An Iterative Global Optimization Algorithm for Potential Energy Minimization

N. P. Moloi and M. M. Ali*
School of Computational and Applied Mathematics,
Witwatersrand University, Wits - 2050, Johannesburg

March 22, 2003

Abstract

In this paper we propose an algorithm for the minimization of potential energy functions. The new algorithm is based on the differential evolution algorithm of Storn and Price [1]. The algorithm is tested on two different potential energy functions. The first function is the Lennard Jones energy function and the second function is the many-body potential energy function of Tersoff [2, 3]. The first problem is a pair potential and the second problem is a semi-empirical many-body potential energy function considered for silicon-silicon atomic interactions. The minimum binding energies of up to 30 atoms are reported.

Keywords: Many-body, pair potential, differential evolution, potential energy, iterative, global optimization.

1 Introduction

Potential energy functions are becoming an increasingly important means of various investigations in nanoscale devices within the semiconductor industry and in structure-based drug design. These functions are often used as global optimization problems in search of the stable structure of atoms or molecules [4, 5]. Therefore, the determination of the global minimum energy configuration, or the ground state structures, for clusters of atoms or molecules, predicted by the potential functions, is an important global optimization problem. The numerical approach to finding the global minimum of these empirical potentials poses a very difficult problem as the number of local minima increases rapidly with the number of atoms. They are very hard in the sense that they have many local minima and that they represent high non-linearities in their mathematical representations.

To date, several methods have been proposed for this type of optimization problem [5, 6]. These are the spatial smoothing techniques [7, 8, 9], variants of simulated annealing [10] and the genetic algorithm (GA) [11]. These algorithms, however, were implemented on the Lennard-Jones (LJ) pair potential. The main thrust of these studies was to find the better configuration (or minimizer) compromising the efficiency of algorithm. Except the numerical studies carried out in [12], very little work has been done on numerical experiments using various global optimization algorithms considering the potential functions as sample global optimization test problems. In [12] the global optimization

*Visitor at the Institute for Mathematics and its Applications, University of Minnesota, USA.

involving Tersoff's many-body potential (MB) for silicon ($Si(B)$ and $Si(C)$ [3]) is reported. Eight different recent global optimization algorithms were used in this study with the conclusion that MB is very hard to optimize and therefore only small problems for silicon clusters of size up to 6 atoms were considered for testing different algorithms. Global optimization of the MB potential using up to 15 atoms was carried out in [13]. The algorithm used was a topographical multi-start algorithm. We note that unlike the LJ problem, which has been extensively studied by the optimization communities, very little experiment has been devoted to the more realistic potential problem such as MB.

In this study we consider the potential functions as sample global optimization problems only for testing our algorithm. We do not consider the physics involved in the problems and therefore no plots of the clusters are given. It was concluded in [13] that the design of an efficient multi-start type algorithm was imperative for potential minimization. We have designed the new algorithm with this observation in mind. This is a local search based differential evolution (DE) algorithm. The new algorithm is iterative in the sense that it uses the optimal structure of the cluster of $n_p - 1$ atoms when minimizing the cluster of n_p atoms. The main thrust of this paper is to devise a global optimization algorithm which is robust and efficient in finding the global minimum of silicon (Si) and Lennard-Jones clusters. We demonstrate the robustness of the new algorithm on both of these problems. The organization of the paper is as follows. In section 2 we present our new algorithm. In the same section we also briefly present the DE algorithm. Section 3 contains the statement of the problems. Section 4 contains the numerical results and conclusion is made in section 5.

2 Local Search Based Differential Evolution Algorithm

The DE algorithm is a direct method and it has a role to play where the gradient of the function is not available. Design of DE certainly brought a new dimension to the direct search techniques in the field of global optimization. Indeed, in [15] it was found that DE is very efficient in locating the global minimizer for some test problems when comparing with some other direct search global optimization methods. However, the robustness of DE falls off as the number of dimension increases. In this study we have adapted the DE method for problems of large dimension for which the gradients are readily available. We incorporate the localization of search within the global exploration feature of DE. In particular, we would like to integrate the complementary strengths of DE and the multi-start in a suitable framework, to minimize the potential functions. Although the new method is designed for potential minimization it can be adapted for general global optimization problems. Therefore, we have presented the local search based differential evolution algorithm for general problem and for the potential problem. We call these two algorithms respectively as DELG and DELP. It is easy to describe our new algorithms based on the description of DE. We begin by briefly describing the DE algorithm.

2.1 Differential Evolution (DE)

Differential Evolution (DE) [1] is a population set based algorithm [15] designed for minimizing a function of real variables. It is extremely robust in locating the global minimum. Like the other population set based direct search methods [15], DE also attempts to guide an initial set $S = \{x_1, x_2, \cdots, x_N\}$ of points in the search region $\Omega \subset I\!R^n$, n being the dimension of the problem, to the vicinity of the global minimum through repeated cycles of selection, reproduction (mutation and crossover) and acceptance. DE attempts to replace all points in S by new better points at each generation. The mechanism of the DE algorithm is to progress in an epoch or generation base. During each epoch k_{iter} of DE, N function values are evaluated. We now describe how this is done. In each generation, N competitions are held to determine the members of S for the next generation. The i-th ($i = 1, 2, \cdots, N$) competition is held to replace x_i in S. Considering x_i as the target point, a trial point y_i is found from

two points (parents), the point x_i, i.e., the target point and the point $\hat{x}_i$ determined by the mutation operation. In its mutation phase DE randomly selects three distinct points $x_{p(1)}, x_{p(2)}$ and $x_{p(3)}$ from the current set S. None of these points should coincide with the current target point x_i. The weighted difference of any two points is then added to the third point which can be mathematically described as :

$$\hat{x}_i = x_{p(1)} + F \times (x_{p(2)} - x_{p(3)}) \,, \tag{1}$$

where $F \leq 1$ is a scaling factor. The trial point y_i is found from its parents x_i and $\hat{x}_i$ using the following crossover rule :

$$y_i^j = \begin{cases} \hat{x}_i^j & \text{if } R^j \leq C_R \text{ or } j = I_i \\ x_i^j & \text{if } R^j > C_R \text{ and } j \neq I_i \,, \end{cases} \tag{2}$$

where I_i is a randomly chosen integer in the set I, i.e., $I_i \in I = \{1, 2, \cdots, n\}$; the superscript j represents the j-th component of respective vectors; $R^j \in (0, 1)$, drawn randomly for each j. The entity C_R is a constant (eg. 0.5). The ultimate aim of the crossover rule (2) is to obtain the trial vector y_i with components coming from the components of target vector x_i and mutated vector $\hat{x}_i$. This is ensured by introducing C_R. Notice that for $C_R = 1$ the trial vector y_i is the replica of the mutated vector $\hat{x}_i$, i.e., only the mutation operation is used in reproduction. For more detailed discussions on the scaling parameter F, the population size N and on the controlling parameter C_R the authors refer [15]. In the acceptance phase the function value at the trial point, $f(y_i)$, is compared to $f(x_i)$, the value at the target point. If $f(y_i) < f(x_i)$ then y_i replaces x_i in S, otherwise, S retains the original x_i. This process, per generation, continues until all members of S are targetted. The algorithm stops when some stopping condition is met. The typical stopping condition is given by

$$f_{\max} - f_{\min} \leq \epsilon, \tag{3}$$

where ϵ is a small number, say $\epsilon = 10^{-3}$, and $f_{\max}$ and $f_{\min}$ are, respectively, the current maximum and the minimum function values in S.

2.2 The Local Search Based DE Algorithm (DELG) : General Case

Structurally, DELG is similar to DE except the fact the population set S for DELG contains only the local minimizers. Mutation and crossover are similar to DE but they use distinct local minimizers in S. Initially, the N local minimizers (not necessarily distinct) of the set S are generated in the following way; iteratively sample x_i from Ω and perform a local minimization from the point x_i giving $x_i^* \in S$, $i = 1, 2, \cdots, N$. Alternatively, a set S^{ini} consisting of N random points can be generated first and then the set S consisting of the corresponding local minimizers can be generated. Like DE, DELG attempts to replace points (minimizers) in S with better points (minimizers). However, N local searches are needed per generation. In each generation of DELG the local minimizers in S are targetted and attempts are made to replace them with better minimizers. Like DE, DELG needs multiple epochs/generations to improve the estimate of the optimal solution. The number of generations increases until some stopping condition is met. Let k_{iter} be the number of generation. For the targetted local minimizer x_i^*, at k_{iter}-th generation, DELG first generates a point y_i using mutation (1) and crossover (2) and then obtains the corresponding local minimizer y_i^*. If $f(y_i^*) < f(x_i^*)$ then x_i^* is replaced by y_i^*. This process is continued until all x_i^* in S are targetted. However, unlike DE, the mutation and the crossover in DELG are carried out using local minimizers in S. For instance, the mutation rule (1) is implemented on three distinct random minimizers $x_{p(1)}^*, x_{p(2)}^*$ and $x_{p(3)}^*$ from the current set S and the crossover rule (2) is carried out between the targetted minimizer x_i^* and the mutated point $\hat{x}_i$. If the distinct number of local minimizers in S is less than 4 at any epoch of the DELG algorithm, it skips the mutation and performs the crossover only. However, in this case the

crossover rule (2) is not used. Rather, for every targetted $x_i^* \in S$, it uses

$$y_i^j = \alpha_j x_{p(1)}^{*(j)} + (1 - \alpha_j) x_{p(2)}^{*(j)}, \quad j = 1, 2, \cdots, n, \tag{4}$$

to obtain the elements of y_i, where α_j are uniform random numbers in $[-0.5, 1.5]$. See references [15, 16] for the motivation of α_j. Again two distinct local minimizers $x_{p(1)}^*$ and $x_{p(2)}^*$, both different from x_i^*, are selected randomly from S. The generation k_{iter} is completed when all x_i^* are targetted. The DELG algorithm stops when all minimizers in S have converged. We have taken the following convergence criterion.

$$f_{\max}^* - f_{\min}^* \leq \epsilon, \tag{5}$$

where $f_{\max}^*$ and $f_{\min}^*$ are, respectively, the current worst minimizer and the best minimizer in S. We now present the DELG algorithm.

Algorithm-1 : The Local Search Based Differential Evolution Algorithm (DELG)

Step 1 **Determine the initial set** S. Generate the set $S^{ini} = \{x_1, x_2, \cdots, x_N\}$ by obtaining the random point $x_i \in \Omega$. Generate the initial set $S = \{x_1^*, x_2^*, \cdots, x_N^*\}$ of local minimizers by performing a local minimization starting at x_i in S^{ini}, giving x_i^* in S, $i = 1, 2, \cdots, N$. Set $k_{iter} = i = 1$ and go to Step 4.

Step 2 **Determine the candidate point for local minimization**. If the number of distinct local minimizers in S is less than 4 then use the mutation rule (4) to obtain y_i. Otherwise, obtain y_i using mutation (1) and crossover (2).

Step 3 **Generate minimizers to replace minimizers in** S. Produce y_i^* by local minimization starting at y_i. Select the trial minimizer y_i^* for the (k_{iter}+1)-th generation using the acceptance criterion : replace $x_i^* \in S$ with y_i^* if $f(y_i^*) < f(x_i^*)$, otherwise retain x_i^*. Set $i = i + 1$. If $i < N$ then go to Step 2.

Step 4 **Stop the algorithm**. Rank order the local minimum values in S from the best to the worst. If (5) is satisfied the stop, otherwise set $i = 1$, $k_{iter} = k_{iter} + 1$ and go to Step 2.

The DELG algorithm presented above can be implemented to any global optimization problem where the gradient based local search is permitted. Indeed, implementation of DE and DELG using 14 test problems is carried out in [17] with the conclusion that DELG is superior. We now present our main DELP algorithm for the minimization of potential functions, namely the LJ and MB potentials.

Remark 1 : In the computer implementation the population set S is represented by an array A. The array A contains the N members of S with their function values. Therefore, A has N rows and $n + 1$ columns.

2.3 The Local Search Based DE Algorithm for Potential Minimization (DELP)

In this section we present the DELP algorithm for energy minimization of clusters of atoms interacting through a well-defined potential function. DELP is a modified version of DELG. This modification takes advantage of the special structure of potential functions in that the optimal cluster of, say $n_p - 1$ atoms is used in the optimization of the cluster involving n_p atoms. We consider the smallest cluster to have three atoms. To optimize the cluster of n_p atoms, DELP iteratively optimize clusters of 3 atoms, 4 atoms, and up to $n_p - 1$ atoms. Therefore, a single run of the DELP is needed to optimize all clusters of atoms up to the cluster of n_p atoms. There are $n_p - 2$ optimization subproblems in the optimization of the cluster consisting of n_p atoms. This optimization problem is solved iteratively

with the inner level (the smallest subproblem) formed by the problem of 3 atoms and the outer level (the largest subproblem) formed by n_p atoms. In these subproblem optimizations the optimal structure in the (j-1)-th subproblem is used in the optimization of jth subproblem, $j = 1, 2, \cdots, (n_p - 2)$. We now describe how this is done. For this let us define $n^{(p)}$ be the dimension of the subproblem with n_p number of atoms. For instance $n^{(3)}$ is the dimension of 3 atoms problem. The size N of the population set S is dependent on the dimension $n^{(p)}$ of the problem. The population set for this subproblem with n_p atoms will be

$$S = \{x_1^*, x_2^*, \cdots, x_N^*\},$$

where $N = 10n^{(p)}$ and $x_i^* \in I\!R^{n^{(p)}}$. The corresponding array A have $10n^{(p)} \times (n^{(p)} + 1)$ elements. After this subproblem has been solved its final minimizers x_i^* in S, except the function values, are passed into the generation process of the initial S of the next subproblem with (n_p+1) atoms. As the number of atoms increases in the subproblems, so is the size N of the set S. The additional coordinates of points in S^{ini} that are coming from the previous subproblem, as well as the coordinates of additional points are generated randomly. For instance, the subproblem with (n_p+1) atoms, S^{ini} have $10n^{(p+1)}$ members. Hence, $10n^{(p)} \times n^{(p)}$ coordinates in S^{ini} will be coming from the final S of the previous subproblem. The remaining $10n^{(p+1)} \times n^{(p+1)} - 10n^{(p)} \times n^{(p)}$ coordinates in S^{ini} are generated randomly within the search region Ω. The initial population set S for this subproblem is then formed by performing local searches from these $10n^{(p+1)}$ points in S^{ini}. The array A stores these minimizers with its final column containing the corresponding minimum values. Since DELP iteratively optimize each subproblem, the population sets S^{ini} and S are indexed respectively as S_j^{ini} and S_j for the j-th subproblem. We now present the DELP algorithm for potential minimization. Let the maximum number of atoms considered be p.

Algorithm-2 : The Local Search Based Differential Evolution Algorithm for Potential Minimization (DELP)

Step 1 **Initialize the subproblem counter, the number of atoms in the subproblem and the population sets**. Let S_j^{ini} denotes the population set of points, and S_j the population set of minimizers for the subproblem j. Set $j = 1$, $i = 1$, $k_{iter} = 1$, $n_p = 3$ and $N = 10n^{(p)}$.

Step 2 **Determine the initial set** S. If $j = 1$ generate the initial set S_j; proceed exactly the same way as in Step 1, Algorithm-1. Otherwise, generate the extended coordinates of points (minimizers) in S_{j-1}^{ini} and the full coordinates of the additional $N - 10n^{(p-1)}$ points in determining the set S_j^{ini}, and obtain $S_j = \{x_1^*, x_2^*, \cdots, x_N^*\}$ of local minimizers by performing local minimizations starting at each point in S^{ini}. Go to Step 4.

Step 3 **Determine the candidate point for minimization, Generate minimizers to replace minimizers in** S_j. Perform Step 2 and Step 3 of Algorithm-1.

Step 4 **Stop the algorithm for subproblem** j. Rank order the local minimum values in S_j from the best to the worst. If (5) is satisfied then go to Step 5, otherwise set $i = 1$, $k_{iter} = k_{iter} + 1$ and go to Step 3.

Step 5 **Stop the algorithm**. If $j = p - 2$ or $n_p = p$ then stop. Otherwise set $j = j + 1$, $n_p = n_p + 1$ and go to Step 2.

The Algorithm 1 (DELG) is embedded in the Algorithm 2 (DELP) as the DELP is iterative. We will present results and compare the performances of these algorithms with those of some previously proposed algorithms using LJ and MB problems, but first we present LJ and MB mathematically.

3 The Potential Energy Function

In this section we present two potential functions namely the LJ and MB potential functions. Both the potentials are differentiable. While the gradient of LJ function is easy to calculate (see [17]), the gradient of MB is explicitly given in [13]. The model parameter values for different parameters in MB can be found in [13, 3] and therefore will not also be repeated here.

3.1 The LJ Potential

The LJ potential energy between two atoms separated by distance r is given by

$$LJ(r) = \frac{1}{r^{12}} - \frac{2}{r^6}.$$

The total potential energy f_{LJ} of a cluster of n_p atoms is defined by

$$f_{LJ}(X_1, \cdots, X_{n_p}) = \sum_{i<j} LJ\left(\|X_i - X_j\|\right), \tag{6}$$

where $X_i \in I\!R^3$ represents the coordinates of the i-th atom and the norm is Euclidean.

3.2 Tersoff's MB Potential

The binding energy in the Tersoff formulation [3] is written as a sum over atomic sites in the form

$$E_i = \frac{1}{2} \sum_{j \neq i} f_c(r_{ij})(V_R(r_{ij}) - \beta_{ij} V_A(r_{ij})), \quad \forall i \tag{7}$$

where r_{ij} is the distance between atoms i and j (or X_i and X_j), V_R is a repulsive term, V_A is an attractive term, $f_c(r_{ij})$ is a switching function and β_{ij} is a many-body term that depends on the positions of atoms i and j and the neighbours of atom i. More details of each of these quantities can be found in [13, 3]. The term β_{ij} is given by

$$\beta_{ij} = (1 + \gamma^{n_1} \xi_{ij}^{n_1})^{-1/2n_1} \tag{8}$$

where n_1 and γ are known fitted parameters [3]. The term ξ_{ij} for atoms i and j (i.e., for bond ij) is given by

$$\xi_{ij} = \sum_{k \neq i,j} f_c(r_{ik}) g(\theta_{ijk}) E(r_{ij}, r_{ik}), \tag{9}$$

where

$$E(r_{ij}, r_{ik}) = \exp\left(\lambda_3^3 (r_{ij} - r_{ik})^3\right) \tag{10}$$

and θ_{ijk} is the bond angle between bonds ij and ik and g is given by

$$g(\theta_{ijk}) = 1 + c^2/d^2 - c^2/[d^2 + (h - \cos\theta_{ijk})^2]. \tag{11}$$

The quantities λ_3, c, d and h which appear in (10) and (11) are also known fitted parameters. The terms $V_R(r_{ij})$ and $V_A(r_{ij})$ are given by

$$V_R(r_{ij}) = Ae^{-\lambda_1 r_{ij}} \tag{12a}$$
$$V_A(r_{ij}) = Be^{-\lambda_2 r_{ij}} \tag{12b}$$

where A, B, λ_1 and λ_2 are given fitted parameters. The switching function $f_c(r_{ij})$ is given by

$$f_c(r_{ij}) = \begin{cases} 1\,, & r_{ij} \leq R - D \\ \frac{1}{2} - \frac{1}{2}\sin[\pi(r_{ij} - R)/(2D)]\,, & R - D < r_{ij} < R + D\,, \\ 0\,, & r_{ij} \geq R + D \end{cases} \tag{13}$$

values of R and D can also be found in [13, 3]. We consider two different MB problems, due to two different parameterizations of Tersoff potential for silicon. The two sets of parameter values respectively for $Si(B)$ and $Si(C)$ are taken from [3]. In order to calculate the minimum potential energy for, say n_p atoms we need to calculate the energy for each atom. Each atom, say atom i, has its own potential energy, E_i, given by (7). Therefore, to determine the potential energy of a single atom one has to calculate (7) which involves the calculation of (8)-(13) for each neighbour of that atom. Notice that the energy of a atom depends upon the distances and angles subtended with respect to the other atom and therefore different atoms have different energies. To formulate the problem we consider the atomic positions in two and three dimensional space as variables. It is clear that the total energy, say $f_{MB}(X_1, \cdots, X_{n_p})$, is a function of atomic coordinates and it is given by

$$f_{MB}(X_1, \cdots, X_{n_p}) = E_1(X_1, \cdots, X_{n_p}) + E_2(X_1, \cdots, X_{n_p}) + \cdots + E_{n_p}(X_1, \cdots, X_{n_p}), \tag{14}$$

for which the global minimum f^*_{MB} has to be found.

3.3 Problem Formulation

We have reduced the dimension of both potentials described by (6) and (14) in the following way. We first fix an atom at the origin and choose our second atom to lie on the positive x-axis. The third atom is chosen to lie in the upper half of the x-axis. Since the position of the first atom is always fixed and the second atom is restricted to the positive x-axis, this gives a minimization problem involving three variables for three atoms. For four atoms, additionally three variables (the Cartesian co-ordinates of the 4-th atom) are required to give a minimization problem in six independent variables. For each further atom, three variables (coordinates of the position of the atom) are added to determine the energetics of clusters. Let x be the variable of the problem which has three components for three atoms, six components for 4 atoms and so on.

The search region for all the problem is constructed in the following way. The first variable due to the second atom is taken as $x_1 \in [0, 4]$ and the second and third variables are such that $x_2 \in [0, 4]$ and $x_3 \in [0, \pi]$. The coordinates x_i due to any other atom are taken to lie on

$$\left[-4 - \frac{1}{4}\left\lfloor\frac{i-4}{3}\right\rfloor,\ 4 + \frac{1}{4}\left\lfloor\frac{i-4}{3}\right\rfloor\right], \tag{15}$$

where $\lfloor r \rfloor$ is the nearest least integer with respect to $r \in IR$. For instance, three variables (x_4, x_5 and x_6) due to the 4-th atoms lie on $[-4, 4]$ and the variables for the 5-th atoms on $[-4.25, 4.25]$ and so on. The entire region Ω will be the union of these subregions. The function due MB to be minimized now can be defined by

$$f_{MB}(x) = E_1(x) + \cdots, +E_{n_p}(x), \tag{16}$$

where $x \in \Omega \subset IR^{(3n_p-6)}$. The $f_{LJ}(x)$ can also be defined similarly [17].

4 Computational Results

In this section we present the numerical results obtained by DELG and DELP on both the problems. We also compare the results obtained by these algorithm with those obtained by TDE algorithm [14]

suggested for large scale potential minimization. Implementation of these algorithms require setting of some parameter values. They inherit these parameters from the original DE algorithm. Therefore, we set these parameter values according to the suggestions in [13, 14, 15]. For instance a good value for the population size N lies in $[7n, 15n]$. We have used $N = 10n$. The value of C_R is taken to be 0.5 [15]. A good choice of the scaling factor F can be calculated using a formula given in [14, 15]. All computations were carried out on a SGI-Indy Irix-5.3. A limited memory BFGS algorithm (LBFGS) of Lui and Nocedal [18] was used as the local search algorithm. The LBFGS is designed for large scale local optimization problems. The performance is measured by criteria based on the number of function evaluation (FE), the cpu time (cpu) and the total success (ts) in obtaining the global minimum. To show the robustness of our algorithm we first compare all algorithms on the LJ problem. We consider clusters of up to 10 atoms. There are 8 problems and 10 runs are conducted for each problem. This gives a total of 80 test runs[1]. Except a few cases, both DELG and DELP were successful in finding the global minimum for almost all runs. DELP fails only 5 times out of 360 runs and DELG 4 times out of 80 runs, all occurring on 6 atom cluster. However, the total successful runs, out of 80 runs, for DE and TDE are 59 and 71 respectively. Therefore, DELG and DELP methods are superior to the other two methods. The total number of successes for DELP (3 failures at inner levels and 2 failures at the outer levels for $n_p = 6$) and FELG are 78 and 76 respective. The results on FE and cpu are summarised in Table 1. In this table FE and cpu respectively represents the average number of function evaluations and average cpu times. The average is taken over successful runs, out of 10 runs, for which the algorithm obtained the best known minimizer [17]. From this table and the

Table 1: Comparison of FE and cpu using 80 runs for LJ

	DE		TDE		DELG		DELP	
n_p	FE	cpu	FE	cpu	FE	cpu	FE	cpu
3	1980	0.09	2524	0.18	1707	0.22	1973	0.24
4	5700	0.35	10716	1.23	8678	1.47	10359	1.57
5	19440	1.56	40968	4.54	18948	3.74	31443	5.47
6	119760	12.93	271044	14.11	106692	22.47	67953	15.52
7	525150	69.69	1360444	34.01	160757	40.32	103449	23.22
8	866700	142.40	9557877	154.44	593776	165.47	207776	51.89
9	3736740	685.78	14942176	717.55	1206893	382.09	1463183	470.48
10	37887600	8196.02	29063121	6012.07	6176557	3557.61	4653885	1711.68
Total	43163070	9108.82	55248870	6938.13	8177985	4173.39	6540021	2280.07

discussion above it is quite clear that both DELG and DELP are superior to DE and TDE in terms of FE, cpu and ts. Since both DELG and DELP were successful almost in all runs we compare them using the total FE and cpu given on the last row of Table 1. DELP is superior to DELG in terms of FE and cpu respectively by 20% and 45%. However, DELP produces cumulative FE and cpu. For instance, FE needed for $n_p = 10$ is the total number of function evaluations from the beginning (first cluster to the last cluster) to the end and therefore FE for $n_p = 10$ includes the FE for all subproblems. Therefore, FE and cpu for DELP are respectively given in brackets on the second last row of Table 1. If we compare total FE and cpu for DELG with those of DELP given in brackets we see that DELP is superior to DELG by 43% and 59% respectively in terms of FE and cpu. These results motivate us

[1]When DELP is applied to each of the 8 problems 10 times total number of test runs involved is 360. When DELP solves the 10 atom problem it iteratively solves 8 subproblems starting from 3 atom up to 10 atom problems. Therefore when we apply DELP 10 times to the 10 atom problem there are altogether a total of 80 test runs. When we apply DELP 10 times to the 9 atom problem there are a total of 70 test runs involved. Similarly, a total of 60 test runs for 8 atom problem, 50 for 7 atom problem and so on up to a total of 10 test runs involved in the 3 atom problems. We took average FE and cpu for all successful test runs.

to conduct more test runs using bigger clusters of atoms of 11 to 15 atoms. We ran DELG and DELP 5 times on each of these problems. The number of problems is now 5 and a total of 25 test runs[2]. DELP was successful in all test runs (including the inner ones) and DELG failed 6 times. DELP was superior (in successful runs) to DELG by 45% in FE and by 60% in cpu. Moreover, the robustness of DELG gradually fell off as we added more atoms in the cluster. For example, we ran both DELG and DELP on LJ using clusters for 16 to 30 atoms. In these problems DELP again outperforms DELG in all respect, i.e., in terms of ts, FE and cpu. We ran these two algorithms 5 times on each of the 5 problems. In these 75 runs DELP was successful (including the inner level runs) in all runs, and DELG was successful in 55 runs. We further tested DELP using LJ 5 times on each problems of clusters of 16 to 30 atoms. In these runs DELP only fails only 4 times, all only at the inner levels. These failures occur for 19 and 23 atom problems. However, the best minimum obtained in these runs were -72.60 and -92.63 respectively for $n_p = 19$ and $n_p = 23$, against their best known minimum respective values -72.66 and -92.84. The total number of failures for DELG was 26.

Table 2: Global minimum values for MB found by DELP

	f^*_{MB}			f^*_{MB}	
n_p	$Si(C)$	$Si(B)$	n_p	$Si(C)$	$Si(B)$
3	-5.33	-7.87	17	-57.72	-72.10
4	-8.63	-15.70	18	-60.94	-78.54
5	-12.43	-20.39	19	-64.37	-82.86
6	-15.80	-25.65	20	-67.60	-87.31
7	-18.66	-30.39	21	-71.43	-90.95
8	-22.27	-36.26	22	-74.51	-94.8
9	-26.20	-40.67	23	-77.60	-99.63
10	-29.26	-45.19	24	-82.55	-103.44
11	-32.83*	-46.89	25	-86.19	-109.28
12	-36.36*	-53.02	26	-89.56	-112.87
13	-41.54*	-55.38	27	-92.11	-116.53
14	-44.67*	-59.43	28	-95-56	-120.14
15	-48.73*	-63.62	29	-98.75	-124.32
16	-53.99	-67.86	30	-102.22	-128.03

The above numerical experiments on LJ problems motivated us to apply DELP to optimize a more complicated potential, namely the MB potential. Our previous experiment with this potential using only 15 atoms [12, 13] has shown that MB is a much harder problem than LJ. In these studies using MB for 100 independent runs we noticed that $Si(C)$ was difficult than $Si(B)$, it took on average more FE and cpu. Indeed, it was difficult to locate the very best minimum for $Si(C)$, for all 15 atoms considered. Unlike LJ, the best known minimum values for MB are not known except the minimum values for up to 15 atoms reported in [13]. We have used DELP for the optimization of MB potential using up to 30 atoms involving 84 variables. The minimum values have been found by DELP in 5 independent runs are given in Table 2. In Table 2 the minimum values with asterisk marks indicates the best minimum found by DELP than that reported in [13], where results of up to 15 atoms are given.

[2]DELP has 75 test runs as defined in footnote 1 and DELP was successful in all these runs

5 Conclusion

We have developed a new global optimization algorithm for large scale problems and the algorithm has been implemented on difficult optimization problems involving potentials functions. The MB potential function is complex in that the interacting forces are many-body and angle-dependent. The global optimization of silicon potentials for up to 30 atoms consisting of up to 86 variables is carried out. The new algorithm was able to produce better minima for some problems than those previously found. The algorithm can be applied to large scale optimization problems in other areas of application such as problems in biological chemistry and in plasma physics. However, the new algorithm has become rapidly slow as we have added more than 30 atoms in the cluster. This is partly because of the size of the array A. Therefore, designing a more efficient algorithm for even larger problems will form the basis of our future research. In particular, we are working in designing a new algorithm for large scale potential problem that will combine the current features of DELP with as much as prior knowledge as possible on the problem domain. In this way we would be able to work with a managable size of the array A. One of our objective is to inform the global optimization communities of the more realistic MB potential and we hope that more researchers will investigate this potential alongside the LJ potential.

Acknowledgements

This paper was written at IMA, University of Minnesota, USA, where the second author participated the IMA Program on Optimization for a year. Financial support from IMA is acknowledged. The authors thank Professor Kar Wong of Witwatersrand University for several helpful comments on the first draft of this work.

References

[1] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", Journal of Global Optimization, vol. 11, pp. 341-359, 1997.

[2] J. Tersoff, "New Empirical approach for the Structure and Energy of Covalent Systems", Physics Review B, vol. 37, pp. 6991-7000, 1988.

[3] J. Tersoff, "Empirical Interatomic Potential for Silicon with improved Elastic Properties", Physics Review B, vol. 38, pp. 9902-9905, 1988.

[4] M. M. Ali and R. Smith, "The Structures of Small Cluster Ejected by Ion Bombardment of Solids", Vacuum, vol. 44, pp. 377-379, 1993.

[5] H. A. Scheraga, "Recent Developments in the Theory of Protein Folding : Searching for the Global Minimum", Biophysical Chemistry, vol. 59, pp. 329-339, 1996.

[6] P. M. Pardalos, D. Shalloway and G. L. Xue, "Optimization Methods for Computing Global Minima of Non-convex Potential Energy Function", Journal of Global Optimization, vol. 4, pp. 117-133, 1994.

[7] J. J. Moré and Z. Wu, "Global Smoothing and Continuation for Large Scale Molecular Optimization", MCS-P539-1095, 1995.

[8] J. Kostrowicki, L. Piela, B. J. Cherayil and A. Scheraga, “Performance of the Diffusion Equation Method in Searches for Optimum Structures of Clusters of Lennard-Jones Atoms”, Journal of Physical Chemistry, vol. 95, pp. 4113-4119, 1991.

[9] M. Locatelli and F. Schoen, “Fast Global Optimization of Difficult Lennard-Jones Clusters”, Computational Optimization and Applications, vol. 21, pp. 55-70, 2002.

[10] G. L. Xue, “Molecular Conformation on the CM-5 by Parallel Two-level Simulated Annealing”, Journal of Global Optimization, vol. 4, pp. 187-208, 1994.

[11] D. M. Deaven, N. Tit, J. R. Morris and K. M. Ho, “Structural Optimization of Lennard-Jones Clusters by a Genetic Algorithm”, Chemical Physics Letters, vol. 256, pp. 195-198, 1996.

[12] Ali, M. M., Storey, C. and Törn, A. (1997), “Application of Stochastic Global Optimization Algorithms to Practical Problems”, *Journal of Optimization Theory and Applications, Vol. 95*, 545-563.

[13] M. M. Ali and A. Törn, “Optimization of Carbon and Silicon Cluster Geometry for Tersoff Potential using Differential Evolution”, in Optimization in Computational Chemistry and Molecular Biology : Local and Global Approaches, C.A. Floudas and P.M. Pardalos (Eds.), Kluwer Academic Publisher, 2000, pp.287-300.

[14] M. M. Ali and A. Törn, “Topographical Differential Evolution Using Pre-calculated Differentials”, in Stochastic and Global Optimization, G. Dzemyda, et. al. (Eds.), Kluwer Academic Publisher, 2002, pp.1-17.

[15] M. M. Ali and A. Törn, “Population set based Global Optimization Algorithms : Some modifications and Numerical Studies”, to appear in Computers and Operations Research, 2003.

[16] Y. F. Hu, K. C. Maguire, D. Cokljat and R. J. Blake, Parallel Controlled Random Search Algorithms for Shape Optimization, in Parallel Computational Fluid Dynamics, Eds., D. R. Emerson, A. Ecer, J. Periaux and N. Satofuka, pp.345-352, North Holland, 1997.

[17] N. P. Moloi, “A Local Search Based Differential Evolution Algorithm for Potential Minimization”, *MSc Dissertation*, School of Computational and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa, 2002.

[18] D. C. Lui and J. Nocedal, “On the Limited Memory BFGS Method for Large Scale Optimization”, Mathematical Programming, vol. 45, pp. 503-528, 1989.