Evolutionary and Gradient-Based Algorithms for Lennard-Jones Cluster Optimization

Sibylle D. Müller **Nicol N. Schraudolph** **Petros Koumoutsakos**
Institute of Computational Science
Swiss Federal Institute of Technology
8092 Zürich, Switzerland
muellers,nic,petros@inf.ethz.ch

Abstract

Finding the equilibrated configuration of atomic clusters modeled by the Lennard-Jones potential poses a challenging task to numerical optimization strategies as the number of local minima grows exponentially with the number of atoms in the cluster. We use this massively multimodal problem to test different evolutionary, deterministic and randomized gradient methods with respect to their global search behavior. The randomized gradient method was designed to combine the advantages of gradient and stochastic direct optimization.

1 Introduction

The Lennard-Jones (LJ) potential models van der Waals interactions between noble gas atoms. The LJ potential is given by

$$E = 4\epsilon \sum_{i=1}^{N-1} \sum_{j>i}^{N} \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \qquad (1)$$

where ϵ is the pair well depth and $2^{1/6}\sigma$ is the pair separation at equilibrium. r_{ij} is the distance between atoms i and j. Throughout this study, reduced units $\epsilon = 1$ and $\sigma = 1$ are used. Thus, the potential can be expressed as

$$E = 4 \sum_{i=1}^{N-1} \sum_{j>i}^{N} g_{ij} \qquad (2)$$

where

$$g_{ij} := \left[\left(\frac{1}{r_{ij}} \right)^{12} - \left(\frac{1}{r_{ij}} \right)^{6} \right]. \qquad (3)$$

The aim is to find a spatial distribution of atoms with minimal energy. The atomic coordinates r_{ij} are the optimization parameters and the number of atoms is N which results in $n = 3N$ parameters.

Determining the global minima of LJ clusters poses a challenging task for numerical optimization techniques as the number of local minima on the potential energy surface grows exponentially with the number of atoms N in the cluster [14]. For $N = 98$, there exist an estimated number of local minima in the order of 10^{40} [14]. Global optimization techniques to minimize the potential usually combine global and local search algorithms: a global search algorithm, e.g., genetic algorithm [1, 2, 18], simulated annealing [17], basin-hopping [15], or Monte Carlo suggests points from where a local optimization (e.g., conjugate gradient method) is started. With the local minima information, the search is continued on the global level and so on. One distinguishes between biased (incorporating problem-specific knowledge in initializing the search or in the search process) and unbiased search. An example for biased search is to define initial configurations according to structural motifs which are frequently found in atomic aggregates of the considered size. While biased search is found to converge faster, it more likely gets trapped in local optima. To find global minima, it is suggested to use an unbiased technique.

We employ unbiased search techniques, in particular (i) stochastic direct (= evolutionary), (ii) deterministic gradient (= conjugate gradient), and (iii) randomized gradient (= asynchronous conjugate gradient) optimization techniques. Their use is motivated as follows:

1. Since the search space is multimodal, stochastic methods may be beneficial for the search for the global optimum.

2. Since the gradient is available analytically, it should be used to enhance the speed to find the global optimum.

3. The combination of randomness and usage of gradient information appears appealing as it might amplify both advantageous effects.

We compare the different algorithms with each other and with the putative global minima reported in [16, 9] of configurations with up to $N = 309$ atoms. It is not our primary goal to outperform these reported global optima, but rather to assess the differences between the selected search strategies.

In Section 2, we present evolutionary algorithms in comparison to the deterministic conjugate gradient method, while Section 3 shows how the asynchronous conjugate gradient method compares with the deterministic gradient technique. The work is concluded in Section 4.

2 Optimization Using Evolutionary Algorithms

2.1 Chosen Evolutionary Methods

Our evolutionary test bed consists of evolution strategies (ES), namely the (1+1)-ES with 1/5 success rule [8, 13], the evolution strategy with covariance matrix adaptation CMA-ES [3, 4], and its extension for a larger population size, here called parallel CMA-ES (pCMA-ES) [6, 5]. As one of the first developed evolution schemes, the (1+1)-ES with 1/5 success rule is relatively easy to implement and has been investigated thoroughly from a theoretical viewpoint. The CMA-ES has been found to converge much faster for nonseparable and badly scaled functions than evolution strategies with other adaptation schemes [4]. Its extension to the pCMA-ES makes it possible to vary population sizes in a larger range than in the CMA-ES, which is especially favorable if the population can be chosen large, e.g., on massively parallel computers, which results in a reduction of time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ [6, 5].

We compare these evolution strategies with the deterministic conjugate gradient technique (CG) as described in the subroutine called `frprmn` in Numerical Recipes [7], p. 417.

The initial cluster is designed as a cubic lattice in which each edge contains $N^{1/3}$ ($N = 8, 27$) atoms. The optimization of this cluster by CG results in a suboptimal structure, that is, the distance between atoms takes up its equilibrium value, but the cubic lattice structure remains. To allow the search for a global optimum, the atoms are randomly perturbed. This perturbed cluster is the initial configuration for optimization using the four methods above. The search is terminated as soon as the difference between the current function value and the global optimum is smaller than 0.0001 or the difference between objective functions from one generation to the next is less than 0.0001 in 50 subsequent generations. The performance of evolution strategies is measured by averaging the results of 100 and 20 runs for $N = 8$ and $N = 27$, respectively, while for CG one run is sufficient. In all runs, the initial cluster configuration is the same. Runs differ only in the seed of the random number generator. The performance of ESs can be measured by the success rate. For $N = 8$, the success rate is defined as the ratio of the number of runs that yielded the global optimum and the total number of performed runs (= 100). Because the global optimum was never reached for $N = 27$, we define the success rate in this case as the ratio of the number of runs that yielded better results than CG and the total number of performed runs (= 20).

2.2 Results

Both optima found by the different strategies and success rates are reported in Table 1.

Table 1: Global optima [16], conjugate gradient optima, and success rates in % for the (1+1)-ES, the $(2_I, 10)$-CMA-ES, and the $(16_I, 64)$-pCMA, respectively, for the optimization of clusters with sizes $N = 8$ and $N = 27$. For $N = 8$, the success rate is the ratio of the number of runs that yielded the global optimum and the total number of performed runs (= 100). For $N = 27$, the success rate is the ratio of the number of runs that yielded better results than CG and the total number of performed runs (= 20).

N	Global opt. [16]	CG	Success rates [%] for (1+1),	CMA,	pCMA
8	-19.82149	-18.97606	50	43	39
27	-112.87358	-106.1820	60	85	80

The different ESs perform similarly for the $N = 8$ cluster yielding the global minimum in almost half of the cases on average. Note that the success rates depend on the initial configuration of the cluster. For $N = 27$, the success rates are measured relative to the performance of conjugate gradient. Again, a different initial configuration is likely to yield different numbers for both CG and ES. The fact that the global optimum in this configuration is not found by any ES run is not surprising given that the function is highly multimodal and only a limited number of optimization runs with a limited number of function evaluations could be afforded.

An interesting result occured for the pCMA-ES using $N = 27$ atoms. Here, different population sizes ($\mu = 16, \lambda = 64$) and ($\mu = 27, \lambda = 104$) were tested. While the $(16_{\mathrm{I}}, 64)$ strategy yielded a success rate of 80%, the $(27_{\mathrm{I}}, 104)$ method had difficulties to converge to any local optimum. The undesirable behavior for larger populations may be due to the multimodality of the function. Originally designed for local optimization, the pCMA-ES is expected to adapt the mutation distribution the better, the larger the population. However, we believe that in multimodal function landscapes the mutation distribution is disturbed by information from several local minima. This effect may be increased as the population size increases. Instead of exploiting the information about one optimum, the pCMA-ES with large populations acquires mutation distributions from multiple optima that are useless for local optimization. This assumption needs to be validated in future studies.

CPU times for one run are approximately 1 s for $N = 8$ for all strategies, and for $N = 27$ they vary between 1 s (CG) and $2 - 3$ min for $(16_{\mathrm{I}}, 64)$-pCMA-ES. The computational effort for the eigenvalue/eigenvector analysis in the CMA-ES increases with $\mathcal{O}(n^3)$ [7]. Since one function evaluation is cheap compared with the solution of the eigensystem, one has to expect high computational cost for optimizing large clusters. For the same reason, analyses for clusters with $N > 27$ are omitted. In general, our results are not conclusive as to whether ES or CG is the better search method.

3 Optimization Using the Asynchronous Conjugate Gradient Method

3.1 Asynchronous Conjugate Gradient Method

In a next step, we combine two features of optimization techniques, namely stochasticity and usage of gradient information, in the hope that this is beneficial for the LJ cluster optimization which is characterized by multimodality and availability of the gradient.

The asynchronous conjugate gradient method is a partially randomized conjugate gradient method. We apply it to the LJ optimization because we hope that by introducing stochasticity in the optimization algorithm its performance is increased for optimizing large clusters.

The move from deterministic to randomized conjugate gradient is comparable to the move from deterministic to stochastic direct optimization methods (e.g., ES): In both cases, it is intended to obtain an increase in robustness, e.g., against getting trapped in local optima etc.

A difference between randomized gradient and stochastic direct techniques can be seen in the following respect: In randomized gradient algorithms, the stochasticity is introduced in the *order* in which subproblems of the objective function are optimized; in stochastic direct methods, e.g., ES, the stochasticity is inserted (usually) in the mutation step.

To use the asynchronous conjugate gradient algorithm, the objective function needs to be reformulated. Therefore, we write the potential for LJ cluster optimization as

$$E_{aCG} = \underbrace{4 \sum_{j=1|j\neq 1}^{N} g_{1j}}_{=:f_1} + \underbrace{4 \sum_{j=1|j\neq 2}^{N} g_{2j}}_{=:f_2} + \ldots + \underbrace{4 \sum_{j=1|j\neq N}^{N} g_{Nj}}_{=:f_N} \tag{4}$$

where g_{ij} as in Equation 3, and f_1 to f_N are subproblems or batches.

The algorithm of aCG reads as follows:

```
DO until termination
    $s$ = permute $(1, \ldots, N)$
    DO i = 1,N
        Minimize $f_{s(i)}$ using CG.
    ENDDO
ENDDO
```

Therefore, the asynchronous conjugate gradient algorithm is characterized by minimizing subproblems f_{s_i} instead of minimizing the potential E. This means, that in aCG the atom positions are incrementally (= asynchronously) updated and so is the potential.

The relationship of the potentials can be expressed as $E_{aCG} = 2E$. This can be seen when Equation 2 is rewritten as follows:

$$E = 4\sum_{j>1}^{N} g_{1j} + 4\sum_{j>2}^{N} g_{2j} + \ldots + 4 \sum_{j>N-1}^{N} g_{(N-1)j}. \tag{5}$$

However, the factor of 2 effectuates just a scaling of the potential and therefore does not affect the optimization.

Note also that for minimizing $f_{s(i)}$, we compute the search direction and perform a line search in that direction (as in deterministic CG). However, the difference to deterministic CG is that in aCG the gradient of $f_{s(i)}$ needs to be computed (and not the gradient of the potential E as in deterministic CG).

The asynchronous conjugate gradient method (aCG) is not to be confused with the more familiar stochastic approximation of gradients, which has recently been applied to the conjugate gradient method [10, 11, 12] although both types of algorithms employ stochasticity. While *stochastic conjugate gradient* (sCG) involves two levels of randomization, aCG is characterized by one level of randomization for Lennard-Jones optimization. In this sense, asynchronous CG can be seen as intermediate step between deterministic CG and stochastic CG. In our application, the first level of randomization in aCG is introduced by perturbing the order of the indexes. In sCG, a second level of randomization is involved: Besides perturbing the atom indexes, sCG would approximate f_1 to f_N by taking into account the interactions of one atom to a randomly selected *subset* of atoms (whereas aCG takes into account the interactions of one atom to *all* other $N-1$ atoms). A more detailed description of sCG can be found in [10, 11].

3.2 Results

We compare the deterministic conjugate gradient method (CG) with the asynchronous conjugate gradient method (aCG).

All cluster configurations start from a cubic lattice in which each edge of length 1 contains $N^{1/3}$ ($N = 8, 27, 64, 81, 125, 216, 343$) atoms. To avoid symmetry, the atoms are randomly perturbed by adding to the coordinate values of the atoms uniformly random numbers multiplied with 0.01. With this configuration, we perform one CG and one aCG run. With a new random initialization, we produce a new initial configuration, and perform another CG and aCG run, etc., until we obtain statistics for 100 runs for both optimization methods.

Our statistics, summarized in Tables 2 and 3, include for both methods:

- the number of iterations to reach the goal, averaged over 100 runs,
- the lowest number of iterations out of 100 runs,
- the optimum value, averaged over 100 runs,
- the lowest optimum value out of 100 runs,
- the CPU time per run, averaged over 100 runs,
- the success rate, that is, the ratio of runs yielding the global optimum to the total number of runs (=100),
- the number of failures (= crashes), that is, the ratio of number of runs not converging to any optimum to the total number of runs (the results of failed runs are not included in the statistics).

The conjugate gradient subroutine of [7], p. 417, called `frprmn`, contains termination criteria that are determined by the following parameter settings: convergence tolerance ftol $= 10^{-20}$ for both CG and aCG; small number EPS $= 10^{-16}$ for both CG and aCG; maximum number of iterations (one iteration includes determining the search direction and performing line search) ITMAX $= 5000$ for CG and ITMAX $= 4$ for aCG. The difference in the setting of the maximum number of iterations originates from the fact that in aCG we are not interested in a fully converged solution but in an approximation of the solution (recall that the optimized subproblems f_{s_i} are also only approximations of the true potential).

The figures given in Tables 2 and 3 imply the following conclusions:

Number of failures. The number of failures is 0 for all configurations for aCG and in the range of 0-2 % for CG [1]. The rare occuring failures of CG could be due to symmetry in the configuration. Although the difference is small, aCG is less prone to failure than CG.

Cost. The cost of the algorithm is measured in terms of average number of iterations and CPU time. The number of iterations in aCG is always larger than in CG. While the number of iterations in CG is in the order of N, in aCG the average number of iterations becomes drastically larger with increasing configuration size. For a fair comparison, however, one should recall that one iteration includes determining the search direction and performing line search, which accordingly requires to compute several times the objective function and its derivative: While CG requires the potential E (with interactions between all atoms) and its derivative, aCG needs only a subset of the potential, f_{s_i} (with interactions between *one* atom and

[1] Please note that the numbers for CG reported here are obtained with a repeated call to the conjugate gradient subroutine `frprmn` [7]. Without this, the number of failures is slightly higher.

Table 2: Statistics over 100 runs for CG and aCG for clusters of $N = 8, 27, 64$ atoms.

$N = 8$	CG	aCG
Average no. iterations	32	982
Lowest no. iterations	21	320
Average optimum value	-19.34319	-19.33511
Lowest optimum value	-19.82149	-19.82149
Global optimum [16]	-19.82149	-19.82149
CPU per run [s]	0.00361	0.08111
Success rate [%]	26	35
Number of failures [%]	2	0
$N = 27$	CG	aCG
Average no. iterations	95	16534
Lowest no. iterations	52	5076
Average optimum value	-107.0278	-107.0899
Lowest optimum value	-111.8644	-111.2919
Global optimum [16]	-112.8736	-112.8736
CPU per run [s]	0.12958	2.35984
Success rate [%]	0	0
Number of failures [%]	0	0
$N = 64$	CG	aCG
Average no. iterations	158	87116
Lowest no. iterations	61	16896
Average optimum value	-310.3006	-310.8228
Lowest optimum value	-316.0594	-319.1035
Global optimum [16]	-329.6201	-329.6201
CPU per run [s]	0.91769	17.8794
Success rate [%]	0	0
Number of failures [%]	1	0

Table 3: Statistics over 100 runs for CG and aCG for clusters of $N = 125, 216, 343$ atoms.

$N = 125$	CG	aCG
Average no. iterations	201	228140
Lowest no. iterations	69	92500
Average optimum value	-677.9642	-676.5854
Lowest optimum value	-693.5336	-693.5863
Global optimum [16]	-721.3032	-721.3032
CPU per run [s]	5.88375	79.7644
Success rate [%]	0	0
Number of failures [%]	0	0
$N = 216$	CG	aCG
Average no. iterations	242	523229
Lowest no. iterations	113	206496
Average optimum value	-1257.128	-1253.199
Lowest optimum value	-1288.924	-1274.222
Global optimum [9]	-1340.711	-1340.711
CPU per run [s]	21.33470	351.7713
Success rate [%]	0	0
Number of failures [%]	0	0
$N = 343$	CG	aCG
Average no. iterations	283	949602
Lowest no. iterations	121	489804
Average optimum value	-2089.456	-2085.439
Lowest optimum value	-2160.145	-2132.781
Global optimum	not known	not known
CPU per run [s]	49.05392	1013.271
Success rate [%]	-	-
Number of failures [%]	0	0

all other neighbors) and its derivative — thus, one iteration in aCG is clearly computationally less expensive than in CG. This difference is considered in the measurement of the average CPU time per run. Here, it can be seen, that aCG is by a factor of 14-21 slower than CG, *independent* of the configuration size. That means, the optimization time grows in N for both CG and aCG, or in other words, the two algorithms have the same time complexity. In this respect, CG and aCG perform equally well.

Performance. The performance can be measured in terms of the success rate and the average and lowest optimum values. The success rate is larger than 0 only for the $N = 8$ configuration. For $N = 8$, aCG with a success rate of 35 % performs better than CG with a success rate of 26 %. For larger configurations, neither CG nor aCG is able to find the global optimum in 100 runs. This is not surprising given that the number of local optima rises exponentially with N. Therefore, to assess the performance of both algorithms for $N > 8$, we compare their average and lowest optimum values. The average optimum found by aCG is better than CG for $N = 27, 64$ and worse for $N = 125, 216, 343$, while the lowest optimum found by aCG is better than CG for $N = 64, 125$ and worse for $N = 27, 216, 343$. These figures are not conclusive as to whether CG or aCG performs better.

In summary, the deterministic and asynchronous conjugate gradient algorithms yield similar results. The time complexity of both algorithms is the same but aCG is slower than CG. The goal that aCG would outperform CG for larger clusters was not accomplished, although aCG achieved a small gain in robustness for small clusters.

4 Conclusions

Lennard-Jones clusters were optimized using (i) stochastic direct (= evolutionary), (ii) deterministic gradient (= conjugate gradient), and (iii) randomized gradient (= asynchronous conjugate gradient) search algorithms. Each strategy shows advantages and disadvantages with respect to run time or performance, but the differences are so marginal that we cannot conclude which of these methods is best for this application. It remains to future investigations if a hybridization (= combination of global and local search in a separate/sequential way) of different search paradigms shows a better performance than our approach to combine stochasticity and gradient usage in a single search method.

Acknowledgments

The authors wish to thank Dr. Jens H. Walther, Institute of Computational Science, ETH Zürich, for many helpful discussions.

References

[1] Barrón, C., Gómez, S., Romero, D., Saavedra, A., "A Genetic Algorithm for Lennard-Jones Atomic Clusters," *Applied Mathematics Letters*, Vol. 12, pp. 85-90, 1999.

[2] Deaven, D.M., Tit, N., Morris, J.R., Ho, K.M., "Structural Optimization of Lennard-Jones Clusters by a Genetic Algorithm," *Chemical Physics Letters*, Vol. 256, pp. 195-200, 1996.

[3] Hansen, N., Ostermeier, A., "Convergence Properties of Evolution Strategies with the Derandomized Covariance Matrix Adaptation: The $(\mu/\mu_I, \lambda)$-CMA-ES," *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT'97)*, pp. 650-654, 1997.

[4] Hansen, N., Ostermeier, A., "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, Vol. 9, No. 2, pp. 159-195, 2001.

[5] Hansen, N., Müller, S.D., Koumoutsakos, P., "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)," *Evolutionary Computation Journal*, MIT Press, Vol. 11, No. 1, pp. 1-18, 2003.

[6] Müller, S.D., Hansen, N., Koumoutsakos, P., "Improving the Serial and Parallel Performance of the CMA-Evolution Strategy with Large Populations," *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)*, Lecture Notes in Computer Science, Vol. 2439, Springer, Berlin, pp. 422-431, 2002.

[7] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., "Numerical Recipes in Fortran 77: The Art of Scientific Computing (Vol. 1 of Fortran Numerical Recipes)," Cambridge University Press, 1996.

[8] Rechenberg, I., "Evolutionsstrategie '94," Fromann-Holzboog, Stuttgart, 1994.

[9] Romero, D., Barron, C., and Gomez, S., "The optimal geometry of Lennard-Jones clusters: 148-309," *Computer Physics Communications*, Vol. 123, No. 1-3, pp. 87-96, 1999.

[10] Schraudolph, N.N., Graepel, T., "Conjugate Directions for Stochastic Gradient Descent, *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2002)*, pp. 1351-1358, 2002.

[11] Schraudolph, N.N., Graepel, T., "Towards Stochastic Conjugate Gradient Methods," *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP '02)*, Singapore, 2002.

[12] Schraudolph, N.N., Graepel, T., "Combining Conjugate Direction Methods with Stochastic Approximation of Gradients," *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, Key West, FL, pp. 7-13, 2002.

[13] Schwefel, H.-P., "Evolution and Optimum Seeking," John Wiley and Sons, New York, 1995.

[14] Stillinger, F.H., "Exponential Multiplicity of Inherent Structures," *Physical Review E*, Vol. 59, pp. 48-51, 1999.

[15] Wales, D.J., Doye, J.P.K., "Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms," *Journal of Physical Chemistry A*, Vol. 101, pp. 5111-5116, 1997.

[16] Wales, D.J., Doye, J.P.K., Dullweber, A., Naumkin, F.Y., "The Cambridge Cluster Database," URL http://brian.ch.cam.ac.uk/CCD.html.

[17] Wille, L.T., "Minimum-Energy Configurations of Atomic Clusters: New Results Obtained by Simulated Annealing," *Chemical Physics Letters*, Vol. 133, No. 5, pp. 405-410, 1987.

[18] Wolf, M.D., Landman, U., "Genetic Algorithms for Structural Cluster Optimization," *Journal of Physical Chemistry A*, Vol. 102, pp. 6129-6137, 1998.