PROJECT REPORT

A C++ Implementation of a Genetic Algorithm for Energy Minimization in LAMMPS

Jobst Beckmann[a] and Julius Beerwerth[a]

[a]Chair and Institute of General Mechanics, RWTH Aachen University, Aachen, Germany

ABSTRACT
In this work, we present a genetic and a hybrid genetic algorithm for minimizing the potential energy of atomic structures. Afterwards, we test the algorithms on three test structures and compare the results to the conjugate gradient method. The idea behind using a genetic algorithm is to avoid converging into the nearest local minimum by exploring a larger space of possible solutions.

KEYWORDS
LAMMPS; energy minimization; genetic algorithm; molecular dynamics;

Jobst Beckmann Email: jobst.beckmann@rwth-aachen.de
Julius Beerwerth Email: julius.beerwerth@rwth-aachen.de

Contents

1. Introduction & Motivation

In the field of molecular dynamics, energy minimization plays an important role during the initialization of a simulation. Energy minimization describes the process of computationally finding a minimum energy conformation of a system of atoms. It is used to determine the proper arrangement of the atoms in space, since the initial positions of the atoms might not be energetically favorable. Especially for systems with randomly initialized atom positions, energy minimization is necessary to prevent the system from heating up or even exploding right after the start of the simulation.
Three major computational methods are common to perform energy minimization: steepest descent, conjugate gradient and newton-raphson. All of these methods belong to the category of line search methods. In general, a line search method determines a descent direction along which the energy will be reduced and then computes a step size to determine how far to move along the descent direction. The advantage of these methods is that they converge relatively fast to a local minimum. The disadvantage of line search methods is that the minimum they converge into is highly dependent on the initial positions of the atoms.
To prevent converging to the nearest minimum, multiple stochastic approaches have been explored in the past like a basin-hopping strategy [11] or genetic algorithms [3]. In recent years genetic algorithms have continuously shown to be well-suited for optimizing atomic structures, especially crystalline structures, in multiple works [8][12][9]. This work presents a genetic algorithm and a genetic algorithm in combination with the conjugate gradient method as protocols for energy minimization. The methods have been implemented inside the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS), which is a code for molecular dynamics simulations [10]. This way it is possible to seamlessly minimize the potential energy of a system, using the presented algorithms, right before starting the simulation.

2. Methods & Materials

In this section, we give an overview of the optimization problem we want so solve. Furthermore, we present genetic algorithms in general as well as our implementation specifically.

2.1. *Problem description*

In order to make proper calculations with a molecular system, it is important that this system starts in a stable configuration. Since the structure provided as an input for LAMMPS might not be in such a stable configuration, it is essential to first perform an energy minimization. To achieve this, we evaluate the potential energy of the system (see section 2.2) and try to tweak the system to find a lower energy state. Current solvers available in the LAMMPS application like conjugate gradients or steepest descent tend to get stuck in a local minimum and are therefore unable to find an optimal starting configuration. We propose that a genetic algorithm can solve this problem by being able to use its genetic operations like crossover and mutation to exit a local minimum and finding a better solution.

2.2. *Energy function*

The energy function allows us to calculate the potential energy for every possible configuration of atoms in a molecular system. To achieve this the different energy types in the system (namely pair, bond, angular, dihedral, improper and fix energies respectively) are calculated for every possible atom combination and are then added up. The different energy terms represent interactions between atoms in the molecular structure. LAMMPS [1] calculates the energy as follows:

$$\begin{aligned} E(r_1, r_2, \dots, r_N) = &\sum_{i,j} E_{pair}(r_i, r_j) + \sum_{ij} E_{bond}(r_i, r_j) + \\ &\sum_{ijk} E_{angle}(r_i, r_j, r_k) + \sum_{ijkl} E_{dihedral}(r_i, r_j, r_k, r_l) + \\ &\sum_{ijkl} E_{improper}(r_i, r_j, r_k, r_l) + \sum_{i} E_{fix}(r_i) \end{aligned}$$

This equation does of course only capture a simplified view of the underlying physics, but is well suited for molecular simulations as it can be computationally cheaply evaluated. [5] A major part of the atom interactions is the van-der-Waals potential. It describes the forces of attraction and repulsion based on the distance between the atoms. A common method to calculate this pair energy is the Lennard-Jones potential [6].

$$E_{pair}(r) = E_{lj}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{1}$$

Where r is the distance between two atom centers, ϵ is the depth of the potential well and σ is the distance of zero energy interaction.

2.3. *Genetic Algorithm*

Genetic algorithms provide us with a method to solve an optimization problem. They are inspired by processes like natural selection or survival of the fittest found in nature. To mimic these processes, genetic algorithms are based on a group of possible solutions of our optimization problem called a population. The possible solutions of our optimization which make up the population are called individuals. The individuals in the population are then selected, create offspring and are mutated in order to find a better solution in the next generation. The process can be repeated until a fixed amount of generations or a termination criterion is reached. This process is summarized in algorithm 1.

Here x^0 represents the vector containing the initial coordinates of each atom, $n_{\text{individual}}$

Algorithm 1 Genetic Algorithm

```
1: given initial values $x^0$ and parameters $n_{\text{individual}}, p_{\text{mutate}}, d_{\text{max}}$
2: Initialization()
3: repeat
4:     Selection()
5:     Crossover()
6:     Mutation()
7:     FitnessCalculation()
8: until termination criterion is satisfied
```

the number of individuals, p_{mutate} the probability that an atom mutates and d_{max} the maximum distance an atom can travel during a mutation.

2.3.1. Initialization

Since we want to find the atom configuration with minimal energy, a solution of our optimization problem is characterized by the atom positions. We can think of the coordinates of a single atom as the genes of an individual in the genetic algorithm. The aggregate of all atoms (genes) forms the chromosome of the individual. At the start of the minimization, all individuals in the population must be given a chromosome. We present two different methods to achieve this. LAMMPS can already provide an initial atom configuration based either on user input or generated randomly from a seed. We can use this configuration as the initial chromosome for the entire population and possibly mutate it to provide distinct individuals. The other method is to generate random positions for the initial chromosomes. Depending on the task, one or the other should be chosen. If a general structure is already provided, random initialization should be disabled, as solutions that deviate too much from the provided configuration are uninteresting. Otherwise, a random initialization will increase the genetic diversity of the population and help the algorithm to find a better solution. After each individual has been assigned a chromosome, the fitness of each individual is calculated once, which completes the initialization process.

2.3.2. Fitness function

We need the ability to evaluate the different individuals in our population. To achieve this, we define a fitness function that takes the chromosome of an individual and returns a scalar score, thereby giving us the information on the fitness of the individuals in question. The fitness function of a genetic algorithm heavily depends on the optimization problem. In our case, we want to minimize the energy of all possible atom configurations. For this reason, we define our fitness function to be the energy function discussed in section 2.2.

2.3.3. Selection

A major avenue for the genetic algorithm to find a better solution in the next generation is the combination of previous solutions to form offspring. In order to transfer beneficial genes to the next generation, we generally want to find fit individuals to be parents. We must be careful not to just let the fittest individuals create children, because this can lead to a premature convergence in a local minimum. One has to keep in mind that unfit individuals might also carry valuable genes that we want to carry over into the next generation. We need a selection method that satisfies these considerations. A very common method is the fitness proportional selection [2]. Here, individuals are chosen based on a probability that is calculated using the fitness score. One can think of it like a wheel, where different individuals take up a portion of the wheel based on their proportional fitness. To determine a parent, the wheel is rotated and the individual at the position the wheel stops is selected. This also called a Roulette Wheel Selection. If two parents are selected during each spin, the method is called Stochastic Universal Sampling. However, we have selected a different selection method, the Tournament Selection. In this method, a set of individuals is randomly picked from the population. These individuals are then compared in terms of fitness and the best one is picked as one of the parents. The method is then repeated to find a second parent. The main benefit of tournament selection is its lower computational complexity. Because the population does not need to be sorted beforehand, the computational complexity is linear [4]. It can also easily work with both positive and negative fitness values, which is very important for our application.

2.3.4. Crossover

After the two parents have been selected, they are used to create a child for the next generation. Usually one of two methods is used. With n-point-crossover, the chromosome of each parent is divided at n locations. The parts are then swapped to create a new chromosome belonging to the child. With uniform crossover, the decision which parent's gene transfers to the child is made individually for each gene by performing a coin flip [7]. We have found that uniform crossover works well for the problem at hand.

2.4. ***Hybrid Genetic Algorithm***

Genetic algorithms presented in other works for the optimization of atomic structures do not implement the genetic algorithm as a standalone method for minimizing the energy, but as a combination of a genetic algorithm and a local minimization method. The idea behind this approach is to combine the relatively fast convergence of a local minimization (e.g. a line search method) with the larger solution space of a genetic algorithm.
Therefore, a local minimization is carried out right before the fitness calculation on each individual. The maximal number of steps for the local minimization can be specified by the user. The chromosome of each individual is then updated to the atom positions resulting from its corresponding local minimization. Afterwards, the

fitness calculation is carried out as usual, but considering the updated atom positions resulting from the local minimization. This general steps can be seen in algorithm 2.

Here n_{min} represents the maximal number of iterations carried out per local

Algorithm 2 Hybrid Genetic Algorithm

```
1: given initial values $x^0$ and parameters $n_{\text{individual}}, p_{\text{mutate}}, d_{\text{max}}, n_{\text{min}}$
2: Initialization()
3: repeat
4:     Selection()
5:     Crossover()
6:     Mutation()
7:     LocalMinimization()
8:     FitnessCalculation()
9: until termination criterion is satisfied
```

minimization. All of the established line search methods (Steepest Descent, Conjugate Gradients and Newton-Raphson) for energy minimization are well-suited to be used as the local minimization method. The implementation presented here uses the conjugate gradient method for the local minimization, because it has been shown to be a good middle ground between the Steepest Descent method (slower convergence) and the Newton-Raphson method (higher computational complexity). It is also set as the default method for energy minimization in LAMMPS.

3. Results

The algorithms were tested on three test structures. The structures were made up of a gas in different configurations. For all three structures, the pair potential was set to the Lennard-Jones potential and the units were set to lj. This means that all quantities are unitless, and the boundary conditions were set to periodic.

3.1. *Setup*

A simple two-dimensional structure with a quadratic bounding box length of 12 units is shown in figure 1. Three different atoms are present, denoted by different colors (red, blue and yellow). The structure consists of 50 atoms of the first type (red), 10 atoms of the second type (blue) and 5 atoms of the third type (yellow). The genetic algorithm was run with 100 individuals, a mutation rate of 20 percent and a maximal mutation distance of 0.1 units. A comparison with the conjugate gradient method on the same structure is shown in table 1. We can observe that conjugate gradients runs into a local minimum the genetic algorithm can avoid. We can also already see one of the major downsides of the genetic algorithms, the large run-time that is required.
The second structure is shown in figure 2. The bounding box was a cube with a side length of 5 units. The atom positions have been initialized randomly using a seed to ensure they are consistent in every run. The system was made up of two types of atoms spread throughout the bounding box. In total, there were 20 atoms of type one (red) and 5 atoms of type two (blue) present. The idea behind the second structure was to test the algorithms on a simple 3D configuration of atoms.

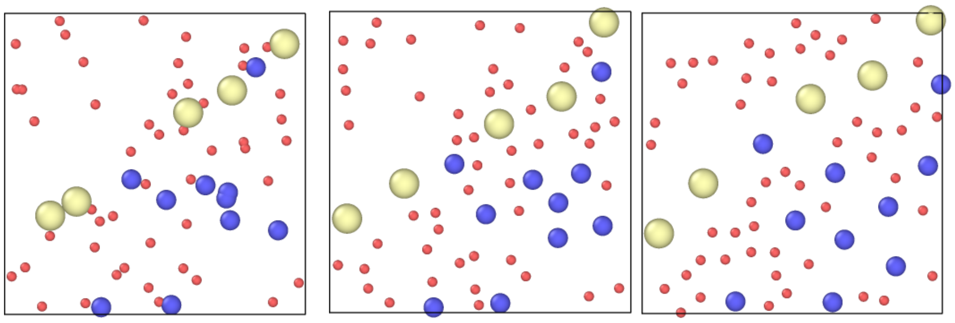

(a) i = 0. (b) i = 200. (c) i = 2000.

Figure 1. Simple 2D test case with three different atoms shown after 0, 20 and 2000 iterations of the genetic algorithm

The third structure is shown in figure 3. Again, the bounding box was chosen to be a cube, but with a different side length of 30 units. This time, the two types of atoms were initialized in two different regions. The first region is formed by a cylinder centered along the z-axis with a diameter of 15 units. The second region was defined as the difference between the bounding box and the first region. Atoms of the first type have been assigned randomly generated positions inside the second region, and atoms of the second type have been assigned randomly generated positions inside the first region. In total, there were 1000 atoms of type 1 and 150 atoms of type 2. The idea behind structure 3 was to illustrate further how running into a local minimum can be disadvantageous, and how using a genetic algorithm can prevent this.

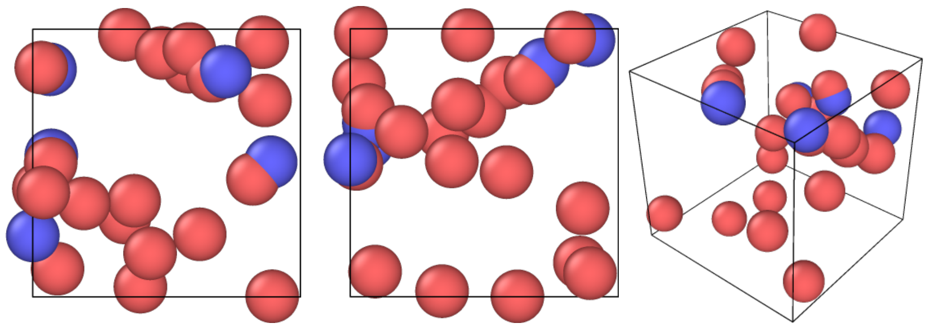

(a) Top view. (b) Front view. (c) Perspective 3D view.

Figure 2. Structure 2 seen from different angles. Atoms of type 1 are colored red, atoms of type 2 are colored blue. The atom positions have been assigned completely random.

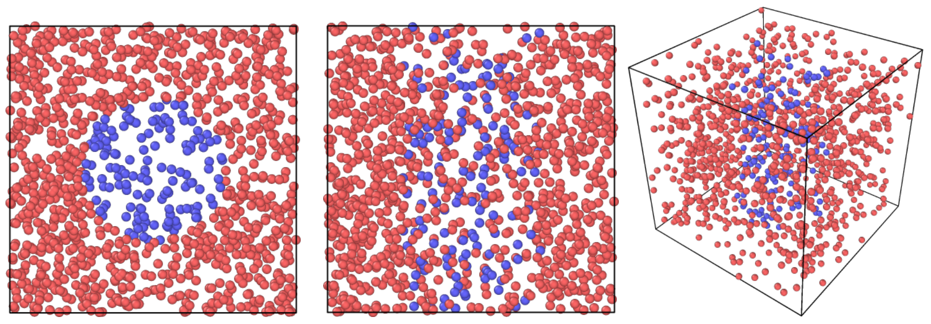

(a) Top view. (b) Front view. (c) Perspective 3D view.

Figure 3. Structure 3 seen from different angles. Atoms of type 1 are colored in red, atoms of type 2 are colored in blue. Atoms of type two have been randomly assigned positions within a cylinder centered at the z-axis with diameter of 15 units. Atoms type 1 have been assigned random positions outside the cylinder.

3.2. *Energy minimization*

Energy minimization was performed on structure 2 using the conjugate gradient method, the genetic algorithm and the hybrid genetic algorithm. For all three minimization methods, the energy tolerance was set to 1e-4 and the force tolerance to 1e-6. For the first run using the conjugate gradient method, the maximal number of iterations and evaluations was set high enough for the method to converge. The conjugate gradient method needed 34 iterations to converge up to a potential energy of ca. -0.74. Figure 4 shows structure 2 after 0, 17 and 34 iterations of the conjugate gradient method. It can be observed, that the atoms spread out but are still relatively close to their initial position.
For the genetic algorithm the number of individuals was set to 128, the probability for mutations to 10% and the maximal distance an atom can move during mutations to 0.05. All parameters were chosen empirically based on what resulted in the lowest value for the potential energy. The genetic algorithm was set to run for 2000 iterations, as running it for more iterations did not lead to further decrease of the potential energy. The genetic algorithm minimized the potential energy to a value of -3.01. Therefore, the atom configuration the genetic algorithm found has a smaller potential energy than the one in the conjugate gradient method. Figure 5 shows structure 1 after 0, 1000 and 2000 iterations of the genetic algorithm. Comparing the result of the genetic algorithm with the run of the conjugate gradient method, we see that the minimum both methods found is different. It can also be observed, that the atoms moved less

using the genetic algorithm than they did while using the conjugate gradient method, due to the small search radius.
For the hybrid genetic algorithm, the number of individuals was set to 128 and the probability for mutations to 10%. The maximal distance an atom can move during a mutation was increased to 0.5 units, as our tests with the hybrid genetic algorithm showed that a stronger mutation rate is beneficial for this method. The number of local minimization steps was set to 100 in order to guarantee that the local minimization method converges. The hybrid genetic algorithm was set to run for 500 iterations, as the potential energy did not improve within the given tolerance afterwards. The hybrid genetic algorithm minimized the potential energy to a value of -4.03. Comparing the potential energy of all three runs, it is evident that the configuration the hybrid genetic algorithm found has the lowest potential energy. Figure 6 shows structure 2 after 0, 250 and 500 iterations of the hybrid genetic algorithm. It can be seen, that the atoms moved so much, that the initial configuration is not recognizable at all during the final iteration. Furthermore, instead of simply spreading out the atoms, atoms of type 1 (red) have been clumped together with the atoms of type 2 (blue) taking up the remaining space.

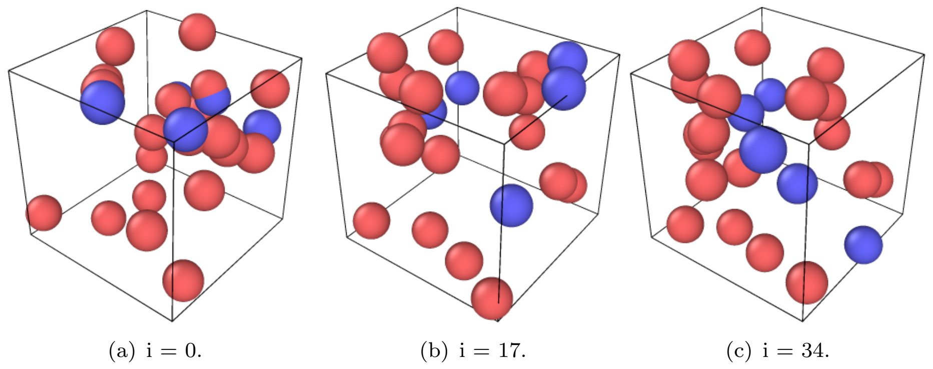

(a) i = 0. (b) i = 17. (c) i = 34.

Figure 4. Structure 2 seen in the perspective 3D view after 0, 17 and 34 iterations of the conjugate gradient method.

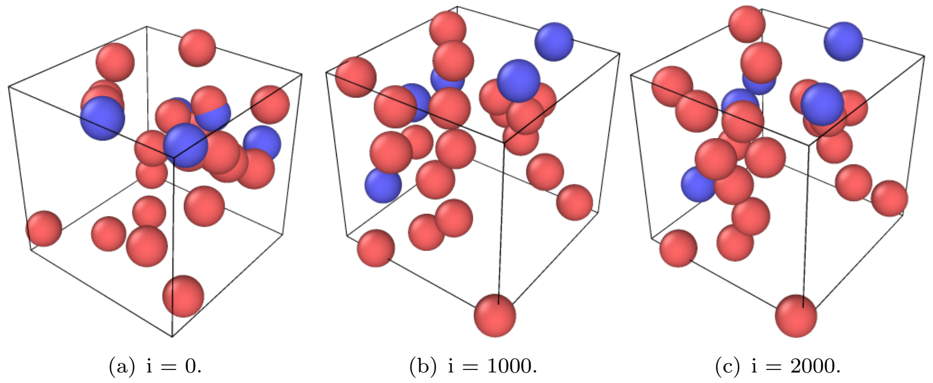

(a) i = 0. (b) i = 1000. (c) i = 2000.

Figure 5. Perspective 3D view of structure 2 after 0, 1000 and 2000 iterations of the genetic algorithm.

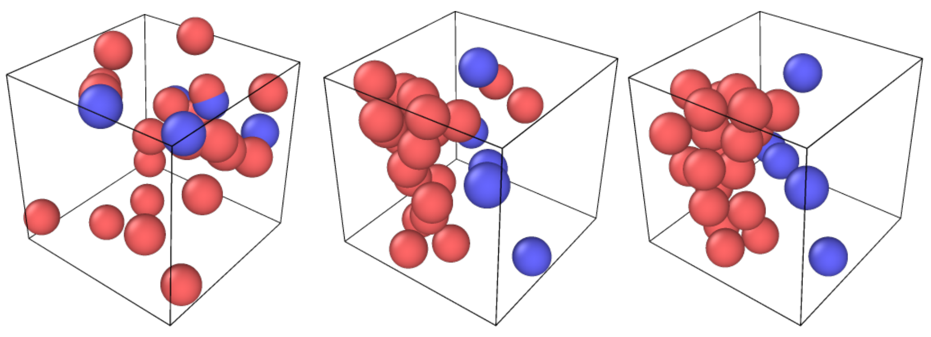
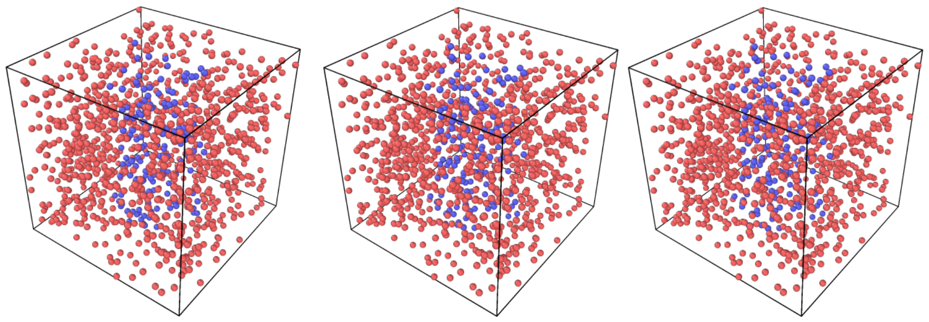

(a) i = 0. (b) i = 250. (c) i = 500.

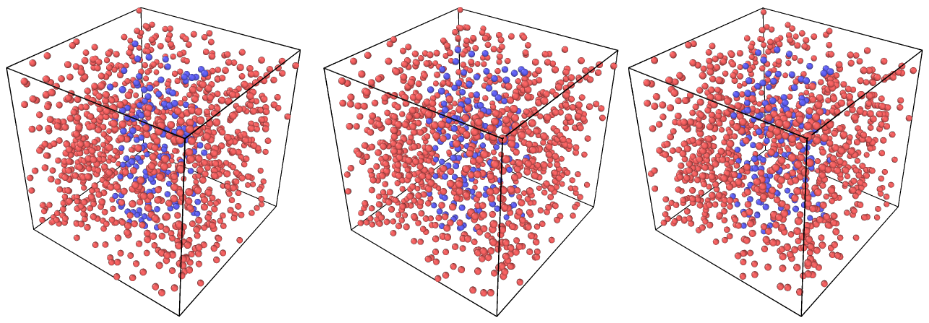
Figure 6. Structure 2 after 0, 250 and 500 iterations of the hybrid genetic algorithm seen from the perspective 3D view.

Structure 3 was minimized in terms of potential energy as well, using the conjugate gradient method, the genetic algorithm and the hybrid genetic algorithm. Again, the energy tolerance was set to 1e-4 and the force tolerance to 1e-6 for all three methods. For the conjugate gradient method, the maximal number of iterations was set high enough for the method to converge. This time, the conjugate gradient method needed 51 iterations to achieve this and to reduce the potential energy to a value of ca. -0.36. Figure 7 shows structure 3 after 0, 25 and 51 iterations of the conjugate gradient method. Comparing the initial atom configuration to the final atom configuration, it is hard to see the difference. Only when comparing the positions of individual atoms can it be observed that the atoms have moved a bit. This example illustrates how the conjugate gradient method can get stuck in a local minimum.
For the genetic algorithm the number of individuals was set to 128, the probability for mutations to 10% and the maximum distance an atom can be moved by mutating to 0.05 units. This time the genetic algorithm was set to run for 5000 iterations. The energy was reduced to a final value of ca. -0.4. Comparing this result to the potential energy the conjugate gradient method resulted in, it can be noted again that the genetic algorithm found a slightly better atom configuration in terms of potential energy. Figure 8 shows structure 3 after 0, 2500 and 5000 iterations of the genetic algorithm. Surprisingly, the genetic algorithm runs into the same problem as the conjugate gradient method, being stuck at the starting configuration of the system. Similar to the result of the conjugate gradient method, it seems like most of the atoms did not move at all.
For the hybrid genetic algorithm applied to structure 3 the number of individuals was set to 128, the probability for mutations to 10%. The maximal distance of atom movement by mutation was set to 7.5 units. The number of local minimization steps was set to 100 to guarantee convergence of the local minimization. For this test, the hybrid genetic algorithm was set to run for only 10 iterations. The potential energy was reduced to a value of ca. -0.8. Again, it can be noted that the hybrid genetic algorithm found an atom configuration with a lower potential energy than the conjugate gradient method as well as the genetic algorithm. Figure 9 shows structure 3 after 0, 5 and 10 iterations of the hybrid genetic algorithm. Here the advantage of the hybrid genetic algorithm becomes obvious. It can be observed that the two atom types do not stay separated, but mix up, breaking out of the initial configuration completely. This illustrates that the hybrid genetic algorithm does not get stuck in a local minimum that depends on the starting configuration, for the given example.



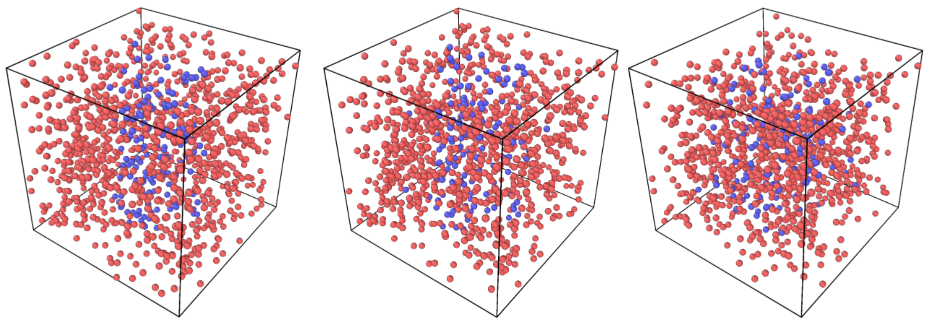
(a) i = 0. (b) i = 25. (c) i = 51.

Figure 7. Structure 3 seen from the perspective 3D view after 0, 25 and 51 iterations of the conjugate gradient method.



(a) i = 0. (b) i = 2500 (c) i = 5000.

Figure 8. Structure 3 after 0, 2500 and 5000 iterations of the genetic algorithm seen from the perspective 3D view.



(a) i = 0. (b) i = 5. (c) i = 10.

Figure 9. Perspective 3D view of structure 3 after 0, 5 and 10 iterations of the hybrid genetic algorithm.

3.3. *Run-time comparison*

Last but not least, we want to have a look at the run-times of the three methods. Table 1 summarizes the number of iterations and the run-times of the three methods for all structures. The purpose of this table is to compare the run-times qualitatively. Firstly, it can be noted that the total run-time of both the genetic and the hybrid genetic is orders of magnitudes greater than the run-time of the conjugate gradient method. This is due to number of individuals, as the genetic and the hybrid genetic algorithm have to compute the potential energy and perform local minimization for each individual in each iteration respectively. Also note that the implementations of the genetic and the hybrid genetic algorithm have not been parallelized, although this would certainly be possible.
Secondly, it can be observed that the genetic and the hybrid genetic algorithm scale a lot worse than the conjugate gradient method. This becomes clear, when comparing the CPU time per iteration for structure 2 and 3 for all methods. While the CPU time per iteration grows by a factor of five for the conjugate gradient method, for the genetic algorithm the CPU time per iteration increases by a factor of about 24 and for the hybrid genetic algorithm by a factor of about 32. Again, this can be explained by considering the number of individuals. As the size of the system grows, the time for computing the potential energy and performing local minimization also grows. This effect amplifies for the genetic and the hybrid genetic algorithm as they have to calculate the potential energy and perform a local minimization for each individual respectively.
In addition, one could think of simply reducing the number of individuals to reduce the total CPU time by reducing the CPU time per iteration, but this would simply lead to less genetic diversity causing the genetic and hybrid genetic algorithms to need more iterations to reach the same decrease in potential energy.

Table 1. Number of iterations, CPU time, CPU time per iteration and final potential energy summarized for each energy minimization method and all structures.

Method	Structure	#Iterations	CPU time	CPU time per Iteration	Final potential energy
Conjugate Gradient	1	29	0.0060 s	2.07×10^{-4} s	8390.04
	2	34	0.0032 s	9.41×10^{-5} s	-0.74
	3	51	0.0250 s	4.90×10^{-4} s	-0.36
Genetic	1	2000	4.5139 s	0.0026 s	54.65
	2	2000	4.63 s	0.0023 s	-3.01
	3	5000	276.03 s	0.055 s	-0.40
Hybrid Genetic	2	500	21.52 s	0.043 s	-4.03
	3	10	13.83 s	1.38 s	-0.80

4. Discussion

The results presented in the previous section confirm that it is possible to successfully implement a simple genetic algorithm for energy minimization in LAMMPS. The genetic as well as the hybrid genetic algorithm were able to minimize the potential energy of a given atom configuration further than established line search methods like the conjugate gradient method. It also became clear that a genetic algorithm on its own does not lead to the desired results exempt in the most simple applications. The first test case with structure 1 shows convergence to a good solution, where conjugate gradients gets stuck in a local minimum. This does not necessarily hold for more complex tasks, though. Because of the sensitivity of the problem to the atom positions the distance an atom can be moved by mutations had to be decreased, in order to get the genetic algorithm to converge. Therefore, the space of possible solutions is not explored further than the conjugate gradient method does. This became clear when testing the algorithms on structure 2 and structure 3. The hybrid genetic algorithm on the other hand was able to rely on the local minimization method to converge into a local minimum. Knowing this made it possible to move atoms throughout the whole simulation box during mutating. This way, the hybrid genetic algorithm was able to explore a much broader space of possible solutions than a classical line search method could.
The results presented in this work are in line with the findings of similar works from the literature. Early works in this direction like [3] from Deaven and Ho as well as recent works like [9] by Maldonis et al. have already shown that genetic algorithms in combination with a local minimization method can be used successfully for optimizing atomic structures. The fact that most of the genetic algorithms presented in the literature (e.g. [3][8][9][13]) for energy minimization are hybrid genetic algorithms, incorporating a local minimization method within the genetic algorithms, also falls in line with our results.
We have shown, that a simple implementation of a genetic algorithm, without local minimization methods, is able to successfully minimize the potential energy of atomic structures. Nevertheless, it also becomes clear why it is beneficial to combine a genetic algorithm with a local minimization method for energy minimization. Furthermore, this work implements the genetic algorithm within the software LAMMPS for molecular dynamics simulations. Therefore, the genetic algorithm presented here can be used for minimizing the energy of a system seamlessly before running a simulation. In the future, it would make sense to expand on our examples and use the genetic structures on different materials. In addition, better methods for mutation and crossover could be found to make the genetic algorithm more powerful. One of the major drawbacks of the presented implementation is the lack of parallelization. The evaluation of the potential energy for the genetic algorithm, as well as the local minimization procedures for the hybrid genetic algorithm are carried out in sequence, as parallelizing the algorithm within LAMMPS was outside the scope of this work. This, and the fact that these calculations have to be carried out as many times as the user-specified number of individuals at every iteration, leads to the fact that the presented implementation is not at all competitive to the well-established line search strategies in terms of runtime.
To summarize, this work shows that a genetic algorithm on its own is able to minimize the potential energy of atomic structures. But to do so, the search space has to be decreased so much that it does not bring a major improvement compared to a classical line search method. This can be countered by combining the genetic algorithm with

a local minimization method into a hybrid genetic algorithm. While a hybrid genetic algorithm does increase the explored solution space by allowing for arbitrarily large search spaces, it is not competitive in terms of run-time with the classical line search methods.

5. Usage

Once you have cloned the repository and build the LAMMPS executable, using the genetic and the hybrid genetic algorithm is quite simple. In general one can simply change the minimization style to either the genetic or the hybrid genetic algorithm using the `min_style` command respectively.

```
min_style genetic
min_style ga_hybrid
```

LAMMPS will then use the user-specified minimization style with the default values for the parameters. The default number of Individuals `numIndividuals` is set to 128, the default probability for mutations `p` to 0.1 (10 %), the default maximal number of local minimization steps `localMinSteps` to 100 and the default search radius `dmax` to 0.1 units. All of the parameter can be changed by the user using the `min_modify` command and key value pairs as follows.

```
min_modify numIndividuals 64
min_modify p 0.2
min_modify localMinSteps 200
min_modify dmax 0.5
```

References

[1] *LAMMPS minimize command.* `https://docs.lammps.org/minimize.html`, . – Accessed: 2022-02-23

[2] Chahar, Vijay ; Katoch, Sourabh ; Chauhan, Sumit: A Review on Genetic Algorithm: Past, Present, and Future. In: *Multimedia Tools and Applications* 80 (2021), 02. `http://dx.doi.org/10.1007/s11042-020-10139-6`. – DOI 10.1007/s11042–020–10139–6

[3] Deaven, D M. ; Ho, K M.: Molecular geometry optimization with a genetic algorithm. In: *Physical Review Letters* (1995)

[4] Fang, Yongsheng ; li, Jun: A Review of Tournament Selection in Genetic Programming, 2010. – ISBN 978–3–642–16492–7, S. 181–192

[5] Jorgensen, William L. ; Tirado-Rives, Julian: Potential energy functions for atomic-level simulations of water and organic and biomolecular systems. In: *Proceedings of the National Academy of Sciences* 102 (2005), Nr. 19, 6665–6670. `http://dx.doi.org/10.1073/pnas.0408037102`. – DOI 10.1073/pnas.0408037102. – ISSN 0027–8424

[6] Lim, Teik-Cheng: The Relationship between Lennard-Jones (12-6) and Morse Potential Functions. In: *Zeitschrift für Naturforschung A* 58 (2003), 11. `http://dx.doi.org/10.1515/zna-2003-1104`. – DOI 10.1515/zna–2003–1104

[7] Lingaraj, Haldurai: A Study on Genetic Algorithm and its Applications. In: *International Journal of Computer Sciences and Engineering* 4 (2016), 10, S. 139–143

[8] Lonie, David C. ; Zurek, Eva: XtalOpt: An open-source evolutionary algorithm for crystal structure prediction. In: *Computer Physics Communications* (2011)

[9] Maldonis, Jason J. ; Xu, Zhongnan ; Song, Zhewen ; Yu, Min ; Mayeshiba, Tam ; Morgan, Dane ; Voyles, Paul M.: StructOpt: A modular materials structure optimization suite incorporating experimental data and simulated energies. In: *Computational Materials Science* (2019)

[10] Thompson, A. P. ; Aktulga, H. M. ; Berger, R. ; Bolintineanu, D. S. ; Brown, W. M. ; Crozier, P. S. ; Veld, P. J. '. ; Kohlmeyer, A. ; Moore, S. G. ; Nguyen, T. D. ; Shan, R. ; Stevens, M. J. ; Tranchida, J. ; Trott, C. ; Plimpton, S. J.: LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. In: *Comp. Phys. Comm.* (2022). `https://www.lammps.org`

[11] Wales, David J. ; Doye, Jonathan P.: Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. In: *Journal of Physical Chemistry A* (1997)

[12] Wang, Yanchao ; Lv, Jian ; Zhu, Li ; Ma, Yanming: CALYPSO: A method for crystal structure prediction. In: *Computer Physics Communications* (2012)

[13] Wu, S. Q. ; Ji, M. ; Wang, C. Z. ; Nguyen, M. C. ; Zhao, X. ; Umemoto, K. ; Wentzcovitch, R. M. ; Ho, K. M.: An adaptive genetic algorithm for crystal structure prediction. In: *Journal of Physics Condensed Matter* (2014)

Appendix A. LAMMPS input scripts

```
# Initialization

log     min_lj_01_cg.log

units   lj
dimension   2
atom_style  atomic
pair_style  lj/cut 2.5
boundary  p p p

# System definition

region            myreg block -6 6 -6 6 -0.5 0.5
create_box        3 myreg

create_atoms      1 random 50 3456 myreg
create_atoms      2 random 10 56774 myreg
create_atoms      3 random 5 34565 myreg


# Simulation settings

mass              1 1
mass              2 1
mass              3 2
pair_coeff        1 1 1.0 1.0
pair_coeff        2 2 0.5 3.0
pair_coeff        3 3 1 6
# Run

dump    1 all custom 1 min_lj_01_cg_*.dump id type x y z vx vy vz

thermo    10
min_style cg

minimize  1.0e-4 1.0e-6 2000 100000
```
Listing 1 Input script for minimizing structure 1 using the conjugate gradient method

```
# Initialization

log     min_lj_01_ga.log

units   lj
dimension   2
atom_style  atomic
pair_style  lj/cut 2.5
boundary  p p p

# System definition

region            myreg block -6 6 -6 6 -0.5 0.5
create_box        3 myreg

create_atoms      1 random 50 3456 myreg
create_atoms      2 random 10 56774 myreg
create_atoms      3 random 5 34565 myreg


# Simulation settings

mass              1 1
mass              2 1
mass              3 2
pair_coeff        1 1 1.0 1.0
pair_coeff        2 2 0.5 3.0
pair_coeff        3 3 1 6
# Run

dump    1 all custom 1 min_lj_01_ga_*.dump id type x y z vx vy vz

thermo    10
min_style genetic
min_modify dmax 0.1 numIndividuals 100 p 0.2 localMinSteps 10

minimize  1.0e-4 1.0e-6 2000 100000
```

Listing 2 Input script for minimizing structure 1 using the genetic algorithm

```
# Initialization
log     min_lj_02_cg.log

units   lj
dimension   3
atom_style  atomic
pair_style  lj/cut 2.5
boundary  p p p

# System definition

region            myreg block -2.5 2.5 -2.5 2.5 -2.5 2.5
create_box        2 myreg
create_atoms      1 random 20 341341 myreg
create_atoms      2 random 5 127569 myreg

# Simulation settings

mass              1 1
mass              2 1
pair_coeff        1 1 1.0 1.0
pair_coeff        2 2 0.5 3.0

# Minimization

dump    10 all custom 1 min_lj_02_cg_*.dump id type x y z

thermo    10
thermo_style custom step cpu etotal
min_style cg

minimize  1.0e-4 1.0e-6 1000 10000
```

Listing 3 Input script for minimizing structure 2 using the conjugate gradient method

```
# Initialization
log     min_lj_02_genetic.log

units   lj
dimension   3
atom_style  atomic
pair_style  lj/cut 2.5
boundary  p p p

# System definition

region            myreg block -2.5 2.5 -2.5 2.5 -2.5 2.5
create_box        2 myreg
create_atoms      1 random 20 341341 myreg
create_atoms      2 random 5 127569 myreg
```

```

# Simulation settings

mass              1 1
mass              2 1
pair_coeff        1 1 1.0 1.0
pair_coeff        2 2 0.5 3.0

# Run

dump    100 all custom 100 min_lj_02_genetic_*.dump id type x y z

thermo    100
thermo_style custom step cpu etotal
min_style genetic
min_modify dmax 0.05 numIndividuals 128 p 0.1

minimize  1.0e-4 1.0e-6 2000 100000
```
Listing 4 Input script for minimizing structure 2 using the genetic algorithm

```
# Initialization
log     min_lj_02_ga_hybrid.log

units   lj
dimension   3
atom_style  atomic
pair_style  lj/cut 2.5
boundary  p p p

# System definition

region            myreg block -2.5 2.5 -2.5 2.5 -2.5 2.5
create_box        2 myreg
create_atoms      1 random 20 341341 myreg
create_atoms      2 random 5 127569 myreg

# Simulation settings

mass              1 1
mass              2 1
pair_coeff        1 1 1.0 1.0
pair_coeff        2 2 0.5 3.0

# Run

dump    100 all custom 1 min_lj_02_ga_hybrid_*.dump id type x y z

thermo    100
thermo_style custom step cpu etotal
min_style ga_hybrid
min_modify dmax 0.5 numIndividuals 128 p 0.1 localMinSteps 100

minimize  1.0e-4 1.0e-6 500 100000
```
Listing 5 Input script for minimizing structure 2 using the hybrid genetic algorithm

```
# Initialization
log     min_lj_03_genetic.log

units   lj
dimension   3
atom_style  atomic
pair_style  lj/cut 2.5
boundary  p p p

```

```
# System definition

region       mybox block -15 15 -15 15 -15 15
create_box  2 mybox
region    mycylin cylinder z 0 0 7.5 INF INF side in
region    mycylou cylinder z 0 0 7.5 INF INF side out
create_atoms  1 random 1000 341341 mycylou
create_atoms  2 random 150 127569 mycylin

# Simulation settings

mass              1 1
mass              2 1
pair_coeff        1 1 1.0 1.0
pair_coeff        2 2 0.5 3.0

# Run

dump    1 all custom 10 min_lj_03_cg_*.dump id type x y z

thermo    1
thermo_style custom step cpu etotal
min_style cg
min_modify dmax 0.1

minimize  1.0e-4 1.0e-6 100 10000
```
Listing 6 Input script for minimizing structure 3 using the conjugate gradient method

```
# Initialization
log     min_lj_03_genetic.log

units   lj
dimension   3
atom_style  atomic
pair_style  lj/cut 2.5
boundary  p p p

# System definition

region       mybox block -15 15 -15 15 -15 15
create_box  2 mybox
region    mycylin cylinder z 0 0 7.5 INF INF side in
region    mycylou cylinder z 0 0 7.5 INF INF side out
create_atoms  1 random 1000 341341 mycylou
create_atoms  2 random 150 127569 mycylin

# Simulation settings

mass              1 1
mass              2 1
pair_coeff        1 1 1.0 1.0
pair_coeff        2 2 0.5 3.0

# Run

dump    1 all custom 1 min_lj_03_genetic_*.dump id type x y z

thermo    100
min_style genetic
min_modify dmax 0.05 numIndividuals 128 p 0.1

minimize  1.0e-4 1.0e-6 5000 100000
```
Listing 7 Input script for minimizing structure 3 using the genetic algorithm

```
# Initialization
log     min_lj_03_ga_hybrid.log

units   lj
dimension   3
atom_style  atomic
pair_style  lj/cut 2.5
boundary  p p p

# System definition

region      mybox block -15 15 -15 15 -15 15
create_box  2 mybox
region    mycylin cylinder z 0 0 7.5 INF INF side in
region    mycylou cylinder z 0 0 7.5 INF INF side out
create_atoms  1 random 1000 341341 mycylou
create_atoms  2 random 150 127569 mycylin

# Simulation settings

mass              1 1
mass              2 1
pair_coeff        1 1 1.0 1.0
pair_coeff        2 2 0.5 3.0
neigh_modify  every 1 delay 5 check yes

# Run

dump    1 all custom 1 min_lj_03_ga_hybrid_*.dump id type x y z

thermo    1
min_style ga_hybrid
min_modify dmax 7.5 numIndividuals 128 p 0.1 localMinSteps 100

minimize  1.0e-4 1.0e-6 10 10000
```
Listing 8 Input script for minimizing structure 3 using the hybrid genetic algorithm