Genetic Algorithms for Solving the Global Geometry Optimization Problem

Evaluating Initialization and Crossover Strategies for Lennard-Jones Cluster Optimization

Emīls Dzintars[1]

Supervisor(s): Peter Bosman [1] [2], Anton Bouter [2], Vanessa Volz [2]

[1]**EEMCS, Delft University of Technology, The Netherlands**
[2]**Centrum Wiskunde Informatica, Amsterdam, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Emīls Dzintars
Final project course: CSE3000 Research Project
Thesis committee: Peter Bosman, Anton Bouter, Vanessa Volz, Thomas Abeel

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Discovery of new materials is essential in a lot if different fields, such as, space exploration, maritime industry and others. To stop new materials undergoing spontaneous reactions or reacting with the environment, they have to stable or at least metastable. That is where Global Geometry Optimization comes in, which tries to find global minimum on its Potential Energy Surface. But modeling atoms is difficult as there is complex interplay of forces between each atom. A common simplification is to use the Lennard-Jones potential, which treats atoms as 3D points and models their interactions based on distance. To find global minimum, Genetic Algorithm will be used and this research plans to study multiple different crossover strategies(sphere cut splice, uniform, two point, one point, arithmetical, plane cut splice crossover) and initialization strategies(box, sphere, octahedron initialization) to determine, which is the best for finding minimum for Lennard-Jones clusters. After analyzing results, it is inconclusive, which crossover produces best results, but it is clear that arithmetical produces worst results and octahedron initialization produces best results for clusters with size 10 to 15.

1 Introduction

The discovery of new materials plays a crucial role in advancing various fields, particularly those that require a diverse range of materials to test under different conditions. Such advancements can significantly enhance current technologies and capabilities, for example, in space exploration [11] [15] and the maritime industry [10].

To effectively utilize a new material and harness its properties, it must be stable or at least metastable. If a material is unstable, it may undergo spontaneous reactions or interact with its environment, potentially altering its structure and, consequently, its properties [14]. A key factor in achieving stability is ensuring that the atomic configuration of a cluster corresponds to a global minimum on its potential energy surface (PES). This is where Global Geometry Optimization (GGO) becomes essential, identifying the molecular structure with the lowest potential energy.

Modeling atomic systems is inherently challenging due to the complex interplay of forces acting on each atom. A common simplification is to use the Lennard-Jones potential, which treats atoms as point particles and models their interactions based on distance. Equation (1) potential is widely applied in areas such as protein folding [12].

$$V(r) = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \tag{1}$$

$V(r)$	Intermolecular potential between the two atoms
ε	Well depth; a measure of how strongly the two particles attract each other
σ	Finite distance at which the inter-particle potential is zero
r	Distance of separation between both atoms

[9]

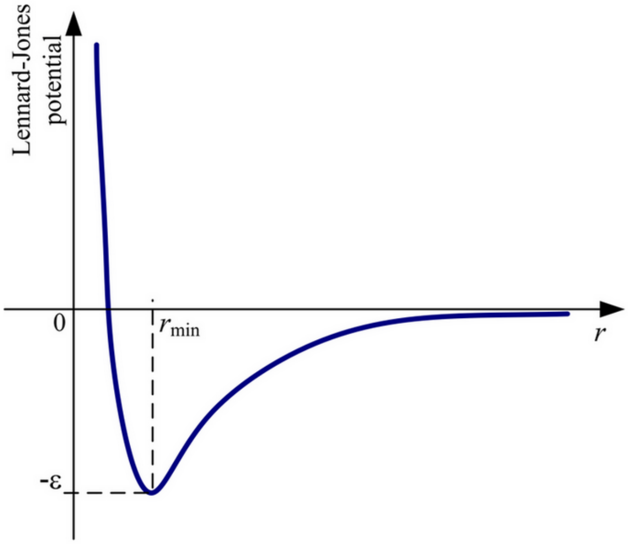


Figure 1: Lennard Jones potential plot

Simplest algorithm to find global minimum for Lennard-Jones clusters would be Random Search Algorithm, where it would generate a random cluster, calculate the potential energy of the cluster, compare it to the last lowest potential energy cluster and replace if it is lower. Additional improvement would be to add local optimization after generating a new cluster, but it would still be ineffective and useful for only small Lennard-Jones clusters.

To solve this model, Genetic algorithm(GA) will be used, which is also used in all kinds of other problems, such as economics, engineering, politics, management, and engineering [7]. It is considered to be population based metaheuristic algorithm, which mimics the Darwinian theory of survival of fittest in nature. Each iteration a new population is created from the previous population using genetic operations(mutation and crossover). Mutation operator takes only one parent as an input and produces one or multiple children, which are used in the future generations and are similar but not equal to the parent. Crossover is similar to mutation operator, but instead of one parent it has two parents and uses characteristics from both parents. A basic GA algorithm starts by randomly initializing its elements in the domain. Calculate fitness of each element. Select elements for crossover or mutation or both, and produces a new population, which then is used in the next iteration until a stopping condition is reached. [7].

First well-known study to use genetic algorithms for molecular structures was Bernd Hartke for small silicon clusters[4] [5], where clusters were encoded as simple bit strings, mutations are bit flips and used one-point crossover.

With time researchers have created different GAs with different genetic operators and operator sequences, such as arimethical crossover, plane-cut-crossover, sphere-cut-crossover, twist operator, interior operator and other operators [13]. The most notable is plane-cut-crossover, which is used in a many research papers with positive results [1]. In addition there are also multiple initialization strategies, such as, box, sphere, octahedron initialization.

Although there are a lot of different studies conducted on GAs, none of them have compared multiple initialization strategies and crossover strategies on Lennard Jones clusters. That is why the study plan is to:

- **Compare the run times of different crossover methods.**
- **Compare the run times of different initialization methods.**

Context: These comparisons are performed using a genetic algorithm to find the global minimum of Lennard-Jones clusters.

This could help understand effectiveness, advantages, disadvantages of certain crossovers and initialization strategies or validate best practices, which are used today.

In these experiments, all variations of the genetic algorithm are applied solely to carbon atoms, with Python libraries using angstroms (Å) as units of length and electronvolts (eV) as units of energy.

This paper is organized into seven sections: Introduction, Baseline Genetic Algorithm, Strategies, Experimental Setup and Results, Discussion, Responsible Research, and Conclusion.

- **Baseline Genetic Algorithm:** Outlines standard algorithm used in this study, detailing the implemented methods and any modifications applied.
- **Strategies:** Describes the crossover and initialization strategies evaluated in the experiments, which are compared against the baseline and each other.
- **Experimental Setup and Results:** Presents the experimental design, along with visualized results and performance data.
- **Discussion:** Interprets the results, providing reasoning, comparisons to the baseline, and analysis of observed trends.
- **Responsible Research:** Addresses the reproducibility of the research and discusses its ethical considerations.
- **Conclusion:** Summarizes the key findings and offers recommendations for future research.

2 Baseline Genetic Algorithm

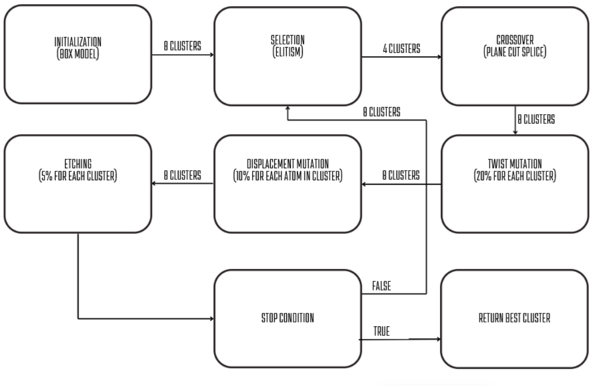


Figure 2: illustration of Baseline Genetic algorithm.

The baseline genetic algorithm was taken from previous projects, with minor modifications to improve performance and simplify the integration of experimental changes. As previously mentioned, genetic algorithms are population-based metaheuristic methods inspired by Darwinian natural selection, where the principle of survival of the fittest guides the evolution process.

Before starting the algorithm, initial population has to be generated. Throughout the entire project, the population size remained constant at 8. Default method for generating population is the box model, where x, y, z coordinates are uniformly generated from 0 to box length, where box length gets larger as cluster size increases. In each iteration, the algorithm generates a new population based on the fittest individuals from the previous generation. The iteration begins with parent selection for crossover. In this setup, elitism is used [19]: four parents with the highest fitness values are selected (top 50%). From these, eight valid children are generated through crossover. To check if generated children are valid, algorithm checks if all atoms are at least 0.15 Å from each other. Since the crossover function requires two parents to produce one child, it is called eight times per iteration. Parents are randomly chosen from the four selected individuals.

The plane-cut splice crossover method is used in the baseline algorithm [1]. Once the eight offspring are created, each has a chance to undergo mutation. Three types of mutations are applied, each with distinct probabilities:

1. **Twist Mutation**: A random plane is generated to divide a cluster into two parts. One part is then rotated around the normal of this plane. This mutation is applied with a 20% probability per cluster [13] [5].
2. **Displacement Mutation**: Each atom in a cluster has a 10% chance of receiving a small random displacement vector [13].
3. **Etching Mutation**: This mutation has two variants [17]:

- **Etching Addition**: A random atom is temporarily added to the cluster, followed by a local optimization step. The added atom is then removed.
- **Etching Subtraction**: A random atom is removed first, followed by local optimization, and then the atom is reinserted.

Each variant has a 2.5% probability, resulting in a combined 5% chance for etching mutation.

Because mutation types are applied on the basis of independent probabilities, a single cluster may be subjected to multiple mutations in a single iteration.

After the mutation phase, the algorithm checks for stopping conditions. In the default implementation: the algorithm terminated if either the iteration count reached 100, or if the best cluster had not changed over the last 10 iterations (i.e., convergence). However, this stopping criterion was not ideal for benchmarking, as reaching it did not guarantee that the global minimum had been found—it was possible for the algorithm to remain trapped in a local minimum for 10 iterations.

To address this, a new stopping condition was implemented. The algorithm now compares the fitness value of the current best cluster to the known global minimum obtained from a reference database [16] from Cambridge. If the fitness value is within a defined margin (10^{-6} eV) of the global minimum, the algorithm stops; otherwise, it proceeds to the next iteration. This margin was chosen to be as small as possible and to be able to counteract the floating point error.

3 Strategies

In the experiment 6 different crossover methods and 3 different initialization strategies were used. All of the methods have been previously documented and used in different articles. For all of the methods, documentation was provided, but not a implementation, so it is likely that methods used in this experiment are slightly different. Furthermore, all crossover strategies return only one child. Although some crossovers are capable of producing two children, only one is randomly selected and returned.

3.1 Crossovers

- **One Point Crossover**

  One-point crossover[2] between two parents is performed by selecting a random breakpoint—between the 2nd and 3rd elements in the figure (see Figure 3). The segments before and after the breakpoint are exchanged between the parents to produce two offspring: the first segment from Parent 1 is combined with the second segment from Parent 2, and vice versa. In the experiment, the atom sequence is randomized and scrambled if a valid child cannot be generated. Sorting the parent atoms was deliberately avoided, as it would resemble the plane-cut splice crossover.

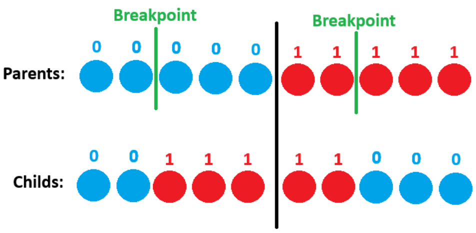


Figure 3: illustration of One Point Crossover

- **Two Point Crossover**

  Two-point crossover [5], similar to one-point crossover, uses two breakpoints instead of one, as described. This results in each parent being divided into three segments. These segments are then recombined to produce new offspring. As with the one-point crossover, the atom sequence is randomized.

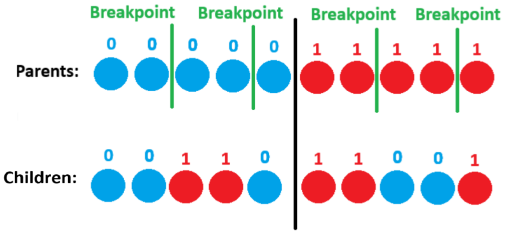


Figure 4: illustration of Two Point Crossover

- **Uniform Crossover**

  In uniform crossover[5] [13], the atom sequences for both parents are randomly generated, as in one-point and two-point crossover. A single child is produced by comparing corresponding elements from each parent and randomly selecting one of them for each position. Experiment employed a 70/30 bias in favor of the higher fitness parent [13].

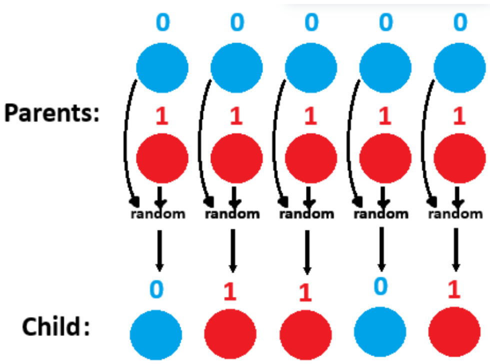


Figure 5: illustration of Uniform Crossover

- **Sphere cut splice Crossover**

In the Sphere Cut Splice Crossover [13] method, atomic positions actually matter. Each parent structure is divided into two regions: an inner sphere and an outer sphere. A child is generated by combining the inner sphere of one parent with the outer sphere of the other. The inner sphere's radius is selected randomly, while the outer sphere's radius is adjusted to ensure all atoms are accounted for. This crossover is repeated until valid offspring with the correct number of atoms are produced.

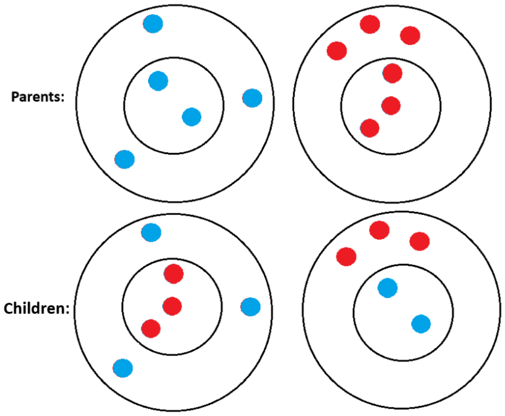


Figure 6: illustration of Sphere cut splice Crossover. Circles were used instead of spheres to better illustrate the process.

- **Arithmetical Crossover**

In arithmetical crossover [13], atoms are arranged in a random sequence, similar to one-point, two-point, and uniform crossover methods. A single child is produced by averaging the positions of the corresponding n-th atoms from both parents. This averaging process is repeated until a valid child cluster is obtained.

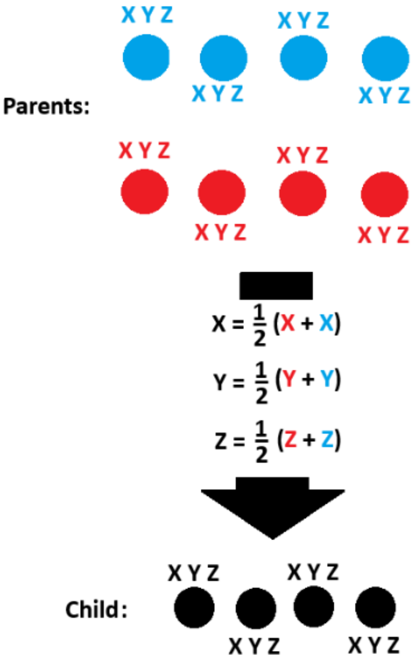


Figure 7: illustration of Arithmetical Crossover.

- **Plane Cut Splice Crossover**
  Similar to the Sphere Cut Splice Crossover , the Plane Cut Splice Crossover [13] also relies on the spatial positions of atoms. However, instead of using spheres, this method uses an infinite plane to divide each parent into two regions. A randomly generated plane slices through space, partitioning both parents. A child is then created by combining one region from each parent—one from each side of the plane. This crossover process is repeated until valid clusters with the correct number of atoms are produced.

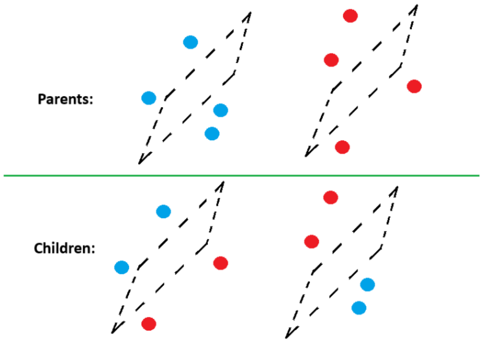


Figure 8: illustration of Plane Cut Splice Crossover.

3.2 Initialization

- **Box initialization**
  In the *box initialization* method, the x, y, and z coordinates of atoms are generated uniformly within

a cube ranging from 0 to the box length. The box length is defined as:

$$\text{box\_length} = 2\left(\frac{1}{2} + \left(\frac{3 \times \text{num\_atoms}}{4\pi\sqrt{2}}\right)^{1/3}\right) \tag{2}$$

This Equation (2) ensures that the box size increases with the number of atoms, preventing spatial biases that could otherwise arise in larger clusters.

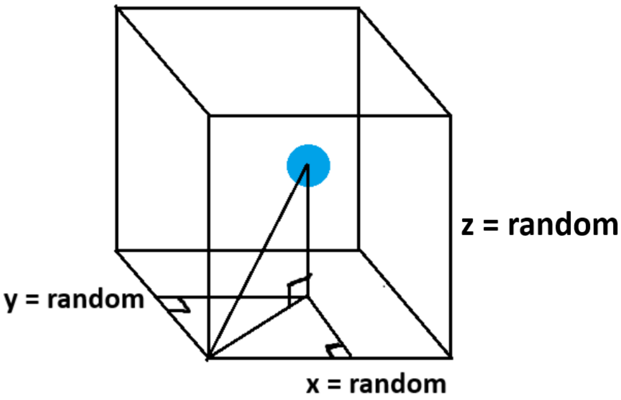


Figure 9: illustration of Box initialization.

- **Sphere initialization**

  Rather than generating atoms uniformly within a box, the *sphere initialization* method places atoms uniformly within a sphere of radius R. The radius is defined as:

$$R = r_e\left(\frac{1}{2} + \left(\frac{3 \times \text{num\_atoms}}{4\pi\sqrt{2}}\right)^{1/3}\right) \tag{3}$$

[13] This Equation (3) ensures that the spatial distribution scales larger with the cluster size.

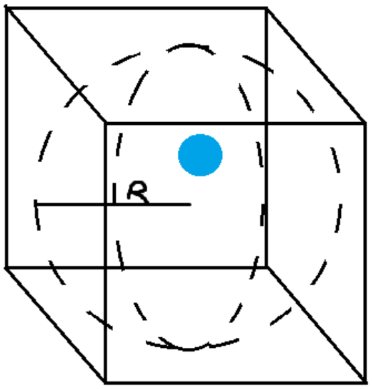


Figure 10: illustration of Sphere initialization.

- **Octahedron initialization**

  Octahedron initialization [18] differs from the box and sphere models by using fixed, patterned positions instead of fully random atom placements. However, using fixed positions can lead to identical clusters across the initial population. To address this, the method generates all possible positions for a higher-degree octahedron (example of different degree octahedrons in figure 11) and then randomly selects a subset of these positions to place the atoms. This approach introduces controlled randomness while maintaining the geometric structure.

  As shown in Figure 11, the structure consists of multiple layers containing varying numbers of atoms arranged in a rectangular layout. Each layer is separated by a distance of 0.5 Å (0.05 nm), and within each layer, atoms are positioned to form small rectangles with side lengths of 0.5 Å. All layers are centered along the same two axes. The maximum extent along each axis is calculated and used to define a simulation box of corresponding dimensions.

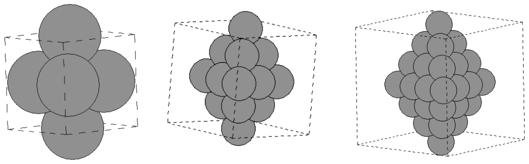


Figure 11: Rendered different degree octahedron positions.

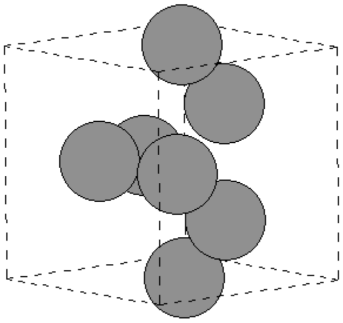


Figure 12: Final positions for Lennard-Jones cluster 7. Different 12 positions were randomly removed.

4 Experimental Setup and Results

In this experiment, multiple variants of the genetic algorithm were evaluated, focusing on crossover and initialization strategies. The goal was to compare their performance, including potential runtime improvements. This constitutes a comparative performance evaluation, structured as two separate experiments: one testing different crossover methods and the other testing various initialization strategies, both using the baseline genetic algorithm as a mold to put in the corresponding strategy.

The algorithm and all the code were created in Python using the Atomic Simulation Environment [8] library, which provides a structured implementation of atoms and the Lennard-Jones potential. All simulations were performed on a desktop computer (see Appendix A for hardware specifications).

In the experiment, six different crossover strategies and three different initialization strategies were evaluated.

For the first part of the experiment, the baseline setup was used. However, instead of relying solely on the plane cut splice crossover method, five additional crossover strategies were introduced. Cluster sizes ranging from 10 to 31 were tested, with each size run 10 times for every crossover strategy. These runs were computationally manageable, which made it feasible to test all strategies extensively at smaller sizes. Also cluster size 31 have multiple deep local wells, which might increase run time for some configurations of the algorithm.

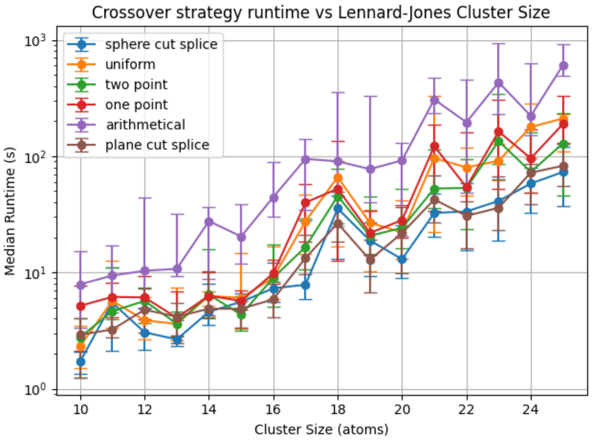
Beyond size 31, the computation time increased significantly, making it impractical to run each cluster 10 times for every crossover strategy up to size 40. Therefore, a selected subset of larger cluster sizes was tested instead: 33, 35, 36, 40. Due to time constraints, larger Lennard-Jones clusters were excluded, and most of sizes between 31 and 40 were skipped to conserve resources.

In the second part of the experiment, three initialization strategies were tested: Box model initialization, Sphere model initialization, Octahedron initialization. Since there were fewer initialization strategies than crossover strategies and baseline crossover method (plane cut splice) is not the worst, the computational cost was lower. Consequently, the same set of cluster sizes used in the crossover strategy tests was applied here as well and also larger cluster sizes: 43, 48, 51, 55.

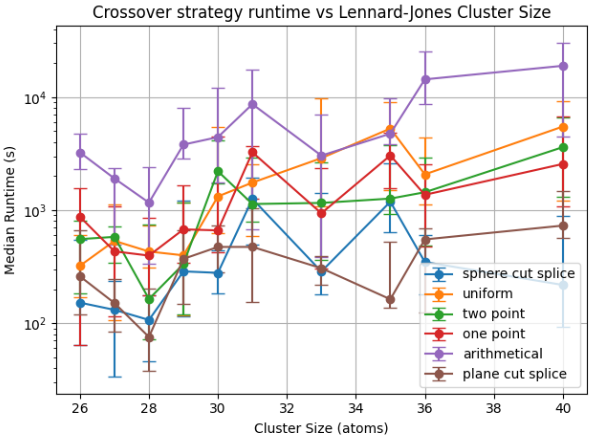
To ensure reproducibility and fairness, the np.random.seed(29 + run) function from the NumPy library [3] was used. Since all sources of randomness in the genetic algorithm—including crossover and initialization strategies—originate from NumPy, setting the seed based on the run index (e.g., np.random.seed(30) for run 1, np.random.seed(31) for run 2) ensures consistent randomness within each run. This means that running the genetic algorithm with the plane cut splice crossover on run 1 will always yield the same output. It also guarantees that the initialization strategy produces the same initial population across all crossover methods within the same run.

In addition, simple parallelization was used: up to 4 instances of genetic algorithm worked on different strategies and clusters. Each instance reserved 10-15% of CPU resources and a few MB of RAM. Further increasing number of instances might affect run times of other instances.

The results of running different crossovers are chaotic and the order of the best crossovers change often based on the cluster size, but it can be observed that arithmetical often is above all other crossovers in Figure 13.



(a) Lennard-Jones Clusters 10–25



(b) Lennard-Jones Clusters 26–40

Figure 13: Run time for different crossover strategies. Cluster size is on x-axis. Log run time of the algorithm is on y-axis. Dots are medians, upper whisker excludes top 20% worst runs, lower whisker excludes top 20% best runs.

To eliminate any biases, statistical tests were also used. All pairwise combinations of crossovers and their distributions were compared and only crossover pairs with p value below 0.05 were considered as it means that there is a significant difference between these two crossover distributions (for example, for cluster sizes 20, 21 in Table 1). Among these results even in the full table, the arithmetical crossover was always one of the pairs and the mean difference was always negative, indicating that the mean run time for the arithmetical crossover was always higher and other crossovers did not have significant difference between each other.

For initialization strategies, all run times seem to be close together, except for octahedron initialization for cluster sizes from 10 to 15, where it seems to be faster than the competition.

Cluster	Method	p	ΔMean (s)	95% CI (s)
20	Arith. vs PCS	0.0146	-75.44	(-140.67, -10.22)
	Arith. vs SCS	0.0064	-81.74	(-146.96, -16.51)
	Arith. vs TP	0.0399	-67.19	(-132.42, -1.97)
	Arith. vs UN	0.0332	-68.76	(-133.98, -3.53)
21	Arith. vs OP	0.0028	-230.49	(-401.90, -59.07)
	Arith. vs PCS	0.0000	-307.68	(-479.10, -136.27)
	Arith. vs SCS	0.0000	-327.71	(-499.13, -156.30)
	Arith. vs TP	0.0002	-276.53	(-447.94, -105.12)
	Arith. vs UN	0.0345	-179.86	(-351.27, -8.45)

Table 1: All pairwise comparisons between initialization strategies for cluster sizes 20, 21, where p value was below 0,05. Arith. = Arithmetical, PCS = Plane Cut Splice, SCS = Sphere Cut Splice, TP = Two Point, UN = Uniform, OP = One Point. For the complete table of cluster sizes 10–40, refer to Appendix C.

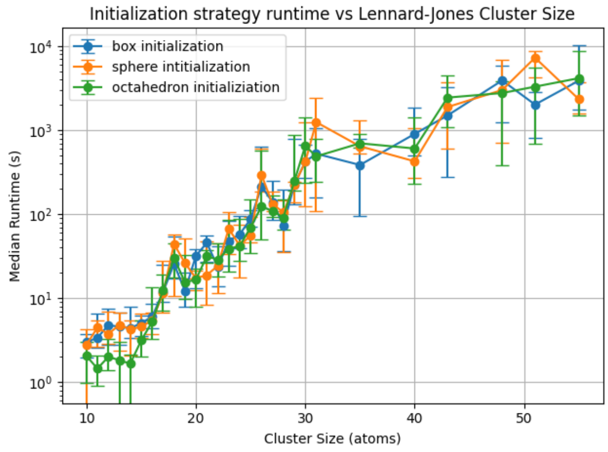


Figure 14: Run time for all of the initialization strategies. Cluster size is on x-axis. Log run time of the algorithm is on y-axis. Dots are medians, upper whisker excludes top 20% worst runs, lower whisker excludes top 20% best runs.

Statistical tests were also performed for initialization strategies to remove bias. The same configuration, which was mentioned for crossover strategies, was also used for initialization strategies. In Table 2, the octahedron initialization seems to be better than the sphere initialization for clusters 12, 13 and appears to be better than the box initialization for clusters 13, 14. However, not all pairs that include octahedron initialization with cluster sizes from 10 to 15 have a p-value below 0,05 threshold.

Also in Table 2 it can be observed that sphere initialization is the worst for cluster size 51 and performs worse than box initialization for cluster size 19.

5 Discussion

For smaller clusters (10-15), the octahedron initialization seems to be a winner, but in statistical tests some pairwise comparisons between the octahedron initialization and other initializations had a p value above 0,05. In addition, when the cluster size increases, run times get closer together.

Cluster	Method	p	ΔMean (s)	95% CI (s)
12	Octa. vs Sphere	0.0350	3.8935	(0.23, 7.56)
13	Box vs Octa.	0.0078	-3.4237	(-6.06, -0.79)
	Octa. vs Sphere	0.0214	3.0140	(0.38, 5.65)
14	Box vs Octa.	0.0002	-5.6762	(-8.81, -2.54)
19	Box vs Sphere	0.0173	18.5503	(2.95, 34.15)
51	Box vs Sphere	0.0011	4322.1139	(1669.32, 6974.91)
	Octa. vs Sphere	0.0231	3023.1741	(370.38, 5675.97)

Table 2: All pairwise (clusters 10-55) comparisons between initialization strategies for all cluster sizes, where p value was below 0,05. Arith. = Arithmetical, PCS = Plane Cut Splice, SCS = Sphere Cut Splice, TP = Two Point, UN = Uniform, OP = One Point. For the complete table of cluster sizes 10–40, refer to Appendix C.

This can be attributed to the initialization strategy, which generates an initial population that often falls within the global minimum well. To assess this, a method was developed to check whether the best cluster in the first iteration was within 0.1 Å (0.01 nm) of the true global minimum (see Figure 15).

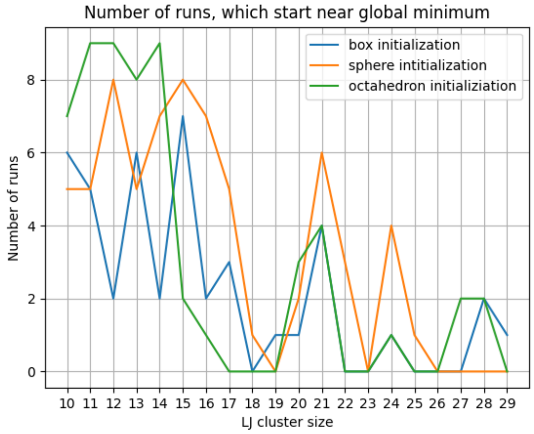


Figure 15: Graph of number of runs near global minimum.

From 15 it can be seen that octahedron almost always starts in the global minimum well for cluster sizes 10-14 and drastically decreases after size 15, which could explain fast run times for all the cluster sizes below 15 for octahedron initialization.

Since initialization is performed only once per execution, it is reasonable that the runtime differences between initialization strategies for larger clusters are minimal, as all other algorithmic steps are executed over a thousand times. Additionally, for smaller clusters, there is a higher likelihood of generating an initial population within the global minimum well, making it plausible that some initialization strategies can exploit this advantage more effectively than others.

From statistical tests (Table 1, Appendix C) and from the plot (Figures 13), the arithmetical crossover was the worst crossover strategy, as it was a pair in all pairwise comparisons with a p value below 0,05 and with a mean value greater than the other pair.

The worst performing crossover is easy to identify, but determining the best one is less straightforward, as the remaining five crossovers show no significant differences in their distributions. The uniform, one-point, and two-point crossovers use a shuffle method to randomize the atom sequence in memory when valid child clusters cannot be generated with the current atom sequence, which could explain similarities between them. However, it is harder to explain non significant distributions between uniform, one-point, two-point crossovers, which manipulates atom sequence to create new children, and plane cut splice, sphere cut splice crossovers, which manipulates atom cartesian positions to create new children. Exploring other variables (such as population size, selection strategies, and mutation methods) could lead to different results, as it is unlikely that the current configuration represents the optimal setup of the genetic algorithm for solving the Lennard-Jones cluster optimization problem.

Furthermore, despite 5 weeks of continuous testing of crossover strategies, it was unfortunately not possible to complete all 10 runs for each crossover on every cluster. Specifically, Cluster 33 has 9 runs per crossover, while Clusters 35 and 40 each have 8 runs per crossover. This may impact the results by increasing variance. On the positive side, all runs for the initialization strategies were successfully completed.

It is also important to acknowledge the limitations of the p-value, which played a key role in analyzing the results for both crossover and initialization strategies. The p-value is susceptible to false positives, and over-reliance on it can promote questionable research practices. Moreover, even trivial effects can appear statistically significant given a sufficiently large sample size [6].

6 Conclusion

Based on the experimental results, arithmetical crossover performs the worst among the tested crossover methods for solving the global optimization problem of Lennard-Jones clusters. However, it remains unclear which of the remaining crossovers (Sphere Cut Splice, Plane Cut Splice, Uniform, One-Point, or Two-Point) is the most effective.

For Lennard-Jones clusters of sizes 10 to 14, the octahedron initialization strategy proves to be the most effective, as it has a higher probability of generating initial populations near the global minimum. For clusters larger than 15 atoms, no single initialization strategy clearly outperforms the others.

To enhance the baseline genetic algorithm in future work, several improvements can be considered: introducing multi-threading to enable simulations of larger clusters, ensuring all cluster sizes are included without omissions, increasing the number of runs per configuration, running the algorithm on more powerful hardware, and take into account other variables (such as population size and mutation, selection strategy).

7 Responsible Research

Significant consideration was taken to ensure that the research conclusions remain as unbiased as possible. To avoid cherry-picking the results, no data was excluded from the dataset throughout the study. Due to inherent randomness in the algorithm and its initialization, it was essential to mitigate runtime variance. Therefore, each cluster size and algorithm configuration were run at least 10 times, or more, when there was extra time.

Additionally, factors related to the system, such as background processes and hardware load, can introduce additional variability. To reduce this impact, efforts were made to standardize the system environment across runs, even when the runs occurred on different days and on parallel runs. Increasing the number of repetitions also helped counteract this variability and ensure more reliable results.

During the research process, large language models (LLMs) were used as tools to assist with various tasks, such as improving code snippets, interpreting error messages, checking grammar, generating latex syntax for plots and images, modifying Overleaf templates, generating bibtex references and providing a second opinion on whether certain text made sense. Overleaf built in grammatical checker and ChatGPT was used for this. However, their suggestions were not followed blindly. For python code, for example, it is much slower to iterate through all atoms using python ”for” loops, but with help of numpy and other libraries, the code was optimized to improve performance. To make sure the method functions the same as before, it was tested to ensure it produced same output as before. Generating ideas, analysis was avoided as there was enough information from found research articles, provided templates, information from brightspace platform and ideas for analysis was discussed in the meetings and with other students. All of the text was written by one person. In writing process LLM was used to refine individual sentences for clarity or fluency, and grammar suggestions in Overleaf were reviewed and selectively accepted or ignored during the writing process. See Appendix B for prompts.

References

[1] D. M. Deaven and K. M. Ho. Molecular geometry optimization with a genetic algorithm. *Physical Review Letters*, 75(2):288–291, 1995.

[2] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

[3] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy,

Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[4] Bernd Hartke. Global geometry optimization of clusters using genetic algorithms. *The Journal of Physical Chemistry*, 97(39):9973–9976, 1993. Received: May 11, 1993.

[5] Roy L. Johnston. Evolving better nanoparticles: Genetic algorithms for optimising cluster geometries. *Dalton Transactions*, pages 4193–4207, 2003. Received 21st May 2003, Accepted 15th August 2003, First published online 1st September 2003.

[6] Samuel C. Karpen. P value problems. *American Journal of Pharmaceutical Education*, 81(9):6570, 2017.

[7] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80:8091–8126, 2021. Received: 27 July 2020 / Revised: 12 October 2020 / Accepted: 23 October 2020.

[8] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, Eric D Hermes, Paul C Jennings, Peter Bjerre Jensen, James Kermode, John R Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W Jacobsen. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017.

[9] LibreTexts. Lennard-jones potential. Accessed: 2025-05-22.

[10] A. P. Mouritz, E. Gellert, P. Burchill, and K. Challis. Review of advanced composite structures for naval ships and submarines. *Composite Structures*, 53(1):21–41, 2001.

[11] Prachi Patel. Materials for space exploration take a giant leap. *ACS Central Science*, 9(4):582–585, 2023.

[12] Peter Schwerdtfeger and David J. Wales. 100 years of the lennard-jones potential. *Journal of Chemical Theory and Computation*, 20(9):3379–3405, 2024. Published April 26, 2024.

[13] Frederico T. Silva, Mateus X. Silva, and Jadson C. Belchior. A new genetic algorithm approach applied to atomic and molecular cluster studies. *Frontiers in Chemistry*, 7:707, 2019. Published: 05 November 2019.

[14] Adrian P. Sutton. *Concepts of Materials Science*. Oxford University Press, Oxford, United Kingdom, 1st edition, 2021.

[15] Odile Patrick Thalia. Advanced materials for space exploration. *Research Output Journal of Biological and Applied Science*, 3(1):18–22, 2024.

[16] D. J. Wales, J. P. K. Doye, A. Dullweber, M. P. Hodges, F. Y. Naumkin, F. Calvo, J. Hernández-Rojas, and T. F. Middleton. The cambridge cluster database. http://www-wales.ch.cam.ac.uk/CCD.html, n.d. Accessed: 2025-06-02.

[17] Matthew D. Wolf and Uzi Landman. Genetic algorithms for structural cluster optimization. *The Journal of Physical Chemistry A*, 102(30):6129–6137, July 1998. Received: March 11, 1998; In final form: May 1, 1998.

[18] Yuhong Xiang, Haiyan Jiang, Wensheng Cai, and Xueguang Shao. An efficient method based on lattice construction and the genetic algorithm for optimization of large lennard-jones clusters. *The Journal of Physical Chemistry A*, 108(14):3586–3592, April 2004. Received: December 10, 2003; In final form: February 11, 2004.

[19] Eckart Zitzler, Kalyanmoy Deb, Carlos Artemio Coello Coello, David Corne, and Lothar Thiele, editors. *Evolutionary Multi-Criterion Optimization: First International Conference, EMO 2001, Zurich, Switzerland, March 7–9, 2001, Proceedings*, volume 1993 of *LNCS*, Berlin, Heidelberg, March 2001. Springer. CR Subject Classification (1998): F.2, G.1.6, G.1.2, I.2.8.

A Computer specifications

- **CPU:** AMD Ryzen 5 5600 6-Core Processor
- **Memory:** Corsair DDR4 3200MHz OC, 16GB
- **GPU:** AMD Radeon RX 6700 XT
- **Motherboard:** MSI B350M Gaming Pro
- **Operating System:** Windows 11
- **Storage:** Western Digital Blue 7200RPM 1TB HDD

B Questions and answers from chatGPT

This section includes all prompts, which were used in some process in the research.

- **Question:** Can you rewrite this sentence:
  "To make sure the method functions the same as before, it was tested if it produces the same output."

  Answer: "The method was tested to ensure it produces the same output as before."
- **Question:** Can you itemize this in LaTeX?
  Answer:

```
\begin{itemize}
    \item \textbf{CPU:}
    AMD Ryzen 5 5600 6-Core Processor
    \item \textbf{Memory:}
    Corsair DDR4 3200MHz, 16GB
    \item \textbf{GPU:}
    AMD Radeon RX 6700 XT
    \item \textbf{Motherboard:}
    MSI B350M Gaming Pro
    \item \textbf{Operating System:}
    Windows 11
    \item \textbf{Storage:}
    Western Digital Blue 7200RPM
    1TB HDD
\end{itemize}
```

- **Question:** how to make overwrite if the file exists

  ```
  def save_last_pop(self):
      path_base = f"./data/ga/
      LJ{self.num_atoms}/pop"
      os.makedirs(path_base, exist_ok=True)

      data = {}
      for i, cluster in enumerate(self.cur_clusters):
          data[f"pop{i}"] = cluster.positions
      with open(path_base, "w") as f:
          json.dump(data, f, indent=4)
  ```

  Answer:

  ```
  file_path = os.path.join(path_base,
  "last_pop.json")
  ```

- **Question:** can you rewrite this sentence ”But not all pairs, which include octahedron initialization, from cluster size 10-15 have p value below 0,05.”

  Answer: However, not all pairs that include octahedron initialization with cluster sizes from 10 to 15 have a p-value below 0.05

- **Question:** what is the difference between results and discussion sections in a research paper

  Answer: Results: To present the data and findings of your research objectively, without interpretation. Discussion: To interpret and analyze the meaning and implications of the results in the context of your research question and the broader field.

- **Question:** can you rewrite this ”From these tests only pairs with p value below 0.05 were the ones, which include arithmetical crossover and mean diff was always negative, which means arithmetical crossover mean run time was always higher.”

  Answer: ”Among these tests, only the pairs with p-values below 0.05 involved arithmetical crossover, and the mean difference was consistently negative, indicating that the mean run time for arithmetical crossover was always higher.”

- **Question:** does reject == True indicate that the distributions are not the same using tukey

  Answer: Yes, exactly. When reject == True in the Tukey HSD test results, it means: There is a statistically significant difference between the two group means at the specified significance level (usually alpha = 0.05).

- **Question:** ”np.where(mask[:, None], pos1, pos2)” what does [:, None] mean

  Answer: The expression [:, None] (or equivalently [:, np.newaxis]) in NumPy is used to add a new axis to an array, effectively reshaping it.

- **Question:** can you itemize this in overleaf ”The Baseline Genetic Algorithm section outlines a standard algorithm used in this study, detailing the methods implemented and any modifications applied.

  The Strategies section describes the crossover and initialization strategies evaluated in the experiments, which are compared against the baseline and each other.

  In the Experimental Setup and Results section, the experimental design is presented, along with visualized results and performance data.

  The Discussion section interprets the results, providing reasoning, comparisons to the baseline, and analysis of observed trends.

  The Responsible Research section addresses the reproducibility of the research and discusses its ethical considerations.

  Finally, the Conclusion summarizes the key findings and offers recommendations for future research.”

  Answer:

  ```
  \begin{itemize}
      \item \textbf{Baseline Genetic Algorithm:}
      Outlines the standard algorithm used
      in this study, detailing the
      implemented methods and any
      modifications applied.

      \item \textbf{Strategies:}
      Describes the crossover and
      initialization strategies
      evaluated in the experiments,
      which are compared against the
      baseline and each other.

      \item \textbf{Experimental Setup
      and Results:} Presents the
      experimental design, along
      with visualized results
      and performance data.

      \item \textbf{Discussion:}
      Interprets the results,
      providing reasoning, comparisons
      to the baseline, and analysis
      of observed trends.

      \item \textbf{Responsible Research:}
  ```

```
    Addresses the reproducibility of the
    research and discusses its ethical
    considerations.

    \item \textbf{Conclusion:}
    Summarizes the key findings
    and offers recommendations
    for future research.
\end{itemize}
```

- **Question:** how to rewrite this sentence ”Algorithm does not choose only one mutation per cluster, all of the mutations can happen on a single cluster if probability generation deems it .”

  Answer: ”The algorithm does not limit each cluster to a single mutation; multiple mutations can be applied to the same cluster if determined by the probability generation.”

- **Question:** can you rewrite this sentence ”However, this stopping criterion was not ideal for benchmarking purposes, as it did not mean it had found the global minimum, because it was possible to get stuck in a local minimum for 10 iterations.”

  Answer: ”However, this stopping criterion was not ideal for benchmarking, as reaching it did not guarantee that the global minimum had been found—it was possible for the algorithm to remain trapped in a local minimum for 10 iterations.”

- **Question:** how to make figures on top of the next page

  Answer: To force a figure to appear at the top of the next page in LaTeX, you'll want to:

  Use the clearpage or newpage command before the figure to ensure it moves to a new page.

  Use the [t] placement specifier (top of page).

  Avoid [H] (from the float package)

- **Question:** i get this error ”Process finished with exit code -1073741819 (0xC0000005)”

  Answer: The error code -1073741819 (0xC0000005) is a Windows access violation, meaning your Python process tried to access restricted memory and crashed. This is not a typical Python error — it usually suggests:

  A C-extension module (like matplotlib, numpy, etc.) crashed.

  A corrupted .json file or data issue triggered a crash in a compiled library.

  Running out of memory or corrupted Python/IDE environment.

- **Question:** how to make y axis log scale

  ```
   plt.errorbar(x_axis, y_axis,
   yerr=stds, fmt='-o', capsize=5, label=name)

  # Labels and title
  plt.xlabel('Cluster Size (atoms)')
  plt.ylabel('Average Runtime (s)')
  plt.title('GA Runtime vs Lennard-Jones
  Cluster Size')
  plt.grid(True)
  plt.legend()
  plt.tight_layout()
  plt.show()"
  ```

  Answer: To make the Y-axis logarithmic in your Matplotlib plot, you can simply add:

  ```
  plt.yscale('log')
  ```

- **Question:** can you create overleaf cite for this ”Journal of Physics Condensed MatterVolume 29, Issue 277 June 2017 Article number 273002” ”ISSN 09538984 DOI 10.1088/1361-648X/aa680e” ”The atomic simulation environment—a Python library for working with atoms To cite this article: Ask Hjorth Larsen et al 2017 J. Phys.: Condens. Matter 29 273002”

  Answer:

  ```
  @article{larsen2017ase,
    author      = {Hjorth Larsen},
    title       = {The atomic simulation environment|a
    Python library for working with atoms},
    journal     = {Journal of Physics: Condensed Matter},
    volume      = {29},
    number      = {27},
    year        = {2017},
    pages       = {273002},
    doi         = {10.1088/1361-648X/aa680e},
    issn        = {0953-8984}
  }
  ```

- **Question:** how to generate num atoms amount of positions given that positions has more than num atoms of coordinates

  Answer:

  ```
       indices = np.random.choice(
       len(all_positions), size=num_atoms,
       replace=False)
  ```

- **Question:** can you create overleaf reference for this ”Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)Volume 1993, Pages 67 - 812001 1st International Conference on Evolutionary Multi-Criterion Optimization, EMO 2001Zurich7 March 2001through 9 March 2001Code 118929”

  Answer:

  ```
       @inproceedings{EMO2001,
    author      = {Unknown},  % Replace with
    actual author(s) if known
    title       = {Proceedings of the 1st
    International Conference on Evolutionary
    Multi-Criterion Optimization (EMO 2001)},
    booktitle   = {Lecture Notes in Computer
    Science},
  ```

```
  series       = {LNCS},
  volume       = {1993},
  pages        = {67--81},
  year         = {2001},
  publisher    = {Springer},
  address      = {Zurich, Switzerland},
  month        = mar,
  note         = {Includes subseries:
  Lecture Notes in Artificial Intelligence
  and Bioinformatics},
  isbn         = {978-3-540-41747-3},
  % Replace or remove if not needed
  doi          = {},  % Add DOI if available
  eventtitle   = {1st International
  Conference on Evolutionary
  Multi-Criterion Optimization (EMO 2001)},
  eventdate    = {2001-03-07/2001-03-09},
  eventlocation= {Zurich, Switzerland}
}
```

- **Question:** how to include another tex file in latex
  Answer:

  ```
  \input{chapter1}  % This includes
  chapter1.tex
  ```

- **Question:** Im benchmarking genetic algorithm on lennard jones clusters. I am running ga on different cluster size. I want to plot cluster size on x axis and on y axis average run time and I want to include variance.
  Answer:

  ```
  plt.xlabel('Cluster Size (atoms)')
  plt.ylabel('Average Runtime (s)')
  plt.title('GA Runtime vs Lennard-Jones
  Cluster Size')
  plt.grid(True)
  plt.legend()
  plt.tight_layout()
  plt.show()
  ```

- **Question:** how to append lists with 3d coordinates in python
  Answer:

  ```
  coords = []  # empty list
  point = [1.0, 2.0, 3.0]
  coords.append(point)
  ```

- **Question:**

  ```
  can you write this in latex "Rather
  than generating atoms uniformly
  in a box, sphere initialization
  generates atoms uniformly in a sphere
  with a radius R.
  R = re * (0.5 + ((3 * num_atoms) /
  (4 * np.pi * np.sqrt(2))) ** (1 / 3))"
  ```

  Answer:

  ```
  Rather than generating atoms uniformly
  within a box, the
  \textit{sphere initialization} method
  places atoms uniformly within a
  sphere of radius $R$. The radius
  is defined as:

  \[
  R = r_e \times \left(0.5 +
  \left(\frac{3 \times \text{num\_atoms}
  }{4 \pi \sqrt{2}} \right)^{1/3} \right)
  \]
  ```

- **Question:** create cite for this in overleaf ”Evolving better nanoparticles: Genetic algorithms for optimising cluster geometries Roy L. Johnston School of Chemistry, University of Birmingham, Edgbaston, Birmingham, UK B15 2TT Received 21st May 2003, Accepted 15th August 2003 First published as an Advance Article on the web 1st September 2003”
  Answer:

  ```
       @article{johnston2003evolving,
    title     = {Evolving better nanoparticles:
    Genetic algorithms for optimising
    cluster geometries},
    author    = {Roy L. Johnston},
    journal   = {Nanoscale},
    year      = {2003},
    note      = {Received 21st May 2003,
    Accepted 15th August 2003, First published
    online 1st September 2003},
    institution = {School of Chemistry,
    University of Birmingham, Edgbaston,
    Birmingham, UK B15 2TT}
  }
  ```

- **Question:** how to make sure the figure is with the text and not on other page
  Answer: Use [H] from the float package
  The standard [h] placement is a suggestion, so LaTeX might still move your figure elsewhere to optimize layout. The [H] placement from the float package forces the figure exactly at that location.
- **Question:** how to cite this website in overleaf

  ```
  "https://chem.libretexts.org/Bookshelves/
      Physical_and_Theoretical_
      Chemistry_Textbook_Maps/
      Supplemental_Modules_(Physical_and_
      Theoretical_Chemistry)
      /Physical_Properties_of_Matter
      /Atomic_and_Molecular_Properties
      /Intermolecular_Forces/
      Specific_Interactions/
      Lennard-Jones_Potential"
  ```

  Answer:

  ```
  @online{libretextsLJ,
    author       = {{LibreTexts}},
  ```

```
  title        = {Lennard-Jones Potential},
  year         = {2023},
  url          = {https://chem.libretexts.org
  /Bookshelves/

  Physical_and_Theoretical_
  Chemistry_Textbook_Maps/

  Supplemental_Modules_
  (Physical_and_Theoretical_Chemistry)
  /Physical_Properties_of_Matter/
  Atomic_and_Molecular_Properties/
  Intermolecular_Forces/
  Specific_Interactions/
  Lennard-Jones_Potential},
  note         = {Accessed: 2025-05-22}
}
```

- **Question:** how to write what each value is in latex

```
    "\[
V(r) = 4\varepsilon \left[ \left(
\frac{\sigma}{r} \right)^{12}
- \left( \frac{\sigma}{r} \right)^6 \right]
\]"
```

Answer:

```
\[
V(r) = 4\varepsilon \left[
\left( \frac{\sigma}{r} \right)^{12}
- \left( \frac{\sigma}{r} \right)^6 \right]
\]

\begin{align*}
V(r) & : \text{Potential energy as
a function of distance } r \\
\varepsilon & : \text{Depth of
the potential well} \\
\sigma & : \text{Finite distance
at which the inter-particle potential is zero} \\
r & : \text{Distance between
the particles}
\end{align*}
```

- **Question:** python how to check for nan

Answer:

```
x = float('nan')
print(math.isnan(x))
```

- **Question:** how to make the scatter plot and x axis as integers

```
  "

    plt.plot(x_axis, y_axis, label=name)
plt.xlabel("Time (s)")
plt.ylabel("Potential Energy")
plt.title("Comparison of Energy
Convergence LJ" + str(lj))
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()"
```

Answer:

```
plt.scatter(x_axis, y_axis,
label=name)  # scatter instead of plot

# Optional: force x-ticks
to show as integers
plt.xticks(x_axis)

plt.xlabel("LJ cluster size (lj)")
plt.ylabel("Success Ratio
(found_min / total)")
plt.title("Comparison of
Success Ratio Across LJ Sizes")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

- **Question:** How to shuffle numpy 2d array for cluster1.positions

Answer:

```
a = np.array(cluster1.positions)
np.random.shuffle(a)
```

- **Question:** python how to speed up this function

```
"def uniform(cluster1, cluster2):
    align_cluster(cluster1)
    align_cluster(cluster2)

    pos1 = np.array(cluster1.positions)
    pos2 = np.array(cluster2.positions)

    while True:
        np.random.shuffle(pos1)
        np.random.shuffle(pos2)

        g = []
        for i in range(len(pos1)):
            if random.random() < 0.5:
                g.append(pos1[i])
            else:
                g.append(pos2[i])
        if configuration_validity(g):
            return np.array(g)"
```

Answer:

```
def uniform(cluster1, cluster2):
    align_cluster(cluster1)
    align_cluster(cluster2)

    pos1 = np.array(cluster1.positions)
    pos2 = np.array(cluster2.positions)

    size = len(pos1)

    while True:
        # Shuffle indices once, not full arrays
        idx1 = np.random.permutation(size)
        idx2 = np.random.permutation(size)

        # Vectorized 50/50 mask
        mask = np.random.rand(size) < 0.5

        # Vectorized choice based on mask
        g = np.where(mask[:, None],
        pos1[idx1], pos2[idx2])

        if configuration_validity(g):
            return g
```

- **Question:** how to fix this

```
"    def __init__(
        self,
        mutation=None,
        num_clusters=8,
        displacement_length=0.2,
        num_selection=0,
        preserve=True,
        local_optimizer=BFGS,
        calculator=LennardJones,
        crossover=(cluster1, cluster2,
        box_length, num_atoms) =>
        plane_cut_splice(cluster1,
        cluster2, box_length, num_atoms)
    )"
```

Answer:

```
def __init__(
    self,
    mutation=None,
    num_clusters=8,
    displacement_length=0.2,
    num_selection=0,
    preserve=True,
    local_optimizer=BFGS,
    calculator=LennardJones,
    crossover=lambda cluster1,
    cluster2, box_length, num_atoms:
    plane_cut_splice(cluster1,
    cluster2, box_length, num_atoms)
):
```

C Crossover comparison full table

Cluster	Method	p	ΔMean (s)	95% CI (s)
10	Arith. vs SCS	0.0226	-7.80	(-14.87, -0.73)
11	Arith. vs PCS	0.0481	-8.98	(-17.92, -0.05)
12	Arith. vs OP	0.0013	-18.55	(-31.71, -5.39)
	Arith. vs PCS	0.0004	-19.98	(-33.14, -6.81)
	Arith. vs SCS	0.0002	-21.06	(-34.23, -7.90)
	Arith. vs TP	0.0006	-19.56	(-32.72, -6.39)
	Arith. vs UN	0.0004	-20.18	(-33.34, -7.01)
13	Arith. vs OP	0.0000	-15.09	(-23.62, -6.56)
	Arith. vs PCS	0.0001	-14.63	(-23.16, -6.10)
	Arith. vs SCS	0.0000	-16.32	(-24.85, -7.79)
	Arith. vs TP	0.0000	-15.33	(-23.86, -6.80)
	Arith. vs UN	0.0000	-15.29	(-23.82, -6.76)
14	Arith. vs OP	0.0034	-16.49	(-28.95, -4.04)
	Arith. vs PCS	0.0029	-16.69	(-29.15, -4.24)
	Arith. vs SCS	0.0010	-18.11	(-30.57, -5.66)
	Arith. vs TP	0.0101	-14.95	(-27.40, -2.50)
	Arith. vs UN	0.0025	-16.90	(-29.36, -4.45)
15	Arith. vs OP	0.0001	-17.64	(-28.25, -7.03)
	Arith. vs PCS	0.0001	-17.71	(-28.32, -7.10)
	Arith. vs SCS	0.0001	-17.91	(-28.52, -7.29)
	Arith. vs TP	0.0000	-19.07	(-29.69, -8.46)
	Arith. vs UN	0.0009	-15.58	(-26.20, -4.97)
19	Arith. vs OP	0.0346	-175.08	(-342.02, -8.15)
	Arith. vs PCS	0.0214	-185.25	(-352.19, -18.31)
	Arith. vs SCS	0.0255	-181.58	(-348.52, -14.64)
	Arith. vs UN	0.0382	-172.92	(-339.85, -5.98)

Table 3: All pairwise crossover comparisons with p value below 0.05 for clusters 10-19. Arith. = Arithmetical, PCS = Plane Cut Splice, SCS = Sphere Cut Splice, TP = Two Point, UN = Uniform, OP = One Point.

Cluster	Method	p	ΔMean (s)	95% CI (s)
20	Arith. vs PCS	0.0146	-75.44	(-140.67, -10.22)
	Arith. vs SCS	0.0064	-81.74	(-146.96, -16.51)
	Arith. vs TP	0.0399	-67.19	(-132.42, -1.97)
	Arith. vs UN	0.0332	-68.76	(-133.98, -3.53)
21	Arith. vs OP	0.0028	-230.49	(-401.90, -59.07)
	Arith. vs PCS	0.0000	-307.68	(-479.10, -136.27)
	Arith. vs SCS	0.0000	-327.71	(-499.13, -156.30)
	Arith. vs TP	0.0002	-276.53	(-447.94, -105.12)
	Arith. vs UN	0.0345	-179.86	(-351.27, -8.45)
22	Arith. vs OP	0.0256	-154.69	(-296.98, -12.40)
	Arith. vs PCS	0.0006	-214.54	(-356.83, -72.25)
	Arith. vs SCS	0.0010	-206.84	(-349.13, -64.56)
	Arith. vs TP	0.0082	-174.23	(-316.52, -31.94)
	Arith. vs UN	0.0156	-163.38	(-305.67, -21.10)
24	Arith. vs OP	0.0069	-252.48	(-455.51, -49.45)
	Arith. vs PCS	0.0005	-309.53	(-512.56, -106.50)
	Arith. vs SCS	0.0003	-318.33	(-521.36, -115.30)
	Arith. vs TP	0.0021	-279.49	(-482.52, -76.46)
25	Arith. vs OP	0.0000	-494.97	(-768.69, -221.25)
	Arith. vs PCS	0.0000	-617.74	(-891.46, -344.02)
	Arith. vs SCS	0.0000	-601.61	(-875.33, -327.89)
	Arith. vs TP	0.0000	-552.41	(-826.13, -278.70)
	Arith. vs UN	0.0000	-512.38	(-786.09, -238.66)
26	Arith. vs OP	0.0000	-2671.56	(-4042.07, -1301.04)
	Arith. vs PCS	0.0000	-3163.70	(-4534.22, -1793.18)
	Arith. vs SCS	0.0000	-3116.92	(-4487.44, -1746.40)
	Arith. vs TP	0.0000	-2959.40	(-4329.92, -1588.89)
	Arith. vs UN	0.0000	-2906.73	(-4277.25, -1536.21)
27	Arith. vs OP	0.0316	-1202.73	(-2336.92, -68.55)
	Arith. vs PCS	0.0005	-1717.31	(-2851.50, -583.13)
	Arith. vs SCS	0.0008	-1677.65	(-2811.84, -543.47)
	Arith. vs TP	0.0149	-1309.06	(-2443.25, -174.88)
	Arith. vs UN	0.0340	-1192.07	(-2326.25, -57.88)
28	Arith. vs OP	0.0292	-1209.82	(-2339.73, -79.91)
	Arith. vs PCS	0.0018	-1569.35	(-2699.26, -439.44)
	Arith. vs SCS	0.0017	-1578.68	(-2708.58, -448.77)
	Arith. vs TP	0.0137	-1314.91	(-2444.82, -185.00)
29	Arith. vs OP	0.0012	-6011.28	(-10183.29, -1839.28)
	Arith. vs PCS	0.0004	-6504.31	(-10676.31, -2332.31)
	Arith. vs SCS	0.0004	-6464.67	(-10636.67, -2292.67)
	Arith. vs TP	0.0004	-6446.13	(-10618.13, -2274.13)
	Arith. vs UN	0.0006	-6295.06	(-10467.06, -2123.06)
30	Arith. vs OP	0.0212	-7631.49	(-14495.36, -767.62)
	Arith. vs PCS	0.0039	-8999.02	(-15862.89, -2135.15)
	Arith. vs SCS	0.0036	-9059.11	(-15922.98, -2195.24)
31	Arith. vs OP	0.0025	-6251.30	(-10870.76, -1631.84)
	Arith. vs PCS	0.0000	-8481.85	(-13101.31, -3862.38)
	Arith. vs SCS	0.0001	-7883.80	(-12503.26, -3264.33)
	Arith. vs TP	0.0004	-7159.16	(-11778.62, -2539.69)
	Arith. vs UN	0.0005	-7041.07	(-11660.54, -2421.61)
36	Arith. vs OP	0.0000	-19744.10	(-27500.35, -11987.86)
	Arith. vs PCS	0.0000	-21242.92	(-28999.17, -13486.68)
	Arith. vs SCS	0.0000	-21376.37	(-29132.61, -13620.12)
	Arith. vs TP	0.0000	-20183.73	(-27939.97, -12427.48)
	Arith. vs UN	0.0000	-18483.57	(-26239.81, -10727.33)
40	Arith. vs OP	0.0061	-15994.63	(-28618.50, -3370.76)
	Arith. vs PCS	0.0010	-19122.67	(-32127.24, -6118.09)
	Arith. vs SCS	0.0003	-19553.45	(-31873.09, -7233.82)
	Arith. vs TP	0.0081	-15186.48	(-27506.11, -2866.84)
	Arith. vs UN	0.0127	-14518.80	(-26838.44, -2199.17)

Table 4: All pairwise crossover comparisons with p value below 0.05 for clusters 20-40. Arith. = Arithmetical, PCS = Plane Cut Splice, SCS = Sphere Cut Splice, TP = Two Point, UN = Uniform, OP = One Point.