Theor Chem Acc (2004) 112: 7–15
DOI 10.1007/s00214-003-0552-1

Theoretical
Chemistry Accounts

Regular article

Structural optimization of atomic clusters by tabu search in descriptor space

Joey Cheng, René Fournier

Department of Chemistry, York University, 4700 Keele Street, Toronto, Ontario M3J 1P3, Canada

Received: 4 June 2003 / Accepted: 21 June 2003 / Published online: 17 March 2004

Abstract. We devised a new algorithm – tabu search in descriptor space – for searching the global energy minimum structure of atomic clusters. In each cycle, the algorithm generates many cluster structures randomly, transforms them to a standard form by "symmetrization of interatomic distances", and calculates structural descriptors for each. Clusters are then screened according to a model energy obtained by interpolation in descriptor space, and only a small fraction (10% or less) are retained for energy evaluation. This cycle is repeated many times. In a final step, clusters are sorted by increasing energy and optimized by conjugate gradient. This method requires between 10 and a 100 times fewer energy evaluations than a good genetic algorithm for locating the global minimum of n-atom clusters ($n < 35$) described by a Lennard-Jones potential. It seems a very promising method for global optimization on energy surfaces calculated by first-principles.

Keywords: Global optimization – Tabu search – Lennard-Jones – Clusters – Structure

1 Introduction

The properties of small n-atom clusters (roughly $n < 100$) differ from bulk properties and show specific size effects [1]. These properties can be very sensitive to structure. But little is known about the structure of small clusters apart from a few very small clusters ($n < 10$), and some plausible candidate structures or guiding principles for larger ones ($n > 10$), for example, the tendency of rare-gas clusters to maximize coordination [2], and rules for stability specific to fullerenes [3]. This makes it very difficult to understand size-dependent properties of clusters.

Correspondence to: René Fournier
e-mail: renef@yorku.ca

The lowest-energy structures are usually the most abundant in experiments on clusters. So, the relevance of theoretical studies depends critically on discovering the lowest-energy cluster isomers [4, 5]. But this is a very difficult problem. There are at least 1506 distinct local minima for the 13-atom Lennard-Jones (LJ) cluster [6] and good reasons to expect roughly as many local minima for more realistic potentials modeling various elements. The number of distinct minima on the potential-energy surface of clusters A_n, grows exponentially with n [6, 7]. The number of minima of n-atom clusters of most elements should be in the thousands for $n = 13$–15 and probably on the order of 10^4–10^6 for $15 < n < 25$. Mathematically, finding plausible isomers of A_n amounts to doing a global minimization of energy with respect to nuclear coordinates.

Apart from the large number of possible isomers, what makes cluster geometry optimization so difficult is the computer cost of reliable energy calculations. For example, the time for one energy evaluation of Li_{15} by linear combination of atomic orbitals Kohn–Sham (KS) theory with a large basis set on our computer is roughly 1 h. This could go down by orders of magnitude if we used semiempirical theory or a much faster computer, but it would not change the essence of the problem: the computing cost for one energy evaluation is extremely large compared to that of all other operations in a global optimization – generation of random numbers, calculation of interatomic distances, update of geometries, etc. Therefore, it is of paramount importance to devise global optimization methods that require as few energy evaluations as possible, especially if one uses a first-principles method for evaluating the energy and forces.

Local optimization is relatively simple and many good algorithms have now become standard [8]. Typically, the energy gradient is used to update geometry in a way that lowers energy for most steps in the optimization. The nearest minimum is often reached in a small number (approximately $3n$) of steps.

Global optimization is much harder because it has two contradictory requirements: the search has to explore the entire space of variables so as not to miss the

global minimum, but it has to be more thorough in low-energy regions for the sake of efficiency. A number of global optimization algorithms have been used for clusters, including simulated annealing (SA) [9], genetic algorithms (GA) [10, 11, 12, 13], and many others [14, 15, 16, 17, 18, 19, 20]. Even with the best of those optimization methods, very many energy evaluations are required before finding the global minimum. For instance, a combination of GA and conjugate gradient (CG) local optimization typically requires approximately 100,000 energy evaluations before finding the global minimum of a 30-atom LJ cluster. Because of that, energy evaluations must be very fast, which practically rules out first-principles energy methods [5]. By and large, global optimizations of clusters have been limited to empirical potentials and semiempirical energy surfaces: these methods can model realistically only a few elements. Aside from their disappointing performance, we generally find current cluster optimization algorithms unsatisfactory for three theoretical reasons.

First, most algorithms use only the energy evaluated at the last structure (as in SA) or the last few structures (as in GA) in deciding what structure should be considered next. This is wasteful. Instead, it would seem advantageous to take into consideration all the energies and structures visited in previous steps and make maximum use of the available information.

Second, algorithms often generate geometries $\vec{R}$ that are unimportant for reasons obvious to a person but not to a computer algorithm. Roughly speaking, these unimportant geometries fall into two categories which we call "redundant geometries" and "impossible geometries". Redundant geometries are the multitude of geometries that can be obtained by small displacements from a local minimum. They can be problematic near the end of a standard SA optimization when the simulation temperature is very low and the algorithm steps through a long sequence of similar structures. Impossible geometries are those that have a very high energy for reasons obvious to anyone who has some knowledge of physical chemistry. An example is any geometry where one interatomic distance is much smaller than the corresponding diatomic bond length. The various types of "basin-hopping" algorithms [17] avoid redundant and impossible geometries by doing a local optimization on every candidate structure generated by a global optimizer such as GA [10, 11]. This way, the GA effectively operates only within the set of local minima, not the entire set of possible structures. This makes the task much easier for the GA, but it comes at a high price: every GA step gets replaced by "GA plus local optimization", which increases the number of energy evaluations per GA step by a factor of roughly $(3n + 1)$. In addition, every local optimizaton requires approximately $3n$ evaluations of the gradient.

Third, there are some problems, for various algorithms, associated with how new structures are created. In SA it is difficult to allow the random walker to go over high energy barriers between minima in systems with strong directional bonds (such as C_n and Si_n). The simplest solution to this, going to very high simulation temperatures, does not work because atoms start evaporating from the cluster. In GA, it is hard to devise a procedure for creating a child cluster from two parent clusters such that meaningful structural features are passed on from the parents to the child. The standard method is to cut each parent along a dividing plane and assemble a m-atom fragment of parent A with an $(n - m)$-atom fragment of parent B to create an n-atom child [11]. The Cartesian coordinates of the child's atoms look similar to those of the parents, but the relative atomic positions near the dividing plane are not at all like those in the parents. Child clusters almost always have an unphysically high energy (they are "impossible geometries"). The usual solution to this problem, local optimization of the child cluster, is only partly successful: it tends to preserve the structural features of the parents far from the dividing plane but not near it.

We devised a method that avoids the theoretical problems already mentioned and that shows very promising performance on small ($n \leq 40$) LJ clusters. It consists of several parts and is a meta-algorithm more than an algorithm. It has two drawbacks: it is more complicated, and requires more user intervention, than most algorithms. But it has a significant efficiency advantage: it is at least 1 order of magnitude more economical than the current best methods. It uses the main idea of tabu search [21], which is to avoid trying the same solution more than once. The other defining feature of our method is that it uses structural descriptors such as "mean atomic coordination" and "mean nearest neighbor interatomic distance" to describe cluster structure, in addition to atomic coordinates. These descriptors are used to implement the rules for avoiding previous structures, and also as interpolation variables for calculating a model energy from those of previously visited structures. We call our method tabu search in descriptor space (TSDS).

The next section gives a detailed description of the TSDS algorithm that we used for optimization of LJ clusters. In Sect. 3, we compare the performance of TSDS for optimization of LJ clusters with 10–40 atoms to that of two standard methods: a GA (one of the best methods known for cluster optimization) [10], and a random search (which is inefficient but very simple). In Sect. 4 we discuss the prospects of using TSDS with KS density functional theory (DFT) and draw some conclusions.

2 Tabu search in descriptor space

The main steps for the optimization of a n-atom cluster by TSDS are outlined in the following. Additional details for some of the steps are given in separate subsections for clarity.

1. Input the number of atoms, n, and parameters that control the optimization: d_0, $d_{\min}$, $d_{\max}$, N_i, N_c, K, δ_i, δ_f, σ_i, σ_f, ρ_i, and ρ_f.
2. Generate initial random atomic coordinates for N_i clusters (see Sect. 2.3) and set $M = N_i$.

3. Modify each of the M cluster geometries by an operation that we call "symmetrization of interatomic distances" (SID, see Sect. 2.4) to bring it into a standard form. This makes the initial main set $\mathscr{M}$ of clusters.
4. Set the cycle counter variable $c := 1$.
5. Set the three main control parameters of TSDS for this cycle, δ, σ, and ρ, as functions of c (Sect. 2.8).
6. Calculate, or retrieve, the energy U_i $(i = 1, M)$ for all clusters in the main set $\mathscr{M}$ and set $U_0 = \min\{U_i\}$.
7. Assign to each cluster in $\mathscr{M}$ a Boltzmann weight $w_i = \exp[(U_0 - U_i)/\delta]$.
8. Go through steps 8.1–8.4 for $k = 1, K$ to generate a temporary set $\mathscr{K}$ of candidate clusters for this cycle.
8.1. Select one of the M clusters in $\mathscr{M}$ with probability equal to $w_i / \sum_i^M w_i$.
8.2. Select a geometric operation at random among these three: (a) sequence of single atom hops (Sect. 2.5); (b) single atom relocation (Sect. 2.6); (c) multiple atomic displacements (Sect. 2.7). We pick these operations with probabilities equal to 0.7, 0.15, and 0.15, respectively. Then create a new candidate cluster "k" by doing the selected operation on the parent cluster i followed by SID.
8.3. Calculate the descriptors for the kth candidate cluster (Sect. 2.1).
8.4. Calculate a model energy for the kth candidate cluster by interpolation in descriptor space (Sect. 2.2), and then add the coordinates, descriptors, and model energy of this candidate to $\mathscr{K}$.
9. Assign a penalty score (Sect. 2.8) to each cluster in $\mathscr{K}$. Select the one with lowest score and calculate its energy. Add this cluster's coordinates, energy, and descriptors to $\mathscr{M}$, reset $\mathscr{K}$ to an empty set, and update the counters $M := M + 1$ and $c := c + 1$.
10. If $c < N_c$, go to 5. Otherwise, go to 11.
11. Put all clusters in order of increasing energy. Go down the list and do a CG optimization for each cluster "i" that satisfies these two conditions: (a) $U_i < U_0 + U_t$, where U_t is a tolerance typically set equal to 8.0 (8 times the diatomic bond energy) for LJ clusters; and (b) the penalty score (Eq. 11) calculated by summing over the set of structures that previously satisfied conditions (a) and (b) is smaller than some threshold value. For condition (b), we used $\sigma = 0.20$ in Eq. (11) and a penalty threshold of 2.0. This way of doing the final screening of structures limited the number of times we rediscovered some local minima to roughly 3, and typically yielded ten distinct local minima. We updated U_0 whenever a lower energy minimum was found.

2.1 Descriptors

Descriptors are useful because they carry more insight about structure than atomic Cartesian coordinates and only a few descriptors (six in our case) are normally needed instead of $3n$ Cartesians. The six descriptors we used are as follows. We define the coordination number of an atom, c_k, as the number of atoms located within a sphere a radius $d_{\max}$ centered around atom k. Four descriptors are derived from these coordination numbers: the mean, root-mean-square (rms) scatter, minimum, and maximum of atomic coordinations.

$$\mathrm{Mean}(c) = (1/n)\sum_{k=1}^{n} c_k \tag{1}$$

$$\mathrm{Rms}(c) = \left(\sum_{k=1}^{n}[c_k - \mathrm{mean}(c)]^2/n\right)^{1/2} \tag{2}$$

$$c_- = \min\{c_k\} \tag{3}$$

$$c^+ = \max\{c_k\} \tag{4}$$

Two descriptors are derived from the moments of inertia $I_a \geq I_b \geq I_c$.

$$\zeta = \frac{(I_c - I_b)^2 + (I_b - I_a)^2 + (I_a - I_c)^2}{I_a^2 + I_b^2 + I_c^2} \tag{5}$$

$$\eta = (2I_b - I_a - I_c)/I_a \tag{6}$$

These descriptors quantify the departure from spherical shape (ζ) towards a more oblate or more prolate structure (η). They were found to be useful for analyzing trends in cluster isomer energies in Ag [24] and Li [25] clusters.

2.2 Energy estimate by interpolation

At any point in the search (except the very beginning) atomic coordinates of every cluster i in the main set are kept in memory along with six descriptors D_{li} $(l = 1, 6)$ and the energy U_i. When a candidate cluster k is formed, its energy is estimated by interpolation over the energies of the M clusters that make up the current main set $\mathscr{M}$. The distance in descriptor space between any two clusters i and k is defined as

$$\Delta_{ik} = \left[\sum_{l=1}^{6}(D_{li} - D_{lk})^2/(D_{l,\max} - D_{l,\min})^2\right]^{1/2}, \tag{7}$$

where $D_{l,\max} = \max\{D_{li}\}$ is the largest value of descriptor l among clusters of the main set, and $D_{l,\min}$ is the smallest value. Having $(D_{l,\max} - D_{l,\min})$ in the denominator amounts to normalizing each descriptor so that its range is equal to 1. This way, we ensure that all descriptors are given the same importance in the interpolation and in the score (Sect. 2.8). The interpolation that we used is a simple version of a general formula discussed by Bettens and Collins [26]:

$$\tilde{U}_k = \frac{\sum_i^M U_i \Delta_{ik}^{-4}}{\sum_i^M \Delta_{ik}^{-4}} \ . \tag{8}$$

As the search progresses and M increases, the desciptor space gets filled with clusters having known energies U_i; therefore, energy predictions by interpolation become progressively better. This is particularly useful when the cost of energy evaluation is much higher than the cost of interpolating on a few hundred numbers, as is the case for KS DFT. Note that the choice of descriptors is

important for the accuracy of energy predictions by interpolation: if there is no correlation between descriptors and energy, the predictions are unreliable except for the special case where the descriptors of a candidate are nearly identical to those of a cluster in the main set.

2.3 Making random clusters

The first atom is put at the origin. For each of the remaining atoms "j" ($j = 2, N$), we do the following. A reference atom "r" is selected at random among the $j-1$ previous ones; next, a distance "d" is generated randomly in the interval $[d_{\min}, d_{\max}]$; then, a random unit length vector (x, y, z) is generated. We assign coordinates $x_j = x_r + dx$, $y_j = y_r + dy$, and $z_j = z_r + dz$ to the new atom j.

2.4 Symmetrization of interatomic distances

The main idea is to think of the set of atom pairs as made of two subsets: bonded pairs ($d_{ij} \le d_{\max}$) and nonbonded pairs ($d_{ij} > d_{\max}$). For each pair ij we calculate the interatomic distance d_{ij} and set a target distance $t_{ij} = d_0$ if $d_{ij} \le d_{\max}$ and $t_{ij} = 1.1 d_{\max}$ if $d_{ij} > d_{\max}$. The factor 1.1 could in fact be any number larger than 1 but not too large; it does not have to be precisely 1.1. Then we enter a cycle of several (typically 50) iterations. At each iteration, we copy the coordinates of all atoms $\vec{R} = (\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_n)$ into a temporary vector $\vec{Y} = (\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n)$. Next, we go through every atom pair ij and update the temporary coordinates so as to bring interatomic distances closer to their target values. If $t_{ij} > d_{\max}$ and $d_{ij} > d_{\max}$ we do nothing; otherwise, we elongate bond ij if it is too short and shorten it if it is too long, and update $\vec{Y}$. Of course, any change in one bond length affects several bond lengths. That is why we need an iterative procedure with a temporary copy of coordinates at each iteration, and also is why we need a mechanism that makes atoms i and j repel whenever their distance falls below $d_{\max}$ if those atoms initially had $d_{ij} > d_{\max}$. A similar technique was used in the optimization of peptide conformations [27]. A natural choice for d_0 would be 1.000 (in units of equilibrium diatomic bond length) but we found that $d_0 = 1.017$ gives slightly better optimization runs and that is what we used in the LJ cluster calculations reported here.

2.5 Operation 1: Atomic hops

This operation moves a single atom to a new position through a sequence of hops. First, we pick an atom j at random. Then, we calculate the distances d_{jk} between atom j and all other atoms and form the set of neighbors of j, i.e., atoms k for which $d_{jk} \le d_{\max}$. Next we pick a random unit length vector (x, y, z), and pick a random integer K in $[1,\ J]$, where J is the smallest of four and the number of neighbors of atom j. We calculate normalized scalar products $p_k = (1/d_{jk})(x, y, z) \cdot [(x_k, y_k, z_k) - (x_j, y_j, z_j)]$ for each neighbour k. We put the neighbours of j in increasing order of p_k and select the K neighbors with largest value of p_k. The net result of all this is to select K neighbors which lie closest to a certain direction from atom j and which are often neighbors of each other. Then we calculate a "mirror point" $\vec{r}_m$ as the average of the positions of the K neighbors that were selected:

$$\vec{r}_m = (1/K) \sum_{n=1}^{K} \vec{r}_n \ . \tag{9}$$

We then let atom j "hop" from its current position $\vec{r}_j$ to the "reflection point" $\vec{r}_r = 2\vec{r}_m - \vec{r}_j$. This is analogous to the kind of moves used in simplex optimizations [28]. The hop is accepted if $|\vec{r}_r - \vec{r}_j| > d_{\min}$, otherwise it is rejected and a new hop is attempted. Next, we calculate all distances between atom j at its new position and the remaining $(N-1)$ atoms. If all of those distances are greater than $d_{\min}/2$, we stop the hopping sequence and perform a SID operation. Otherwise, we let atom j take another random hop. By continuing the hopping sequence until all d_{jk} are greater than $d_{\min}/2$ we ensure that atom j ultimately lands in a position where it has at least one neighbour and no atom in its immediate vicinity. Otherwise, the SID could fail or it could create a geometry that looks very different from the input geometry.

2.6 Operation 2: Single atom relocation

First we find the six parameters of a bounding box that contains all atoms, x_u, x_l, y_u, y_l, z_u, and z_l. For instance, $x_u = \max\{x_i\}$ is the largest of all x atomic coordinates. We pick one atom at random and move it to a random position inside the bounding box.

2.7 Operation 3: Multiple atomic displacements

We move every atom in a different random direction by a distance $d_r = s d_0$. We use $d_{\max} = 1.16 d_0$, so the factor s should be at least 0.08 to allow bond breaking (after SID) and it should not be much more than 0.20 or else too many bonds would break on average. After a few tests we fixed the value of the factor s to 0.16.

2.8 Scoring candidate clusters

We assign scores S_k to each candidate cluster k before calculating its energy. The screening (step 9) eliminates all candidates except the one with lowest score (the best), so that only one energy evaluation is required per cycle. The score is a sum of three contributions:

$$S_k = S_k^{(1)} + S_k^{(2)} + S_k^{(3)} \ , \tag{10}$$

$$S_k^{(1)} = \sum_{i}^{M} \exp(-\Delta_{ik}^2/\sigma^2) \ , \tag{11}$$

$$S_k^{(2)} = -\exp[(U_0 - \tilde{U}_k)/\delta] \ , \tag{12}$$

and $S_k^{(3)}$ is a random number taken from a uniform distribution in $[0, \rho]$. The distance in descriptor space between clusters i and k, Δ_{ik}, is given by Eq. (7), and $\tilde{U}_k$ is given by Eq. (8).

The term $S_k^{(1)}$ is large when the candidate cluster has descriptors that are numerically close to those of clusters already in the main set. This is the tabu part of the algorithm: it discourages the search from revisiting previous structures. The term $S_k^{(2)}$ ranges between -1 and 0 and tends toward -1 when the energy predicted by interpolation, $\tilde{U}_k$, is equal to the lowest energy found so far. This ensures that the search is more intense in regions of descriptor space where the energy is lower on average (this is like the "intensification" part of a tabu search). The two terms $S_k^{(1)}$ and $S_k^{(2)}$ work together to safisfy the conflicting requirements of a good global optimization: exploration of the entire space of solutions, and intensification of the search in regions of space where good solutions have been found. The term $S_k^{(3)}$ is maybe not essential, but it adds randomness to the search and favors exploration. Logically, a global optimization should emphasize exploration at the beginning and intensification toward the end. We accomplish this by continuously varying the three key parameters of TSDS (σ, δ, and ρ) from some large initial values (σ_i, δ_i, and ρ_i) to small final values (σ_f, δ_f, and ρ_f). The decrease was chosen to be exponential between the first cycle ($c = 1$) and last cycle ($c = N_c$), in analogy to the typical cooling schedules used in SA.

$$a = \ln(\delta_f/\delta_i) \tag{13}$$

$$\delta = \delta_i \exp[a(c-1)/(N_c - 1)] \tag{14}$$

Similar formulas apply to σ and ρ.

2.9 TSDS control parameters

The parameters of Table 1 give good results for optimization of LJ clusters and were used to generate the results reported here. Note that N_c, δ_i and δ_f are proportional to the number of atoms in the cluster. Other parameters are not as important and were kept at fixed values after a few tests. For example the probabilities for choice of operations 1–3, were set to 0.70, 0.15, and 0.15, and the parameters that set the tolerance for energy and the norm of gradient in CG optimizations were set to 0.001.

3 LJ clusters

LJ clusters are a standard testing ground for optimization algorithms. The energy and the structure of the lowest known minima of each size compiled by Wales et al. [23] are a very useful resource and most likely contain the true global minima for n at least up to 40. We optimized n-atom LJ clusters (n = 10–40) by TSDS, by GA, and by a simple random search (RS). For each cluster size, we did 20 independent runs of GA and RS and 40 runs of TSDS to accumulate meaningful statistics on how the algorithms perform. We used the structure initialization described in Sect. 2.3 and CG local optimization for all three methods. The initialization (Sect. 2.3) is slightly different from the one of Ref. [10]. We used the same GA parameters as Roberts et al. [10] (population size 10, number of children per generation 8, mutation probability 0.1) except for the number of generations, N_{gen}, which we chose in a slightly different way: we set $N_{gen} = n$ for $n \leq 26$ and $N_{gen} = 2n$ for $n \geq 27$. The GA found the global minimum 100% of the time for $n \leq 26$, so the results would be unchanged if we used $N_{gen} = 2n$. The number of CG optimizations in RS was set at $40n$.

With these control parameters, RS and GA found the global minimum in every run for $n \leq 24$. The TSDS algorithm failed to find the global minimum in two out of 40 runs (5% of cases) at $n = 17$, and 2.5% of cases at $n = 18$ and $n = 21$. But as shown in Table 2, TSDS typically found the global minimum after many fewer energy evaluations than RS and GA. The TSDS evaluates the energy $100 + 20n$ times in the global optimization part. After that, in most of the runs with $n < 25$, the global minimum is found after only three or fewer CG local optimizations. For instance, referring to $n = 24$ in Table 2, we have $735 - 100 - 23 \times 20 = 175$, which is only 2 or 3 times larger than a typical number of steps in a local optimization.

The largest clusters for which the global minimum was found at least 80% of the time are $n = 25$ for RS, $n = 31$ for GA, and $n = 27$ for TSDS. At $n = 25$, the average numbers of energy evaluations before finding the global minimum were 6.0×10^5 by RS, 8.2×10^4 by GA, and 8.0×10^2 by TSDS. TSDS could be favored in these

Table 1. Parameters used in optimization of Lennard-Jones (*LJ*) clusters

$d_{min} = 0.933$	$d_{max} = 1.167$	$d_0 = 1.017$
$N_i = 100$	$N_c = 20n$	$K = 10$
$\delta_i = 1.00n$	$\sigma_i = 0.25$	$\rho_i = 1.00$
$\delta_f = 0.01n$	$\sigma_f = 0.05$	$\rho_f = 0.01$

Table 2. Mean number of energy evaluations (random search, *RS*, genetic algorithm, *GA*, tabu search in descriptor space, *TSDS*) before finding the global minimum of LJ clusters n = 10–24

n	RS	GA	TSDS
10	102423	11967	369
11	109672	18296	400
12	115850	15150	423
13	160355	18267	398
14	170565	27401	476
15	229306	31866	492
16	175195	33908	521
17	197657	42989	688[a]
18	215869	51687	593[b]
19	253033	54276	582
20	235812	61829	619
21	320096	66721	722[b]
22	412352	70235	710
23	311767	90200	670
24	346404	83863	735

[a] For $n = 17$, TSDS missed the global minimum twice in 40 runs
[b] For $n = 18$, 21, TSDS missed the global minimum once in 40 runs

comparisons because, in a TSDS, local optimization is done only at the very end and it is done in a particular order. But even the mean total number of energy evaluations per run shows a clear advantage for TSDS at $n = 25$: it is 1.0×10^6 for RS, 1.5×10^5 for GA, and 3.9×10^3 for TSDS. By that measure, TSDS is 38 times more efficient than GA at $n = 25$. Generally, the total number of energy evaluations per run is roughly 2–5 times larger than the number of energy evaluations up to the point where the global minimum is found. This is a consequence of choosing a number of cycles that scales with the number of atoms, i.e., a number of cycles that gives a fair chance of finding the global minimum. If we used a quasi-Newton method for local optimization, such as Broyden–Fletcher–Goldfarb–Shanno instead of CG, the performance of GA and RS relative to TSDS might improve, but probably not by more than a factor of 2.

At larger cluster sizes, the frequency of success in finding the global minimum can fall anywhere between 0 and 1. In order to make a fair comparison across different cluster sizes and methods, we calculate N_{90}, which is an estimate of the mean number of energy evaluations required to hit the global minimum with 90% probability. If a method fails to find the global minimum in a fraction, v_f of the runs, the probability of failing in each one of m independent runs is $(v_f)^m$. So, N_{90} can be estimated using the mean number of energy evaluations before the lowest minimum in each run was found, N_l, and the fraction of runs in which the true global minimum was missed, v_f:

$$N_{90} \approx N_l[\ln(0.1)/\ln(v_f)] \ ,$$

where v_f was restricted to lie between 10^{-5} and $1–10^{-5}$. This definition is not very meaningful when $v_s = 1 - v_f$ is outside the range [0.1, 0.9] but that is unavoidable. The factor multiplying N_l makes sense when v_s is inside [0.1, 0.9], for example, it is 14.2 when $v_s = 0.15$ ($0.85^{14.2} = 0.10$) and 1.21 when $v_s = 0.85$ ($0.15^{1.21} = 0.10$). At $n > 29$ we find $v_s < 0.9$ for all three methods. The N_{90} at $n = 30$ is 1.7×10^7 for RS, 5.0×10^5 for GA, and 3.0×10^3 for TSDS. Here again, the ratio N_{90}(TSDS)/N_{90}(GA) would double or triple if we used total numbers of energy evaluations, but TSDS would still show a significant advantage.

A semilogarithmic plot of N_{90} as a function of cluster size for the three methods is shown in Fig. 1. Points are omitted on the RS and TSDS curves when $v_s = 0$. At $n > 25$ RS fails too often to allow meaningful comparisons. The GA-to-TSDS ratio of N_{90} is typically between 10 and 100 for $25 < n < 35$. We see a quick deterioration of the RS method at $n > 24$. This is similar to what Roberts et al. found in a study with Morse potentials [10]. TSDS loses some of its advantage relative to GA as clusters get bigger ($n > 30$), and GA is the only algorithm with which we found the global minimum at every cluster size in at least one run. In 40 runs, TSDS could not find the global minimum of $n = 35$ and 38 once.[1] In its current version, the TSDS becomes unreliable at around $n \geq 35$. There is no guarantee that it could find the global minimum at $n > 40$ even with many more independent runs. It remains to be seen whether TSDS can be applied successfully to very large clusters, but it clearly has an advantage over GA for $n < 35$. We must point out that the advantage in TSDS is in the number of energy evaluations, and this translates into a computing time advantage only if the time for evaluating energies is much larger than the time for doing TSDS operations (e.g., calculating energy estimates by interpolations). So, TSDS has no practical value for LJ cluster optimization, but it looks very promising for optimization when the energy is modeled by complicated potentials or first-principles methods.

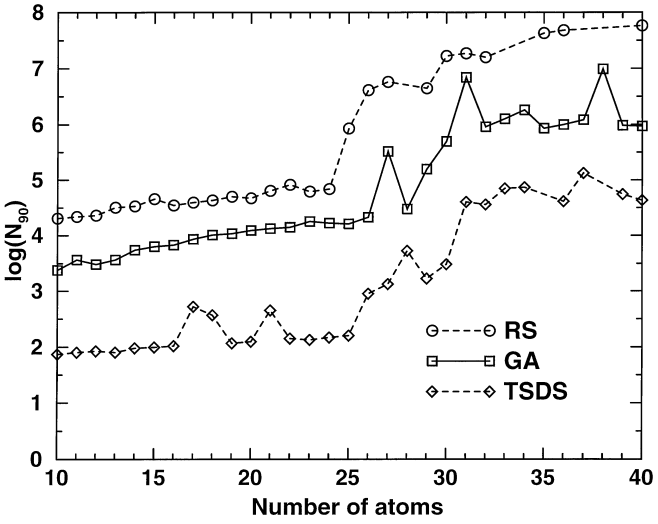

Fig. 1. Logarithm of the estimated number of energy evaluations required to find the global minimum of Lennard-Jones clusters with 90% probability

Part of the reason for the success of TSDS is that it avoids CG optimization until the very end and does SID operations instead. If SID gave structures that were almost identical to local minima, it could afford a speed-up of almost $3n$ (a typical number of steps in a CG optimization). But SID operations do not generate optimized structures. The mean energy difference E(SID) − E(CG) between structures generated by SID and the corresponding ones after CG optimization are shown for a few cluster sizes in Table 3. This difference is a few times larger than the typical energy separation between isomers on average. That is not very accurate,

Table 3. Absolute and percentage deviation between energies of local minima and structures generate by symmetrization of interatomic distances

n	Absolute deviation	Percentage deviation
10	1.57	5.8
15	1.32	2.7
20	1.44	1.9
25	1.75	1.8
30	1.82	1.4
35	2.49	1.6

[1] We did find the global minimum of $n = 35$ in approximately 10% of runs when we set $N_i = 200$ and $N_c = 40n$

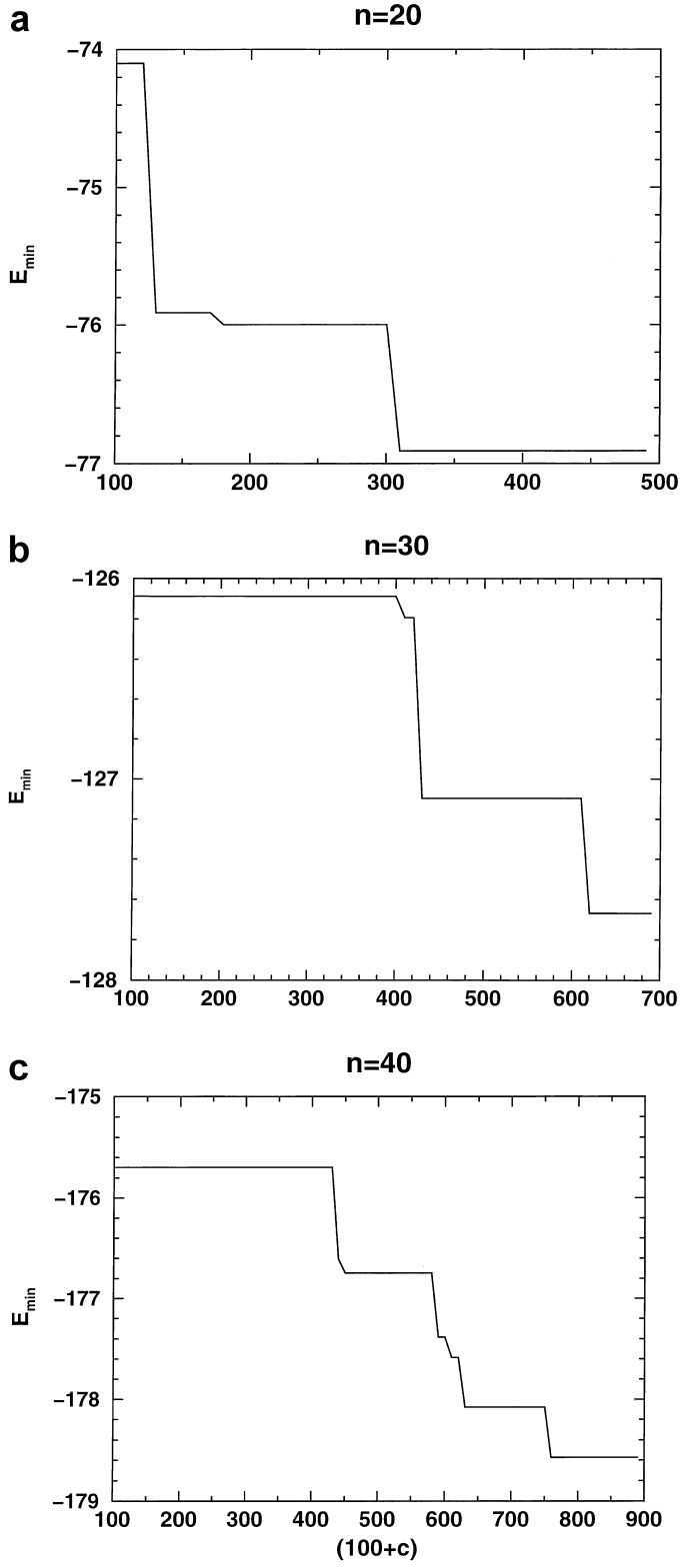

Fig. 2a–c. Lowest energy among all symmetrization of interatomic distances (SID) structures of a run, as a function of iteration number, in one representative tabu search in descriptor space (*TSDS*) run with 20-, 30-, and 40-atom clusters

but it is accurate enough to drive the search toward the lowest energy structures. We noticed in all our runs that $E(\mathrm{SID}) - E(\mathrm{CG})$, expressed in percentage of the energy, is either quite small (less than 1%) and nearly constant, or very large (more than 2%, sometimes 5% or 10%) and variable. This is a potential problem and more work will be needed on improving the SID procedure. But already, the speed-up due to SID is substantial.

The SID operation in TSDS plays a role similar to that of a local optimization in "basin-hopping" methods like the combined GA–CG of Deaven and Ho [11, 17]. In the GA–CG algorithm, arbitrary cluster structures are mapped to the nearest local minimum by CG. So, every valley on the potential surface gets represented by a single point. As pointed out by Wales and Doye [17], this effectively transforms the energy surface seen by GA from a continuously varying function of atomic coordinates to a discontinous, piecewise constant, function. This way the GA has to solve a discrete optimization problem, for which it is very well suited, instead of the original continuous problem. The SID operation should also, ideally, map every structure in a connected subset of space to a single representative structure so as to transform the continuous optimization problem into a discrete problem which TSDS can handle more easily. But the energies of SID structures differ from those of local minima by an amount that is not constant. In that sense, the SID operation is not optimal because it distorts the original problem. But of course the great advantage of SID over CG is that it requires no evaluation of energy or forces.

Another source of speed-up for TSDS is that descriptors and the interpolation (Eq. 8) effectively provide a model energy surface which allows screening of candidate clusters. The energy is evaluated for only one out of K candidates which can potentially give a speed-up of up to K. In practice, the interpolation is not very accurate, and increasing K from 10 to 50 gives little or no improvement in performance. We did many runs for $n = 30$ with a single candidate per cycle ($K = 1$ instead of $K = 10$), which effectively turns the screening off. Another way to turn the screening off, while keeping $K = 10$, would have been to use descriptors that have no correlation whatsoever with energy. The N_{90} for these runs is 2.4 times larger than with $K = 10$, which shows that screening works but gives only a modest speed-up.

The evolution of the lowest energy found among SID structures as a function of the number of cycles in representative runs for $n = 20, 30$, and 40 is shown in Fig. 2. This energy typically improves only a few times during a run. The lowest minimum found in the run of Fig. 2c was -182.588 (the true global minimum is -185.250). The difference between -178.6 (Fig. 2c) and -182.588, 4.016, is the energy-lowering produced by CG optimization of the best SID structure; in this particular case it was a little larger than usual. It may look like the combined TSDS–SID optimization was not very effective in this run because it decreased the energy by roughly 3.0 between the beginning and end (Fig. 2c) compared with a decrease of 4.0 due to CG optimization at the very end. But one has to realize that TSDS–SID does more than just lower the energy: it generates a set of

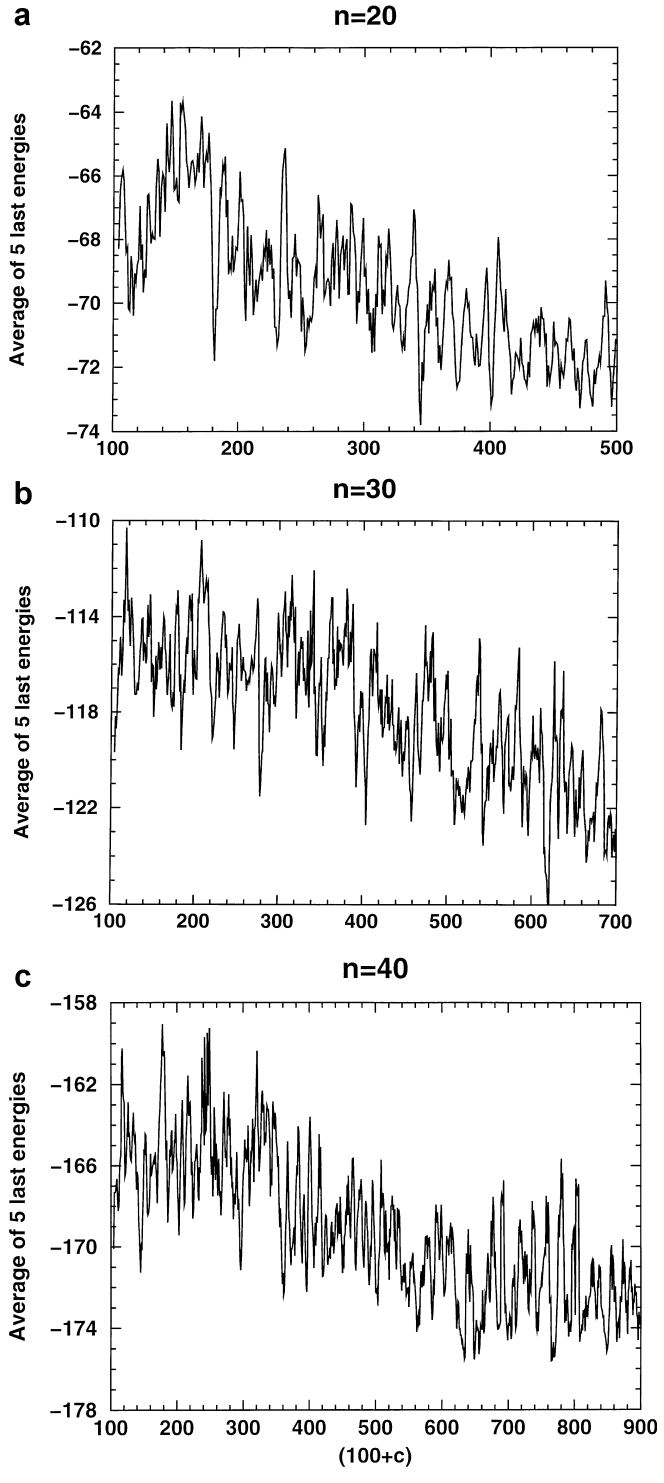

Fig. 3. Running average of the energies of the last five SID structures, as a function of iteration number,in one representative TSDS run with 20-, 30-, and 40-atom clusters

distinct structures, with only one or a few structures per valley on the potential surface. It helps to have representative structures for each valley as close to the bottom of the valley as possible, but it is equally important to have (ideally) precisely one representative structure for each low-energy valley. This is almost realized in our algorithm. In the run of Fig. 2c, for instance, we find that at least 650 of the 800 SID structures generated were distinct. As intended, the penalty term (Eq. 11) prevented the search from revisiting structures in most of the 800 cycles.

Plots of the running average of energies of the five last SID structures generated in a run are shown in Fig. 3 for the same three runs as in Fig. 2. The process by which clusters are generated has much randomness built in and this shows in Fig. 3. However, the gradual decrease of δ, σ, and ρ (Eqs. 13, 14) produces a systematic decrease in the average energy of clusters generated as the search progresses. The best structure is sometimes found near the beginning of a run (out of luck) but the mean energy of clusters created always goes down between the beginning and end of a TSDS run. Notice that energies in Fig. 3 are much higher than in Fig. 2. This is simply because there is a large variabiliy in the energies of SID structures and the probability of making five consecutive low-energy SID structures is very small. Since TSDS actively tries to avoid structures that were already visited, the quantity plotted in Fig. 3 is not guaranteed to decrease. In fact, if the runs were long enough, the running average of energies would have to go up eventually, when most of the low-energy isomers are found.

4 Conclusions

At each iteration the TSDS itself requires much more computing time than it takes for calculating the energy of a LJ potential, which makes it impractical for optimizing LJ clusters. But TSDS finds the global minima of n-atom ($n < 40$) LJ clusters with many fewer energy evaluations than one of the best current cluster optimization methods (GA–CG). Further, it takes roughly 10^4–10^5 less computing time than needed for evaluating the energies by KS DFT. This makes TSDS a very promising method for KS DFT studies of small clusters of almost any element. Prior to this study, we used a different, and less efficient, version of TSDS along with KS DFT energies to search low-energy isomers of Si_n ($10 \leq n \leq 16$) [29] and Li_n ($9 \leq n \leq 20$) clusters [25]. We were able to rediscover, with only a few hundred local spin density energy calculations, the lowest known minima [5] of Si_n ($n = 10, 11, 12$), and fell short of the lowest minima for $n = 13$–16 by 1.2 eV or less [29]. Note that the energy needed for breaking one Si–Si bond is typically approximately 2.0 eV, so 1.2 eV is not such a big energy for Si_n clusters. For Li_n we discovered several new structures, many of which have energies lower than previously reported in the literature, and some of which have triplet or higher spin multiplicity [25]. Work is under way to further improve TSDS, incorporate it in a KS DFT computer code, and test it over a wider class of clusters and molecules.

Acknowledgements. This work was supported by Research Corporation and the Natural Sciences and Engineering Research Council of Canada.

References

1. (a) Knickelbein MB (2001) Phys Rev Lett 86: 5255; (b) Knickelbein MB (1999) Annu Rev Phys Chem 50: 79; (c) Jortner J (1992) Physics and chemistry of finite systems: from clusters to crystals Vol 1. Kluwer, Dordrecht, pp 1–17; (c) Cox AJ, Louderback JG, Bloomfield LA (1993) Phys Rev Lett 71: 923; (e) Riley SJ (1992) Ber Bunsenges Phys Chem 96: 1104
2. (a) Farges J, De Feraudy MF, Raoult B, Torchet G (1983) J Chem Phys 78: 5067; (b) Martin TP (1996) Phys Rep 273: 199
3. Fowler PW, Manolopoulos DE (1995) An atlas of fullerenes. International series of monographs on chemistry no. 30. Clarendon Press, Oxford
4. Shvartsburg AA, Jarrold MF, Liu B, Lu ZY, Wang CZ, Ho KM (1998) Phys Rev Lett 81: 4616
5. Shvartsburg AA, Liu B, Jarrold MF, Ho KM (2000) J Chem Phys 112: 4517
6. Chekmarev SF (2001) Phys Rev E 64: 036703; Tsai CJ, Jordan KD (1993) J Phys Chem 97: 11227
7. Stillinger FH (1999) Phys Rev E 59: 48
8. (a) Schlick T Reviews in computational chemistry, (1992) In: Lipkowitz KB, Boyd DB (eds) Vol. 3. VCH, New York, p 1; (b) Schlegel HB (1987) Adv Chem Phys 67: 249
9. Donnelly RA (1987) Chem Phys Lett 136: 274
10. Roberts C, Johnston RL, Wilson NT (2000) Theor Chem Acc 104: 123
11. Deaven DM, Ho KM (1995) Phys Rev Lett 75: 288
12. Deaven DM, Tit N, Morris JR, Ho KM (1996) Chem Phys Lett 256: 195
13. (a) Holland J (1975) Adaptation in natural and artificial systems. University of Michigan Press, Anm Arbor; (b) Holland J (1992) Sci Am 267 (July): 66
14. Northby JA (1987) J Chem Phys 87: 6166
15. Amara P, Straub JE (1996) Phys Rev B 53: 13857
16. Pillardy J, Piela L (1997) J Comput Chem 18: 2040
17. Wales DJ, Doye JPK (1997) J Phys Chem A 101: 5111
18. Leary H, Doye JPK (1999) Phys Rev E 60: 6320
19. Wales DJ, Scheraga HA (1999) Science 285: 1368
20. Liu P, Berne BJ (2003) J Chem Phys 118: 2999
21. (a) Cvijovic D, Klinowski J (1995) Science 267: 664; (b) Hong SD, Jhon MS (1997) Chem Phys Lett 267: 422
22. Morales LB, Garduño-Juárez R, Aguilar-Alvarado JM, Riveros-Castro FJ (2000) J Comput Chem 21: 147
23. Wales DJ, Doye JPK, Dullweber A, Naumkin FY (1997) The cambridge cluster database. http://brian.ch.cam.ac.uk/CCD.html
24. Fournier R (2001) J Chem Phys 115: 2165
25. Fournier R, Cheng JBY, Wong A (2003) J Chem Phys 119: 9444
26. Bettens RPA, Collins MA (1999) J Chem Phys 111: 816, and references therein
27. Chandrasekhar J, Saunders M, Jorgensen WL (2001) J Comput Chem 22: 1646
28. (a) Nelder JA, Mead R (1965) Comput J 7: 308; (b) Jurs PC (1996) Computer software applications in chemistry, 2nd edn, Wiley, New York pp 136–145
29. Cheng JBY (2002) MSc thesis. York University