	2000	-428.08356	149	2 out of 3
81	8	3	0.01/0.33	2000	-434.34364	408	2 out of 3
82	8	3	0.01/0.33	2000	-440.55042	1402	1 out of 3
83	8	3	0.01/0.33	2000	-446.92409	1904	1 out of 3

Continued on next page

Table 5.1 – continued from previous page

parameters					results		
particles	n_{ind}	e	$p_{\mathrm{mut1}}/p_{\mathrm{mut2}}$	n_{gen}	energy	$n_{\mathrm{g\_min}}$	successful runs
84	8	3	0.01/0.33	2000	-452.65721	570	2 out of 3
85	8	3	0.01/0.33	2000	-459.05580	131	2 out of 3
86	8	3	0.01/0.33	2000	-465.38449	1484	1 out of 3
87	8	3	0.01/0.33	2000	-472.09816	809	1 out of 3
88	8	3	0.01/0.33	2000	-479.03263	203	3 out of 3
89	8	3	0.01/0.33	2000	-486.05391	441	2 out of 3
90	8	3	0.01/0.33	2000	-492.43391	185	3 out of 3
91	8	3	0.01/0.33	2000	-498.81106	88	3 out of 3
92	8	3	0.01/0.33	2000	-505.18531	987	3 out of 3
93	8	3	0.01/0.33	2000	-510.87769	93	2 out of 3
94	8	3	0.01/0.33	2000	-517.26413	997	3 out of 3
95	8	3	0.01/0.33	2000	-523.64021	488	2 out of 3
96	8	3	0.01/0.33	2000	-529.87915	663	2 out of 3
97	8	3	0.01/0.33	2000	-536.68138	257	2 out of 3
98	12	5	0.02/0.33	600	-543.66536	237	2 out of 50
99	8	3	0.01/0.33	2000	-550.66653	763	2 out of 3
100	8	3	0.1/0.33	10000	-557.03982	520	4 out of 5
* .... seeded runs							

Table 5.1: In this table all ground state energies in natural units of the Lennard-Jones potential (rounded to five decimal places) are listed. The number of individuals n_{ind}, the elitism parameter e, the mutation parameters p_{mut1} and p_{mut2}, the maximum number of generations for each run n_{gen}, the minimum number of generations until the corresponding structure was found over all runs $n_{\mathrm{g\_min}}$ and the contingent of successful runs are given as well.

Figure 5.1 shows the binding energies per particle, E/N, of these clusters as a function of particle number N. Since the cluster energy can be expressed as a sum of a bulk term and a surface term, i.e. $E = c_0N + c_1N^{2/3}$, for sufficiently large particle numbers, we fitted a curve

$$E/N = a_0 + a_1N^{-1/3}, \tag{5.1}$$

with $a_0 = -7.97422$, $a_1 = 11.3074$ to data points with $N \geq 15$.

Some part of the growth pattern for Lennard-Jones clusters is shown in the visualization section at the end of this chapter (figure 5.4). It is noted that with increasing N, a growing layer of particles is added to the 13-particle Mackay icosahedron at the core.

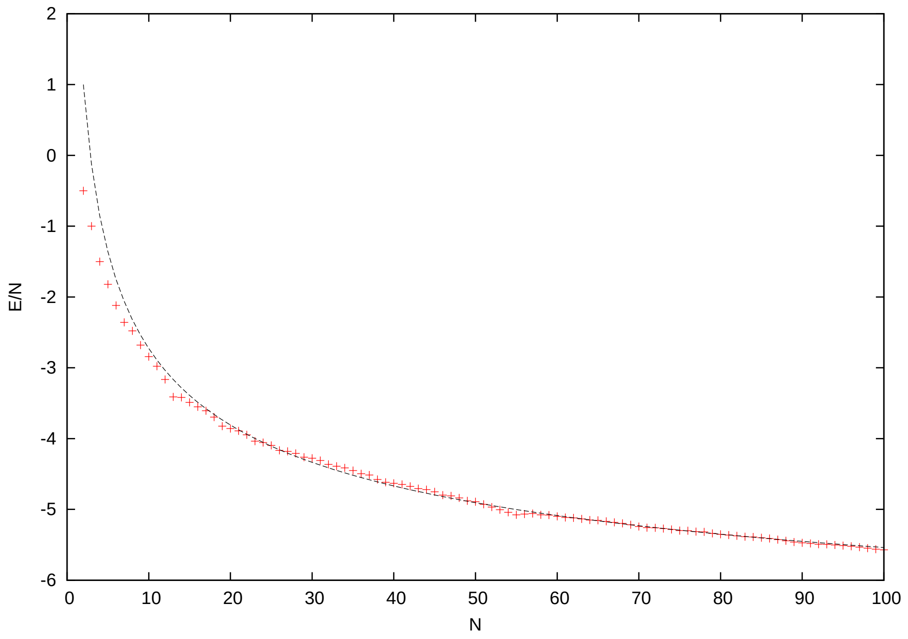


Figure 5.1: Binding energies per particle, E/N, as a function of N, $2 \leq N \leq 100$, for Lennard-Jones clusters; the dashed line represents a fit to the data (with $N \geq 15$) using the functional form of equation 5.1, with parameters $a_0 = -7.97422$, $a_1 = 11.3074$.

5.1.2 $\alpha = 18$

For this potential, we ran simulations for particle numbers $2 \leq N \leq 60$. As pointed out in [72], for very small clusters $N < 10$, the resulting geometries are the same as those encountered using the standard Lennard-Jones potential. However, for larger N values, the shorter range of the $(\alpha = 18)$-potential induces a significant change in the structures. For example, at $N = 13$, the in the Lennard-Jones case particularly stable perfect Mackay icosahedron is not the ground state for this potential.

The shorter range of the potential causes the potential energy hypersurface to be more rugged and therefore calculations tend to be significantly more time consuming

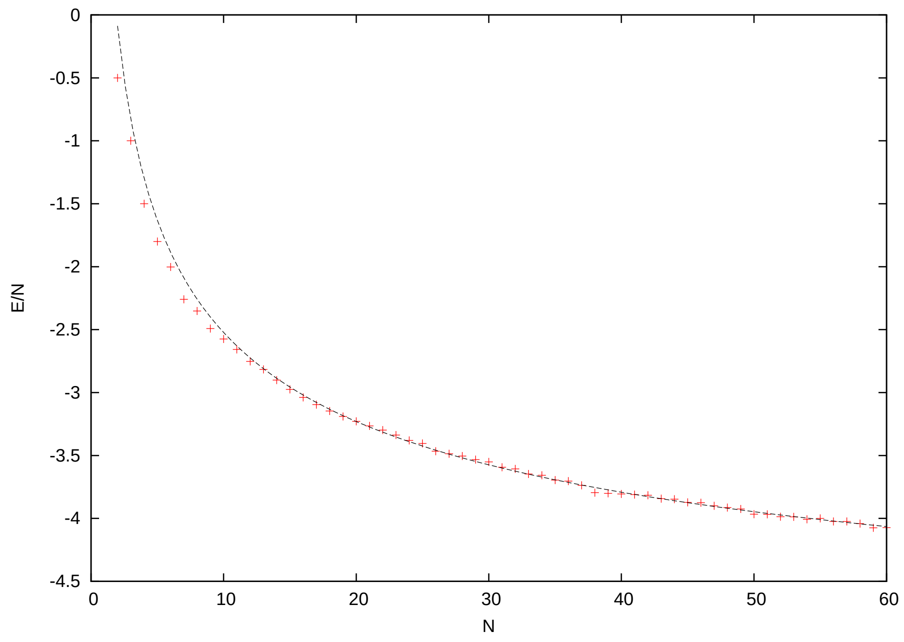


Figure 5.2: Binding energies per particle, E/N, as a function of N, $2 \leq N \leq 60$, for $\alpha = 18$ clusters; the dashed line represents a fit to the data (with $N \geq 15$) using the functional form of equation 5.1, with parameters $a_0 = -5.95226$, $a_1 = 7.38958$.

than in the standard Lennard-Jones case.

As shown in the following table, our algorithm could reproduce most of the particle configurations reported in [72]. In addition, for several cases the algorithm produced configurations with even lower ground state energy than the ones reported in in [72].

Table 5.2: Ground state energies of $\alpha = 18$ clusters

parameters					results		
particles	n_{ind}	e	$p_{\mathrm{mut1}}/p_{\mathrm{mut2}}$	n_{gen}	energy	$n_{\mathrm{g\_min}}$	successful runs
2	8	3	0.01/0.33	10	-1.00000	0	3 out of 3
3	8	3	0.01/0.33	10	-3.00000	0	3 out of 3
4	8	3	0.01/0.33	10	-6.00000	1	3 out of 3
5	8	3	0.01/0.33	10	-9.00029	1	3 out of 3
6	8	3	0.01/0.33	100	-12.01171	11	3 out of 3
7	8	3	0.01/0.33	100	-15.81238	4	3 out of 3
8	8	3	0.01/0.33	100	-18.81303	3	3 out of 3

Continued on next page

Table 5.2 – continued from previous page

parameters					results		
particles	n_{ind}	e	$p_{\text{mut1}}/p_{\text{mut2}}$	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
9	8	3	0.01/0.33	100	-22.42715	2	3 out of 3
10	8	3	0.01/0.33	100	-25.74640	16	3 out of 3
11	8	3	0.01/0.33	100	-29.23267	2	3 out of 3
12	8	3	0.01/0.33	200	-33.02595	23	3 out of 3
13	8	3	0.01/0.33	200	-36.60489	30	1 out of 3
14	8	3	0.01/0.33	500	-40.61001	91	2 out of 3
15	8	3	0.01/0.33	500	-44.61539	82	3 out of 3
16	8	3	0.01/0.33	500	-48.62077	63	3 out of 3
17	8	3	0.01/0.33	500	-52.62615	47	3 out of 3
18	8	3	0.01/0.33	500	-56.63179	85	3 out of 3
19	8	3	0.01/0.33	500	-60.60114	128	3 out of 3
20	8	3	0.01/0.33	500	-64.56643	113	2 out of 3
21	8	3	0.01/0.33	1500	-68.55246	458	2 out of 3
22	8	3	0.01/0.33	1500	-72.55790	166	2 out of 3
23	8	3	0.01/0.33	1500	-76.76435	43	2 out of 3
24	8	3	0.01/0.33	1500	-81.11050	850	1 out of 3
25	8	3	0.01/0.33	5000	-85.11895	513	2 out of 3
26	8	3	0.01/0.33	5000	-90.12737	3100	1 out of 3
27	8	3	0.01/0.33	20000	-94.13163	231	2 out of 3
28	8	3	0.01/0.33	20000	-98.13618	7929	2 out of 3
29	8	3	0.01/0.33	20000	-102.44906	560	1 out of 3
$^{-}30$	8	3	0.01/0.33	20000	-106.53802	4212	1 out of 3
31	8	3	0.01/0.33	20000	-111.40195	356	3 out of 3
32	8	3	0.01/0.33	20000	-115.40657	819	3 out of 3
33	8	3	0.01/0.33	20000	-120.34364	61	3 out of 3
$^{-}34$	8	3	0.01/0.33	20000	-124.34829	2594	2 out of 3
35	8	3	0.01/0.33	20000	-129.32885	7902	2 out of 3
36	8	3	0.01/0.33	20000	-133.33383	1072	3 out of 3
37	8	3	0.01/0.33	20000	-138.25002	2209	2 out of 3
38	8	3	0.01/0.33	20000	-144.21675	528	1 out of 3
39	8	3	0.01/0.33	20000	-148.22109	6199	1 out of 3
$^{-}40$	8	3	0.01/0.33	20000	-152.22544	726	3 out of 3

Continued on next page

Table 5.2 – continued from previous page

parameters					results		
particles	n_{ind}	e	$p_{\text{mut1}}/p_{\text{mut2}}$	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
41	8	3	0.01/0.33	20000	-156.24226	2987	2 out of 3
$^{-}$42	8	3	0.01/0.33	20000	-160.25023	1798	3 out of 3
43	8	3	0.01/0.33	20000	-165.25180	7600	2 out of 3
44	8	3	0.01/0.33	20000	-169.26026	535	2 out of 3
$^{-}$45	8	3	0.01/0.33	20000	-174.26870	5987	2 out of 3
46	8	3	0.01/0.33	20000	-178.28141	10391	1 out of 3
47	8	3	0.01/0.33	20000	-183.27824	12711	1 out of 3
$^{-}$48	8	3	0.01/0.33	20000	-187.86271	10609	1 out of 3
$^{-}$49	8	3	0.01/0.33	20000	-192.30747	2459	2 out of 3
50	8	3	0.01/0.33	30000	-198.30838	3529	4 out of 10
$^{-}$51	8	3	0.01/0.33	30000	-202.31685	4497	2 out of 3
52	8	3	0.01/0.33	30000	-207.32530	1866	1 out of 3
53	8	3	0.01/0.33	30000	-211.33377	358	2 out of 3
$^{-}$54	8	3	0.01/0.33	30000	-216.34221	24880	2 out of 3
$^{-}$55	8	3	0.01/0.33	30000	-219.95547	20690	1 out of 3
56	12	5	0.01/0.33	1000	-225.35122	575	1 out of 20
57	8	3	0.01/0.33	30000	-229.37686	27424	3 out of 3
$^{-}$58	12	5	0.01/0.33	100000	-234.38534	433	2 out of 8
59	8	3	0.01/0.33	30000	-240.38800	28806	1 out of 3
60	12	5	0.01/0.33	100000	-244.39258	629	6 out of 8
$^{-}$ .... minimum energetically lower than in [72]							

Table 5.2: Ground state energies and GA parameters for $\alpha = 18$ clusters.

The binding energies per particle as a function of the particle number are shown in figure 5.2.

Figure 5.6 shows some of the computed cluster structures. For $N = 13-20$, a growth pattern (indicated by different colors) can be observed. The cluster geometries for $N = 30$, 42 and 48 particles are different from those presented in [72] and have lower energies.

5.2 Dzugutov Clusters

Since the Dzugutov potential [70] has attractive and repulsive components, the potential energy hypersurface for these clusters is even more complicated than in the cases described before. We have investigated Dzugutov clusters for particle numbers $N = 2$ to $N = 50$ and have compared our results to the ones published in [71], which are also available online [58]. Our algorithm managed to reproduce all minima except the one at $N = 45$, which corresponds to a very elongated geometry with energy -92.91414, while the local minimum our algorithm found has an energy value of -92.88544 (for a comparison of the corresponding structures, see figure 5.7).

Table 5.3: Ground state energies of Dzugutov clusters

parameters					results		
particles	n_{ind}	e	$p_{\text{mut1}}/p_{\text{mut2}}$	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
2	8	3	0.01/0.33	10	-0.58144	0	3 out of 3
3	8	3	0.01/0.33	10	-1.74431	1	3 out of 3
4	8	3	0.01/0.33	10	-3.48862	2	3 out of 3
5	8	3	0.01/0.33	100	-5.19500	9	3 out of 3
6	8	3	0.01/0.33	100	-6.89547	6	3 out of 3
7	8	3	0.01/0.33	100	-9.11048	12	3 out of 3
8	8	3	0.01/0.33	100	-10.80097	13	3 out of 3
9	8	3	0.01/0.33	100	-13.00434	16	3 out of 3
10	8	3	0.01/0.33	500	-15.18230	22	3 out of 3
11	8	3	0.01/0.33	500	-17.35419	28	3 out of 3
12	8	3	0.01/0.33	500	-19.97396	14	3 out of 3
13	8	3	0.01/0.33	500	-23.20683	37	3 out of 3
14	8	3	0.01/0.33	500	-24.80015	43	3 out of 3
15	8	3	0.01/0.33	500	-26.89233	35	3 out of 3
16	8	3	0.01/0.33	500	-28.93500	24	3 out of 3
17	8	3	0.01/0.33	500	-31.07137	30	3 out of 3
18	8	3	0.01/0.33	500	-33.38890	33	3 out of 3
19	8	3	0.01/0.33	500	-36.38730	100	3 out of 3
20	8	3	0.01/0.33	500	-38.18598	52	3 out of 3
21	8	3	0.01/0.33	500	-40.16478	23	3 out of 3
22	8	3	0.01/0.33	500	-42.23248	56	3 out of 3

Continued on next page

Table 5.3 – continued from previous page

parameters					results		
particles	n_{ind}	e	$p_{\text{mut1}}/p_{\text{mut2}}$	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
23	8	3	0.01/0.33	500	-45.09705	353	1 out of 3
24	8	3	0.01/0.33	500	-46.96753	384	1 out of 3
25	8	3	0.01/0.33	500	-49.35195	71	2 out of 3
26	8	3	0.01/0.33	1500	-51.16203	217	3 out of 3
27	8	3	0.01/0.33	1500	-53.32167	147	3 out of 3
28	8	3	0.01/0.33	5000	-55.48276	288	3 out of 3
29	8	3	0.01/0.33	5000	-58.09194	4616	1 out of 3
30	8	3	0.01/0.33	5000	-59.90790	1798	2 out of 3
31	8	3	0.01/0.33	5000	-62.38719	5491	2 out of 3
32	8	3	0.01/0.33	5000	-64.06448	3208	2 out of 3
33	8	3	0.01/0.33	20000	-66.89364	11301	2 out of 3
34	8	3	0.01/0.33	20000	-68.69681	2290	3 out of 3
35	8	3	0.01/0.33	20000	-71.13407	9151	1 out of 3
36	8	3	0.01/0.33	20000	-73.05979	7646	2 out of 3
37	8	3	0.01/0.33	30000	-75.47329	12195	1 out of 10
38	8	3	0.01/0.33	20000	-78.21269	655	2 out of 3
39	8	3	0.01/0.33	20000	-80.02804	746	3 out of 3
40	8	3	0.01/0.33	20000	-82.00776	2543	2 out of 3
41	8	3	0.01/0.33	30000	-84.19359	2689	4 out of 6
42	8	3	0.01/0.33	30000	-86.88835	562	2 out of 6
43	8	3	0.01/0.33	30000	-88.70720	1429	4 out of 6
44	8	3	0.1/0.33	50000	-91.02178	4861	2 out of 6
$^{+}$45	8	3	0.01/0.33	50000	-92.88544	-	0 out of 20
46	10	3	0.01/0.1	100000	-95.60089	50584	2 out of 10
47	10	3	0.01/0.1	100000	-97.42453	2486	1 out of 10
48	8	3	0.01/0.33	30000	-99.87780	1982	3 out of 6
49	10	3	0.1/0.33	100000	-101.99289	1958	2 out of 5
50	8	3	0.01/0.33	30000	-104.36619	6388	2 out of 6
$^{+}$......... minimum energetically higher than in [58]							

Table 5.3: Ground state energies and GA parameters for Dzugutov clusters.

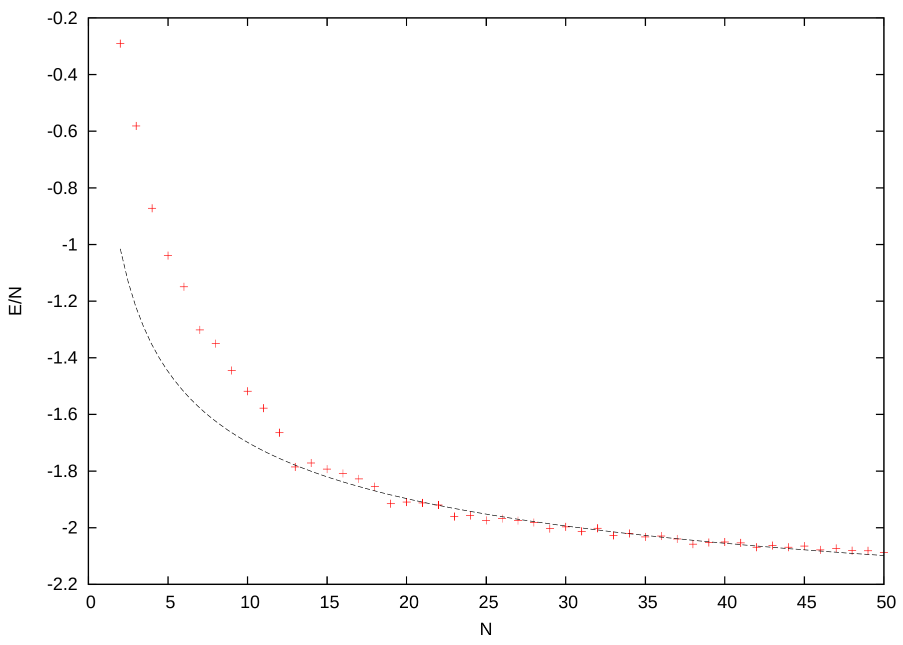


Figure 5.3: Binding energies per particle, E/N, as a function of N, $2 \leq N \leq 50$, for Dzugutov clusters; the dashed line represents a fit to the data (with $N \geq 15$) using the functional form of equation 5.1, with parameters $a_0 = -2.66114$, $a_1 = 2.07367$.

The structures of small Dzugutov clusters with up to 21 particles are based on Mackay icosahedra, just like for Lennard-Jones clusters. For larger cluster sizes, the structures are still based on 13-atom icosahedra, but do not follow an overlayer growth pattern. They rather tend to form non-compact elongated structures, a few examples of these are shown in figure 5.7. The geometry for $N = 38$ is typical for cluster sizes around 40 particles. The global minimum for $N = 45$ [58], which has not yet been identified by our algorithm can be seen as an expansion the 33 particle structure. The structure with 50 particles represents the configurations consisting of face sharing icosahedra and a cavity, which are dominant in this cluster size region.

5.3 2D-System

For our 2D-system we ran calculations for a fixed number $w = 8$ of potential wells in x-direction. As particle numbers, we chose integer multiples of w, namely $N = 24, 32, 40$. For each number of particles N, we searched for the energetically most favorable configurations at 20 density points $\rho^* = N\sigma^2/A$ in the interval $[0.1, 2]$. The results are presented in the following tables (energies for equivalent structures where reproducible up to at least two digits).

24 Particles

Table 5.4: Ground state energies and GA parameters for our 2D system - 24 particles

parameters					results		
ρ^*	n_{ind}	e	p_{mut}	n_{gen}	energy	$n_{\mathrm{g\_min}}$	successful runs
0.1	20	3	0.01	10000	0.01347	1815	1 out of 10
0.2	20	3	0.01	10000	1.38706	43	10 out of 10
0.3	20	3	0.01	10000	7.29447	37	10 out of 10
0.4	20	3	0.01	10000	17.43944	152	9 out of 10
0.5	20	3	0.01	10000	30.10736	115	5 out of 10
0.6	20	3	0.01	10000	44.01761	54	3 out of 10
0.7	20	3	0.01	10000	58.49851	9464	1 out of 10
0.8	20	3	0.01	10000	73.24191	78	8 out of 10
0.9	20	3	0.01	10000	88.03630	339	10 out of 10
1.0	20	3	0.01	10000	102.92844	11	10 out of 10
1.1	20	3	0.01	10000	117.86571	200	9 out of 10
1.2	20	3	0.01	10000	132.82668	683	8 out of 10
1.3	20	3	0.01	10000	147.81078	38	10 out of 10
1.4	20	3	0.01	10000	162.81442	72	9 out of 10
1.5	20	3	0.01	10000	177.82092	103	10 out of 10
1.6	20	3	0.01	10000	192.84476	2447	5 out of 10
1.7	20	3	0.01	10000	207.87000	1652	4 out of 10
1.8	20	3	0.01	10000	222.89690	225	9 out of 10
1.9	20	3	0.01	10000	237.91910	1591	3 out of 10
2.0	20	3	0.01	10000	252.95082	704	5 out of 10

For $N = 24$ particles and low densities up to $\rho^* = 0.7$, the energetically most favorable structure we found is a rather regular one. Each well contains three particles and the interparticle distances in y-direction are equal. For densities between $\rho^* = 0.8$ and $\rho^* = 1.0$, the wells contain $[\,4\,|\,2\,|\,3\,|\,4\,|\,3\,|\,2\,|\,4\,|\,2\,]$ particles and the particles seem to arrange around a central hexagon. For $1.1 \leq \rho^* \leq 1.5$, a structure with

[5 | 2 | 3 | 4 | 2 | 3 | 4 | 1] particles in the wells is dominant. At even higher densities $1.6 \leq \rho^* \leq 1.8$ the particle distribution over the wells remains the same, but the configuration in y-direction changes to a structure characterized by a central three-particle alignment. For the highest densities $\rho^* = 1.9, 2.0$ that we considered, a [3 | 3 | 3 | 3 | 4 | 2 | 2 | 4] structure seems to be the most favorable one. Examples for all these particle alignments can be found in figure 5.8.

32 Particles

Table 5.5: Ground state energies and GA parameters for our 2D system - 32 particles

parameters					results		
ρ^*	n_{ind}	e	p_{mut}	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
0.1	20	3	0.01	10000	0.02988	1921	1 out of 10
0.2	20	3	0.01	10000	2.55851	154	10 out of 10
0.3	20	3	0.01	10000	11.12631	93	10 out of 10
0.4	20	3	0.01	10000	24.64652	13	10 out of 10
0.5	20	3	0.01	10000	41.25811	27	10 out of 10
0.6	20	3	0.01	10000	59.57621	23	10 out of 10
0.7	20	3	0.01	10000	78.70509	16	10 out of 10
0.8	20	3	0.01	10000	98.05634	18	10 out of 10
0.9	20	3	0.01	10000	117.67909	18	10 out of 10
1.0	20	7	0.01	20000	137.46087	990	9 out of 10
1.1	20	7	0.01	20000	157.29585	298	10 out of 10
1.2	20	7	0.01	20000	177.19281	179	10 out of 10
1.3	20	7	0.01	20000	197.13286	920	10 out of 10
1.4	20	7	0.01	20000	217.11848	403	10 out of 10
1.5	20	7	0.01	20000	237.12577	319	10 out of 10
1.6	20	7	0.01	20000	257.14706	18	10 out of 10
1.7	20	7	0.01	20000	277.17422	234	10 out of 10
1.8	20	7	0.01	20000	297.21241	803	6 out of 10
1.9	20	7	0.01	20000	317.25406	4513	3 out of 10
2.0	20	7	0.01	20000	337.29056	280	10 out of 10

In the $N = 32$ case, we found even more different structures to be energetically most favorable in different density domains. For $\rho^* = 0.1$, an alignment with [6 | 3 | 3 | 6 | 3 | 4 | 3 | 4] particles in the wells had the lowest energy value, while for $\rho^* = 0.2$ this was the case for a more symmetric [5 | 4 | 3 | 6 | 3 | 4 | 3 | 4] - structure. In the regime of $0.3 \leq \rho^* \leq 0.6$, a regular structure with four particles in every well and equal distances in y-direction seems to be dominant. For $0.7 \leq \rho^* \leq 0.9$, the energetically most favorable configuration we found contains [5 | 4 | 3 | 6 | 3 | 4 | 3 | 4] particles in the wells, similar to the $\rho^* = 0.2$ case, but with a slightly different alignment in y-direction. At $\rho^* = 1.0$ we observe a change to a [5 | 4 | 3 | 6 | 3 | 4 | 4 | 3] structure. In the rather large density range of $1.1 \leq \rho^* \leq 1.7$ a [5 | 4 | 3 | 6 | 2 | 5 | 3 | 4] alignment, which can be characterized by a central six-particle rhombus, seems to be dominant. For higher densities, the particle numbers per well stay the same, but the configuration in y-direction changes: at $\rho^* = 1.8$ and $\rho^* = 2.0$ a structure with a central two-particle alignment and columns in y-direction is energetically most favorable. For $\rho^* = 1.9$ a structure with yet another particle configuration in y-direction was found. For visualizations of these structures, see figures 5.9 and 5.10.

40 Particles

Table 5.6: Ground state energies and GA parameters for our 2D system - 40 particles

parameters					results		
ρ^*	n_{ind}	e	p_{mut}	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
0.1	20	3	0.01	10000	0.04885	18791	1 out of 10
0.2	20	3	0.01	10000	3.40908	649	10 out of 10
0.3	20	3	0.01	10000	14.45650	24	10 out of 10
0.4	20	3	0.01	10000	31.61355	36	10 out of 10
0.5	20	3	0.01	10000	52.37797	40	10 out of 10
0.6	20	3	0.01	10000	75.08839	10	10 out of 10
0.7	20	3	0.01	10000	98.65277	12	10 out of 10
0.8	20	3	0.01	10000	122.70257	9	10 out of 10
0.9	20	3	0.01	10000	147.12781	8	10 out of 10
1.0	20	3	0.01	10000	171.79135	7	10 out of 10
1.1	20	3	0.01	10000	196.59867	18	10 out of 10
1.2	20	3	0.01	10000	221.50024	6	10 out of 10
1.3	20	3	0.01	10000	246.42798	165	10 out of 10
Continued on next page							

Table 5.6 – continued from previous page

parameters					results		
ρ^*	n_{ind}	e	p_{mut}	n_{gen}	energy	$n_{\text{g\_min}}$	successful runs
1.4	20	3	0.01	10000	271.39071	18	10 out of 10
1.5	20	3	0.01	10000	296.37869	14	10 out of 10
1.6	20	3	0.01	10000	321.38917	9	10 out of 10
1.7	20	3	0.01	10000	346.41903	28	10 out of 10
1.8	20	3	0.01	10000	371.46391	13	10 out of 10
1.9	20	3	0.01	10000	396.51368	24	10 out of 10
2.0	20	7	0.01	10000	421.56449	9	10 out of 10

For $N = 40$ particles only four different structures were found to be energetically most favorable in the investigated density regime: For $\rho^* = 0.1$ an alignment with $[\,7\,|\,5\,|\,4\,|\,7\,|\,5\,|\,4\,|\,4\,|\,4\,]$ particles per well was found to have the lowest energy value. Between $\rho^* = 0.2$ and $\rho^* = 0.6$ a $[\,6\,|\,4\,|\,4\,|\,6\,|\,5\,|\,5\,|\,5\,|\,5\,]$-configuration was identified, where the particles arrange in zig-zag patterns. For $0.7 \le \rho^* \le 1.2$, a $[\,7\,|\,4\,|\,5\,|\,6\,|\,5\,|\,4\,|\,5\,|\,4\,]$-structure characterized by a central two-particle alignment was found to have the lowest energy value. For higher densities $1.3 \le \rho^* \le 2.0$ the distribution of particles over the wells changes to $[\,7\,|\,4\,|\,5\,|\,6\,|\,5\,|\,5\,|\,6\,|\,3\,]$ and two two-particle alignments can be identified. For visualizations of these structures, see figure 5.11.

5.4 Visualizations

On the following pages, 3D images of some of the structures computed by our cluster geometry optimizing algorithm are presented. All of these were generated using the program *PyMOL* [80] and employ a perspective camera (i.e. there is a vanishing point).

For all images showing our 2D system (also created with *PyMOL*), particles within the unit cell are shown as red spheres, while their periodic images appear in gray. Please note that mirror symmetries are possible.

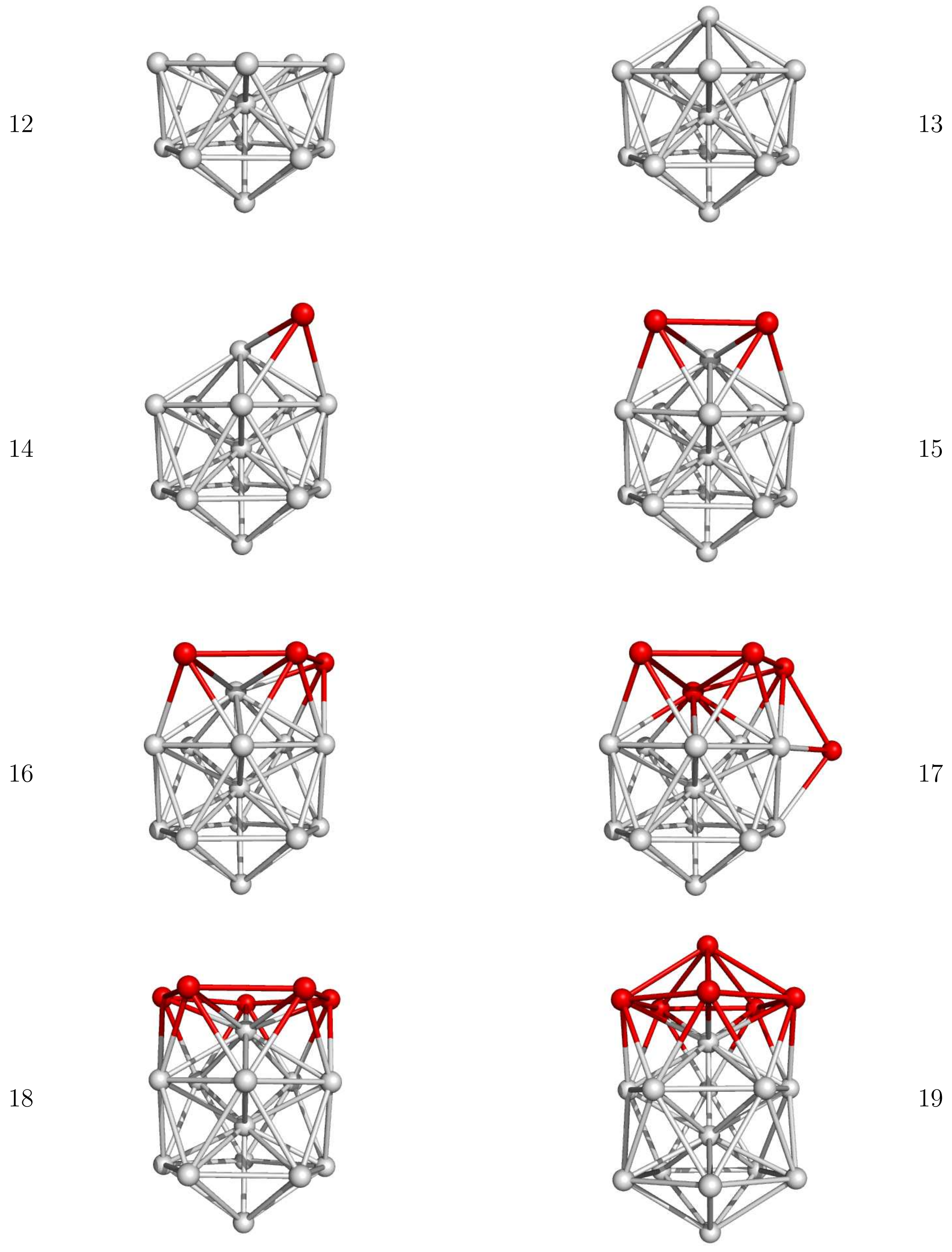


Figure 5.4: Growth pattern for Lennard-Jones clusters with $N = 12 - 19$. Notice the variant structure at $N = 17$.

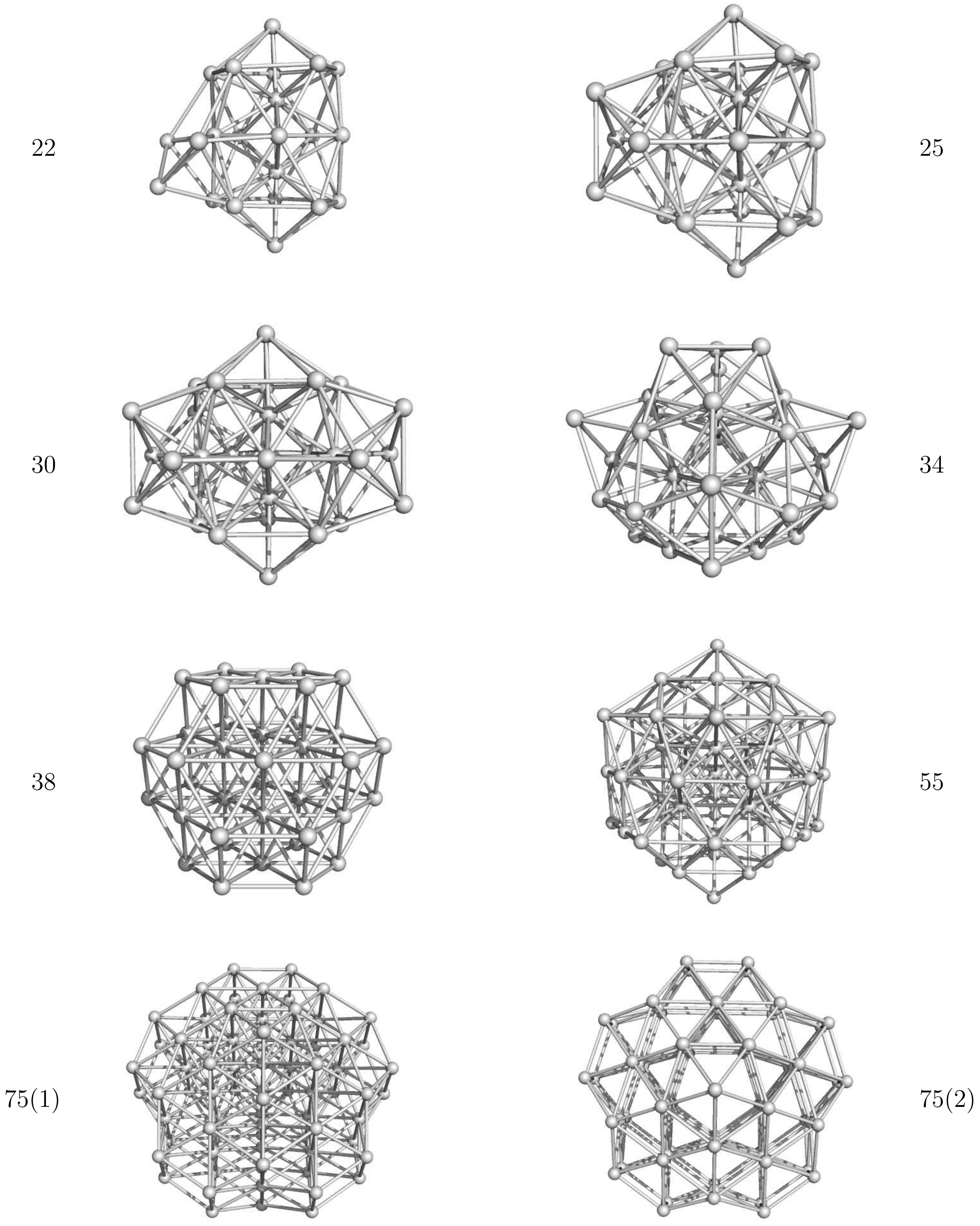


Figure 5.5: Lennard-Jones clusters with 22, 25, 30, 34, 38 (octahedral), 55 (perfect Mackay icosahedron) and 75 (decahedral, two different perspectives) particles.

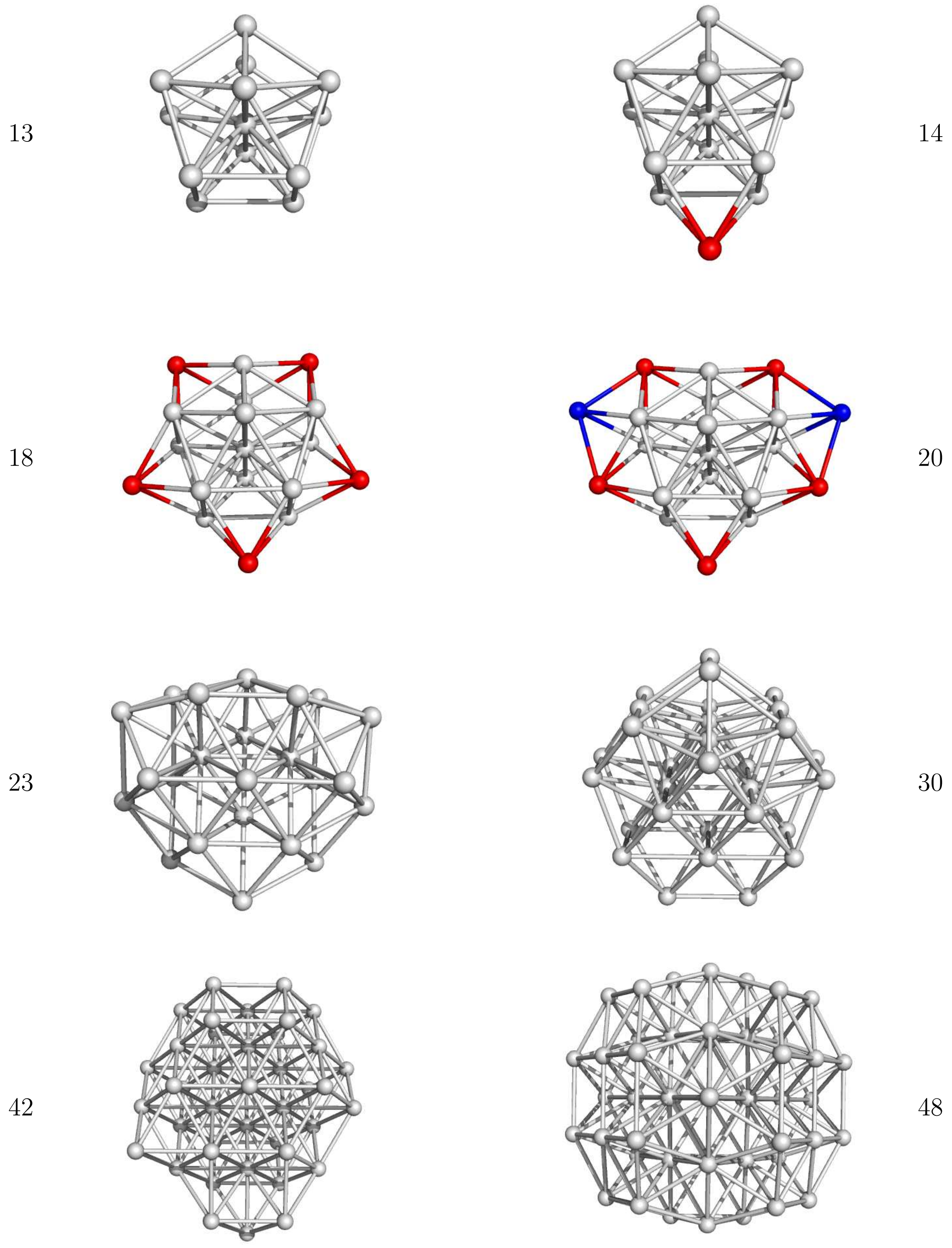


Figure 5.6: Clusters of particle interacting via the ($\alpha = 18$)-potential with 13, 14, 18, 20, 23, 30, 42 and 48 particles. Different colors indicate an observable growth pattern. See text for comments.

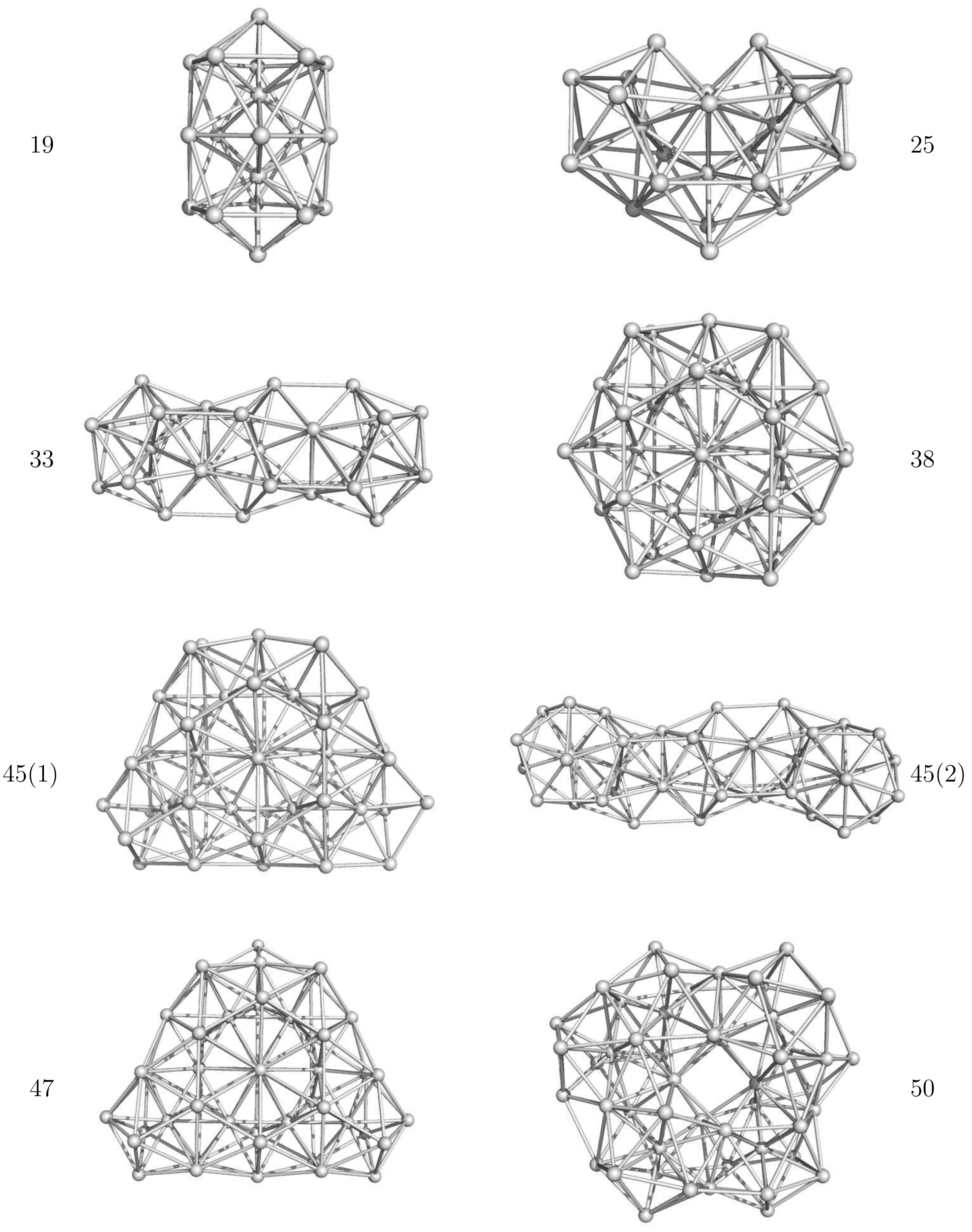


Figure 5.7: Dzugutov clusters with 19, 25, 33, 38, 45 (left: minimum found by our algorithm, right: global minimum [58]), 47, and 50 particles. See text for comments.

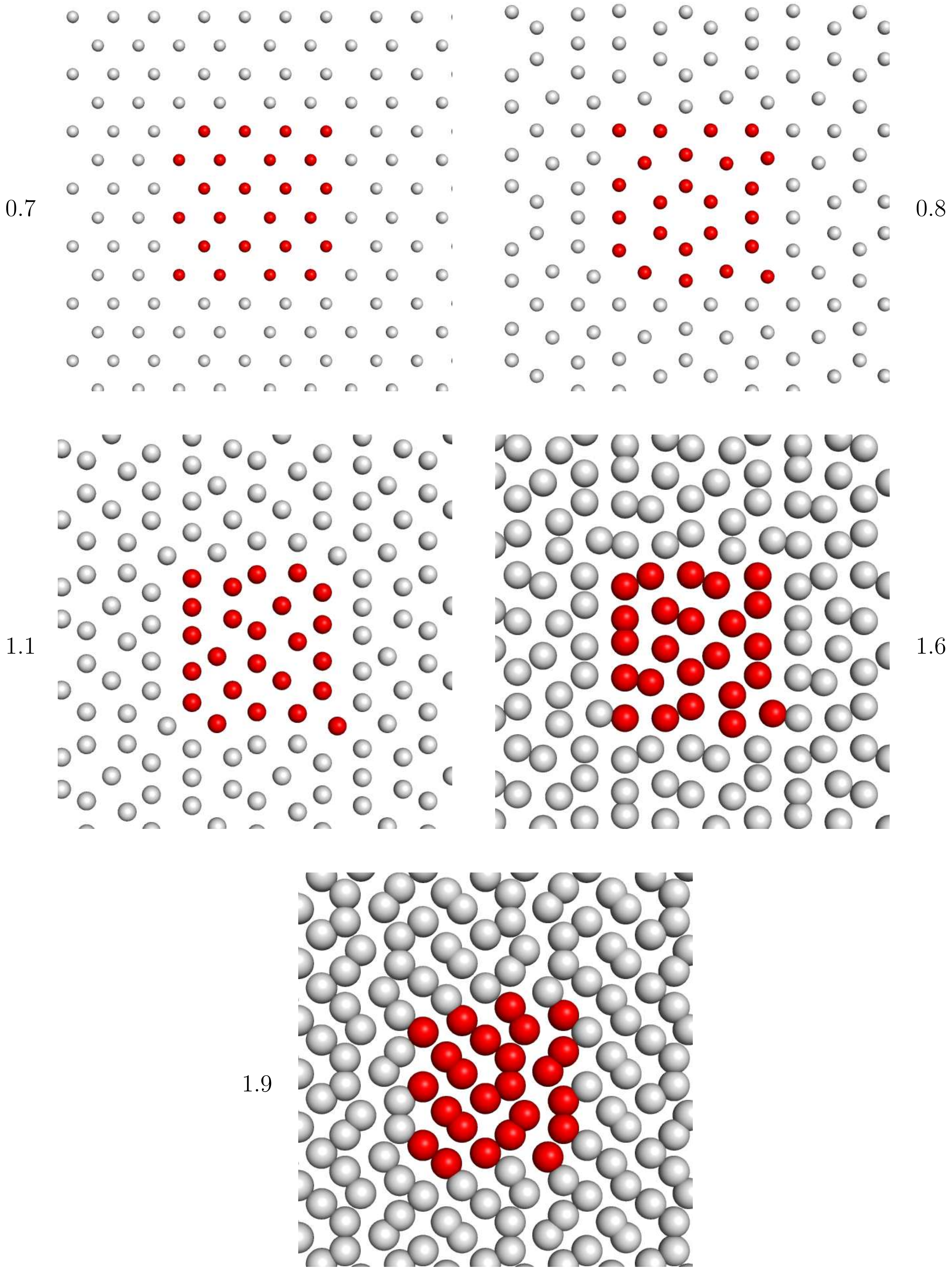


Figure 5.8: 2D System with 24 particles: energetically most favorable structures for $\rho^* = 0.7, 0.8, 1.1, 1.6, 1.9$. Particles in the unit cell are represented by red spheres, their periodic images appear as gray spheres.

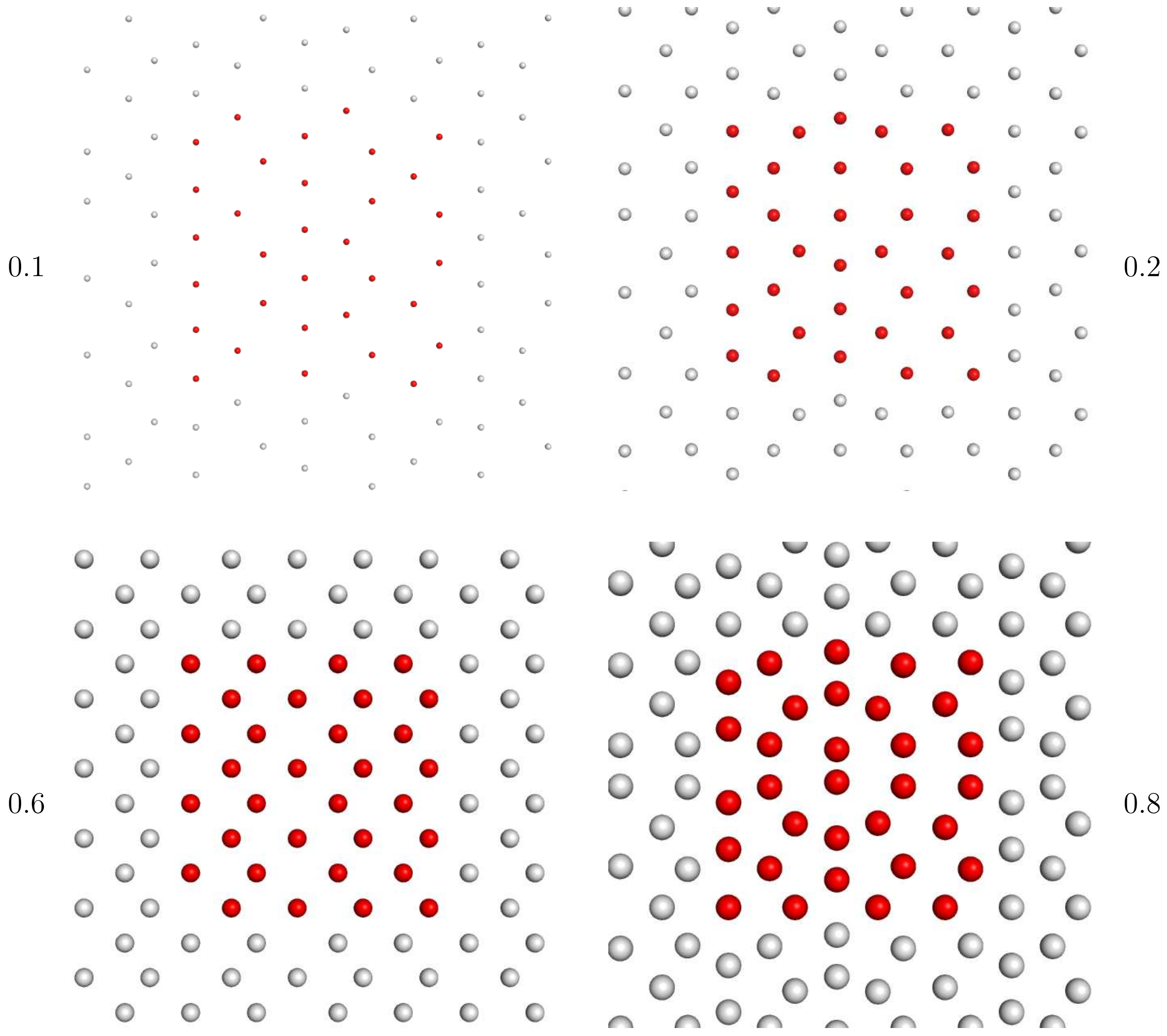


Figure 5.9: 2D System with 32 particles: energetically most favorable structures for $\rho^* = 0.1, 0.2, 0.6, 0.8$. For better visibility, the particles at $\rho^* = 0.1$ and 0.2 are enlarged by factor two.

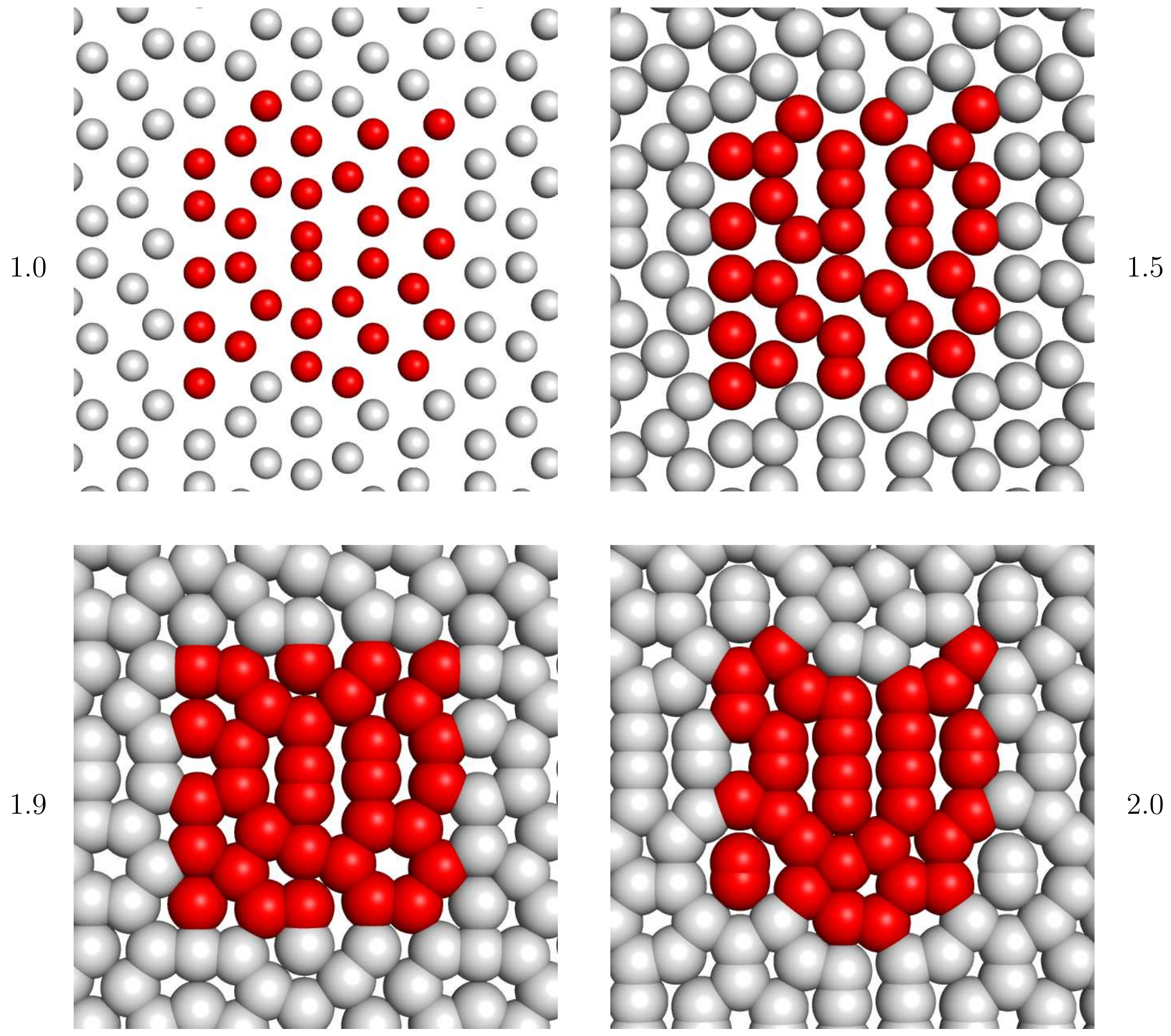


Figure 5.10: 2D System with 32 particles: energetically most favorable structures for $\rho^* = 1.0, 1.5, 1.9, 2.0$.

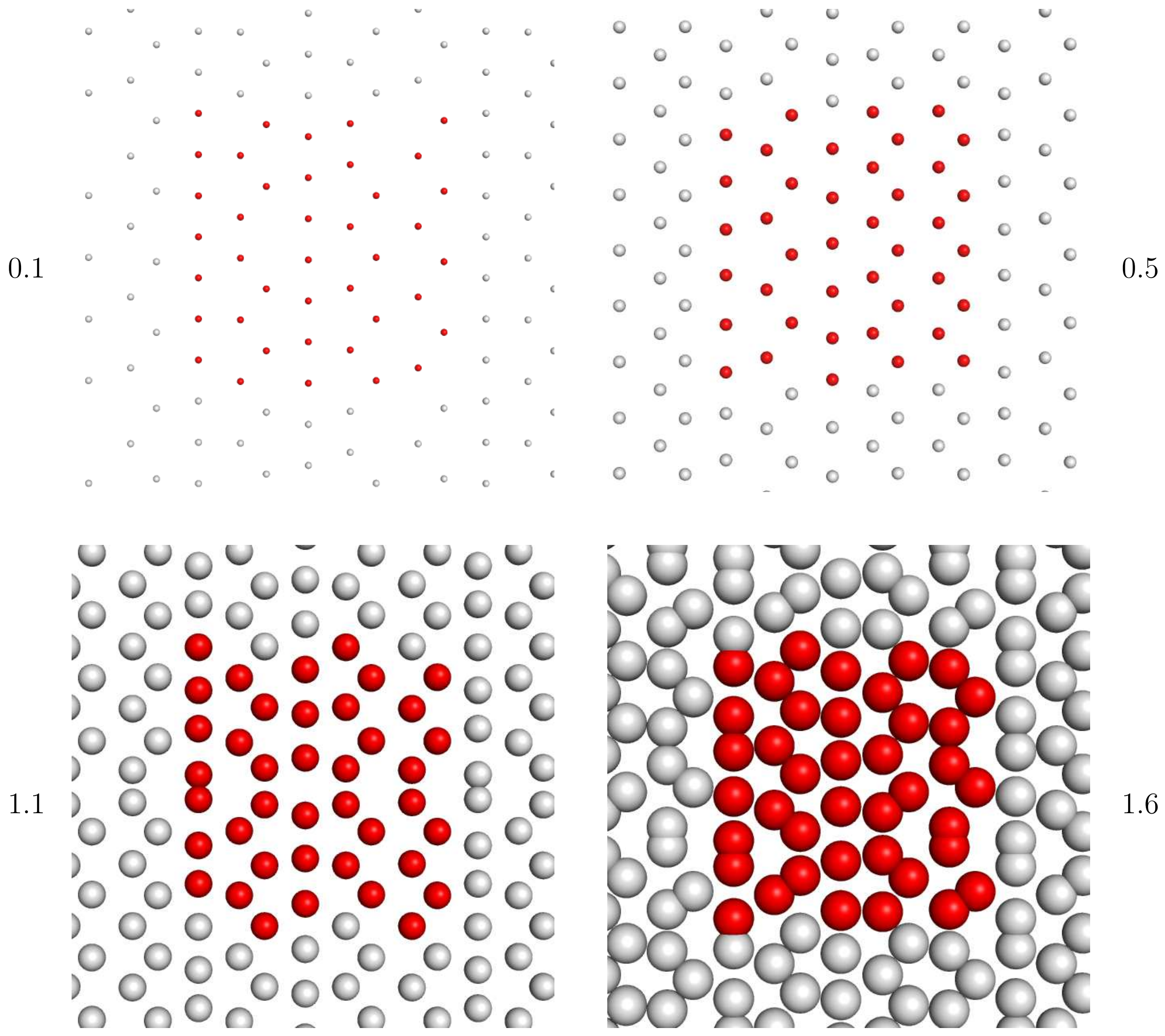


Figure 5.11: 2D System with 40 particles: energetically most favorable structures for $\rho^* = 0.1, 0.5, 1.1, 1.6$. For better visibility, the particles at $\rho^* = 0.1$ are enlarged by factor two.

6 Summary

In this work we used a genetic (phenotype) algorithm as a search strategy for energetically favorable geometrical configurations of (finite numbers of) particles. The pairwise interaction of the particles in the investigated systems is given via potential functions depending on the interparticle distance. Our implementation of such an algorithm is mainly based on the ideas introduced in [7, 35, 39].

The first problem we studied was cluster geometry optimization. As a benchmark for our algorithm, we tried to reproduce the global energy minima published in [58] for Lennard-Jones clusters with particle numbers up to 100 and for Dzugutov clusters with particle numbers up to 50. Therein, we succeeded in all but one case. Furthermore, we were able to reproduce, and in eleven cases improve, the global minima for 2α-α clusters with $\alpha = 18$ published in [72] for particle numbers up to 60.

As our second problem, we investigated a two-dimensional system inspired by certain properties of quasicrystalline materials [73]. This was done using a slightly altered version of the cluster geometry optimizing algorithm. For this system, we obtained tentative global minimum energy configurations for different particle numbers and densities.

We are optimistic that the algorithm we developed (and further improved versions of it) will be useful in future investigations, especially in my planned PhD studies.

Possible improvements of our program are directed mutation operations and especially a more powerful niching method based on the techniques pointed out in [35].

Bibliography

[1] C.N. Likos, *Effective Interactions in Soft Condensed Matter Physics*, Phys. Rep. **348**, 267-439 (2001).

[2] N. Metropolis and S. Ulam, *The Monte Carlo Method*, J. Am. Statistical Association **44**, 335-341 (1949).

[3] C. Darwin, *The Origin of Species by Means of Natural Selection* (John Murray, Albemarle Street, London, 1859).

[4] J.H. Holland, *Adaption in Natural and Artificial Systems* (The University of Michigan Press, Ann Arbor, 1975).

[5] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley, MA, 1989).

[6] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, New York, 1992).

[7] D.M. Deaven and K.M. Ho, *Molecular Geometry Optimization with a Genetic Algorithm*, Phys. Rev. Lett. **75**, 288 (1995).

[8] A.R. Oganov and C.W. Glass, *Crystal structure prediction using* ab initio *evolutionary techniques: Principles and applications*, J. Chem. Phys. **124**, 244704 (2006).

[9] S.M. Woodley and R. Catlow, *Crystal structure prediction from first principles*, Nature Materials **7**, 937-946 (2008).

[10] B. Hartke, *Application of Evolutionary Algorithms to Global Cluster Geometry Optimization*, Struct. Bond. **110**, 33-53 (2004).

[11] D.E. Clark (ed.), *Evolutionary Algorithms in Molecular Design* (Wiley-VCH, Weinheim, 2000).

[12] G.W. Greenwood and A.M. Tyrell, *Introduction to Evolvable Hardware* (Wiley-IEEE Press, 2006).

[13] J. Arifovic, *Genetic algorithm learning and the cobweb model*, J. Econ. Dyn. Contr. **18**, 3 (1994).

[14] T. Riechmann, *Genetic algorithm learning and evolutionary games*, J. Econ. Dyn. Contr. **25**, 1019 (2001).

[15] D. Gottwald, *Genetic Algorithms in Condensed Matter Theory*, Ph. D. thesis, TU Vienna (2005).

[16] D. Gottwald, G. Kahl and C.N. Likos, *Predicting equilibrium structures in freezing processes*, J. Chem. Phys. **122**, 204503 (2005).

[17] G.J. Pauschenwein, *Phase behavior of colloidal systems*, Ph. D. thesis, TU Vienna (2008).

[18] G.J. Pauschenwein and G. Kahl, *Clusters, columns, and lamellae - minimum energy configurations in core softened potentials*, Soft Matter **4**, 1396 (2008).

[19] G.J. Pauschenwein and G. Kahl, *Zero temperature phase diagram of the square-shoulder system*, J. Chem. Phys. **129**, 174107 (2008).

[20] J. Fornleitner, *Ordered Equilibrium Structures of Two-Dimensional Soft Matter Systems*, Ph. D. thesis, TU Vienna (2008).

[21] J. Fornleitner, F. Lo Verso, G. Kahl and C.N. Likos, *Genetic algorithms predict formation of exotic ordered configurations for two-component dipolar monolayers*, Soft Matter **4**, 480 (2008).

[22] J. Fornleitner and G. Kahl, *Lane formation vs. cluster formation in two-dimensional square-shoulder systems - A genetic algorithm approach*, Europhys. Lett. **82**, 18001 (2008).

[23] M. Kahn, *Ordered equilibrium structures of soft particles in layered systems*, Diploma thesis, TU Vienna (2008).

[24] K. Burjorjee, *The Fundamental Problem with the Building Block Hypothesis*, `http://arxiv.org/pdf/0810.3356` (2008).

[25] F. Gray, Pulse code communications, U.S. Patent 2632058 (1953).

[26] S. Forrest and M. Mitchell, in D. Whitley (ed.), *Foundations of Genetic Algorithms 2* (Morgan Kaufmann, San Mateo, CA, 1993).

[27] C.R. Reeves and J.E. Rowe, *Genetic Algorithms: Principles and Perspectives: a Guide to GA Theory* (Kluwer Academic Publishers, 2003).

[28] D.E. Goldberg, *Real-coded genetic algorithms, virtual alphabets, and blocking*, Complex Systems **5**, 139-167 (1991).

[29] S.W. Mahfoud, *Niching Methods for Genetic Algorithms*, Ph.D. dissertation Urbana Champaign: Univ. Illinois (1995).

[30] D.H. Wolpert and W.G. Macready, *No Free Lunch Theorems for Search*, Santa Fe, NM: Santa Fe Institute, Tech. Rep. SFI-TR-05-010 (1995).

[31] D.H. Wolpert and W.G. Macready, *No Free Lunch Theorems for Optimization*, IEEE Trans. Evolutionary Computation **1**, 67-82 (1997).

[32] J.C. Culberson, *On the Futility of Blind Search*, Evolutionary Computation **6**, 109-127 (1998).

[33] B. Hartke, *Global geometry optimization of clusters using genetic algorithms*, J. Phys. Chem. **97**, 9973 (1993).

[34] Y. Zeiri, *Prediction of the lowest energy structure of clusters using a genetic algorithm*, Phys. Rev. E **51**, 2769-2772 (1995).

[35] B. Hartke, *Global Cluster Geometry Optimization by a Phenotype Algorithm with Niches: Location of Elusive Minima, and Low-Order Scaling with Cluster Size*, J. Comp. Chem. **20**, 1752-1759 (1999).

[36] D.J. Wales and J.P.K. Doye, *Global Optimization by Basin-Hopping and the Lowest Energy of Lennard-Jones Clusters Containing up to 110 Atoms*, J. Phys. Chem. A **101**, 5111 (1997).

[37] D.J. Wales, *Energy Landscapes* (Cambridge University Press, Cambridge, 2003).

[38] F.H. Stillinger and T.A. Weber, *Nonlinear optimization simplified by hypersurface deformation*, J. Stat. Phys. **52**, 1429-1445 (1988).

[39] R.L. Johnston, *Evolving better nanoparticles: Genetic algorithms for optimising cluster geometries*, Dalton Trans. **2003**, 4193-4207 (2003).

[40] M.D. Wolf and U. Landmann, *Genetic Algorithms for Structural Cluster Optimization*, Jour. Phys. Chem. A **102**, 6129 (1998).

[41] N.L. Abraham and M.I.J. Probert, *A Periodic Genetic Algorithm with*

Real-Space Representation for Crystal Structure and Polymorph Prediction, Phys. Rev. B **73**, 224104 (2006).

[42] D.J. Wales and T.V. Bogdan, *GMIN: A program for finding global minima and calculating thermodynamic properties from basin-sampling.*, Source code and documentation available online, `http://www-wales.ch.cam.ac.uk/GMIN/`.

[43] R. Fletcher and C.M. Reeves, *Function minimization by conjugate gradients*, Comp. J. **7**, 149-154 (1964).

[44] E. Polak and G. Ribière, *Note sur la Convergence de Methods de Directions Conjugués*, Revue Francàise Informat. Recherche Operationnelle **16**, 35-43 (1969).

[45] M.R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, J. Res. Nat. Bur. Stand. **49** 409-436 (1952).

[46] C.G. Broyden, *The Convergence of a Class of Double-Rank Minimization Algorithms*, J. Inst. Math. Appl. **6**, 76-90 (1970).

[47] R. Fletcher, *A New Approach to Variable Metric Algorithms*, Comp. J. **13**, 317-322 (1970).

[48] D. Goldfarb, *A Family of Variable Metric Updates Derived by Variational Means*, Math. Comp. **24**, 23-26 (1970).

[49] D.F. Shanno, *Conditioning of Quasi-Newton Methods for Function Minimization*, Math. Comp. **24**, 647-656 (1970).

[50] J. Nocedal, *Updating Quasi-Newton Matrices with Limited Storage*, Math. Comp. **35**, 773-782 (1980).

[51] D.C. Liu and J. Nocedal, *On the Limited Memory Method for Large Scale Optimization*, Math. Progr. B **45**, 503-528 (1989).

[52] R.H. Byrd, P. Lu and J. Nocedal *A Limited Memory Algorithm for Bound Constrained Optimization*, SIAM J. Sci. Stat. Comp. **16**, 1190-1208 (1995).

[53] C. Zhu, R.H. Byrd and J. Nocedal, *L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization*, ACM Trans. Math. Software **23**, 550-560 (1997).

[54] H.W. Kroto, J.R. Heath, S.C. O'Brien, R.F. Curl and R.E. Smalley, *C_{60}: Buckminsterfullerene*, Nature **318**, 162-163 (1985).

[55] M.R. Hoare, *Structure and dynamics of simple microclusters*, Adv. Chem. Phys. **40**, 49 (1979).

[56] L.T. Wille and J. Vennik, *Electrostatic energy minimisation by simulated annealing*, J. Phys. A: Math. Gen. **18**, L419 (1985).

[57] A.M. Turing *On computable numbers, with an application to the Entscheidungsproblem*, Proc. London Math. Soc. **42**, 230-265 (1936).

[58] *Cambridge Cluster Database*, `http://www-wales.ch.cam.ac.uk/CCD.html`.

[59] J. E. Lennard-Jones, *Cohesion*, Proc. Phys. Soc. **43**, 461-482 (1931).

[60] G.A. Vliegenthart, J.F.M. Lodge and H.N.W. Lekkerkerker, *Strong and Weak Metastable Liquids Structural and Dynamical Aspects of the Liquid State*, Physica A, **263**, 378 (1999).

[61] J.A. Northby, *Structure and binding of Lennard Jones clusters: $13 \leq N \leq 147$*, J. Chem. Phys. **87**, 6166 (1987).

[62] D.M. Deaven, N. Tit, J.R. Morris and K.M. Ho *Structural optimization of Lennard-Jones clusters by a genetic algorithm*, Chem. Phys. Lett. **256**, 195 (1995).

[63] A.L. Mackay, *A dense non-crystallographic packing of equal spheres* Acta Crystallogr. **15**, 916 (1962).

[64] B. Raoult, J. Farges, M.F. de Feraudy and G. Torchet, *Comparison between icosahedral, decahedral and crystalline Lennard-Jones models containing 500 to 6000 atoms*, Philos. Mag. B **60**, 881 (1989).

[65] J. Pillardy and L. Piela, *Molecular Dynamics on Deformed Potential Energy Hypersurfaces*, J. Phys. Chem. **99**, 11805 (1995).

[66] J.P.K. Doye, D.J. Wales and R.S. Berry, *The effect of the range of the potential on the structures of clusters*, J. Chem. Phys. **103**, 4234-4249 (1995).

[67] R.H. Leary and J.P.K. Doye, *New Tetrahedral Global Minimum for the 98-atom Lennard-Jones Cluster*, Phys. Rev. E **60**, R6320-R6322 (1999).

[68] J.P.K. Doye and D.J. Wales *Magic numbers and growth sequences of small face-centred-cubic and decahedral clusters*, Chem. Phys. Lett. **247**, 339-347 (1995).

[69] J.P.K. Doye, M.A. Miller and D.J. Wales, *The double-funnel energy landscape of the 38-atom Lennard-Jones cluster*, J. Chem. Phys. **110**, 6896 (1999).

[70] M. Dzugutov and U. Dahlborg, *Molecular dynamics study of the coherent density correlation function in a supercooled simple one-component liquid*, J. Non-Cryst. Solids **131-133**, 62-65 (1991).

[71] J.P.K. Doye, D.J. Wales and S.I. Simdyankin, *Global Optimization and the Energy Landscapes of Dzugutov Clusters*, Faraday Discuss. **118**, 159-170 (2001).

[72] S. Mossa, F. Sciortino, P. Tartaglia and E. Zaccarelli, *Ground-State Clusters for Short-Range Attractive and Long-Range Repulsive Potentials*, Langmuir **20**, 10756-10763 (2004).

[73] C. Janot, *Quasicrystals: A Primer* (Oxford Univ. Press, New York, 1992).

[74] J. Mikhael, J. Roth, L. Helden and C. Bechinger, *Archimedean-like tiling on decagonal quasicrystalline surfaces*, Nature **454**, 501-504 (2008).

[75] F.H. Stillinger, *Phase transitions in the Gaussian core system*, J. Chem. Phys. **65**, 3968 (1976).

[76] B. Krüger, L. Schäfer and A. Baumgärtner, *Correlations among interpenetrating polymer coils: the probing of a fractal*, J. Phys. France **50**, 3191-3222 (1989).

[77] M. Luescher, *A portable high-quality random number generator for lattice field theory simulations*, Comput. Phys. Commun. **79**, 100 (1994).

[78] F. James, *RANLUX: a Fortran implementation of the high-quality pseudorandom number generator of Lüscher*, Comput. Phys. Commun. **79**, 100 (1994).

[79] A. Miller, *TOMS778*, `http://users.bigpond.net.au/amiller/toms/toms778.zip` (1999).

[80] *PyMOL Molecular Viewer*, `http://www.pymol.org`.

Acknowledgements

During the years of my studies and the time I spent working on this thesis, a lot of people have supported me. I would like to thank

My supervisor Gerhard Kahl, both scientifically and personally.

The members of the soft matter theory group at TU Wien Julia Fornleitner, Gernot Pauschenwein, Dieter Schwanzer, Daniele Coslovich, Jan Kurzidim, Lukas Strauss and Emanuela Bianchi for providing good company and help whenever it was needed.

Bernhard Zauner, Raphael Angerer, Johannes Strobl, Georg Wachter, Christoph Weber, Daniela Klotz, Julia Urschler, Dominik Seebacher, Martina Lechner, Alexander Gutmann, Robert Frick, Simone Angerer, Daniel Steiner, Karl Rühringer, Klaus Pramberger, Thomas Brandner and Sebastian Salhofer for being friends for all these years.

My parents Maria and Franz for their support and trust in my decisions, my sister Eva-Maria, my grandparents, my uncle Josef and my whole family.

All these walls were never
really there.
Mike Skinner