Efficient Utilization of Local Optimization Methods and Strategies in Local Search Genetic Algorithms for Lennard Jones Clusters

Kaloyan Yanchev[1]

Supervisors: Peter A.N. 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: Kaloyan Yanchev
Final project course: CSE3000 Research Project
Thesis committee: Peter A.N. Bosman, Anton Bouter, Vanessa Volz, Thomas Abeel

Abstract

This paper investigates how the local optimization method and strategy affect the efficiency of genetic algorithms (GAs) for Lennard-Jones (LJ) clusters. Several ASE-implemented optimizers were considered; however, only BFGS, FIRE, and Conjugate Gradient (CG) proved viable for integration. The optimizers were first benchmarked independently, and then the GA execution was benchmarked using the different optimizers, with the main solution quality metric being the number of times the global minimum (GM) is found. While BFGS produced better results, its timing was the highest and steepest scaling; furthermore, the trade-off could not be mitigated by reducing the maximal number of optimization steps. In contrast, FIRE and CG, when run with a doubled population size, produced superior results both in terms of execution time and final cluster energy. Furthermore, a cluster isomerism-based heuristic for applying selective local optimization halved the GA's execution time at the expense of a reduction in solution quality. Nonetheless, when computational time was equalized through doubling the population size, the heuristic-based GA produced equal or better results. These findings suggest that faster methods and suboptimal optimization choices, when combined with increased population diversity, can outperform more powerful but slower GA configurations.

1 Introduction

Global Geometry Optimization (GGO) is a central problem in the field of material science. The objective of the problem is to discover the most stable atomic configuration in 3-dimensional space. For this purpose, two inputs are required: the atomic type of the cluster (number of atoms of each type) and the potential function that models pairwise atomic interactions. Thus the objective is to find the cluster configuration that represents the minimal value for the sum of the potential energies over each atomic pair (total potential energy of the cluster). The GGO problem is proven to be NP-Hard [25], since the number of local minima on the potential energy surface (PES) is exponentially increasing with the number of atoms. This is true even for simple type-agnostic functions such as Lennard-Jones (LJ) [5], presented in Figure 1, which has one critical point – its global minimum.

The study of GGO applies to a wide range of scientific and technological fields. It enables discovery of novel low-energy atomic configurations, resulting in advanced materials. Applications of GGO include designing of high-performance electronic devices, efficient catalysts, and structurally resilient materials for aerospace or energy systems [10; 15]. In biology, LJ optimization techniques have proven informative in protein folding simulation, thus identifying stable folded states that resemble actual biological structures [16]. In pharmacy, GGO of ligand-receptor binding pathways allows for discovering new drugs [19]. In space exploration, GGO of boron nitride and polymer-metal hybrids has a promising outlook on the development of lightweight, efficient radiation shielding materials [17; 21]. Lastly, in the field of photonics, inverse design and GGO techniques enable the creation of coatings capable of near-perfect absorption across the infrared spectrum, useful for energy harvesting, thermal regulation, and imaging [9; 14].

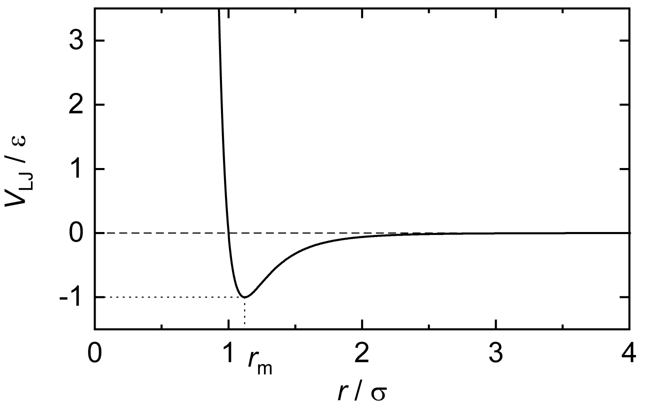

Figure 1: Graph of the Lennard-Jones potential function [22].

Genetic Algorithms (GAs) [7] have proven to be an effective strategy for solving the GGO problem due to their meta-heuristic and population-based approach. The very first works on GAs for this problem have identified the need for local optimization, which narrows down the search space to local minima only [3; 26]. Figure 2 represents the general execution flow of the local search GA. To this end, the L-BFGS [12] has become the standard choice of local optimizer, which is evident in numerous more recent works [8; 13; 18; 20]. Nonetheless, any choices concerning local optimization are hardly ever discussed. It must be noted that Pereira et al. [18] are the only ones that mention a configuration parameter, namely Local Search Length (LSL), which corresponds to the maximum number of steps; however, no value is reported.

While local search significantly improves the results of the GA, it also introduces high costs in terms of execution time. Heiles and Johnston [6] make an important observation: optimizations of solutions that yield the same minimum are redundant, and preventing those significantly reduces computational expenses. Even though predicting the local minimum seems impossible, some clusters are optimized each generation, even though they are later discarded by the GA selection. Furthermore, a number of intricate cluster cases, such as LJ38, prove the existence of multiple deep funnel local minima [5]. In such cases, the utilization of local searches might lead to premature convergence of the GA. All of these problems signal the need for consistent and systematic testing of local optimization methods and strategies to better mitigate the inherited trade-off between result quality and execution time.

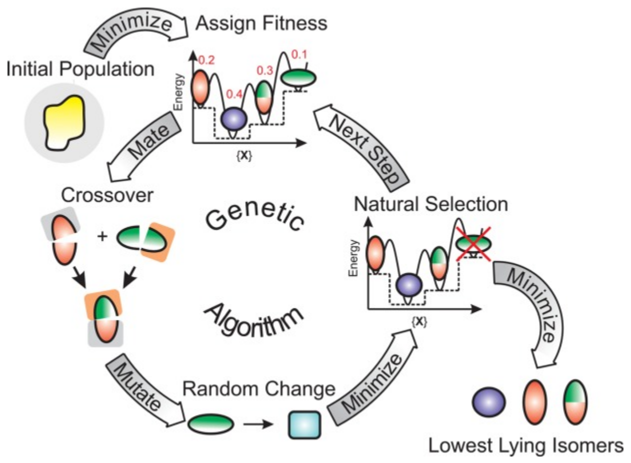

Figure 2: Schematic representation of a local search GA [6].

This paper poses the following central research question: *How does the choice of local optimization method and strategy affect the efficiency of a local search Genetic Algorithm for Global Geometry Optimization of Lennard-Jones clusters?* The primary aim is to identify efficient local optimization configurations and heuristics that reduce execution time without significantly compromising solution quality. There are two possible approaches – either reducing the time for every local optimization or reducing the total number of local optimizations. First, the optimization methods are independently assessed in their convergence behavior, execution time and ability to locate local minima. This benchmarking helps narrow down the set of viable local optimizers for integration and provides insight about their optimal configuration. Afterwards, the selected methods are embedded within the GA framework, and full global optimization runs are performed, providing the main results – the GA's execution time and the energy of its best-found cluster. Finally, a strategy for selective local optimization is implemented and tested.

2 Genetic Algorithm Framework

The GA framework is based on an open-source repository [1] that contains previous research on the subject, developed in Atomic Simulation Environment (ASE) [11], a Python library. Note that the repository has a larger collection of several algorithms that follow an interface and can be run in parallel. Only the general execution flow of the sequential GA is preserved in the modified framework. For simplicity of testing, certain values have been abstracted to parameters, and minor adjustments have been made regarding the correct execution of operations. Because the GA is highly modular and many of the operations and their sequencing are controlled through its parameters, only the default execution flow is discussed. The GA uses 3D Cartesian floating-point representation of atomic coordinates and the fitness value of clusters is their potential energy.

The initial population's atomic configurations are randomly sampled from a cube with a side proportional to the number of atoms in the cluster. The only restriction all throughout the GA, whenever a cluster is generated (initialization and crossover) or disturbed (mutation), is that there is no interatomic distance of less than 0.15 units. After the initial 8 clusters are generated, they are locally optimized using BFGS with a maximum of 10000 optimization steps. Then, the clusters' potential energies are noted and the best half, meaning 4, of the clusters are selected. Using the 4 selected clusters, 4 pairs are randomly drawn such that each pair consists of two different clusters. For each pair, a child is generated using the standard cut and splice crossover method as illustrated in Figure 3. Finally, the population, now containing 8 clusters again (4 preserved parents and 4 generated children), undergoes probability-based mutation.

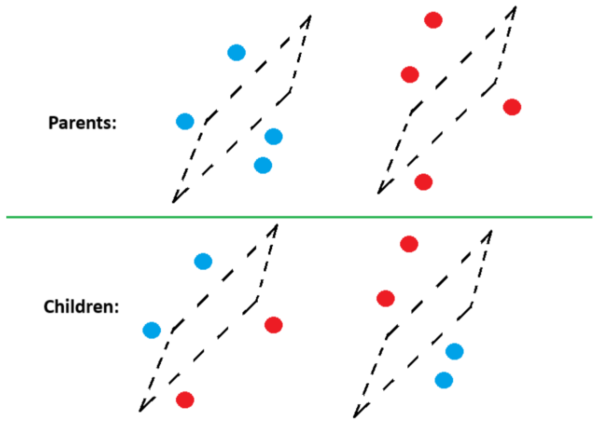

Figure 3: Cut and splice crossover [3].

There are three mutations that can be performed: twist, random displacement and etching. Twist mutation is applied to a whole cluster with a probability of 0.2. The cluster is split in two by a random geometric plane; one side is chosen at random and then it is rotated by a random degree around the normal of the splitting plane. Concerning random displacement, it is applied per atom in a cluster – with a probability of 0.1, the atom is displaced 0.2 units in a randomly drawn direction in 3D space. Finally, etching has two variations, each applied with a probability of 0.025 for a cluster. The first variant initially removes the highest energy atom, then performs local optimization and finally introduces an atom at random, while the other variant initially introduces an atom at random, then performs local optimization and finally removes the highest energy atom.

At this point, the GA's execution has gone full circle and the operations repeat each other – local optimization is performed and its results are treated as the new generation. The GA optimizes for 100 generations at most. However, if no better cluster than the overall best has been found in the last 20 generations, the GA converges and terminates the execution early (before the 100th generation). The established criterion for equating two solutions is a difference of potential energies less than $10^{-6} \times n^2$, where n is the number of atoms in the cluster.

Etching mutation is an operation that utilizes additional local optimization. It emerged from the development of a single-cluster global optimization algorithm called basin hopping [24]. The main idea is that, as illustrated in Figure 4, clusters of adjacent sizes have similar global minimum (GM) symmetries. However, some clusters have a very different GM symmetry than their adjacent ones; in such cases, etching can cause premature convergence on deep funnel local minima. Nonetheless, local optimization in etching mutation is not to be differentiated from the regular one. These additional optimizations can introduce some variance in the execution time; however, given its low probability rate, it should not have a tremendous influence over the total timing. Therefore, local optimization in etching mutation is not to be explicitly studied in this paper.

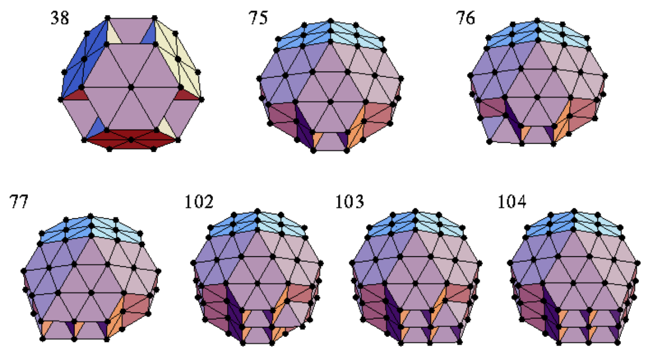

Figure 4: Selected LJ clusters visualizations [24].

3 Independent Benchmarking

The local optimizers implemented in or adapted to ASE are:

- BFGS (Broyden-Fletcher-Goldfarb-Shanno)
- BFGSLineSearch (Quasi-Newton)
- LBFGS (Limited-memory BFGS)
- LBFGSLineSearch (Limited-memory Quasi-Newton)
- GPMin (Gaussian Process Minimizer)
- MDMin (Moledular Dynamics Minimizer)
- FIRE (Fast Inertia Relaxation Engine) [2]
- SciPyFminCG (Conjugate Gradient, CG) – ASE wrapper for the SciPy implemented optimizer

The independent benchmarking is performed on clusters generated using the initialization strategy of the GA. Furthermore, all optimizers considered are benchmarked for optimizing the same initial clusters. However, the cluster sampling is random and the constraints allow for interatomic distances as low as 0.15 units. Since the LJ potential grows exponentially for distances below its optimal one ($\approx$ 1.12), a single low interatomic distance can introduce value-dominating forces and potentials. This means that the optimizers need to handle well the initial huge numerical instability. Nonetheless, those cases are indeed the first local optimizations to be performed inside the GA, so the optimizers do need to be able to handle them. On the other hand, those initial local optimizations are expected to be worst-timed, upper-bound cases, since subsequent clusters should be partially optimized. Finally, it should be noted that due to the nature of the GA, the viability and suitability of the local optimizers is determined from their capability of handling tightly bound atomic interactions.

There is a database of current global minima for the LJ clusters [4], consisting of both atomic configurations and energy levels. However, there are value discrepancies between the energy values in the database and the energy values obtained when a cluster object with the same atomic coordinates is generated in ASE. For this reason, a separate file with the energy values obtained in ASE is generated; also, a tolerance threshold of $10^{-6} \times n^2$, where n is the number of atoms, is considered for concluding the discovery of the global minimum.

3.1 Preliminary Tests

Four of the initial local optimizers are excluded, since single test cases are sufficient to showcase their shortcomings. Those have all been detected using a small cluster size, namely LJ13. The first one, the Gaussian Processes Minimizer (GPMin), throws a runtime error and terminates execution. Molecular Dynamics Minimizer (MDMin), on the other hand, does not converge at all and reaches the maximum number of optimization steps. Lastly, the two quasi-Newtonian line search optimizers present a peculiar case for exclusion. Even though they do manage to optimize some clusters, early on the random sequence generates a very tricky cluster. The problem is that the optimizers cannot perform a single optimization step; however, they also do not terminate, so it is presumed they just enter an infinite loop. Furthermore, this tricky cluster configuration is part of the initial population of the default GA, so this means that they also could not be used for the integrated experiments.

The preliminary tests consist of a small number of trials (10-50) for different cluster sizes and are executed for the remaining four local optimizers. BFGS and L-BFGS produce some non-converging optimizations, meaning that they execute until the maximum number of steps is reached. These are problematic for two reasons: first, those runs take longer than the average ones, and second, since the method has not converged, its result is suboptimal. In fact, the cluster energy in those cases is sometimes on the scale of 10^6, while the global minimum (GM) lies in the negative hundreds. Thus, quartile statistics are preferred for the independent tests to safely ignore such outlier results. Nevertheless, this means that non-converging local optimizations can only have a negative impact on the efficiency of the algorithm. However, the GA discards half of its clusters with the highest energies every generation, so a practical approach should prioritize only the better results from half of the runs. Nonetheless, L-BFGS produces 10/10 non-converging optimizations on LJ55 and 9/10 on LJ75, which is why it is excluded from the independent benchmarking.

3.2 Full-scale Experiments

The full-scale experiments are performed on cluster sizes that showcase the scaling challenges of the problem but require reasonable time. Furthermore, BFGS is executed with a maximum number of optimization steps less than 10000 such that the impact of non-converging optimizations is mitigated. Due to this, an acceptance criterion is defined – more than 75% of all optimizations should be converging in order for the quartile values to be representative. Thus, the experiment consists of the following tests: 1000 trials on LJ38 and on LJ47 and 500 trials on LJ55 and LJ65.

Results show that BFGS produces the highest and worst-scaling execution time; however, it finds lower energy minima. Figure 5 shows that for each of the cluster sizes, CG is the fastest method, tightly followed by FIRE, while BFGS is significantly slower than both. Furthermore, with the increase of cluster size, FIRE and CG are linearly scaling by a small gradient, while BFGS is scaling much more steeply and maybe non-linearly. Table 1, however, illustrates an opposite trend in terms of energy results. Particularly, BFGS produces the lowest median energies, followed by FIRE and finally CG. On top of that, BFGS shows another important quality – it is able to discover superior minima that the other optimizers cannot discover.

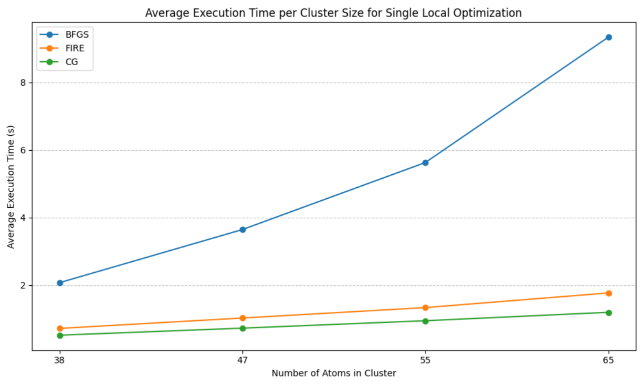

Figure 5: Average execution time per cluster size for a single local optimization by each of the methods in the independent experiment.

Table 1: Median and minimum cluster energy after single local optimization per method per cluster size in the independent experiment.

		BFGS	FIRE	CG
LJ38	median	-161.19	-160.05	-152.45
	minimum	-168.49	-167.50	-166.68
LJ47	median	-208.65	-207.38	-200.34
	minimum	-217.12	-215.64	-214.88
LJ55	median	-251.19	-250.24	-242.64
	minimum	-263.98	-260.36	-259.49
LJ65	median	-305.62	-304.82	-300.19
	minimum	-320.72	-315.97	-316.56

A novel analysis approach is used to study the optimal maximum number of optimization steps. The approach is based on examining the quartile values for each of the cluster sizes, since all three optimizers tested require more optimization steps with the increase in the number of atoms. Furthermore, since the GA uses only half of the clusters with the lower energies, the analysis includes those separately. Figure 6, Figure 7 and Figure 8 visualize the studied data for BFGS, FIRE

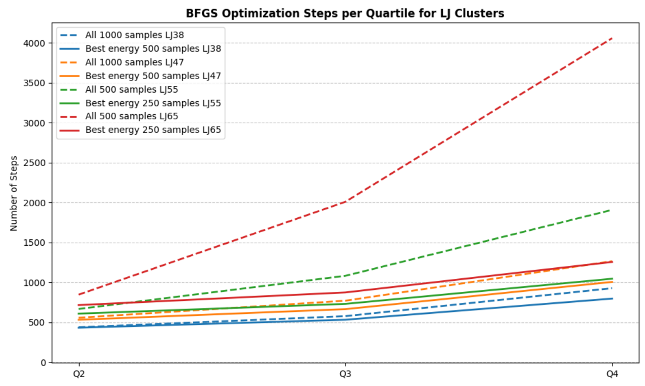

Figure 6: Graph of BFGS optimization steps quartile values.

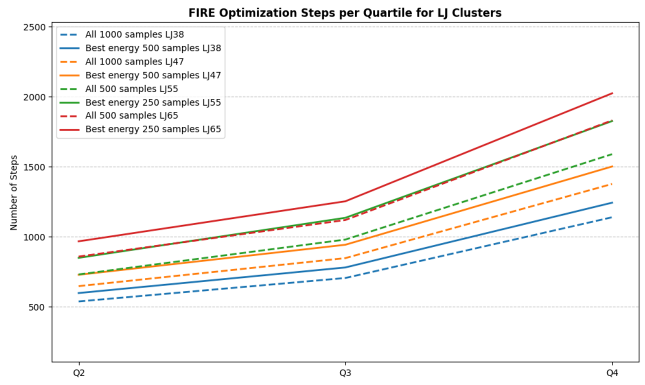

Figure 7: Graph of FIRE optimization steps quartile values.

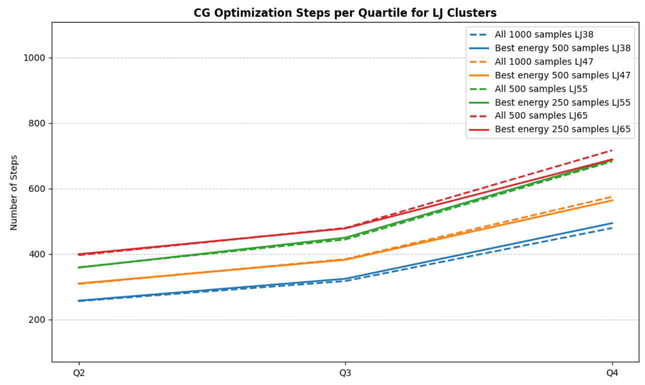

Figure 8: Graph of CG optimization steps quartile values.

and CG, respectively. The idea is to examine the step distribution for the selected results (solid lines) in relation to the step distribution for all results (dashed lines) in order to determine whether limiting the maximum number of optimization steps is suitable. In the case of FIRE, the lower energy results require more steps than the general case, and for CG, both distributions are equal. This means that a reduction in the maximum number of optimization steps is not suitable since that would result in higher energy of otherwise favorable results. BFGS, on the other hand, proves that its maximum number of optimization steps can be safely reduced because it produces lower energy results when it converges early, while longer optimizations yield unfavorable results. These findings have been confirmed by a second experiment using a different randomness seed, thus different initial cluster configurations. Furthermore, for BFGS, the second experiment uses much tighter limits on the number of steps such that the steps distribution for all results is skewed, with more than 25% of all runs not converging. Nonetheless, the distribution for the 50% better results is accepted since more than 75% of its runs converge and is confirmed to be the same as in the first experiment, meaning it can be assumed universal.

4 Integrated Experiments and Results

4.1 Local Optimizers and Configurations

Outcomes of the first integrated experiment conform to the findings in the independent benchmarking. The experiment consists of running 40 GA runs on LJ38 and on LJ47 and 20 GA runs on LJ55 and LJ65. For BFGS, the Q4 step values of the 50% best results from the independent experiments are used as the maximum number of optimization steps. Furthermore, LBFGS is included with step limits below 10000. Figure 9 visualizes the average execution times for full GA optimization, while Table 2 contains the number of times the global minimum (GM) is found. LBFGS, due to its non-convergence on large cluster sizes, scales exponentially with the number of atoms in the cluster without producing any significant results. BFGS now shows linearly scaling timing; however, it is still much higher than that of FIRE and CG and is more steeply scaling. Nonetheless, BFGS still manages to produce significantly better results by finding more GM than FIRE and CG.

Table 2: Number of times the global minimum (GM) is found by the GA per method per cluster size in the integrated experiments.

	LJ38	LJ47	LJ55	LJ65
LBFGS	0	6	0	0
BFGS	1	8	0	1
FIRE	1	6	0	0
CG	0	1	1	0
BFGS-Q4	1	6	0	0
BFGS-Q3	1	4	1	0
BFGS-Q2	0	1	0	0
FIRE2	0	13	3	0
CG2	0	11	2	2

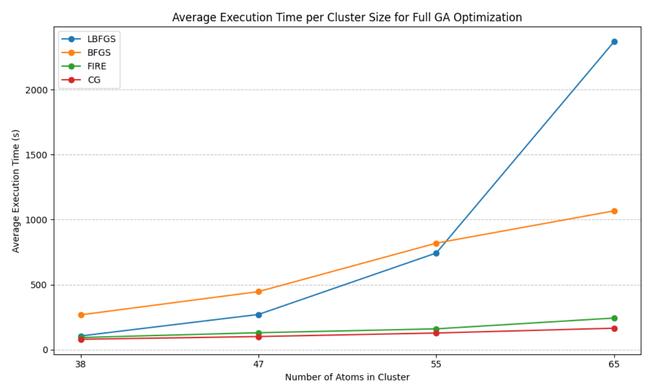

Figure 9: Average execution time per cluster size for a full GA optimization utilizing each of the methods in the integrated experiment.

The second integrated experiment tests reducing the execution time of single BFGS local optimization by reducing the maximum number of optimization steps. The maximum number of optimization steps is based on the quartile values from the first integrated experiment. Figure 10 represents the time reduction obtained by the different configurations. Even though the timing is indeed lower, it is still significantly higher than that of FIRE and CG and is still steeply scaling. Furthermore, Table 2 reports a significant reduction in the number of times the GM is found. Therefore, this time reduction scheme is deemed unfavorable.

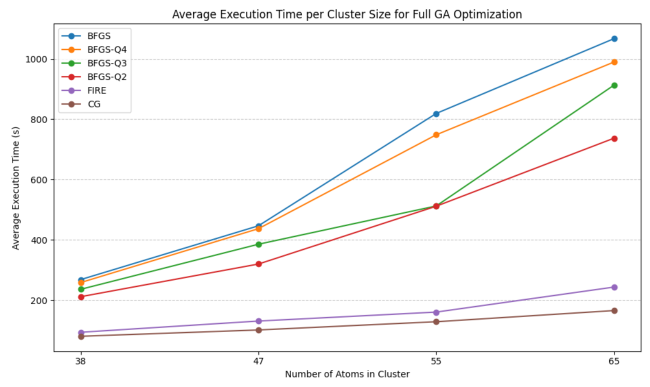

Figure 10: Average execution time per cluster size for a full GA optimization utilizing the reduced steps limits BFGS configurations.

A third integrated experiment is introduced that aims to effectively compare FIRE, CG and BFGS. An objective comparison of the different variations of the GA can be done by equating either the timing or the results. However, equal results cannot really be guaranteed, and the timing of BFGS cannot be reduced without compromising its results; therefore, the timing of FIRE and CG should be extended. To do so effectively, the population size is doubled, providing more diversity and thus a higher probability of finding GM. Figure 11 visualizes the average GA execution timing, proving that FIRE2 and CG2 are still faster than BFGS, even in its best timing configuration, BFGS-Q2. Table 2, on the other hand, reports that the double population versions manage to outperform even BFGS in GM discovery. Therefore, FIRE2 and CG2 surpass BFGS in efficiency since they are superior in terms of both execution timing and results energy.

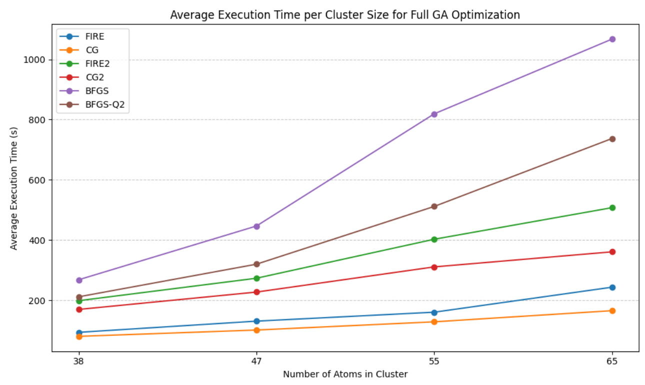

Figure 11: Average execution time per cluster size for a full GA optimization utilizing the doubled population FIRE and CG methods.

4.2 Heuristic Selection

A heuristic-based selection is proposed to halve the number of local optimizations, thus significantly reducing the total execution time. The default GA performs local optimization on all clusters and afterwards selects only the best half. If the energy after optimization can be predicted by a heuristic, then half of the local optimizations are completely useless. Cluster metrics such as potential energy or forces are not suitable heuristics since their values can be dominated by a single atomic pair interaction. Therefore, a heuristic that is independent of the interatomic distances is preferred. Zhao et al. [27] study cluster isomerism by computing the sum of the distances between each atom and the center of mass. An inspection of the isomerism values and the energy after optimization from the previous integrated experiments proves that there is a correlation; however, a single isomerism value maps to a wide range of energy results. Furthermore, a numerical test is performed to establish how accurate the heuristic is by measuring the percentage of clusters being selected by both the heuristic and the default methodology per each generation. Mean aggregates of this analysis result in a 70% accuracy for the heuristic, which means it is a decent candidate that is worth practically examining.

The isomerism heuristic for the GA selection practically halves the execution time at the expense of a reduction in GM discovery. To properly examine efficiency, additional tests are introduced that utilize FIRE and CG with quadrupled population size (32 candidate solutions), such that execution time is equated with the default GA FIRE2 and CG2 configurations. Figure 12 illustrates the time scaling of the default GA configurations and the double population heuristic selection GA. As observed, the timings of hFIRE4 and hCG4 are mainly lower than those of FIRE2 and CG2, respectively; however, they also scale at a lower gradient. Table 3 reports the number of times the GM is discovered by the different configurations of the heuristic selection GA. A comparison between identical configurations of local optimizer and population size proves a noticeable reduction in results. However, comparison between equal timing configurations proves that the heuristic has its own merits and produces more GM in total.

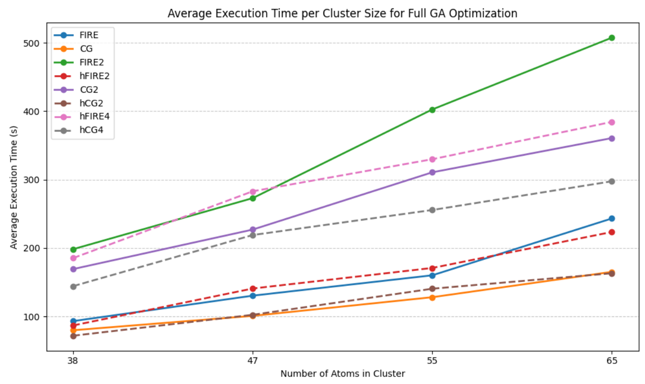

Figure 12: Comparison of average execution time per cluster size for a full global optimization by default and heuristic selection GA.

Table 3: Number of times the global minimum (GM) is found by the heuristic selection GA per method per cluster size.

	LJ38	LJ47	LJ55	LJ65
BFGS	1	5	0	0
FIRE	0	3	1	0
CG	0	0	1	0
FIRE2	1	7	2	1
CG2	0	7	2	0
FIRE4	3	12	2	0
CG4	3	12	5	0

An assessment of the merits of heuristic selection combined with doubled population size is performed. LJ47 is a test case that produces a significant number of discovered GM runs in the default GA. In the heuristic GA, those results are fairly matched in the double population size instances. LJ55, on the other hand, is much harder for GM discovery; however, population increase in the default GA manages to produce more stable results. In the case of the heuristic GA, those results are matched even with the same population size, while CG4 produces a noticeable breakthrough. The largest cluster considered, LJ65, proves to be the hardest test case, with a single stable GM discovery produced by CG2 default GA. Finally, LJ38, the smallest cluster considered, proves to be as challenging as LJ65, since the default GA does not manage to produce a configuration that discovers its GM more than once. The heuristic GA matches those results; however, with FIRE4 and CG4, it performs another breakthrough by discovering the GM 3 times each.

A final experiment is performed that tests all viable configurations of the default and heuristic GA for 50 trials on LJ31. The smaller cluster size is chosen such that more trials can be run as well as the expectation of more GM discoveries that provide a basis for clear comparisons. Table 4 reports the number of GM discoveries for each configuration. For this test case, it can be observed that FIRE, even with the same population size, produces significantly better results by utilizing the heuristic GA. The same, however, is not true for CG. Nonetheless, a comparison between time-equated configurations, meaning a default GA and the corresponding doubled population heuristic GA, shows that the heuristic GA produces significantly better results, thus proving higher efficiency.

Table 4: Number of times the global minimum (GM) is found by the default and heuristic selection GA per method for LJ31.

	default	heuristic
BFGS	7	6
FIRE	1	3
CG	1	1
FIRE2	3	6
CG2	5	2
FIRE4	-	13
CG4	-	9

5 Responsible Research

The highest measures of experiment replicability and result reproducibility have been ensured. First, the full codebase of the project can be found online. Furthermore, on top of the modified GA implementation, it also contains the code for the experiments as well as any auxiliary methods developed for storing, aggregating and visualizing results. Therefore, anyone has the means to replicate the experiments. On the other hand, to ensure reproducibility, all of the experiments utilize randomness generation seeding. Furthermore, to ensure that the results of the different local optimization methods are comparable, they are tested using the same cluster atomic configurations in the independent experiments and the same randomness sequence in the full GA executions. For reproducibility of independent experiments, the randomness sequence is seeded once at the beginning; thus, all of the generated cluster configurations can be derived again. It should be noted that reusing the ASE Atoms objects should be avoided at all costs, since they are dynamically updated. Thus, once a valid atomic configuration has been sampled from the randomness sequence, a different ASE Atoms object is separately generated for each local optimization. The GA implementation, on the other hand, runs every full global optimization by first seeding the randomness sequence using either the default or user-input value. Thus, integrated experiments are performed by using the same set of randomness seeds for the different cluster sizes and tested configurations. Lastly, in order to mitigate execution time variance due to machine state, experiments are executed such that different local optimizers are used in consecutive runs, aiming at equal spread of that variation.

GGO-based methods are undeniably instrumental in enabling discoveries across a spectrum of scientific and engineering fields. These include, but are not limited to, nanomaterials, biological macromolecules, pharmacological agents, radiation shielding systems, and photonic devices. The dual-use nature of many of these applications, however, necessitates careful ethical consideration. Tools that can optimize for catalytic efficiency or structural stability can, with minor adjustments, also be applied toward the design of stealth technologies or synthetic toxins [23]. It is therefore imperative that research in this domain proceed with transparency, ethical foresight, and collaboration across disciplines to ensure beneficial and responsible technological development.

6 Discussion

The experimental results challenge some common assumptions about local optimizers in LJ cluster GAs. Notably, L-BFGS (Limited-memory BFGS) has long been the de facto industry standard, yet findings show it is suboptimal in this context. Although L-BFGS is memory-efficient and widely used, it achieves neither the lowest energies nor the fastest convergence in these tests. In fact, L-BFGS is the worst compared to the other viable optimizers (BFGS, FIRE, CG) on both solution quality and speed as cluster size increases. This suggests that the conventional choice of L-BFGS may need reconsideration, especially when better scaling alternatives are available.

BFGS (Broyden–Fletcher–Goldfarb–Shanno) emerges as the most powerful optimizer for energy minimization. In independent benchmarks, BFGS consistently finds the lowest-energy cluster configurations. However, this accuracy comes at a high computational cost. BFGS has the longest per-run execution time, and its scaling with cluster size is steep. It is attempted to mitigate these disadvantages by limiting the number of optimization steps, but the time–energy trade-off remains unfavorable. In practice, pure BFGS optimizations make the GA evaluations very thorough but also very slow. Thus, while BFGS is the default optimizer in many GA implementations as well as this one, its scaling issues are a major drawback: achieving more descent per step incurs diminishing returns in an evolutionary run.

By contrast, the FIRE (Fast Inertia Relaxation Engine) and Conjugate Gradient (CG) optimizers are much more lightweight. Individually, FIRE and CG converge to decent minima much faster than BFGS. When the GA population is doubled to use the extra time, GAs utilizing FIRE or CG actually outperform the standard BFGS-based GA in terms of wall-clock time and final energy. In other words, trading off a bit of per-optimization accuracy for speed allows exploring many more configurations, which yields better results overall. The GA with doubled population size and FIRE/CG local optimizer often finds the GM equal or more times than the default GA utilizing BFGS. This highlights that saved time can be reinvested productively. However, simply scaling the population is not necessarily the optimal or only way to do this: doubling population size increases selection overhead and may suffer from diminishing returns. More systematic study is needed to identify optimal population size; furthermore, adaptive population growth schemes should also be explored. But fundamentally, data emphasizes that faster optimizers like FIRE and CG have clear advantages when given adequate population diversity.

Concerning GA design and efficient speedup, two major approaches are considered – time reduction per local optimization or global reduction in the number of local optimizations. BFGS tests prove that limiting the maximum number of steps leads to minor execution time reduction per optimization; however, the results quality is significantly decreased. Thus, a heuristic-based selection strategy is implemented and evaluated. It effectively reduces the number of local optimizations in half each generation by optimizing only the cluster selected based on their isomerism value. This heuristic is tested mainly for FIRE/CG-based GAs. Its effect is consistent: it roughly halves the total GA's execution time; however, it also introduces a drop in solution quality. Nonetheless, when the total compute time is equalized by doubling the population size, the heuristic-augmented GA outperforms the default GA. These results indicate that even though the heuristic is prone to choosing suboptimal clusters for optimization, an increased population size and thus diversity still preserve the ability to discover good results. On the other hand, it should be mentioned that isomerism value is a cheap heuristic; however, different versions of it can be derived if the distance values are scaled differently, for example, squared. This means that, based on this principle, multiple different ways to calculate the isomerism heuristic values can be derived, each selecting a different set of clusters. Nonetheless, the value of the heuristic selection is derived from the reduced execution time; however, its reinvestment strategy through doubling population size might not be the most results-optimal solution.

7 Conclusions and Future Work

This work shows that the most effective local optimization strategy in GA-based GGO is one that balances execution speed against the depth of energy descent. In practice, the slower but powerful BFGS optimizer yields the best-minimized structures but at a very high computational cost. In contrast, the FIRE and CG optimizers are far more efficient in CPU time, and when given a larger population (to equalize effort) they achieved comparable or better outcomes than BFGS. The commonly used L-BFGS method proves relatively inefficient in these experiments, finding neither the best energies nor offering speed advantages (despite its prevalence). Finally, the cluster-isomerism heuristic (a selective local-search rule) substantially improves the trade-off: it roughly halves runtime with not so significant quality loss and outperforms the unmodified GA when time is equalized. Taken together, these findings indicate that employing faster local optimizers (and possibly even suboptimal ones) in combination with increased population diversity can surpass conventional BFGS-heavy strategies. They also confirm that the heuristic GA concept is robust across different optimizer configurations.

For future work, several directions are promising. Gained execution time reinvestment strategies should be carefully studied. It is believed that increasing the population size is the best option since the timing is predictable and the population diversity is expanded, improving the chances for finding the GM. Nonetheless, a diligent study should explore the effect of the convergence criterion and the consequences of relaxing it. Furthermore, population size can be examined in depth, resulting in the identification of optimal values. On top of that, it is believed that population diversity is more important than pure population size; thus, isomerism can be used to develop adaptive population growth schemes. Alternatively, dynamic heuristic strategies can be developed that utilize machine learning or adaptive methods to decide in real time which clusters to fully optimize. Concerning heuristic isomerism values, the different ways of calculating it can be further studied to explore how that changes the final results. Overall, future work should focus on incorporating efficient heuristics and self-adaptability strategies, potentially by utilizing isomerism metrics.

References

[1] Mohammed Balfakeih, Ivo Blok, Juan Breton de la Cierva, Martin Damyanov, Aitor Bilbao Pardo, and Kaloyan Yanchev. Parallel library for global geometry optimization. https://github.com/Martin092/CSE_Project, January 2025. Accessed: 2025-05-30.

[2] Erik Bitzek, Pekka Koskinen, Franz Gähler, Michael Moseler, and Peter Gumbsch. Structural relaxation made simple. *Phys. Rev. Lett.*, 97:170201, October 2006.

[3] David M. Deaven and Kai-Ming Ho. Molecular geometry optimization with a genetic algorithm. *Phys. Rev. Lett.*, 75:288–291, July 1995.

[4] Jonathan P. K. Doye. Lennard-jones clusters structures. https://doye.chem.ox.ac.uk/jon/structures/LJ.html, 1997. Accessed: 2025-05-30.

[5] Jonathan P. K. Doye, Mark A. Miller, and David J. Wales. Evolution of the potential energy surface with size for lennard-jones clusters. *The Journal of Chemical Physics*, 111(18):8417–8428, November 1999.

[6] Sven Heiles and Roy L. Johnston. Global optimization of clusters using electronic structure methods. *International Journal of Quantum Chemistry*, 113(18):2091–2109, 2013.

[7] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, 1992.

[8] Roy L. Johnston and Christopher Roberts. Genetic algorithms for the geometry optimization of clusters and nanoparticles. In *Soft Computing Approaches in Chemistry*, pages 161–204, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[9] Jagyeong Kim, Kiwook Han, and Jae W. Hahn. Selective dual-band metamaterial perfect absorber for infrared stealth technology. *Scientific Reports*, 7(1):6740, Jul 2017.

[10] Jun Hee Kim, Thang Viet Pham, Jae Hun Hwang, Cheol Sang Kim, and Myung Jong Kim. Boron nitride nanotubes: synthesis and applications. *Nano Convergence*, 5(1):17, June 2018.

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

[12] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(1–3):503–528, August 1989.

[13] Marco Locatelli and Fabio Schoen. Local search based heuristics for global optimization: Atomic clusters and beyond. *European Journal of Operational Research*, 222(1):1–9, 2012.

[14] Alexandre Mayer, Hai Bi, Sarah Griesse-Nascimento, Benoit Hackens, Jérome Loicq, Eric Mazur, Olivier Deparis, and Michaël Lobet. Genetic-algorithm-aided ultra-broadband perfect absorbers using plasmonic metamaterials. *Optics Express*, 30(2):1167, January 2022.

[15] Amtul Nashim and Kulamani Parida. A glimpse on the plethora of applications of prodigious material mxene. *Sustainable Materials and Technologies*, 32:e00439, July 2022.

[16] Mark T. Oakley, David J. Wales, and Roy L. Johnston. Energy landscape and global optimization for a frustrated model protein. *The Journal of Physical Chemistry B*, 115(39):11525–11529, Oct 2011.

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

[18] Francisco B. Pereira, Jorge M. C. Marques, Tiago Leitão, and Jorge Tavares. Designing efficient evolutionary algorithms for cluster optimization: A study on locality. In *Advances in Metaheuristics for Hard Optimization*, pages 223–250, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[19] Diogo Santos-Martins, Leonardo Solis-Vasquez, Andreas F. Tillack, Michel F. Sanner, Andreas Koch, and Stefano Forli. Accelerating autodock4 with gpus and gradient-based local search. *Journal of Chemical Theory and Computation*, 17(2):1060–1073, Feb 2021.

[20] 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, 2019.

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

[22] TimeStep89. Graph of the lennard–jones potential. https://commons.wikimedia.org/wiki/File:Graph_of_Lennard-Jones_potential.png, September 2020. Image hosted on Wikimedia Commons. Licensed under CC BY 4.0.

[23] Fabio Urbina, Filippa Lentzos, Cédric Invernizzi, and Sean Ekins. Dual use of artificial intelligence-powered drug discovery. *Nature Machine Intelligence*, 4, 03 2022.

[24] David J. Wales and Jonathan P. K. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, July 1997.

[25] L T Wille and J Vennik. Computational complexity of the ground-state determination of atomic clusters. *Journal of Physics A: Mathematical and General*, 18(8):419, June 1985.

[26] Matthew Wolf and Uzi Landman. Genetic algorithms for structural cluster optimization. *Journal of Physical Chemistry A*, 102:6129–6137, 1998.

[27] Jijun Zhao, Ruili Shi, Linwei Sai, Xiaoming Huang, and Yan Su. Comprehensive genetic algorithm for ab initio global optimisation of clusters. *Molecular Simulation*, 42:809 – 819, 2016.