Analyzing The Impact of Mutations on Genetic Algorithms for Finding the Lowest Energy Structure of Atomic Clusters

A Benchmark Study of Mutation Operations on Lennard-Jones Clusters

Stefan Bud
Supervisor(s): Peter A.N. Bosman, Anton Bouter, Vanessa Volz
EEMCS, Delft University of Technology, 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: Stefan Bud
Final project course: CSE3000 Research Project
Thesis committee: Peter A.N. Bosman, Anton Bouter, Vanessa Volz, Thomas Abeel

Abstract

Finding the lowest-energy structure of a cluster of atoms is an NP-Hard problem with applications in materials science. Genetic Algorithms (GAs) have shown promise in solving this problem due to their ability to explore complex energy landscapes. A critical component of GAs are the mutations, which maintain diversity and helps avoid premature convergence. Despite the existence of various mutation operations for atomic clusters, there is a lack of benchmarking to compare their effectiveness. In this paper, we evaluate the performance of eight mutation operations within a GA framework for optimizing Lennard-Jones clusters. We assess each mutation based on its ability to reach the global minimum (accuracy) and the time required to do so (runtime). Experiments are conducted across population sizes of 8, 15, and 20, with multiple mutation probabilities tested. Results show that the Etching mutation consistently provides the highest accuracy, but at a considerable runtime cost. In contrast, Twist and Random Displacement offer fast convergence with moderate success rates.

1 Introduction

A cluster of atoms is a finite group of atoms bound together by physical or chemical forces. Finding the lowest-energy structure of a cluster of atoms is a proven NP-Hard problem [12] that has important implications in the field of materials science. An efficient way of solving this problem would help with discovering new materials with potential applications in areas such as biomedical imaging [7], semiconductor development [4], and aerospace engineering [8]. Numerous methods have been used to tackle this problem, such as Simulated Annealing [11] or Basin-Hopping algorithms [10], but the focus of this paper is on Genetic Algorithms.

One of the earliest adopters of Genetic Algorithms for solving this problem were Xiao and Williams [14]. Their algorithm used a binary encoding for clusters in order to store them as strings, allowing standard genetic operations to be applied to them. Two years later, Zeiri [15] released a Genetic Algorithm in which the atoms are represented by real-valued three-dimensional Cartesian coordinates instead of binary number strings. A significant advancement then came with the work of Deaven and Ho [1] by incorporating a local optimizer into this solution. Their algorithm has become a standard model and the foundation of numerous improvements over the years, including many different types of mutations. Mutations play a crucial role in the algorithm by preventing premature convergence to suboptimal solutions and, as a result of this, increasing the probability of finding the global minimum. To the best of our knowledge, no attempts have been made to benchmark these mutations in order to find the most efficient ones.

The goal of this paper is to benchmark and analyze the performance of various mutation strategies in order to find which one performs best in terms of how often and how fast they can find the global minimum. This will be done by experimenting with different mutations on a base implementation of a Genetic

Algorithm which is able to find the lowest-energy structure for small clusters. The findings of this paper could be of use to future Genetic Algorithm based solutions to this problem.

This report will be presented in the following structure. Chapter 2 will provide a detailed explanation of how the Genetic Algorithm optimizing the structure of a cluster of atoms works. The mutation strategies selected for testing will be presented in Chapter 3. Chapter 4 will present the details of the experimental environment along with the results of the experiments. Chapter 5 will contain a section on responsible research and the reproducibility of the experiments. A discussion on the results, the conclusion and suggestions for future work will be present in Chapters 6 and 7.

2 Methodology

In order to properly evaluate the efficiency of each mutation strategy, an initial Genetic Algorithm was coded together with the other members of Group 48 of the Research Project. The following is a description of the algorithm written in Python. The complete implementation can be found here.

2.1 Representation

The Atomic Simulation Environment (ASE) is a set of tools and Python modules intended for analyzing atomistic simulations. We have opted to use ASE in order to represent our clusters of atoms as an Atoms object. As such, ASE provides methods for easily accessing the atoms' positions and computing the energy of the entire cluster. It should also be noted that each cluster should respect the physical laws, meaning that a valid representation of a cluster of atoms is one where the pairwise distance between any two atoms is not smaller than 0.15 Ångströms ($= 10^{-10}$ meters) .

2.2 Structure

Our implementation follows a standard optimization loop for Genetic Algorithms. The algorithm starts from a randomly initialized population. Each iteration consists of locally optimizing the clusters, computing their fitness, selecting the parents for the next generation, performing crossover to generate the next generation and mutating the new clusters. This process is repeated until any of the stopping criteria are met. The structure is illustrated in Figure 1. Each feature of the Genetic Algorithm is presented below.

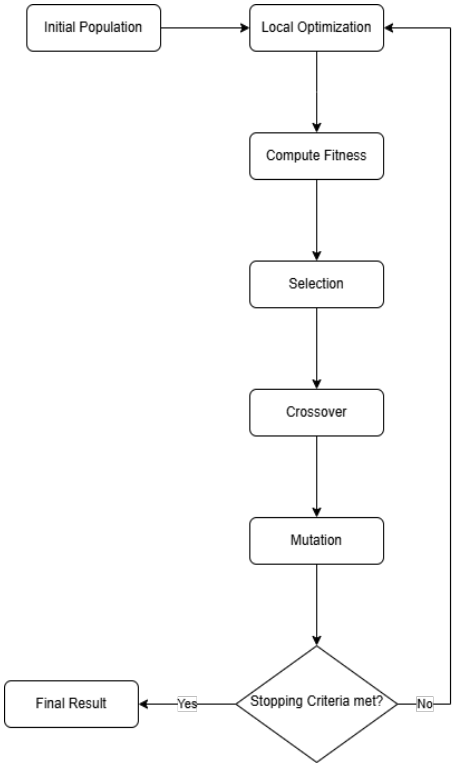

Figure 1: Flowchart illustrating the structure of the Genetic Algorithm used in our experiments.

2.3 Initial Population

The initial population consists of a number of clusters, each containing the same number of atoms. To generate these clusters, we uniformly sample coordinates inside of a bounding cube. The side length of the bounding cube is dependent on the number of atoms and it is given by the formula:

$$2 \cdot \left(0.5 + \left(\frac{3 \cdot \texttt{Number of Atoms}}{4\pi\sqrt{2}} \right)^{\frac{1}{3}} \right)$$

The purpose of using a bounding cube is to ensure that atoms are initialized within a reasonable spatial density. This prevents them from being placed too far apart, which would result in very high energy configurations and an increase in convergence time.

2.4 Local Optimization

Before fitness evaluation, each cluster undergoes local optimization to relax its structure into a nearby local minimum. This is performed by using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, as implemented in ASE, for at most 10000 steps.

2.5 Fitness Function

We then compute the fitness of each cluster using Lennard-Jones Potential (LJ). The reason why this was chosen is because it allows us to abstract away any chemical details, such as the atom type. Since the atom type does not matter when computing the energy, the problem is reduced to a purely geometric form. The fitness of a cluster results from summing the following pairwise Lennard-Jones Potential formula across all pairs of atoms.

$$V_{\mathrm{LJ}}(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$

where:

- $V_{\mathrm{LJ}}(r)$ is the potential energy as a function of the distance r between two particles,
- $\varepsilon = 1$ is the depth of the potential well, representing the strength of the interaction,
- $\sigma = 1$ is the finite distance at which the inter-particle potential is zero,
- r is the distance between the centers of the two particles.

2.6 Selection Strategy

A form of truncation selection is used to reduce the amount of possible parents of the next generation of clusters by only considering 50% of the population with the best fitness. Pairs of parents are then randomly sampled from this selected group for crossover.

2.7 Crossover

Crossover is performed using a cut-and-splice method. First, we translate the center of mass of each of the parents to the origin. Then, Principal Component Analysis is used to align the principal axes of the cluster. After that, a random plane is generated and each parent is split into two halves. Each half from one parent is combined with the opposite half from the other parent. If the children cluster contains the same number of atoms as the parent and its representation is valid, it is accepted to the new population. Otherwise, we generate another random plane. We repeat this process until we have generated a number of children equal to the initial population. Mutations are then performed on these children and they subsequently replace the current population.

2.8 Stopping Criteria

The algorithm terminates if any of the following three stopping criteria are met:

- The global minimum has been reached, defined as the best solution having an energy within 10^{-6} of the known global minimum.

- The maximum number of iterations set at 100 has been reached.
- There has been no improvement to the best fitness function of the population in the past 10 iterations.

3 Mutations

Due to the nature of the potential energy surface of many of the clusters we are trying to optimize [2], it is often the case that an algorithm can get stuck in a funnel leading to a local minima. This is why mutations are a very important part of our algorithm as they help avoid stagnation or convergence to a suboptimal solution.

It should be noted that all the mutated clusters must have a valid configuration. As such, each mutation operation has a maximum number of attempts (set at 1000) to find a valid configuration. If not, the algorithm will proceed without mutating the cluster.

In what follows, we introduce the mutation operations selected for benchmarking in our experiments. These mutations were found in previous implementations of Genetic Algorithms for cluster optimization and have been adapted to our base algorithm.

3.1 Atom Replacement

One of the most basic forms of mutations that we can implement is Atom Replacement. If a cluster is selected to be mutated, around a third of its atoms will be assigned random coordinates [9]. This is implemented by, with a probability of 0.333, regenerating each atom in the same way as how the initial clusters' atoms are generated.

3.2 Cluster Replacement

A very similar, yet more aggressive mutation strategy is Cluster Replacement, where the entire cluster is regenerated instead [5]. The core idea behind this is to try to explore a completely different part of the energy surface. We regenerate a cluster using the same process as for the initial population.

3.3 Center of Mass Spherical (CoM-S)

Originally developed as part of CEO-GA, a cluster optimization algorithm that aims to solve the very same problem that we are working on without the use of local optimizers, CoM-S was designed to quicken the search of the energy space [6]. The way it works is by rotating each atom, with a probability of 0.1, around the center of mass of the cluster. The idea behind it relies on the spherical nature of many of the lowest-energy clusters. Our implementation of CoM-S relies on vector space operations. For each atom, we compute the vector from the cluster's center of mass to the atom and measure its length. A new vector of the same length is then generated in a random direction, effectively rotating the atom around the center of mass to determine its new position.

3.4 Random Displacement

Random Displacement is one of the most prevalent mutation operations used, being present in Deaven and Ho's initial algorithm using Cartesian coordinates [1], as well as more recent algorithms such as the Birmingham Cluster Genetic Algorithm [5]. The idea behind it is very simple: displace a certain number of atoms by a very small amount. Specifically, our implementation is based on the algorithm developed by Zhao et al. [16]: perform a displacement of at most 10% of the average pairwise distance between atoms on all atoms with a probability of 0.33.

3.5 Etching

Etching is a mutation operation that is intended to mimic physical processes leading to the formation of clusters [13]. The way it works is by first adding a fixed number of atoms, in our case 10, to the existing cluster. Then, local optimization is performed on the new cluster for a small number of steps: 100 for our implementation. We are now left with a cluster of $N + 10$ atoms and the goal is to reduce it back to N atoms. This is achieved iteratively: in each iteration, the atom with the highest individual energy is removed, followed by another short local optimization. This process is repeated 10 times in order to achieve the initial number of atoms.

It should also be noted that Etching can also be performed in the reverse order: remove the 10 highest-energy atoms and then iteratively add them back while using a short local optimization. In our implementation, whenever Etching is applied on a cluster, one of those two variants will be chosen with an equal probability of 0.5.

3.6 Neighbor

One of the most intuitive mutation operations is the Neighbor mutation, which involves moving a randomly selected atom into the vicinity of another randomly chosen atom [16]. The underlying idea is straightforward: placing atoms closer together often results in a lower overall energy. Our implementation of this mutation operation considers the neighborhood of an atom to be the sphere around it, with a radius equal to the minimum pairwise distance between any two atoms in the cluster. An atom is selected to be moved and another one is selected as the neighbor. A vector is then generated to represent the direction, starting from the neighbor, of where to move the other atom. The length is decided by uniformly sampling a number between 0 and the neighborhood radius, while also checking that the cluster representation is valid.

3.7 Random Walk

Random Walk is a mutation operation often used in early versions of Genetic Algorithms for cluster optimization [1] [13]. It involves selecting around a fifth of the atoms and taking them on a random walk through space. In our implementation, each atom is selected with a probability of 0.2. They are then taken

on a random walk with a step size of 0.2 Ångströms and a maximum number of 50 steps.

3.8 Twist

The Twist (or Twinning) mutation is one of the most used mutation operations in cluster optimization. Originally proposed by Wolf and Landman [13], it begins by dividing the cluster into two parts, similar to how crossover is performed. Then, randomly select a side and rotate it around the normal vector of the dividing plane by a random number of degrees.

3.9 Note on Atom Permutation Mutation

Another mutation operation frequently used cluster optimization involves swapping the atomic types of two atoms within a cluster [5]. This can be particularly useful for alloy or compound clusters [16]. However, in our case, this mutation is not applicable, as we use Lennard-Jones potential which means that changing atomic types would have no effect on the energy of a cluster and thus it is excluded from our experiments.

4 Results

4.1 Experimental Setup

The Cambridge Energy Landscape Database is a publicly available database of known and verified minimal energy clusters, including Lennard-Jones clusters. This database is used as a ground truth for our experiments.

The aim of the experiments is to find the best performing mutation operation based on two criteria:

- **Accuracy**: How often can the mutation reach the global minimum?
- **Speed**: How much time does it take to reach the global minimum?

Each mutation operation is tested independently under identical conditions. The experiments are ran on population sizes of 8, 15 and 20. Each mutation is tested with different probabilities ranging from 0.005 to 0.3, depending also on the population size. Each experiment is performed 10 times in order to account for stochastic variability. All runs are performed on the DelftBlue supercomputer.

4.2 Population Size 8

When using a low population size of 8, we expect the choice of mutation operation to play a very important role. With fewer clusters per generation, the algorithm's capacity to explore the search space is reduced. This means that the algorithm has an increased reliance on mutations to escape local minima and reach the global minimum efficiently.

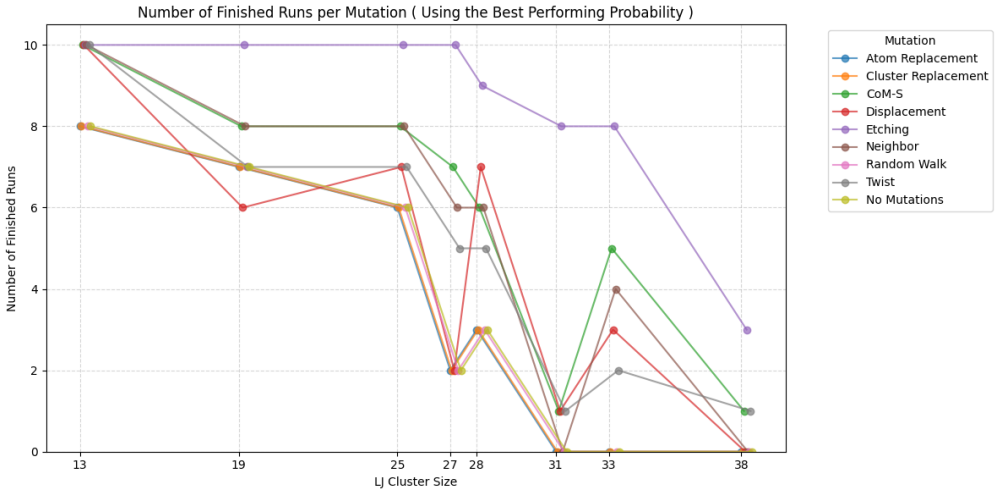

Figure 2: Number of finished runs (runs reaching the global minimum) for each mutation at its best performing probability on population size 8.

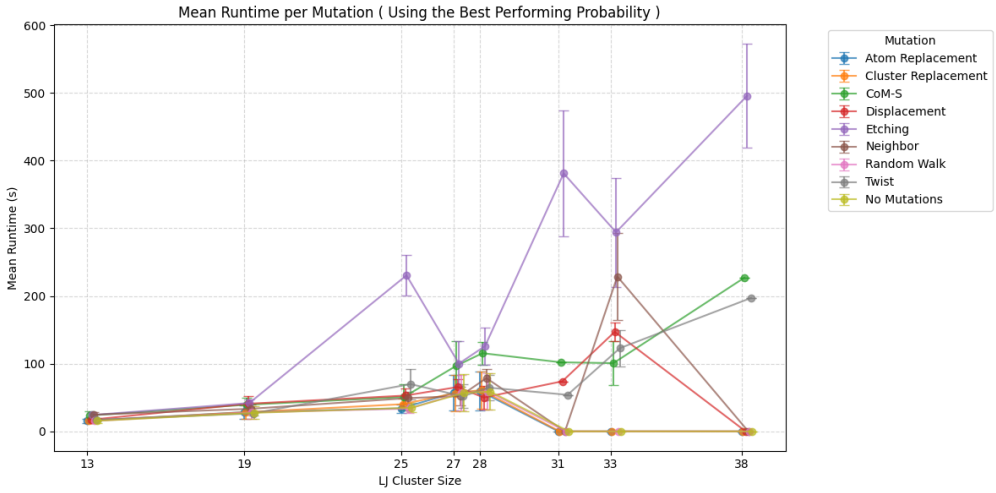

Figure 3: Convergence time of finished runs (runs reaching the global minimum) for each mutation at its best performing probability on population size 8.

Looking at Figure 2, we can see that in terms of accuracy, the Etching mutation shows remarkable results, as it performs better than the others on all cluster sizes tested. One of the most surprising results comes from the size 38 cluster, where the base algorithm using the Etching mutation, albeit with a relatively high mutation probability of 0.3, manages to find the global minimum 3 out of 10 times. The reason why this is so surprising is that finding the lowest energy LJ38 cluster is a very difficult process due to its double-funnel energy landscape [3], which leads most attempts into local minima. Despite these positives, Etching stands out as the poorest performer when it comes to runtime, as shown in Figure 3. On larger clusters, it takes two to four times longer to converge com-

pared to other mutations, while also having a higher variance in converge time. What this tells us is that Etching could be very beneficial by being used with a low probability of happening, in order to avoid high convergence times, and in combination with other mutation operations, so as help maintain high accuracy. This could prove to be a great opportunity for further research on this topic.

At the other end of the spectrum, Atom Replacement, Cluster Replacement and Random Walk mutation operations do not seem to bring any improvement to the algorithm, at least in the case of a low population size, as they only managed to get the same number of finished runs and very similar runtimes as our base algorithm with no mutations.

The remaining four mutation operations display very comparable results. Out of these, CoM-S and Neighbor seem to be the most similar. While CoM-S exhibits an ever so slightly higher accuracy, the Neighbor mutation has a lower runtime, apart from LJ33 where it performs almost as bad as Etching.

The Twist mutation performs slightly worse when it comes to accuracy, but where it stands out is its low, and relatively stable runtimes, being one of the best performers on all cluster sizes. Another interesting result is that Twist was only one of three mutations able to find the LJ38 lowest-potential cluster, but that could also be due to random chance. Random Displacement also displays great speed, but its inconsistent behavior when it comes to accuracy (on LJ27) would suggest to avoid it, at least for a low population size.

4.3 Population Size 15

With a higher population of 15, we aim to find a balance between the exploitative and explorative properties of the mutation operations, allowing us to asses their performance in a more balanced setting. A higher initial population means that not as much emphasis is placed on the explorative ability of the mutations.

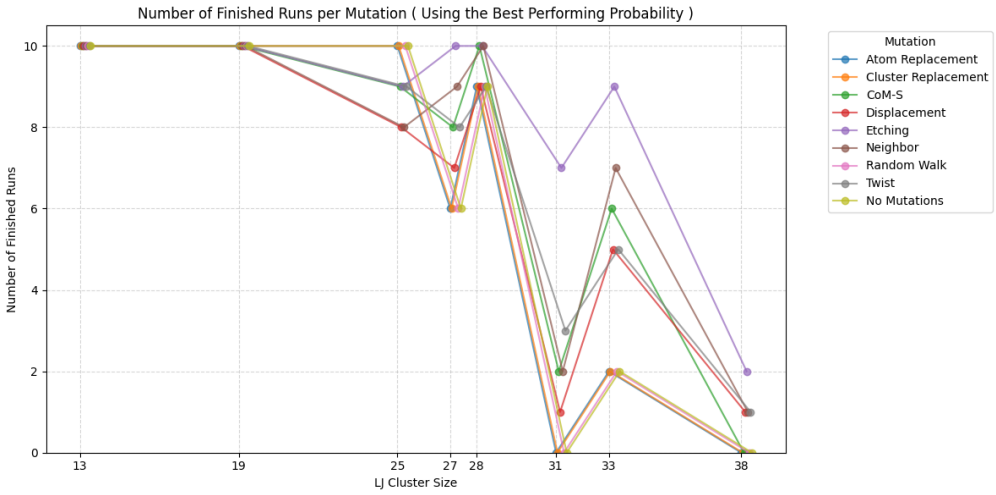

Figure 4: Number of finished runs (runs reaching the global minimum) for each mutation at its best performing probability on population size 15.

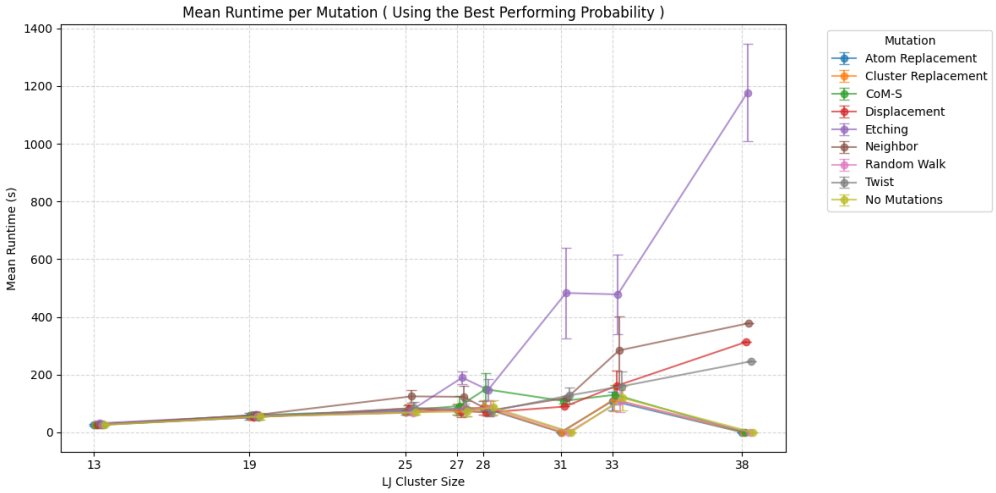

Figure 5: Convergence time of finished runs (runs reaching the global minimum) for each mutation at its best performing probability on population size 15.

In Figure 4 we can see that increasing the population size has also increased the accuracy of all mutation operations, which is to be expected, with the exception of Etching. Despite this, Etching still seems to perform considerably better than all other mutations in terms of accuracy, especially on harder to find solutions such as on LJ31 and LJ38. In terms of speed, Etching is still the slowest to converge, as can be seen in Figure 5.

Another observation that we can make is that, as was the case with the population size set at 8, Atom Replacement, Cluster Replacement and Random Walk perform in almost the exact same way as the initial algorithm running without mutations.

The performance of the last four mutation operations becomes even more similar with the population size set at 15. In terms of Accuracy, the mutation with the better results varies between CoM-S, Neighbor and Twist, with the only constant being Random Displacement, which has the lowest Accuracy. In terms of Speed, the results are so similar that we cannot conclude anything. Despite this we can observe some outliers such as the Neighbor mutation having a higher time to converge on LJ25 and LJ27, and CoM-S performing badly on LJ28. Conversely, the runtimes of Twist and Random Displacement seem to be more stable.

4.4 Population Size 20

Using a high population size of 20, we expect most runs to finish no matter the mutation operations used. The idea behind these experiments is to focus on the convergence times of the mutation strategies in a favorable environment.

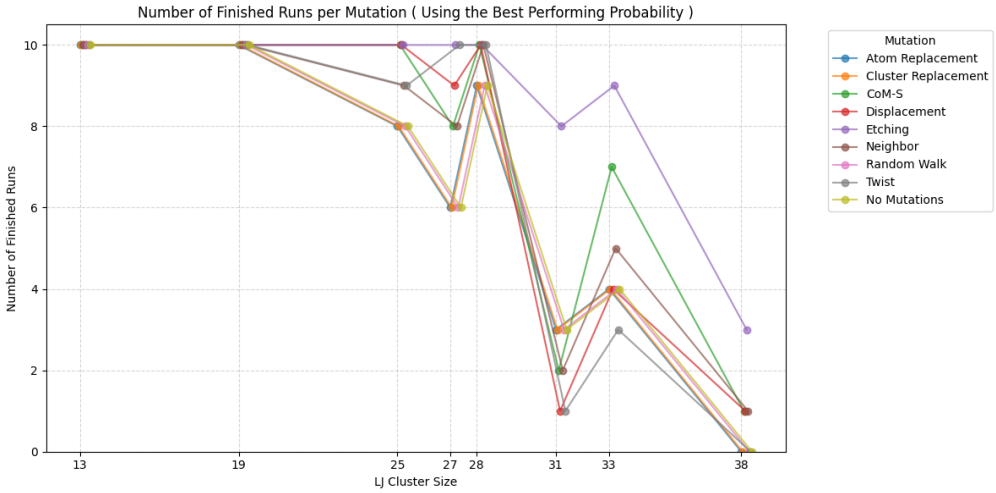

Figure 6: Number of finished runs (runs reaching the global minimum) for each mutation at its best performing probability on population size 20.

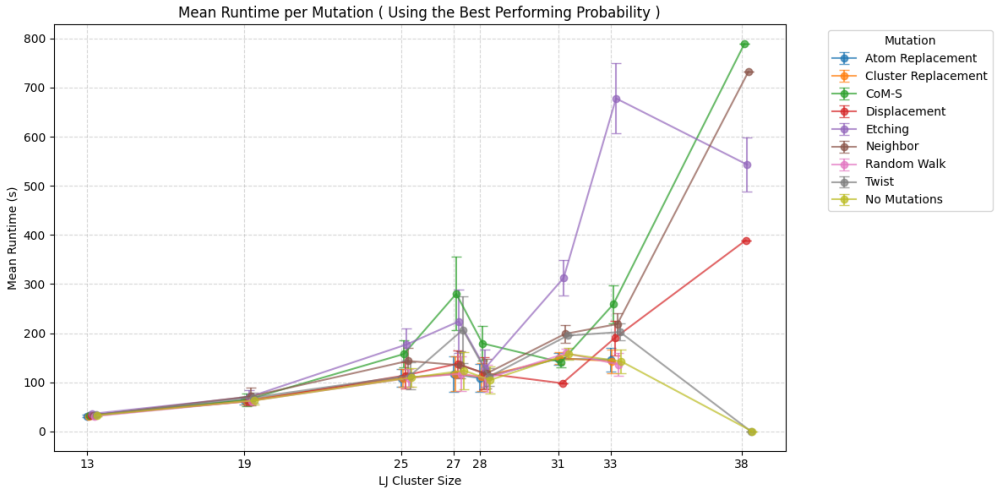

Figure 7: Convergence time of finished runs (runs reaching the global minimum) for each mutation at its best performing probability on population size 20.

By looking at Figure 6, we can deduce that further increasing the population does not affect the accuracy of most mutation operations. Three of the exceptions are Atom Replacement, Cluster Replacement and Random Walk, which is to be expected since they perform the same as using no mutations and a more diverse initial population means a higher chance of starting with a solution which leads to the global minimum through local optimizations. The other exception is Random Displacement, which has a slightly increased accuracy on medium-sized clusters of 25, 27 and 28 atoms, while also having the best performance in terms of speed (apart from the three previously mentioned mutations).

Another interesting result comes with the Twist mutation, which actually

performs worse in terms of accuracy with a higher population of 20 as opposed to 15. This result is very surprising because, despite using a mutation probability as low as 0.005, Twist managed to perform worse than the baseline algorithm with no mutations on the LJ33 cluster. What this tells us is that in spite of the fact that the Twist mutation provides a great balance between accuracy and speed, balancing the population size is a very important factor in its performance.

CoM-S and Neighbor mutations once again seem to perform similarly, displaying a slightly higher accuracy than Random Displacement and Twist on bigger clusters (LJ31, LJ33), but with a significant time overhead of up to around 33%, as can be seen in Figure 7.

Unlike the other mutation operations, Etching showcases a special property which is that its convergence time does not appear to increase when used with a higher population size such as 20, as seen in Figure 8. This could indicate that the Etching mutation may also perform very well on even bigger clusters where big population sizes are needed.

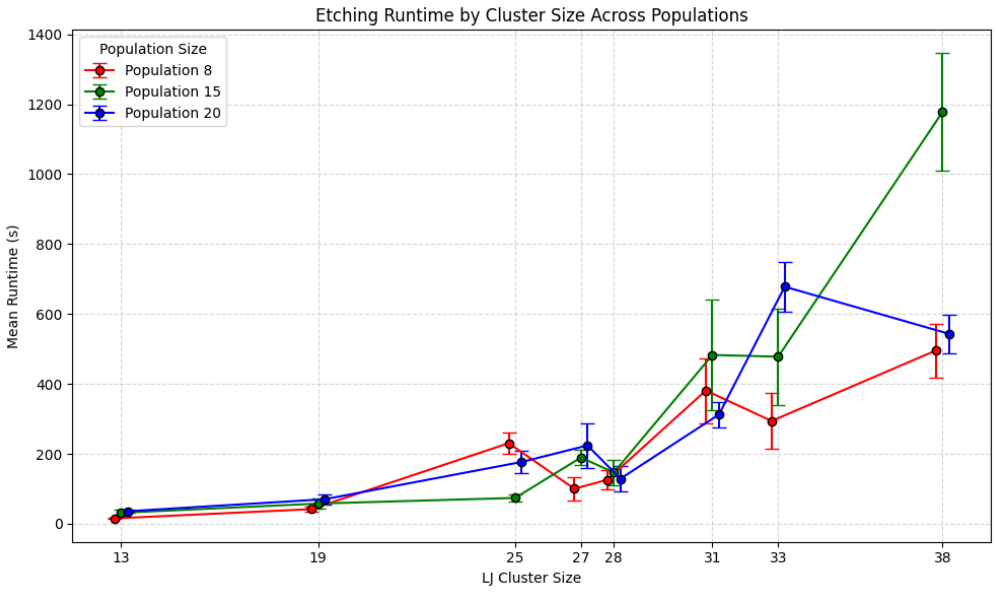

Figure 8: Convergence time of finished runs (runs reaching the global minimum) for Etching at its best performing probability.

4.5 Discussion

Considering all the results, we can confidently say that the Etching mutation performs the best in our situation, at least in terms of Accuracy. It should also be noted that since every use of the Etching mutation entails locally optimizing a cluster ten times, it is heavily reliant on the type of local optimizer used (in our case BFGS). Thus, further research on its performance using different local optimizers, as well as on its Accuracy and Speed on bigger clusters would be beneficial.

Random Displacement and Twist, despite their shortcomings in terms of Accuracy, have the best convergence speeds. This suggests two promising solutions which could be further explored: incorporating restart strategies in order to mitigate premature convergence, or combining these fast mutations with more reliable ones like Etching to improve on the Accuracy without significantly increasing runtime.

CoM-S and the Neighbor mutation display average results, not excelling in either of Accuracy or Speed. However, their performance may significantly improve when combined with other mutations, making it a promising idea for future studies.

Atom Replacement, Cluster Replacement and Random Walk did not display any results to suggest that they could improve the Genetic Algorithm and thus, they should be excluded from future works.

5 Responsible Research

Our project does not deal with collecting, or even using data related to human subjects, so there are no privacy concerns. Instead, our data regarding the lowest-energy Lennard-Jones energy clusters comes from The Cambridge Energy Landscape Database, as stated before. This database is easily accessible by anyone and it contains verifiable results and sources for how they were discovered.

In order to ensure transparency we have made the entire codebase including the initial algorithm, the mutation operations' implementations and all experimental results, not just the best performing ones, on our Gitlab repository.

To promote reproducibility, we used fixed random seeds for all experiments, enabling exact replication of our runs. One concern could come from the fact that we cannot guarantee that repeated experiments on the Delftblue supercomputer will run on the exact same hardware nodes. Despite this, we have not observed deviations of more than 0.1 seconds in runtime which, considering the scale of our results, are negligible.

Additionally, any Genetic Algorithm solving the problem of finding the lowest-energy structure of a cluster has the potential to be affected by a loss of precision on its results due to floating point errors. Thus, all new results should ideally be further validated by real world testing.

Finally, AI-based tools (ChatGPT-4o) were only lightly used for rephrasing and improving clarity during the writing process, using prompts such as "Rephrase this sentence", with all scientific content and analysis produced independently.

6 Conclusions

In this study, we benchmarked a wide variety of mutation operations within a Genetic Algorithm designed to find the lowest-energy structure of Lennard-Jones atomic clusters based on two performance criteria: the ability to reach the global minimum (Accuracy) and the time required to do so (Speed). Our experiments were carried out across varying population sizes (8, 15, and 20) and mutation

probabilities (ranging from 0.005 to 0.3). Each mutation operation was evaluated independently, without combining them with any other mutations. From the obtained results, several key observations can be made.

Firstly, the Etching mutation consistently outperformed all other mutation operations in terms of Accuracy. However, this advantage comes at a significant cost in terms of convergence time, particularly on smaller populations. Despite this, we observed a promising reduction in the runtime of Etching as the population size increased, indicating that it could be particularly effective in high population and large cluster scenarios.

In contrast, Atom Replacement, Cluster Replacement, and Random Walk mutations showed no meaningful improvement over using no mutation at all. This suggests that they are not suitable for this problem setup and can be excluded from future applications.

Mutations like Twist and Random Displacement exhibited the best runtimes while achieving average Accuracy. This suggests that they are good candidates for hybrid strategies, containing multiple mutation operations.

CoM-S and Neighbor mutations displayed average performance in terms of both Accuracy and Speed. While not excelling individually, they may provide value when used in combination with other mutations.

Thus, it is suggested that future developments in the field of Genetic Algorithms for finding the lowest-energy structure of a cluster of atoms should be based on the highest performing mutation operations found in our benchmarks, such as Etching, Twist and Random Displacement.

7 Future Work

Several directions for future work are worth considering to build upon the findings of our benchmarks. Firstly, despite the promising results of the Etching mutation, we believe that it may further be improved by using a different local optimizer.

Another idea for future research would be to experiment with the impact of using a restart strategy in a Genetic Algorithm with some of the better performing mutations in terms of speed such as Twist and Random Displacement. We would expect their accuracy to increase, but we do not know how much of an impact it would have on their time to converge.

It would also be very beneficial to evaluate the performance of hybrid mutation strategies, making use of multiple mutation operations. Particularly, the results of combining Etching and Twist / Random Displacement, or even CoM-S and Neighbor could prove to be very advantageous.

Lastly, it could be interesting to analyze if our results translate to even bigger clusters than 38 atoms (although this is not to be expected while only using one mutation operation at a time), to higher population sizes, or to different energies other than Lennard-Jones.

References

[1] D. M. Deaven and K. M. Ho. "Molecular Geometry Optimization with a Genetic Algorithm". In: *Phys. Rev. Lett.* 75 (2 July 1995), pp. 288–291. DOI: 10.1103/PhysRevLett.75.288. URL: https://link.aps.org/doi/10.1103/PhysRevLett.75.288.

[2] Jonathan P. K. Doye, Mark A. Miller, and David J. Wales. "Evolution of the potential energy surface with size for Lennard-Jones clusters". In: *The Journal of Chemical Physics* 111.18 (Nov. 1999), pp. 8417–8428. ISSN: 0021-9606. DOI: 10.1063/1.480217. eprint: https://pubs.aip.org/aip/jcp/article-pdf/111/18/8417/19025739/8417_1_online.pdf. URL: https://doi.org/10.1063/1.480217.

[3] Jonathan P. K. Doye, Mark A. Miller, and David J. Wales. "The double-funnel energy landscape of the 38-atom Lennard-Jones cluster". In: *The Journal of Chemical Physics* 110.14 (Apr. 1999), pp. 6896–6906. ISSN: 0021-9606. DOI: 10.1063/1.478595. eprint: https://pubs.aip.org/aip/jcp/article-pdf/110/14/6896/19184162/6896_1_online.pdf. URL: https://doi.org/10.1063/1.478595.

[4] Michael Galchenko et al. "Field Effect and Photoconduction in Au25 Nanoclusters Films". In: *Advanced Materials* 31.18 (2019), p. 1900684. DOI: https://doi.org/10.1002/adma.201900684. eprint: https://advanced.onlinelibrary.wiley.com/doi/pdf/10.1002/adma.201900684. URL: https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/adma.201900684.

[5] Roy L. Johnston. "Evolving better nanoparticles: Genetic algorithms for optimising cluster geometries". In: *Dalton Trans.* (22 2003), pp. 4193–4207. DOI: 10.1039/B305686D. URL: http://dx.doi.org/10.1039/B305686D.

[6] Vera A. Kazakova, Annie S. Wu, and Talat S. Rahman. "Cluster energy optimizing genetic algorithm". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. GECCO '13. Amsterdam, The Netherlands: Association for Computing Machinery, 2013, pp. 1317–1324. ISBN: 9781450319638. DOI: 10.1145/2463372.2463536. URL: https://doi.org/10.1145/2463372.2463536.

[7] Sanne M. van de Looij et al. "Gold Nanoclusters: Imaging, Therapy, and Theranostic Roles in Biomedical Applications". In: *Bioconjugate Chemistry* 33.1 (2022). PMID: 34894666, pp. 4–23. DOI: 10.1021/acs.bioconjchem.1c00475. eprint: https://doi.org/10.1021/acs.bioconjchem.1c00475. URL: https://doi.org/10.1021/acs.bioconjchem.1c00475.

[8] Abhishek K. Pathak and Sanjay R. Dhakate. "Carbon Nanomaterial-Carbon Fiber Hybrid Composite for Lightweight Structural Composites in the Aerospace Industry: Synthesis, Processing, and Properties". In: *Advanced Composites in Aerospace Engineering Applications*. Ed. by Norkhairunnisa Mazlan, S.M. Sapuan, and R.A. Ilyas. Cham: Springer International Publishing, 2022, pp. 445–470. ISBN: 978-3-030-88192-4. DOI: 10.1007/978-3-

030-88192-4_23. URL: https://doi.org/10.1007/978-3-030-88192-4_23.

[9] C. Roberts, R. Johnston, and N. Wilson. "A Genetic Algorithm for the Structural Optimization of Morse Clusters". In: *Theoretical Chemistry Accounts* 104 (2000), pp. 123–130. DOI: 10.1007/s002140000117. URL: https://doi.org/10.1007/s002140000117.

[10] Gustavo G. Rondina and Juarez L. F. Da Silva. "Revised Basin-Hopping Monte Carlo Algorithm for Structure Optimization of Clusters and Nanoparticles". In: *Journal of Chemical Information and Modeling* 53.9 (2013). PMID: 23957311, pp. 2282–2298. DOI: 10.1021/ci400224z. eprint: https://doi.org/10.1021/ci400224z. URL: https://doi.org/10.1021/ci400224z.

[11] Michal Roth, Yoni Toker, and Dan T. Major. "Monte Carlo-Simulated Annealing and Machine Learning-Based Funneled Approach for Finding the Global Minimum Structure of Molecular Clusters". In: *ACS Omega* 9.1 (2024), pp. 1298–1309. DOI: 10.1021/acsomega.3c07600. eprint: https://doi.org/10.1021/acsomega.3c07600. URL: https://doi.org/10.1021/acsomega.3c07600.

[12] L. T. Wille and J. Vennik. "Computational complexity of the ground-state determination of atomic clusters". In: *Journal of Physics A: Mathematical and General* 18.8 (1985), pp. L419–L422. DOI: 10.1088/0305-4470/18/8/003.

[13] Matthew D. Wolf and Uzi Landman. "Genetic Algorithms for Structural Cluster Optimization". In: *The Journal of Physical Chemistry A* 102.30 (1998), pp. 6129–6137. DOI: 10.1021/jp9814597. eprint: https://doi.org/10.1021/jp9814597. URL: https://doi.org/10.1021/jp9814597.

[14] Yongliang Xiao and Donald E. Williams. "Genetic algorithm: a new approach to the prediction of the structure of molecular clusters". In: *Chemical Physics Letters* 215.1 (1993), pp. 17–24. ISSN: 0009-2614. DOI: https://doi.org/10.1016/0009-2614(93)89256-H. URL: https://www.sciencedirect.com/science/article/pii/000926149389256H.

[15] Yehuda Zeiri. "Prediction of the lowest energy structure of clusters using a genetic algorithm". In: *Phys. Rev. E* 51 (4 Apr. 1995), R2769–R2772. DOI: 10.1103/PhysRevE.51.R2769. URL: https://link.aps.org/doi/10.1103/PhysRevE.51.R2769.

[16] Jijun Zhao et al. "Comprehensive genetic algorithm for ab initio global optimisation of clusters". In: *Molecular Simulation* 42.10 (2016), pp. 809–819. DOI: 10.1080/08927022.2015.1121386. eprint: https://doi.org/10.1080/08927022.2015.1121386. URL: https://doi.org/10.1080/08927022.2015.1121386.