Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2017-0253

Vol. **25**, No. 3, pp. 911-927
March 2019

Optimizing Atomic Structures through Geno-Mathematical Programming

Antti Lahti[1,2,*], Ralf Östermark[3] and Kalevi Kokko[1,2]

[1] *Department of Physics and Astronomy, University of Turku, FI-20014 Turku, Finland.*
[2] *Turku University Centre for Materials and Surfaces (MatSurf), Turku, Finland.*
[3] *Åbo Akademi University, School of Business and Economics, FIN-20500 Turku, Finland.*

Received 15 December 2017; Accepted (in revised version) 18 April 2018

Abstract. In this paper, we describe our initiative to utilize a modern well-tested numerical platform in the field of material physics: the Genetic Hybrid Algorithm (GHA). Our aim is to develop a powerful special-purpose tool for finding ground state structures. Our task is to find the diamond bulk atomic structure of a silicon supercell through optimization. We are using the semi-empirical Tersoff potential. We focus on a 2x2x1 supercell of cubic silicon unit cells; of the 32 atoms present, we have fixed 12 atoms at their correct positions, leaving 20 atoms for optimization. We have been able to find the known global minimum of the system in different 19-, 43- and 60-parameter cases. We compare the results obtained with our algorithm to traditional methods of steepest descent, simulated annealing and basin hopping. The difficulties of the optimization task arise from the local minimum dense energy landscape of materials and a large amount of parameters. We need to navigate our way efficiently through these minima without being stuck in some unfavorable area of the parameter space. We employ different techniques and optimization algorithms to do this.

AMS subject classifications: 82-08

PACS: 81.05.Cy

Key words: Optimization, geno-mathematical programming, bulk silicon, semi-empirical potential.

1 Introduction

Our interest in especially semiconductor-oxide interface and surface structures is due to their prevalence in modern electronics and devices; these structures heavily affect the

[*]Corresponding author. *Email addresses:* ailaht@utu.fi (A. Lahti), ralf.ostermark@abo.fi (R. Östermark), kokko@utu.fi (K. Kokko)

performance of the different components present in the devices. On the interface there are two different interconnected crystal structures often leading to structures hard to predict. This is what makes them a difficult research object. Finding these structures by hand, i.e. designing the interfaces by trial and error needs special knowledge and takes a lot of time, which is why a flexible, powerful optimizing tool would be beneficial for many research problems across the field. Measuring the interface structures experimentally is also difficult as they are buried in the material. That is why simulations and calculations conducted on many different interface models are indispensable in understanding the nature and behavior of these structures. The scale of these structures is often measured in Ångström's (Å) which is short for 10^{-10}m.

We started a collaboration between the School of Business and Economics at Åbo Akademi University and University of Turku's Materials Research Laboratory in order to develop a tool for optimizing atomic structures, with the algorithm especially tuned for interface structures. In geno-mathematical programming artificial intelligence is connected to mathematical programming methodology on parallel supercomputers. The approach provides a powerful basis for coping with difficult irregular optimization problems and solving them concurrently. We were also interested in the performance of our special-purpose algorithm in the physics based problem of optimizing atomic structures of materials, designed using a modern numerical platform as a base. The optimization is done by minimizing the potential energy, measured in electronvolts (eV), of the structure. The difficulty does not lie in finding a nearby local minimum from a given starting structure, as this can usually be achieved through the steepest descent method in a small amount of steps, but in navigating past all these minima to the global minimum. The energy landscape is filled with these local traps that do not reveal much if any indication on where the true global minimum lies.

Exploring the whole landscape is only doable in very small cases. This is because along with the increasing parameter count the number of local minima of the task rises exponentially with the number of atoms: for example with Lennard-Jones clusters it was shown that the number of minima multiplies by around 2-3 per atom added [23]. This makes almost any interesting interface or surface system hard to study. In this article, we show that even our small silicon case can be problematic if not treated properly.

In theory, good molecular dynamics (MD) simulation should be able to find the global minimum given enough time and proper annealing. In practice, the required time is often very large and might require a lot of parameter fine-tuning while still leaving defects at the end of the simulation. Of course, even then we can never be sure that the result is the true global minimum, unless we know the answer beforehand. Large scale computing is a crucial part of bigger MD simulations and advances in that field continue to be made even today, for example speeding up node and core communication [7, 26], reducing memory usage [7], improving threading [14] and creating faster algorithms for force computations [8]. In our work presented in this paper, all the cores work fairly independently, but in the future the algorithm could be expanded to include more communication between the cores. We did a series of MD simulations of our silicon test case shown in Section 2, where we find that while we cannot guarantee the global minimum, it becomes very likely to be found as the simulation time increases.

Advanced techniques like basin hopping [24], meta dynamics [11], swarm simulation [2,3], genetic and evolution based methods have been developed and used over the years to find the global minimum of a system. They have been implemented to various degrees in software like USPEX [15], GASP [27] and CALYPSO [25] to name a few.

2 Silicon: 20-atoms case

Because finding the one global minimum amongst all the multiple local minima is a challenging task, we chose a well-known case of bulk silicon for our first study. The more difficult case of interfaces is left for future research. The structure of silicon we are trying to find is the well-known diamond structure, which can be represented as a cubic box with a periodic boundary condition, also known as a unit cell. We chose our optimization task to be a $2\times2\times1$ a supercell of these cubic diamond cells. The dimensions of this cell were not part of the optimization. The supercell contains 32 atoms in total, but we fixed all of the border atoms in place as shown with grey atoms in Fig. 1. This leaves 20 atoms for optimization, each having spatial coordinates x, y and z that gives us 60 parameters in total. Unlike in some smaller optimization tasks, we could not solve a problem of this size by brute force. The parameters have only simple box constraints determined by the supercell we are using. The cell dimensions are 10.86 Å, 10.86 Å and 5.43 Å, which leads to atom i's coordinates (x_i, y_i, z_i) having the box constraints given in Eq. (2.1):

$$0\leq x_i\leq 10.86,\quad 0\leq y_i\leq 10.86,\quad 0\leq z_i\leq 5.43. \tag{2.1}$$

We have fixed the border atoms for the following reasons:

- The end goal is to produce a package that searches interface and surface structures, which usually have a fixed known bulk structure surrounding the optimized region.
- It provided a starting point for the optimizer while also making it easier for us to analyze if the produced structure was correct or what kind of techniques would be needed to correct it.
- At the start we did not want to concern our optimizer with the periodic boundary conditions.

This leaves three of the four layers to be optimized. These layers have 8, 4 and 8 atoms. The middle layer has 8 atoms in it too, but four of those are frozen in place on the border of the supercell.

We further divide this optimization task into three different cases with 60, 43 and 19 parameters with the latter two having assumptions that reduce the parameter count. In

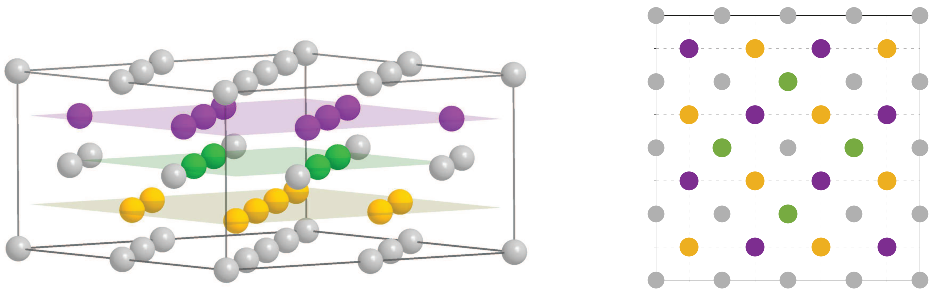
Figure 1: Presentation of our test case target solution from an angled and top-down perspective. The optimized atoms are colored while the fixed atoms are grey and located on the borders of the supercell. This is the ideal diamond structure for silicon we are trying to achieve.

the 60-parameter case all atoms have their positions optimized independently. The 43-parameter case assumes that the atoms reside in 3 layers. All the atoms share 3 height coordinates in total, reducing the parameter count by 17. The final 19-parameter case assumes, in addition that the two 8 atom layers have the atoms arranged in squares with 4 rows and 4 columns. This drops the parameter count from 43 to 19, as the new 8 row/column parameters replace the 32 x- and y-parameters of 16 atoms. In the structure of Fig. 1 these 8 parameters correspond to columns and rows of the yellow and purple atoms. In the global minimum structure these rows and columns are evenly spaced.

Especially the layer assumption is reasonable in many interface and surface models as long as we give some leeway for the atoms within the layer. In the 19-parameter case the row/column assumptions for atoms on a layer can also be useful as the low energy structures often have some form of symmetry that just needs to be found. In general using symmetries can be very advantageous as the reduction in parameters yields a smaller dimensional problem to be explored. The structures produced through these high symmetry cases can also be used as a starting point for runs where the symmetry assumptions are relaxed.

2.1 The Tersoff potential

We tried searching for the diamond structure through traditional molecular dynamics to give us a reference point for comparison. We used the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) software for our simulations [17]. In this test case, we are using the Tersoff-potential [21], which is a fast many body bond-order potential[†]. The Tersoff potential allows fast computations and still describes silicon fairly well in different environments. Comparative testing with alternative potentials and pseudo-potential methods (e.g. the slower but more accurate ReaxFF-potential [22] and VASP [10]) is left for future research.

The potential energy formula consists of a sum of two- and three-body interactions

[†]Potential energy refers to the Tersoff potential throughout the paper

and is implemented in LAMMPS in the following form:

$$
\begin{aligned}
&E=\frac{1}{2}\sum_{i}\sum_{i\neq j} f_C(r_{ij})[f_R(r_{ij})+b_{ij}f_A(r_{ij})],\\
&f_R(r)=Ae^{-\lambda_1 r}, \quad f_A(r)=-Be^{-\lambda_2 r},\\
&f_C(r)=\begin{cases} 1, & r<R-D,\\ \frac{1}{2}-\frac{1}{2}\sin(\frac{\pi}{2}\frac{r-R}{D}), & R-D<r<R+D,\\ 0, & r>R+D,\end{cases}\\
&b_{ij}=(1+\beta^n\zeta_{ij}^n)^{-\frac{1}{2n}},\\
&\zeta_{ij}=\sum_{k\neq i,j} f_C(r_{ik})g(\theta_{ijk})e^{\lambda_3^m(r_{ij}-r_{ik})^m},\\
&g(\theta)=\gamma\left(1+\frac{c^2}{d^2}-\frac{c^2}{d^2+(\cos\theta-\cos\theta_0)^2}\right),
\end{aligned}
$$

where the sums go over all atoms, r_{ij} is the distance between atoms i and j, θ_{ijk} is the bond angle between bonds ij and ik, f_C is a smooth cutoff function, f_R is a repulsive two-body interaction and term $b_{ij}f_A$ is the three-body interaction. The parameters A, B, D, R, β, λ_1, λ_2, λ_3, γ, m, n, c, d and θ_0, are material specific constants. We used the values specified in the Ref. [21] for these parameters.

2.2 Steepest descent and annealing simulations

We wanted to see how good the traditional molecular dynamics methods are at solving this problem. Starting with the steepest descent method, we generated 100000 random initial structures and applied the steepest descent to each of them. The optimization was terminated when a local minimum was reached within the desired energy tolerance; we found out that increasing the tolerance from 10^{-8} eV improved the results only marginally. The energy distribution of the found minimum in these optimization runs is presented in Fig. 2 along with the distribution of optimization steps required to find the minimum. From this figure we see that none of these runs could even come close to finding the global minimum around -148 eV. Indeed, most of the runs stop in the region -135 eV to -130 eV, which is the same region our algorithm easily is stuck into, but more on this later in Section 4.

It is worth noting that if we free the border atoms, the steepest descent method actually becomes noticeably more effective, shifting the minima peak and allowing us to reach lower energies (Fig. 2 dashed line). This must be due to the cell being less rigid, allowing the atoms to move around more freely. However, we chose to keep these atoms frozen, as interface structure calculations do have a more rigid bulk part surrounding the interface. In any future cases, however one should consider keeping the surrounding bulk relatively flexible, in order not to hamper the optimization process.

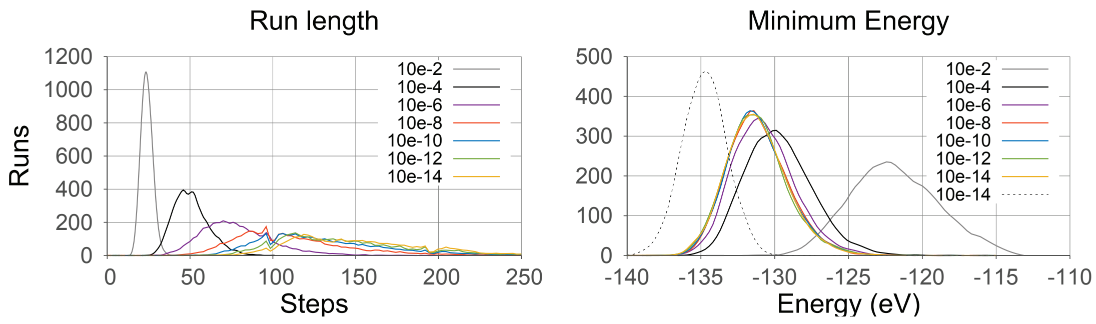

Figure 2: These are the distributions of lowest energy and steps required to achieve that energy from 100000 steepest descent simulations. Different curves correspond to a different energy tolerance used by LAMMPS to end the search. The solid lines correspond to runs with the border atoms fixed in place. The dashed line is from a similar simulation with only one fixed atom. Fixing only one atom made the simulations slightly faster than freeing all of the atoms. The graphs have been smoothed. We see that around the tolerance of 10^{-8} the results start converging.

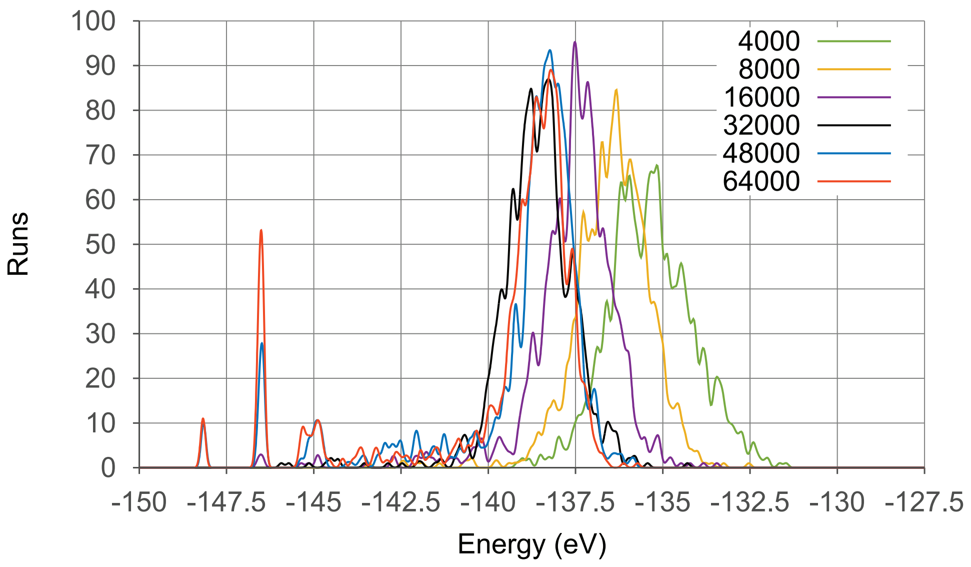

Figure 3: LAMMPS annealing done from random starting positions. At the end of each simulation, we searched the nearest minimum through the steepest descent method. Each line corresponds to a different amount of applied annealing. The graph legend gives the amount of annealing steps done by LAMMPS, which also equals to simulated time in femtoseconds.

After these simulations, we switched to considerably slower annealing simulations [9], where we were able to obtain better results (Fig. 3). In these annealing simulations the system is in a heat bath, which gives the atoms enough kinetic energy to overcome the potential energy barriers. The temperature of the system is slowly reduced to a very low temperature allowing the structure to settle in a local minimum.

In these simulations one step done by LAMMPS corresponds to 1 femtosecond(fs) of simulated time. The calculations now absorb considerably more CPU time, as we set a single annealing to take 4000, 8000, 16000, 32000 48000 and 64000 steps. In contrast most of the steepest descent calculations took between 50 and 300 steps, with the average being around 100. Because the simulations take that much longer, we only did 1000 simulations/annealing time. In Fig. 3, we can see that most runs end with an energy above -140 eV, even if we increase the simulation time.

We see that even this simple case is hard to solve quickly through traditional methods of annealing and steepest descent. Even with 64000 steps, annealing had a success rate of only 1.1%. If we increase the duration of the simulation by tenfold to 640000 steps, the success rate rises noticeably to 41.9%. This run is not shown in Fig. 3 as it would cause a very tall spike at the global minimum energy and render the graph harder to interpret.

We believe this is a good first test case for our optimizer providing enough challenge and a straightforward way to progress into more challenging SiO_2/Si-interfaces and surface structures in the future.

3 Algorithm: methods and techniques

3.1 Introduction to genetic hybrid algorithms

Genetic hybrid algorithms are a combination of two parts, genetic and local search, that complement each others. Genetic algorithms are known for being population based search algorithms, that try to copy the process of natural evolution through natural selection and genetic dynamics. They were first described by Holland in the 70s [6].

In most cases the genetic algorithms are good at exploring the landscape of the whole parameter space and discovering the regions which possibly have the wanted global minimum [4,18]. They do this by trying to use the information gained from the known good solutions and exploring the parameter space widely. The genetic algorithms however sometimes have trouble actually pin pointing the global minimum after they have found the region it belongs to. This is often caused by the algorithm's inability to make the necessary small changes to the system [19].

Different local search methods can cover for this weakness of genetic algorithms [5, 12]. From a given starting point the local search methods are usually very efficient at exploring the region and finding the nearby local minima. The local search methods use the information available from the surrounding area to find a good nearby minimum and continue from there on. The quality of starting point is critical for the method. If the starting point is not in the funnel of the global minimum, then it is very unlikely that the global minimum will be found.

Genetic hybrid algorithm's power comes from the combination of these two methods, where you take advantage of their individual strengths [5,12]. By using the genetics to explore the vast parameter space and using the different local search methods to explore the interesting areas for the global minimum. In structure optimization the genetic methods include operations like exchanging layers between two structures, using parts of known low energy structures to produce very different possible solution structures and different kinds of smaller mutations, like just moving atoms around and switching coordinates/layers in a given structure. The most basic local search methods are nearest minimum locating algorithms like steepest descent and different sequential quadratic programming (SQP) algorithms. When these are combined with methods like basin hopping with its various forms and annealing, we get local search methods that are good at exploring the nearby minima neighborhood.

3.2 GHA platform and our solver

We are using the Genetic Hybrid Algorithm (GHA) as a platform for algorithmic development [16]. It is a computational platform for designing special purpose algorithms for difficult numerical problems, extensively tested in economics and engineering. Through GHA we can access different algorithms and non-linear solvers, like nlpqlp, snopt, fsqp, dncong, cplex, kkt_ql and gurobi, which are powerful tools for mixed-integer non-linear programming problems. We have linked LAMMPS to GHA as a library for easy and fast potential energy evaluation. We can also extract forces from LAMMPS for fast gradient evaluation. For the local search, in addition to the algorithms implemented through GHA, LAMMPS offers annealing and steepest descent. Later in Table 3 we present a comparison done between these algorithms and a sequential quadratic programming (SQP) implementation.

Next we introduce our program structure, parallelization and the essential low-level logic (Fig. 4). The searches we have done were multi-core jobs, but there is no search boosting communication between the cores during the search. All cores are prepared independently from the same parameters in a preprocessor-function. From there each core runs the search for a preset length. The exception to this is when we at times choose to stop the search when one of the cores has found the solution, sending out an interrupt signal that will force all the other cores to stop the search. At the end each core will do their core specific post processing involving mostly clean up of the memory and some result processing. The root level I/O is done last, including most importantly information of the search.

More specifically each core performs the global minimum search in a series of runs. The general structure of the search has been illustrated in Fig. 4. Each of these runs starts with the generation of a large pool of random structures. These are then ranked by their energy and only a population of POP members is retained; typically this is between 4 and 64 in our calculations. With this population we will then perform genetic manipulation, in this case arithmetic cross-over and non-uniform mutations, and continue processing them in series of iteration loops.

Each member of the generated population launches into a series of iteration loops. One loop consisting of a series of mixed evaluator() and accelerator() calls. The standard progression is done through box-constrained optimization(BFGS) using evaluator() calls while the accelerator() is responsible for more radical changes, like different kind of mutations to the structure.

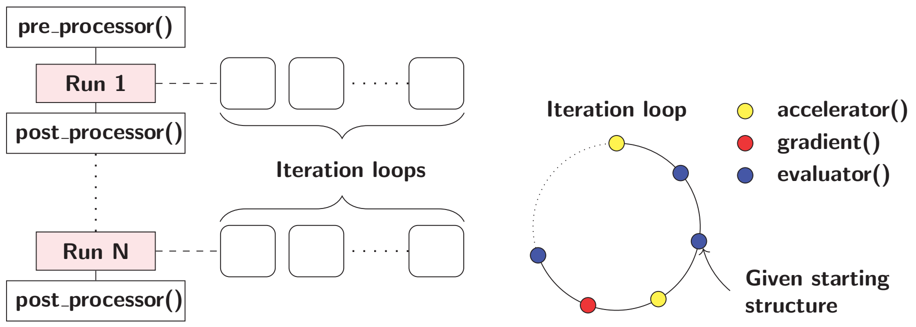

Figure 4: The general structure of the main loops and working of GHA in our case. The main program consists of a series of runs, which launch into multiple iterative loops, one for each population member, that try to solve the problem from different starting positions. The figure is only illustrative as the accelerator()-function is called when it is needed.

After the set amount of iteration cycles only the best structure with lowest energy will continue to the next run cycle; others are discarded after the run. The search ends after the chosen number of completed runs is reached. Since we know the global minimum structure, we also choose to end the search early incase a core reaches the solution.

A very important genetic part of our problem specific accelerator() code are the different implemented mutations that are used to generate new structures from a given structure. The most efficient type of mutations to the structure proved to be the simplest ones where we move one or two atoms simultaneously to a better position. Simultaneous moves of two atoms are especially helpful in breaking the atoms out of strong local minimum potential wells.

Also in the layered 19- and 43-parameter cases it was pretty straight forward to implement a method that crosses the layers from two models with each other. Yet, we can use this type of mutation to nudge the structure from a local minimum to the funnel of another minimum. This is much harder to implement in the 60-parameter case, as the layers are much harder to identify. One of the important reasons for mutations is their ability to produce vastly different structures without starting from scratch so we do not end up exploring only a small portion of the parameter space.

4 Results and comparisons

4.1 Basin hopping

Basin hopping is a well-known algorithm for structure optimization [24]. We implemented a simple form of basin hopping for comparison purposes. To put it shortly, this method tries to jump from a local minimum well to another, eventually hoping to funnel into the global minimum.

We start from a local minimum, chosen by randomly placing the atoms and determining the minimum through steepest descent. Then we choose a nearby structure as

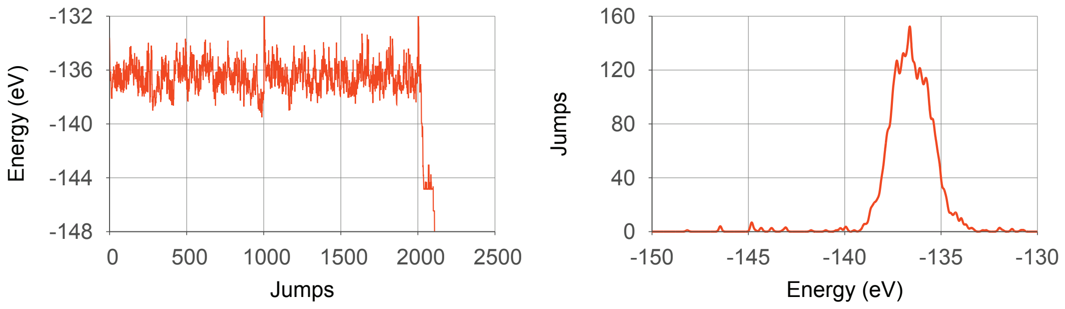

Figure 5: On the left, the energy progression of a typical 2108 jump basin hopping run. The energy spikes shortly at every 1000 jumps because we randomize our current location. We see that after we break the energy barrier of -140 eV we reach the ultimate diamond structure at -148 eV quickly. On the right we present the energy distribution of the terminal jump location, showing where we spend most of the time in the run.

the jump location, coarsely evaluating its energy by assigning it the value of the nearest known local minimum energy. The jump is carried out using the Metropolis algorithm: if the estimated energy is lower than the energy of our current location, we jump. Otherwise the jump is done if a random number is lower than $\exp(-\frac{\Delta E}{kT})$, where ΔE is the energy difference of these two locations, k is Boltzmann's constant and T is a temperature constant affecting the jump chance. If the random number is higher, we try again with a different jump location until a jump is successful. After the jump, we use steepest descent to reach the local minimum and prepare for another jump.

The method has its problems as it too can get trapped into a region of local minima and never reach the funnel of the global minimum. We noticed that in our simple case too, but we were able to avoid the trapping by randomizing our current location every 1000 jumps. The same effect could be achieved by doing a large enough jump – given that the required step length can be set correctly. We did a series of 1000 search runs and were able to find the global minimum consistently, although the time it takes varies a lot. All searches found the solution: on average it would take around 2237 jumps and 38 steps to find the local minimum after the jump. The spread is very high as sometimes we could find the minimum almost right away, in less than 50 jumps, and sometimes it could even take over 10000 jumps. This task as a whole also took a bit over 86 million steps for LAMMPS to complete. Each subsequent jump takes slightly longer on average due to each jump using the known local minima in the jumping process. This list of minima grows as the search progresses leading to an approximately quadratic relation between the search time and the number of jumps made. This could become problematic in a more complex task, requiring more advanced methods for processing and sorting the stored minima. The average 2237 jump search would roughly take around 80 seconds of CPU time. In Fig. 5 we have illustrated the energy distribution of the found minima and the progression of one basin hopping search that took 2018 jumps. We see that most of the jumps land on local minima in the range from -138 eV to -135 eV. Once we get past the apparent barrier around -139 eV the search quickly finds the global minimum.

4.2 Geno-mathematical search

Having implemented the GHA-LAMMPS interface and introduced the problem specific code described above, we conducted a series of independent parallel runs on CrayXC40 at CSC (Helsinki). The parameter cases are shown in Table 1. The simulations with only 1 solution were done using an early mesh interrupt broadcasted when a core found the known optimum solution at -148.17 eV. This was determined by the evaluated potential energy of the structure. Since the number of local solutions close to the global optimum is small and the distance between the global minimum and the nearest local solution is tangible, the advancement to the global solution from a favorable initial point is relatively fast. The nearest local minima are single-atom dislocations at around -146 eV.

Table 1: Results obtained with 1024/4096 parallel cores, where each search found the known global energy optimum $f^* = -1.4817e+02$ eV. The runs with only 1 solution had an interruption signal broadcasted to stop the search once the solution was found. Each core conducts its own search. Each row corresponds to one search done with the given number of cores in parallel. The success rate for the interrupt searches is left out, because only one core finds the solution in each of the cases before the search is forced to stop by the interrupt signal.

		Success	f^*	Average time
n	Cores	rate	energy (eV)	(CPU s/core)
(i)		*Success rate in massive search*		
19	1024	98.9%	-1.4817e+02	1920
19	1024	89.2%	-1.4817e+02	1080
19	1024	99.0%	-1.4817e+02	1740
(ii)		*Solution speed with early mesh interrupt*		
19	1024	-	-1.4817e+02	25.2
19	4096	-	-1.4817e+02	12.6
43	1024	-	-1.4817e+02	1740
43	4096	-	-1.4817e+02	995
60	1024	-	-1.4817e+02	43.2
60	4096	-	-1.4817e+02	63.6

The message of Table 1 is that, when early mesh interrupt is activated, the silicon structure optimization problem is solved to optimum ($f^* = -1.4817e+02$) in at most 30 CPU-minutes from an arbitrary starting point using concurrent search with 1024 parallel cores and different parametrizations. With early mesh interrupt and $n = 19$ and $n = 60$ the processing time is less than 64 CPU-seconds. Studying the sensitivity of CPU-time to mesh size and corroborating the evidence with massively parallel Monte Carlo simulation are left for future research. We also note that the efficiency of our algorithm is critically dependent on the starting point for the local search. Whereas we have applied simple genetic manipulation, more advanced initialization techniques applied in future development efforts, such as variants of e.g. Basin hopping may influence performance significantly.

The $n=43$ search takes significantly longer to complete than the other cases. We believe this must be due to the rigidity problems mentioned earlier in Section 2.2. The constraints on the system cause high-energy barriers that are hard to overcome for the system. We are not sure why this does not affect the $n=19$ case. The significant drop in parameter count may overweight the problems caused by the energy barriers.

We did another series of searches, shown in Table 2, where we fixed the random number seed to core specific values to give us replicable starting structures for the runs. The purpose of these runs is to show how short high population runs compare to longer runs with a smaller population. From these searches we see that using higher population in a short run is much more beneficial than just letting a low population search go on for longer. The results with the interrupt signal can be quite misleading due to how the parallelization behaves in very short runs. On the operating system level different cores start the search at different times. This becomes apparent, when we notice that the similar run with no interrupt signal actually has a core find the solution over twice as fast as with the interrupt signal. This is why we left out the mean time for these runs as it would be heavily influenced by the cores that have not even started yet. If the search time is measured in minutes the differences in core start up aren't significant anymore.

In the long run, the 60-parameter case is slower than the 19-parameter case as expected. The completion time of the cores – the time needed to obtain the global optimum from a unique random starting point – in the long and short 60-parameter run is presented in Fig. 6. The graph suggests that the high population runs are more suitable for short search runs.

Table 2: Runs using the same random variable seed as in Table 3. The purpose of these calculations was to pit long search, with low population count (POP) and many iterations, to short search, with less iterations and high population count. The short search, while having much lower success rate, finds the global minimum faster. The highest reached runtime was nearing 14 hours in the 60-parameter case. The mean time for short runs with interrupt is left out, because it doesn't give useful information due to all searches being stopped when one of the cores finds the solution. Success rate is left out for the same reason, as only one solution is found.

Search Job	n	POP	Cores	Success	Time (s)		Energy (eV)		
				rate	Mean	Fastest	Best	Mean	Variance
Short with interrupt	19	64	4096	-	-	18.1	-1.4817e+02	-119.19	88.39
Short with interrupt	60	64	4096	-	-	116	-1.4817e+02	-122.16	20.22
Short no interrupt	60	64	1024	3.71%	95.3	42.6	-1.4817e+02	-135.95	22.72
Short no interrupt	60	64	4096	3.03%	94.4	41.8	-1.4817e+02	-135.86	21.77
Long no interrupt	19	4	1024	97.9%	1940	12.7	-1.4817e+02	-148.06	1.208
Long no interrupt	60	4	1024	99.5%	5090	66.4	-1.4817e+02	-148.13	0.3406

We did three searches with mesh size 1024 where we fixed the seed for the random number generation like in Table 2 and tried to optimize the structures using only one of three methods: in the first search we only used the steepest descent method from LAMMPS, in the second we used the BFGS/SQP-algorithm [1] and in the third we used only LAMMPS annealing. The purpose of these runs was to see how these methods

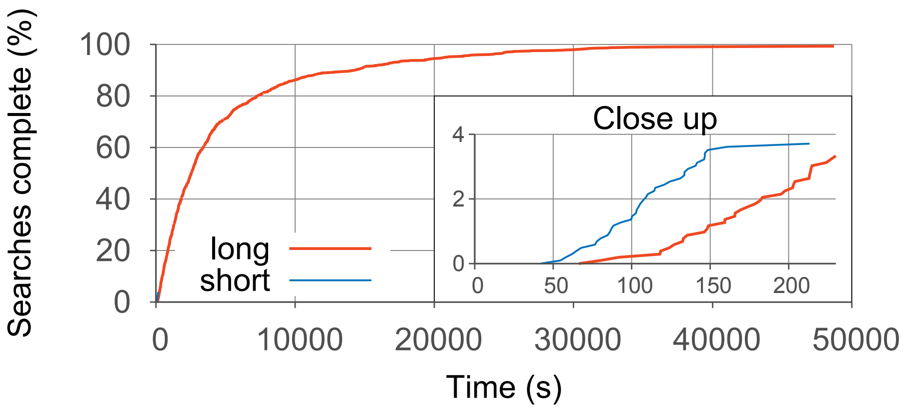

Figure 6: The search completion rate of different cores as a function of time in the case of the long and short 60-parameter run in Table 2. In the close up we see that the short run(blue line) initially has faster solution rate, but the solution rate starts to stagnate towards the end of the run and the long run with a smaller population catches up to it.

perform in isolation compared to our results in Table 2 and each others, given the same starting structures. There was no big difference between the first two searches neither in the obtained energy values nor in the solution time. The results are presented in Table 3.

Table 3: Comparing the steepest descent and BFGS/SQP-algorithms from the same starting position, with mesh size 1024. We also listed a similar simulation that used annealing only.

Search Job	n	Mean time (ms)	Best energy (eV)	Mean energy (eV)	Variance
Steepest descent	60	1.09	-135.945	-129.774	5.74455
BFGS/SQP	60	1.11	-135.397	-129.306	5.39542
LAMMPS annealing	60	11.6	-131.15	-123.974	6.2765

5 Discussion

It is worth noting that often we are not only interested in the global minimum but also in the local minima nearby as they may be more stable under different circumstances. Some of them can also be useful when comparing simulations to data obtained from experiments, as crystals are not perfect in reality. They can sometimes also give useful information about different reaction paths, especially when studying a system with defects, complex interfaces or surfaces. This kind of information can be stored during the simulation and analyzed afterwards. Through replicable core specific random numbers, we can study the solution process of any interesting crystal structure in detail on a single core by specifying the output level of GHA correspondingly.

There is still the issue of identifying the interesting minima, as most of the found local solutions are uninteresting amorphic structures. There are measures for differentiating structure, like the fingerprint function used in USPEX and geometric measures like neighbor counts and the radial distribution function [13].

As we found out with the LAMMPS simulations, fixing some of the atoms actually does not help at all; instead, it hinders the optimization by making some energy barriers harder to bypass. Further testing is required, but it would seem that fixing atoms in place, even if those spots are known correct locations, is not an efficient choice. Restricting their movement to a small area surrounding this location might allow the structure to flex around the energy barriers.

We have probed different methods for generating the initial raw pool of structures. As a word of caution, it is easy to make a structure generation algorithm that leaves out a big portion of the possible structures, but it is harder to control what is left out. We have to be careful not to rule out some possibly successful structures. At the same time, some methods produce mostly unwanted high-energy structures or, for example identical inferior structures. There is also the question of how much we want to refine the initial structures. When two atoms are close to each other, the associated two-body term gives an exponentially growing positive contribution to the total potential energy of the structure. This makes energy-based ranking of the structures slightly problematic, so we have either discarded or modified these converging structures to get a proper evaluation of their energy.

One way to circumvent energy spiking is to design an initial structure having every atom close to the typical silicon bond distance from its neighbor. The procedure is not too time consuming and ensures we are not wasting potentially good structures, since atom pair proximity related energy spikes are avoided. Another way is to develop the structures with unusually high energies a bit, either through applying some steepest descent type of optimization or identifying and moving the problematic atoms.

We tried a method of inserting atoms into the cell one by one. However, we quickly noticed that this was not an effective strategy, as it leads to the atoms clustering in some areas of the cell. We believe this is because the silicon atoms want to form quadruple bonds, which is not possible in their natural location when the simulation area is still half-empty in the beginning of the optimization. As we start inserting the atoms, the only place where they can form these bonds to achieve lower energy is in the region where there already are atoms. This leads to unwanted clustering in some region of the cell. Perhaps this could be circumvented by restricting the placement of an atom to areas far from the previously placed atoms.

Another issue is how good the initial structures should be. With the future in mind, creating an algorithm that generates as good initial structures as possible is desirable. However, we did not want to have too good structures right now, as one of the points of this test case was to assess the performance of the GHA optimizer and develop methods to advance through complex energy landscapes to the global minimum.

Even after we implemented more structure optimization specific techniques the major energy obstacle was between -140 eV and -130 eV. Most of the calculations would end up there easily, but getting forward proved to be challenging. We tried analyzing the structures around the problematic energy area above -140 eV, but the local minima found there did not really resemble the desired diamond structure. It is a known problem in optimization, that the structures have a tendency to turn out fairly disordered and amorphic. That was the case here too. We tried developing some metric to differentiate these structures and find the ones that could potentially lead to the global minimum but we did not find such metric.

If a structure could break the barrier at 140 eV, the optimizer often ended up progressing to the perfect diamond structure around -148.17 eV. We believe that the small number of minima between -140 eV and the global minimum of -148.17 eV and shallowness of the minima in this region make it easy for the system to jump to the funnel of the global minimum. An eyeball test indicates a clear ordered structure with only one or two atomic dislocations and rest of the diamond structure intact around this energy region. That means that most of the minima between -148 eV and -140 eV correspond to structures, which can be corrected with just a few well-done atom displacements. Our calculations confirm this, as passing the barrier around -140 eV usually leads to a rapid advancement to the global minimum.

In the annealing Fig. 3, we see that the graph becomes spiky around this region from -148 eV to -140 eV, with small gaps between the spikes. The spikes are not very sharp though, which we believe is because of the frozen atoms in the cell. Most of the spikes correspond to some 1-2 atom dislocations. For example, ideally, the single atom dislocations should correspond to a unique structure, but that is not the case here. When we introduce a dislocation defect, its location affects the energy of the resulting structure. If the dislocation is near the corner or the edge of the cell, the environment reacts differently as some atoms are unable to move. We tested this by doing simple dislocation defects to the system and noticed a 0.5 eV difference depending on where the dislocation was. With an analogous two-atom dislocation, we noticed differences up to 2 eV.

It is clear that the solution for the type of task considered in this study is very dependent on the initial structure. As we saw in our long searches, a single core could take anywhere from ten seconds to fourteen hours to reach the known global minimum. At the same time, concurrent processing of our algorithm with a large mesh from core specific random numbers with early mesh interrupt activated guarantees – beyond reasonable doubt – the global solution in seconds, when modelling the structure as a full 60-parameter or restricted 19-parameter problem. This is indeed an encouraging and exceptional result in a difficult global optimization problem.

6 Conclusions

We have successfully used the optimizing platform GHA in our case of silicon structure optimization and presented the results. GHA has inbuilt support for the necessary large scale computing and has the required versatility to solve the case, producing results that are comparable to other methods like basin hopping and annealing. With the future in mind, we hope to refine and extend this approach to more relevant material physics problems related to oxides, interfaces and defects.

In order to solve more complicated atomic structures than the one probed in this study, we would have to be able to reduce the processing time significantly and guarantee the optimal solution using only a single processor. This would allow, e.g. massively parallel search of interesting crystal candidates using modern Geno-mathematical techniques, with potential for new discoveries in atomic interface and surface structures.

At the same time, there is a lot of room for improvement of the search algorithm. Incorporating the basin hopping and other popular methods is one way we could try to achieve this. Another possibility is to use elaborate statistical distributions for the starting positions. In the searches presented in this paper, we used the uniform distribution, but distributions generated through, e.g. Copula-theory [20] might prove to be better suited for this case.

Acknowledgments

The Magnus Ehrnrooth Foundation is acknowledged for financial support (A. L.). The computer resources of the Finnish IT Center for Science (CSC) and the FGI project (Finland) are acknowledged. Advice of the experts at CSC on installing LAMMPS on Cray XC40 is gratefully acknowledged.

References

[1] X. Chen. Convergence of the bfgs method for lc1 convex constrained optimization. *J. Control Optim.*, 34:20512063, 1996.

[2] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan*, pages 39–43, 1995.

[3] R. C. Eberhart and J. Kennedy. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ*, pages 1942–1948, 1995.

[4] D. E. Goldberg. Genetic algorithms in search, optimization, and machine learnin. *Addison-Wesley*, 1989.

[5] W. E. Hart. Adaptive global optimization with local search. *Doctoral Dissertation. San Diego: University of California*, 1994.

[6] J. Holland. Adaptation in natural and artificial systems. *The University of Michigan*, 1975.

[7] C. Hu, H. Bai, X. He, B. Zhang, N. Nie, X. Wang, and Y. Ren. Crystal md: The massively parallel molecular dynamics software for metal with bcc structure. *Computer Physics Communications*, 211(Supplement C):73–78, 2017. High Performance Computing for Advanced Modeling and Simulation of Materials.

[8] C. Hu, X. Wang, J. Li, X. He, S. Li, Y. Feng, S. Yang, and H. Bai. Kernel optimization for short-range molecular dynamics. *Computer Physics Communications*, 211(Supplement C):31–40, 2017. High Performance Computing for Advanced Modeling and Simulation of Materials.

[9] S. Kirkpatrick, G. J. C. D., and M. P. Vecchi. Optimization by simulated annealing. *J Phys Cond Matter*, 220:671680, 1983.

[10] G. Kresse and J. Hafner. Ab initio molecular dynamics for liquid metals. *Physical Review B*, 47(1):558–561, 1993.

[11] A. Laio and M. Parrinello. Escaping free-energy minima. *Proceedings of the National Academy of Sciences*, 99(20):12562–12566, 2002.

[12] F. G. Lobo and D. E. Goldberg. Decision making in a hybrid genetic algorithm. *IEEE International Conference on evolutionary Computation, USA: IEEE Press*, pages 122–125, 1997.

[13] A. O. Lyakhov, A. R. Oganov, and M. Valle. How to predict very large and complex crystal structures. *Computer Physics Communications*, 181:1623–1632, 2010.

[14] C. M. Mangiardi and R. Meyer. A hybrid algorithm for parallel molecular dynamics simulations. *Computer Physics Communications*, 219(Supplement C):196–208, 2017.

[15] A. Oganov and C. Glas. Crystal structure prediction using ab initio evolutionary techniques: Principles and applications. *The Journal of Chemical Physics*, 124:244704, 2006.

[16] R. Östermark. A multipurpose parallel genetic hybrid algorithm for non-linear non-convex programming problems. *European Journal of Operational Research*, 152(1):195–214, 2004.

[17] S. Plimpton. Bayesian method for global optimization. *J. Comp. Phys.*, 117:1–19, 1995.

[18] P. Preux and E.-G. Talbi. Towards hybrid evolutionary algorithms. *International Transactions in Operational Research*, 6:557–570, 1999.

[19] C. Reeves. Genetic algorithms and neighbourhood search. *Evolutionary Computing, AISB Workshop*, 865, 1975.

[20] A. Sklar. Fonctions de repartition a n dimensions et leurs marges. *Publ. Inst. Statist. Univ. Paris*, 8:229–231, 1959.

[21] J. Tersoff. New empirical approach for the structure and energy of covalent systems. *Physical Review B*, 37(12):6991–6999, 1988.

[22] A. van Duin, S. Dasgupta, F. Lorant, and W. A. Goddard. Reaxff: A reactive force field for hydrocarbons. *Journal of Physical Chemistry*, A105:9396–9409, 2001.

[23] P. K. Venkatesh, M. H. Cohen, R. W. Carr, and A. M. Dean. Bayesian method for global optimization. *Phys. Rev. E*, 55(5):6219–6232, 1997.

[24] D. J. Wales and J. P. K. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *J. Phys. Chem.*, A 101:5111, 1997.

[25] Y. Wang, J. Lv, L. Zhu, and Y. Ma. Calypso: A method for crystal structure prediction. *Comput. Phys. Commun.*, 183:2063, 2012.

[26] B. Wu, S. Li, Y. Zhang, and N. Nie. Hybrid-optimization strategy for the communication of large-scale kinetic monte carlo simulation. *Computer Physics Communications*, 211(Supplement C):113–123, 2017. High Performance Computing for Advanced Modeling and Simulation of Materials.

[27] T. WW and H. RG. A grand canonical genetic algorithm for the prediction of multicomponent phase diagrams and testing empirical potentials. *J Phys Cond Matter*, 25:495401, 2013.