Global geometry optimization of atomic clusters using a modified genetic algorithm in spacefixed coordinates

J. A. Niesse and Howard R. Mayne

Citation: J. Chem. Phys. **105**, 4700 (1996); doi: 10.1063/1.472311
View online: http://dx.doi.org/10.1063/1.472311
View Table of Contents: http://jcp.aip.org/resource/1/JCPSA6/v105/i11
Published by the American Institute of Physics.

Global geometry optimization of atomic clusters using a modified genetic algorithm in space-fixed coordinates

J. A. Niesse and Howard R. Mayne
Department of Chemistry, University of New Hampshire, Durham, New Hampshire 03824

(Received 6 May 1996; accepted 10 June 1996)

In a recent paper, Gregurick, Alexander, and Hartke [S. K. Gregurick, M. H. Alexander, and B. Hartke, J. Chem. Phys. **104**, 2684 (1996)] proposed a global geometry optimization technique using a modified Genetic Algorithm approach for clusters. They refer to their technique as a deterministic/stochastic genetic algorithm (DS-GA). In this technique, the stochastic part is a traditional GA, with the manipulations being carried out on binary-coded internal coordinates (atom–atom distances). The deterministic aspect of their method is the inclusion of a coarse gradient descent calculation on each geometry. This step avoids spending a large amount of computer time searching parts of the configuration space which correspond to high-energy geometries. Their tests of the technique show it is vastly more efficient than searches without this local minimization. They report geometries for clusters of up to $n=29$ Ar atoms, and find that their computer time scales as $O(n^{4.5})$. In this work, we have recast the genetic algorithm optimization in space-fixed Cartesian coordinates, which scale much more favorably than internal coordinates for large clusters. We introduce genetic operators suited for real (base-10) variables. We find convergence for clusters up to $n=55$. Furthermore, our algorithm scales as $O(n^{3.3})$. It is concluded that genetic algorithm optimization in nonseparable real variables is not only viable, but numerically superior to that in internal candidates for atomic cluster calculations. Furthermore, no special choice of variable need be made for different cluster types; real Cartesian variables are readily portable, and can be used for atomic and molecular clusters with no extra effort. © *1996 American Institute of Physics.* [S0021-9606(96)00435-7]

I. INTRODUCTION

Atomic microclusters[1–7] are composed of two to a few hundred atoms. They may be held together by either bonding or nonbonding interactions. Buckminsterfullerene (C_{60}) is an example of the former; inert gas clusters the latter. The study of the geometry of these species presents a formidable theoretical challenge. This is due to the large number of local minima such clusters can possess. Since the most stable geometry is presumably the global minimum it is usually the object of interest. However, the number of candidate minima increases rapidly with the number of atoms in the system. Even a cluster of only 13 atoms possesses on the order of 10^3 local minima. It is estimated[1,4] that the number grows as rapidly as $\exp(n^2)$. Clearly an exhaustive search in all dimensions is not feasible. It is also extremely improbable that a random walk in the configuration space is likely to find the global minimum. Therefore, several strategies for guiding the search have been devised.

As mentioned above, one difficulty in locating the global minimum of a cluster is the presence of a vast number of local minima. Therefore, one goal of a minimization search for such systems is to efficiently escape from local minima. For the search to be successful, this must be to a minimum of lower potential energy. Techniques which use gradient information to slide downhill typically locate the closest local minimum (in the sense that the minimum acts as an attractor, and all points within its basin are attracted to it). Such methods, which move to a nearby local minimum, are sometimes termed ''greedy''; they solve the local minimization problem at the expense of the global solution.

There have been several recent attempts to introduce methods which allow escape from these local minima. The popular method of simulated annealing (SA)[8,9] has enjoyed success in this arena. The technique is an extension of Metropolis Monte Carlo techniques,[10] in which the system point is allowed to wander over the configuration space at a given temperature. In SA, the temperature is initially high, then is slowly reduced to absolute zero. At the end of the cooling, if the cooling is done infinitely slowly, the system will be in the global minimum. In practice, however, infinitely slow cooling in not realizable. The presence of high barriers at the saddle points between minima may, at low temperatures, make the transition between minima extremely slow. In other words, the system may be kinetically, rather than thermodynamically controlled.

Another strategy has been to allow the system to behave quantum mechanically, leading to the possibility of tunneling. This has been accomplished by using Gaussian wave packets in imaginary time,[11] by using distributed Gaussians,[12] and by combining simulated annealing and quantum Monte Carlo.[13] One problem with such methods is that they rapidly become increasingly difficult to implement as the number of dimensions increases.

Recently, there has been considerable interest in a stochastic minimization global technique which requires no such quantum formulation, and is undeterred by barriers. This is the method of genetic algorithms.[14–17] The genetic algorithm (GA) method is inspired by concepts from Darwinian natural evolution. Populations of candidate solutions compete with each other for survival. Through selection,

breeding, and mutation operations, the fittest individuals pass their genetic characteristics on to later generations. In this way, it is hoped that the ultimate surviving individual (in this case, the cluster geometry) is the fittest possible; that is, the best solution to the optimization problem posed (in this case, the global minimum).

One advantage of the GA method is that it is not ''greedy.'' The genetic operators often create children whose structures differ drastically from their parents. Thus, these operators allow the system to escape from local minima, since they do not attempt to ''creep'' uphill. In this way, there can be extensive searching of the configuration space, with the search appropriately guided by the fitness function.

There have been several recent applications of the GA technique in the chemical literature.[18–31] Many papers[21,25–28,30,31] have dealt with obtaining the equilibrium geometry (that is, the minimum potential energy) of a molecule or cluster. In such cases, the ''fitness'' of a geometry is some function of its potential energy, with low potentials having higher fitness. Deaven and Ho[26] were able to locate the global minimum for C_{60} using GA, although SA was unable to do so. Hartke has applied the method to the Si_4 molecule[21] and the Si_{10} cluster[27] on a semiempirical potential energy surface.[33] Zeiri[25] has used the technique to confirm published[32] geometries of Ar_nH_2 clusters. Mestres and Scuseria[28] have applied the technique to C_8 molecules and Ar_{13} Lennard-Jones clusters.

Very recently, Gregurick *et al.*[31] used a modified GA approach to minimize clusters as large as $n=29$. They call their method DS-GA, for deterministic-stochastic genetic algorithm. This approach incorporates an important innovation. By the addition of a gradient-based local descent for each geometry generated, these authors avoid searching parts of the configuration space which are repulsive. Points on the potential energy surface which happen to be very high in energy (therefore unfit) can, in fact, be ''close'' (in the sense that they are moved there by a gradient-based descent) to geometries which have low potential energy (and are therefore fit). In essence, the authors restricted the individuals in their populations to be the geometries of local minima. Thus, the search becomes a search through a finite (albeit large) number of individuals, rather than over an infinite set of possibilities.

There has been some discussion on the best choice of coding to use in genetic algorithm optimizations. Most of the GA-based approaches to cluster geometry optimization have used the ''standard GA'' to search the geometry space. In the standard GA, variables are coded as binary (base-2) bit strings, and the operations are carried out on these strings. Goldberg[14,17] demonstrates that the ''alphabet'' used should be of as low a ''cardinality'' as possible. That is, the number of possible characters used to convey genetic information should be as small as possible; clearly the binary system fulfills this requirement best.

However, as the number of variables in a problem becomes large, the cost associated with the low cardinality of the binary alphabet may become prohibitive.[16] For instance, suppose the problem depends on N real variables. Roughly speaking, for a worst case scenario, an exhaustive search through these variables requires on the order of N^2 operations. On the other hand, if each of these reals is translated into a binary number (say eight bits), there are now $(8N)$ digits to deal with, and the exhaustive search is now through a space of $(8N)^2$. Unless the search is much more efficient in the binary space, the penalty for binary coding for large N may be significant.

In considering structures such as clusters, there is another important consideration. That is, the choice must be made as to how best to represent the geometries. For instance, in an earlier paper, Hartke[21] used the ''standard GA'' to find the minimum geometry of Si_4. He used a coordinate system which was carefully chosen to be as separable as possible. (Here ''separable'' means, roughly, the ability of a coordinate to be approximately minimized independent of other coordinates.) In the language of GA, such coordinates describe isolated building blocks, which can be either well adapted or not well adapted, and are therefore, relatively clearly related to the fitness. In fact, Hartke states[21] ''Straightforwardly taking the Cartesian or internal coordinates... does not work'', and suggests that the coordinates must represent ''small building blocks.'' This is in accord with the ''principle of meaningful building blocks'' [34] central to GA theory.

However, as a multivariable problem increases in size, and the solution's dependence on the variables becomes increasingly complex, it becomes much more difficult to separate the variables. This, of course, is why normal modes are introduced into the discussion of vibrational motions of molecules; the normal modes are suitable combinations of interatomic distances which are separable for low-amplitude excursions from the global minimum. Unfortunately for the problem at hand, the calculation of such coordinates presupposes knowledge of the global minimum.

A further difficulty is that a separable representation especially chosen to be appropriate to any particular n-atom cluster is not easily generalized to other cluster sizes. Furthermore, even in the case of Si_4 mentioned above[21] the chosen coordinates spanned a restricted search space. It is clear from a later paper by the same author, that selection of such candidate coordinate systems for larger clusters is problematic.[27]

Zeiri, on the other hand, employs the real (base-10) Cartesian space-fixed (SF) coordinates for as many as fourteen atoms as the individuals in his nontraditional GA-based scheme.[25] In this work, the structure of H_2Ar_n clusters was obtained without using the local minimization approach proposed by Gregurick *et al.*[31] The approach of straightforwardly using SF coordinates is tempting. In an atomic cluster containing n atoms, for instance, there are $n(n-1)/2$ interatomic distances needed to describe the geometry. By comparison, there are always $3n$ space-fixed coordinates. Thus for $n>7$, fewer coordinates are needed to describe the cluster in SF coordinates than in internal coordinates. If such additional coordinates as dihedral angles are needed, the number of internal coordinates increases, whereas such coordinates can always be simply obtained from the SF data. For the

sake of evaluating the performance of the algorithm, the number of coordinates required in the SF system is $O(n)$, whereas that required in the internal coordinates is $O(n^2)$.

Clearly, then, for larger systems, the search space is smaller $[O(n^2)]$ for SF coordinates than it is for internal coordinates $[O(n^4)]$. However, the question now becomes that of the efficiency of the search procedure in a space where individual points in that space (e.g., the z coordinate of the ith atom) are not directly related to the potential energy function.

Zeiri[25] finds his results are at worst competitive with those obtained using simulated annealing. Furthermore, use of the Cartesian coordinates provided portability between cluster sizes and required no restriction of the search space. However, the representation in SF coordinates is plainly contrary to the spirit of Goldberg's ''building block hypothesis''; none of the coordinates stands alone as a meaningful building block.[14] Furthermore, many of the variables are interchangeable by symmetry. Can this representation be used efficiently with a Genetic Algorithm approach? While Zeiri has enjoyed success with it, there is no direct timing comparison with other GA approaches available for the system he has chosen. The purpose of this paper is to systematically explore the viability of using the SF coordinates, and to compare with benchmark calculations[31] on Ar_n clusters using more traditional coordinates.

Therefore, we investigate here the feasibility of using the SF (base-10 coded) coordinates in a GA-inspired optimization technique incorporating the coarse minimization proposal of Gregurick *et al.*[31] The computational procedures used are described in Sec. II. Results and Discussion are given in Sec. III, and our conclusions are presented in Sec. IV.

II. METHOD

For the cluster potential energy, we use a pairwise-additive Lennard-Jones potential:

$$V(\mathbf{r})=4\epsilon\sum_{i=1}^{n}\sum_{j>i}^{n}\left(\left(\frac{\sigma}{r_{ij}}\right)^{12}-\left(\frac{\sigma}{r_{ij}}\right)^{6}\right),$$

where $\mathbf{r}$ denotes a vector whose elements are the $3n$ Cartesian coordinates. The values of ϵ and σ used[35] are 0.0123 eV and 3.36 Å.

Each individual, X_i, in the population to be evolved consists of the real (base-10) SF Cartesian coordinates of each of the n atoms in the cluster: $X=(x_1,y_1,\ldots,z_n)$. We choose the number of individuals in the population to be typically 10 or 20. Initially, the coordinates are randomly chosen within a box of size L^3 in the first octant. We choose the first octant so that all variables are $\geqslant 0$. This is needed for the implementation of the geometric mean operation (see below). We take $x_1=L\zeta$, etc., where ζ is a freshly-generated random number between 0 and 1. We have used $L=\sqrt[3]{6n}\sigma$ for this work. A conjugate gradient minimization is performed every generation on each individual to place each structure in the vicinity of the nearest minimum. The conjugate gradient procedure is halted if any $r_{ij}^2=(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2 \geqslant L^2$. Our choice of L^2 for the termination of a conjugate gradient minimization follows that of Gregurick, *et al.*[31] We, as they, utilize this value to prevent dissociation of a cluster during the local minimization. However, we also use L, as described above, when we randomly generate the individuals of a zeroth generation. It is unclear how Gregurick, *et al.*[31] generate their zeroth generation.

Given an individual (that is, a geometry) X_i, we can calculate the potential energy of the cluster for that geometry, $V_i=V(X_i)$. Given the set $\{V_i: i=1,N\}$ we can assign the fitness, f_i, of the ith individual. We use the convention that the f_i are normalized to unity. An intermediate quantity, F_i, is evaluated by taking a function of V_i.

$$F_i=(V_i-V_{\min})/(V_{\max}-V_{\min}), \quad i=1,N.$$

The quantities, $V_{\max}$ and $V_{\min}$ are $\max\{V_i\}$ and $\min\{V_i\}$ respectively. The values of f_i are then found by normalization

$$f_i=\frac{F_i}{\sum_{i=1}^{N}F_i}.$$

The next generation (the ''children'') is formed from the current generation (the ''parents'') as follows. First, the best 20% (that is, those with the highest fitness) of the individuals in the current generation are passed intact to the next generation. (This is known as ''elitism''[14]). The remainder of the population in the next generation is obtained by use of genetic operators on the current generation. These are: (1) inversion; (2) geometric mean; (3) arithmetic mean; (4) n-point crossover; (5) 2-point crossover. They are partially described elsewhere by Zeiri.[25] We give the complete details in Appendices A and B. Of these operators, number 1 transforms one individual into a different individual; numbers 2 and 3 use two parents to ''breed'' one child; numbers 4 and 5 use two parents to produce two children.

All operators are given the same weighting, $w_\alpha=0.2$, for $\alpha=1$ through 5. Following standard Monte Carlo practices,[10] a random number on [0,1] is generated, and used to determine the operator to be selected. The requisite parents (one or two depending on the operator) are then selected weighted by their fitness using fresh random numbers.

A typical run contains 10 or 20 individuals in a population. A run was terminated when either the global minimum was found or the potential energy of the bestfit structure did not change for five generations.

A. Seeding

We also implemented a seeding procedure.[2] For an n-atom cluster, one atom is added to a globally minimized $(n-1)$ cluster. This is carried out in the following manner. The Cartesian coordinates of a minimized $(n-1)$ cluster were previously saved in a data file after having its center of mass translated to the origin. As an $(n-1)$ cluster is read into our algorithm the distance of each atom from the origin is calculated. This determines the distance from the origin, $r_{\max}$, of the furthest atom. The nth atom is then randomly placed upon the sphere centered at the origin with radius$=r_{\max}+2^{1/6}\sigma$. Furthermore, the seed $(n-1)$ structure is randomly rotated about its center of mass by generating three

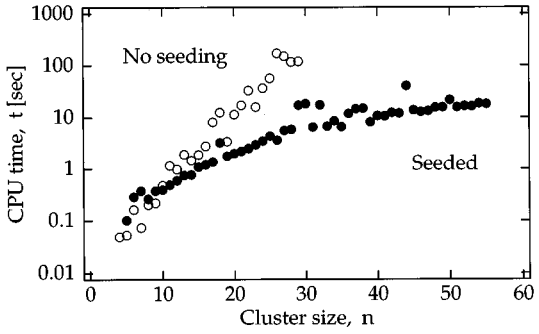

FIG. 1. Plot of CPU time for global minimization of LJ Ar_n cluster as a function of cluster size. Open circles are unseeded calculations (see text); filled circles use the seeding technique described. Each time shown is the best successful result of ten independent runs. Note that the ordinate scale is logarithmic.

random Euler angles. The random Euler rotation generally prevents each of the seeded $(n-1)$ coordinates in each individual from being identical, since the same $(n-1)$ minimized structure is used each time. The n-atom cluster then undergoes a local minimization after which its center of mass is placed at $(x,y,z)=(L/2,L/2,L/2)$. This process is applied, with fresh random numbers, to each individual as the zeroth population is created. All coordinates were allowed to freely evolve during the initial local minimization and in any subsequent generations.

III. RESULTS AND DISCUSSION

The results for the minimization of Ar_n clusters, $n=[4–29]$ without seeding and $n=[5–55]$ with seeding, are shown in Fig. 1. We plot CPU time as a function of cluster size, n. Each result is the best successful run of at most 20 independent runs. In all cases, the global minimum was found to $\pm.001$ reduced energy units,[4] usually within the first 5 runs. For the unseeded runs, we show the best results for populations of 10 and 20 individuals. There was no systematic difference in convergence times for the two population sizes. We report the times for each population size in Table I. Populations larger than 20 were attempted, but did not improve the times and are not reported here. We find that for larger clusters, seeding results in faster convergence.

To implement the seeding method, we started with a nucleus of $n=4$, and built the cluster in increments of one each time. We have pursued this as far as $n=55$. If seeding is used, the upper limit of cluster size which can be minimized using this technique has yet to be found. The CPU times used (on a DEC 2100-500) are given in Table I and shown in Fig. 1. It can be seen from the data presented that the DS-GA using the SF coordinates here is able to find the minimum for a large cluster ''from scratch'' in a reasonable time.

One of the purposes of this paper is to compare the results using SF coordinates without binary coding with the results of Gregurick *et al.*[31] Since our calculations were carried out on a faster machine than theirs, we have, for the purposes of comparison, multiplied our CPU times[36] by a factor of 5.0 in all succeeding figures. (We realize that comparison of CPU times is not straightforward. However, the operations involved in the Gregurick *et al.* paper—potential evaluations, and their derivatives—seem to be reasonably similar in both cases.) The CPU times for cluster minimiza-

TABLE I. Times needed to minimize Ar_n by the SF modified GA. The times are CPU times on a DEC 2100-500. Given are times for unseeded $(n=[4–29])$ and seeded approaches $(n=[5–55])$. The data for the seeded method are the times needed to minimize a cluster of size n starting with a minimized cluster of size $(n-1)$. Results are the best of ten or, at most, twenty independent runs. Figures in parentheses are the number of generations required for convergence. A zero implies convergence upon conjugate gradient minimization of the zeroth generation.

n	10 per pop	20 per pop	10/pop; seeded
4	0.049 (1)	0.103 (0)	NA
5	0.052 (0)	0.157 (1)	0.102(1)
6	0.162 (4)	0.342 (3)	0.284(2)
7	0.073 (1)	0.267 (1)	0.374(2)
8	0.201 (2)	0.366 (2)	0.262(1)
9	0.219 (1)	0.494 (2)	0.375(1)
10	0.478 (1)	0.541 (1)	0.390(1)
11	1.158 (1)	1.176 (1)	0.488(1)
12	1.982 (1)	0.980 (1)	0.589(1)
13	1.899 (2)	3.274 (2)	0.743(1)
14	1.453 (2)	3.874 (2)	0.765(1)
15	1.864 (2)	6.763 (2)	1.076(1)
16	2.736 (2)	7.180 (2)	1.189(1)
17	11.511 (3)	8.196 (1)	1.355(1)
18	12.673 (5)	17.518 (4)	3.220(4)
19	3.387 (2)	15.356 (3)	1.745(1)
20	11.445 (2)	15.276 (2)	1.968(1)
21	17.323 (4)	64.211 (6)	2.190(1)
22	33.170 (3)	42.924 (2)	2.469(1)
23	16.151 (3)	46.166 (4)	2.919(1)
24	36.998 (4)	83.541 (5)	3.453(1)
25	68.748 (6)	56.805 (4)	4.327(1)
26	171.275(10)	325.439(16)	3.617(1)
27	177.368 (9)	151.053 (5)	5.587(5)
28	116.783(20)	457.961(29)	5.869(1)
29	118.968(16)	275.080(15)	17.345(1)
30			18.472(5)
31			6.488(5)
32			17.658(5)
33			6.747(1)
34			8.563(1)
35			6.538(1)
36			11.998(5)
37			14.687(1)
38			15.160(1)
39			8.149(3)
40			10.858(2)
41			10.648(4)
42			12.394(3)
43			12.137(2)
44			12.523(1)
45			13.967(1)
46			12.963(1)
47			13.368(1)
48			15.630(1)
49			15.819(1)
50			21.733(1)
51			15.804(1)
52			16.692(1)
53			16.493(1)
54			18.611(1)
55			18.320(1)

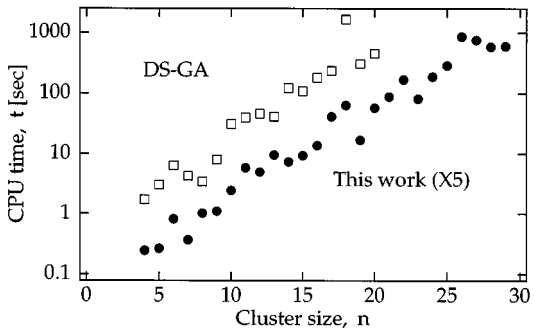

FIG. 2. Plot of CPU time for unseeded clusters, using the DS-GA compared with present method. Open squares are the results of Gregurick *et al.* filled circles are present method multiplied by a factor of 5.0 (see text). Note that the ordinate scale is logarithmic.

tion in which no seeding was employed are compared in Fig. 2. The SF variant of the modified GA performs faster for all cluster sizes reported. Furthermore, Gregurick *et al.* report no converged results for $n \geqslant 20$.

We note that the times needed for $n=6$, and $n=18$ are anomalously high. This observation agrees with that of Gregurick *et al.*[31] This appears to be due to the presence of many relatively low-lying minima in these particular cases. Presumably, the difference in fitness between these and the global minimum is small, making the algorithm work harder to make the final evolutionary step.

In Fig. 3, we compare the times needed to minimize a seeded cluster. The time reported is the CPU time to obtain the optimal energy for cluster size n given a minimized cluster of size $(n-1)$. It can be seen that as the cluster size (and the size of the search space) increases, the SF application becomes preferable.

One way of comparing numerical algorithms is to compare how they scale with the size of the problem. In order to obtain this measure, one plots $\log(t)$ against $\log(n)$ and obtains the best straight line fit. For the data in Figs. 2 and 3, these scalings are given in Table II. In both cases—seeded and unseeded—the SF approach fares better. Only if we compare the data at very low n values is the DS-GA performance better.

TABLE II. Best fit parameters to data in Figs. 2–4. It is assumed the data can be cast in the form, $t \propto n^{\gamma}$. The value of γ is obtained using an unweighted linear least squares fit (Ref. 37) to $\log(t)$ vs $\log(n)$. The integers in brackets $[n_{\min}, n_{\max}]$ denote the range of cluster size over which the fit was taken.

	DS-GA	This work
Unseeded	3.9[4–20]	4.4[4–29]
		3.6[4–20]
Seeded	7.5[21–29]	3.3[5–55]
		3.6[17–41]
		2.2[42–55]
Cumulative	4.5[4–29]	3.3[4–55]

One interesting point to notice from Table II is that the seeding technique increases in efficiency as the second solvation shell ($n=55$) is closed. Presumably this is due to the low number of available second shell sites for the added atom.

Perhaps a more reasonable measure of the performance of the seeding technique is to measure the cumulative time needed to minimize a cluster, Ar_n. For our runs, we define this time as:

$$t_n^{\mathrm{cum}} = t_4 + \sum_{i=5}^{n} t_i ,$$

where t_i is the time needed to minimize the ith cluster starting from the $(i-1)$ structure (or from scratch if no seeding was used, as in the case of $n=4$). Gregurick *et al.*[31] report an overall scaling for their method. It appears that they used a similar definition of their cumulative time. We are able to reproduce their reported scaling law if we further define:

$$t_n^{\mathrm{cum}} = t_{20} + \sum_{i=21} t_i ,$$

for their data. Results of t^{cum} as a function of n are given in Fig. 4. The scalings are given in Table II.

Clearly the SF version of the DS-GA presented here is at worst comparable to, and usually superior to the DS-GA of Gregurick *et al.*[31] This is a rather surprising result in light of Goldberg's discussions[14] of the greater efficiency of GA operations using binary-coded variables, and the ''building

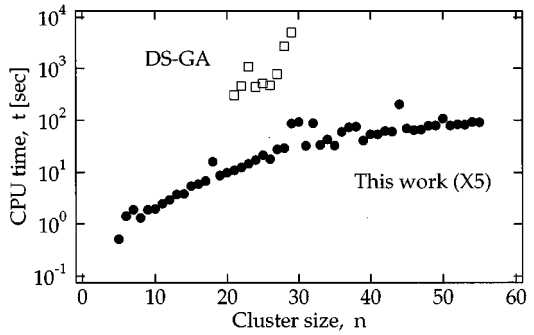

FIG. 3. Plot of CPU time for seeded clusters, using the DS-GA compared with present method. Symbols are as in Fig. 2. Note, that the ordinate scale is logarithmic.

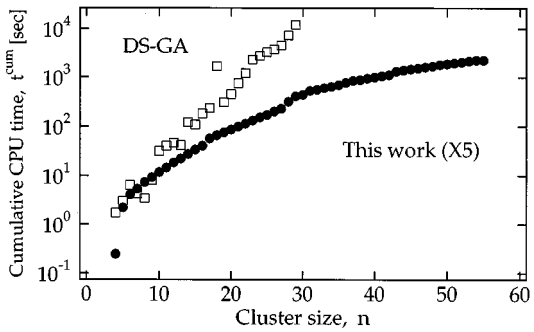

FIG. 4. Plot of cumulative CPU time (see text) for minimization of clusters using the results of Gregurick *et al.* compared with present method. Symbols are as in Fig. 2. Note that the ordinate scale is logarithmic.

block hypothesis.'' It is not clear to us at present exactly why the SF variant performs so well. In future work, we will examine how the particular operators defined here (and originally by Zeiri[25]) create new genetic material.

It is unclear whether our enhanced efficiency is due to the base-10 coding itself or to the more complex operations applicable to base-10 variables. In an attempt to compare our procedure with the more traditional GA approaches, we have carried out comparison calculations on several cluster sizes. The most important operator in the ''traditional'' GA is the one-point crossover.[14,15] We have carried out runs using our base-10 coding together with this single operator. We report here only the results for $n=19$, which we find to be typical. Using an initial population of ten individuals and only the one-point crossover operator, the global minimum was located only twice in a batch of one hundred independent runs. This compares poorly to the usual location of the global minimum at least once in ten runs, as we report here. If the population size was increased to one hundred individuals, the probability of locating the global minimum increased to approximately 30%, but the CPU time of the best run also increased. It appears that the efficiency of our technique is related both to the choice of the real space-fixed variables, and to the use of appropriate operators to search the variable space.

We mention here some of the *caveats* concerning the seeding technique. While the Ar_n clusters used here seem not to go through any phase changes as n goes from 4 to 55, this is not always the case for other species. For certain potential parameters, even such simple clusters as Ar_n can undergo morphology change as a function of n.[7] In such circumstances, genetic information obtained for clusters of phase α may actually be detrimental for clusters of phase β. In addition, there may be several families of morphology (particularly in bonded structures) in which there is little similarity between structures X_n and X_{n+1}, even for small n. We have observed this in silicon clusters.[38]

IV. CONCLUSIONS

We have presented calculations of global potential energy minimizations for Lennard-Jones clusters of argon, Ar_n using a modified genetic algorithm approach. We have used the philosophy of the DS-GA of Gregurick *et al.*[31] in that we allow each geometry created in the search to be immediately quenched to a local minimum. In contrast to their approach, however, we use the atomic space fixed Cartesian coordinates directly as our genetic material. This requires the use of nontraditional genetic operators, which we have adapted from the work of Zeiri.[25]

We find the SF Cartesian version of the DS-GA with real coding is comparable to the DS-GA using internal coordinates with binary coding at low n. However, at high n, the SF version is superior. It is capable of minimizing clusters up to $n=29$ without any seeding. Using seeding, minimized clusters of $n=55$ were readily attainable. It was found that the CPU time required scaled as $O(n^{3.3})$.

The use of genetic algorithms with SF Cartesian coordinates for the optimization of cluster geometries is clearly viable. We are at present pursuing further applications of this approach.

Note added in proof. A very recent GA study of Lennard-Jones clusters [D. M. Deaver, N. Tit, J. R. Morris, and K. M. Ho, Chem. Phys. Lett. **256**, 195 (1996)] located a new global minimum for $n=38$. Since we halted our code when the literature minimum was reached, we missed this geometry. Further runs found the newly-discovered minimum in 291 s CPU time.

APPENDIX A

We give here the details needed to fully understand the genetic algorithm operators used here. We denote the ith geometry of the n-atom cluster as $X_i=(x_1,\ldots,x_n)$, where $x_k=(x_k,y_k,z_k)$ is the displacement of the kth atom. While the distinction between x, y, and z coordinates is important for evaluating the potential, the operators act simply on a string of reals. To emphasize this we relabel this string of reals as $X_i=(c_1,\ldots,c_{3n})$. We use c_k to denote $c_k(i)$ if there is no ambiguity. We summarize below the action of each of the operators. In all the expressions below k runs from 1 to $3n$. In the notation $[c_k(i),c_k(j)]=[c_k(j),c_k(i)]$ simultaneous substitution is implied, with the updated generation on the left hand side, the current generation on the right hand side of the assignment. We also include an example of each perator's behavior.

Inversion: $c_k=c_{q+r-k}$ $r\leqslant k\leqslant q$ (r,q flat on $[1,3n]$)
A single parent, $[\alpha_1,\ldots,\alpha_{k-1},\alpha_k,\alpha_{k+1},\ldots,\alpha_{3n}]$, is required for inversion. For instance if $r=k-2$ and $q=k+1$ the resulting child is $[\alpha_1,\ldots,\alpha_{k+1},\alpha_k,\alpha_{k-1},\ldots,\alpha_{3n}]$.

Arithmetic mean: $c_k(i)=0.5(c_k(i)+c_k(j))$
Two parents, $[\alpha_1,\ldots,\alpha_n]$ and $[\beta_1,\ldots,\beta_{3n}]$, produce one child, $[0.5(\alpha_1+\beta_1),\ldots,0.5(\alpha_{3n}+\beta_{3n})]$.

Geometric mean: $c_k(i)=(c_k(i)\cdot c_k(j))^{1/2}$
Two parents, $[\alpha_1,\ldots,\alpha_{3n}]$ and $[\beta_1,\ldots,\beta_{3n}]$, yield one child, $[\{\mathrm{abs}(\alpha_1\cdot\beta_1)\}^{1/2},\ldots,\{\mathrm{abs}(\alpha_{3n}\cdot\beta_{3n})\}^{1/2}]$. As explained above, we place each randomly generated cluster in the first octant when creating the population for the zeroth generation However, we do not explicitly restrict the coordinates to the

first octant during subsequent generations, and therefore, we take the absolute value of the product before the square root.

N-point crossover: $[c_k(i),c_k(j)]=[c_k(j),c_k(i)]$ if $\zeta>0.5$
$[c_k(i),c_k(j)]=[c_k(i),c_k(j)]$ if $\zeta\leqslant 0.5$
Two parents, $[\alpha_1,\ldots,\alpha_{k-1},\alpha_k,\alpha_{k+1},\ldots,\alpha_{3n}]$ and $[\beta_1,\ldots,\beta_{k-1},\beta_k,\beta_{k+1},\ldots,\beta_{3n}]$, produce two children. For example, $[\beta_1,\ldots,\alpha_{k-1},\beta_k,\beta_{k+1},\ldots,\alpha_{3n}]$ and $[\alpha_1,\ldots,\beta_{k-1},\alpha_k,\alpha_{k+1},\ldots,\beta_{3n}]$ may result, depending on the $3n$ ''fresh'' random numbers, ζ.

2-point crossover: $[c_k(i),c_k(j)]=[S_{s+k}(ij),S_{s+k+3n}(ij)]$ s flat on $[1,3n]$
$S(ij)=(c_1(i),\ldots,c_{3n}(i),c_1(j),\ldots,c_{3n}j))$ and it is understood that $s+k+3n$ is modulo $6n$. The two parents, $[\alpha_1,\ldots,\alpha_{k-1},\alpha_k,\alpha_{k+1},\ldots,\alpha_{3n}]$ and $[\beta_1,\ldots,\beta_{k-1},\beta_k,\beta_{k+1},\ldots,\beta_{3n}]$, yield two children, $[\alpha_{k-1},\alpha_k,\alpha_{k+1},\ldots,\alpha_{3n},\beta_1,\ldots,\beta_{k-2}]$ and $[\beta_{k-1},\beta_k,\beta_{k+1},\ldots,\beta_{3n},\alpha_1,\ldots,\alpha_{k-2}]$ if, for example, $s=k-2$.

APPENDIX B

There are several ways in which the above operators can allow duplication of individuals within a population. To be duplicates here means two structures have not only the same potential energy (degenerate), but have the same coordinates as well. The population may become overly weighted with duplicates of the lowest energy structure because this is chosen most often to be a parent (has the highest fitness). A second reason why duplication is undesirable, particularly within a population as small as that used here, is that duplicate parents can exchange information resulting in a child cluster containing two atoms with identical coordinates. We avoid most duplications by preventing certain choices during some of the breeding schemes. These are:

Inversion: $k\neq 3n$
Arithmetic mean: $i\neq j$
Geometric mean: $i\neq j$
N-point crossover: $i\neq j$ and we ensure that at least one switch occurs.
2-point crossover: $k\neq 3n$ and $i\neq j$

However, duplication may still occur through more complicated, but rare, manipulations spanning more than one generation. We have scanned some runs for structures with the same energy within any one generation. We determined if they are duplicates, and not merely degenerate, by simply subtracting corresponding coordinates. A result of zero for each of the $3n$ pairs indicates duplication. We find, after the above restrictions are implemented, that fewer than 0.1% of generations (10 individuals in population, $n=13$) contained duplicates. As mentioned previously, duplication may result in a child cluster which contains two or more atoms with identical coordinates, causing divide by zero errors upon calculation of the cluster's potential. We prevent these errors by artificially setting any $r_{ij}^2<8.0$ Å^2 to 8.0 Å^2 during the calculation of the potential and its derivatives, without actually altering the coordinates themselves. This ensures that either the offending structure receives an extremely low fitness and is subsequently eliminated from the population or a local minimization produces a viable candidate.

[1] M. R. Hoare, Adv. Chem. Phys. **40**, 49 (1979).
[2] M. R. Hoare and P. Pal, Adv. Phys. **20**, 161 (1971).
[3] A. W. Castleman, Jr. and R. G. Keesee, Annu. Rev. Phys. Chem. **37**, 525 (1986).
[4] J. A. Northby, J. Chem. Phys. **87**, 6166 (1987).
[5] C. D. Maranas and C. A. Floudas, J. Chem. Phys. **97**, 7667 (1992).
[6] R. E. Smalley, Acc. Chem. Res. **25**, 98 (1992).
[7] J. P. K. Doye and D. J. Wales, Science **271**, 484 (1996).
[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Science **220**, 671 (1983).
[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes* 2nd Ed. (Cambridge, NY, 1992).
[10] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods* (Wiley, NY, 1986).
[11] P. Amara, D. Hsu, and J. E. Straub, J. Phys. Chem. **97**, 6715 (1993).
[12] M. Sylvain and R. L. Somorjai, J. Phys. Chem. **95**, 4147 (1991).
[13] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll, Chem. Phys. Letters **219**, 343 (1994).
[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, 1989).
[15] L. Davis, *Handbook of Genetic Algorithms* (Van Nostrand, Reinhold, New York, 1991).
[16] Z. Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs* (Springer-Verlag, New York, 1994).
[17] D. E. Goldberg, Complex Systems **5**, 139 (1991).
[18] E. Fontain, Anal. Chim. Acta, **265**, 227 (1992).
[19] R. S. Judson, J. Phys. Chem. **96**, 10102 (1992).
[20] R. S. Judson and H. Rabitz, Phys. Rev. Lett. **68**, 1500 (1992).
[21] B. Hartke, J. Phys. Chem. **97**, 9973 (1993).
[22] R. S. Judson, E. P. Jaeger, and A. M. Treasurywala, J. Mol. Structure (Theochem) **308**, 191 (1994).
[23] I. Rossi and D. G. Truhlar, Chem. Phys. Lett. **233**, 231 (1995).
[24] Y. Zeiri, E Fattal, and R. Kosloff, J. Chem. Phys. **102**, 1859 (1995).
[25] Y. Zeiri, Phys. Rev. E **51**, R2769 (1995).
[26] D. M. Deaven and K. M. Ho, Phys. Rev. Lett. **75**, 288 (1995).
[27] B. Hartke, Chem. Phys. Lett. **240**, 560 (1995).
[28] J. Mestres and G. Scuseria, J. Comp. Chem. **16**, 729 (1995).
[29] R. S. Judson, M. E. Colvin, J. C. Meza, A. Huffer, and D. Gutierrez, Int. J. Quant. Chem. **44**, 277 (1992).
[30] Y. Xiao and D. E. Williams, Chem. Phys. Lett. **215**, 17 (1993).
[31] S. K. Gregurick, M. H. Alexander, and B. Hartke, J. Chem. Phys. **104**, 2684 (1996).
[32] J. N. Beauregard and H. R. Mayne, Surf. Sci. Lett. **280**, L253 (1993).
[33] B. C. Bolding and H. C. Andersen, Phys. Rev. B **41**, 10568 (1990).
[34] Reference 14, p 80.
[35] G.-Q. Xu, R. J. Holland, S. L. Bernasek, and J. C. Tully, J. Chem. Phys. **90**, 3831 (1989).
[36] Benchmark calculations: (MDBNCH found at ⟨http://www.sissa.it/furio/mdbnch.html⟩ give the following CPU times: DEC 2100/500 37.4s; IBM R/6000 320 H 165 s; a ratio of 4.41:1). The program tested used several potential and derivative calls with nested loops in a molecular dynamics code. The computational effort appears comparable to to that required for this problem.
[37] Software package, IGOR from Wavemetrics, 1989.
[38] J. A. Niesse and H. R. Mayne, Chem. Phys. Lett. (submitted).