Parallel Continuation-Based Global Optimization for Molecular Conformation and Protein Folding*

Thomas F. Coleman† and Zhijun Wu‡

Abstract. This paper presents our recent work on developing parallel algorithms and software for solving the global minimization problem for molecular conformation, especially protein folding. Global minimization problems are difficult to solve when the objective functions have many local minimizers, such as the energy functions for protein folding. In our approach, to avoid directly minimizing a "difficult" function, a special integral transformation is introduced to transform the function into a class of gradually deformed, but "smoother" or "easier" functions. An optimization procedure is then applied to the new functions successively, to trace their solutions back to the original function. The method can be applied to a large class of nonlinear partially separable functions including energy functions for molecular conformation and protein folding. Mathematical theory for the method, as a special continuation approach to global optimization, is established. Algorithms with different solution tracing strategies are developed. Different levels of parallelism are exploited for the implementation of the algorithms on massively parallel architectures.

Abbreviated title: Parallel Continuation-Based Global Optimization

Key words: global/local minimization, numerical continuation, parallel computation, protein folding

AMS (MOS) subject classification: 49M37, 65Y05, 68Q22, 92-08

*To be presented at Supercomputing '94, November, 1994, Washington D.C.

†Department of Computer Science and Center for Applied Mathematics, Cornell University, Ithaca, NY 14853.

‡Advanced Computing Research Institute, Cornell University, Ithaca, NY 14853.

1 Motivation

We are developing massively parallel algorithms and software for molecular conformation, especially protein folding. This paper reports on our recent progress.

The prediction of protein native structures and the understanding of how they fold from sequences of their constituent amino acids is one of the most important and challenging computational science problems of the decade. The protein folding problem is fundamental to almost all theoretical studies of proteins and protein related life processes. It also has many applications in the biotechnology industries such as structure-based drug design for the treatment of important diseases like polio, cancer, and AIDS.

Optimization approaches to the protein folding problem are based on the hypothesis that the protein native structure corresponds to the global minimum of the protein energy. The problem can be attacked computationally by minimizing the protein energy over all possible protein structures. The structure with the lowest energy is presumed to be the most stable protein structure.

Mathematically, for a protein molecule of n atoms, let $x = \{x_i \in \mathbf{R}^3,\ i = 1, \ldots, n\}$ represent the molecular structure with each x_i specifying the spatial position of atom i. Then the computational problem for protein folding is to globally minimize a nonlinear function $f(x)$ for all $x \in \mathbf{S}$, i.e.,

$$\mathbf{min}_{x \in \mathbf{S}}\ \ f(x) \tag{1}$$

where $\mathbf{S}$ is the set of all possible molecular structures, and $f(x)$ is the energy function for the protein defined for all x.

The difficulty with this approach is that global optimization problems are computationally intractable in general, and especially difficult to solve when problem sizes are large and objective functions contain many local minimizers. For protein folding, the problem sizes tend to be very large with possibly thousands of variables, and the objective functions usually have exponentially many local minimizers. Therefore, to solve the optimization problems for protein folding, special algorithms must be developed which exploit the problem structure. In addition, parallel high performance computing is also essential for the solutions to be computationally feasible.

Our work focuses on establishing a new continuation-based approach to global optimization; we develop efficient parallel algorithms and software specifically for molecular conformation and protein folding.

2 The basic approach

The idea behind our approach is the following. To avoid directly minimizing a "difficult" objective function, a smoothing technique is introduced to transform the function into a class of gradually deformed, but "smoother" or "easier" functions. An optimization procedure is then applied to the new functions successively, to trace their solutions back to the original function.

To obtain our smoothing transformation, a parametrized integral transformation is introduced, transforming a given function into a class of new functions corresponding to a set of parameter values. A transformed function is in some sense a coarse approximate to the original function. After applying the transform, the original function becomes smoother with small and narrow minimizers being removed while the overall structure of the function is maintained. This allows a solution tracing procedure to skip less interesting local minimizers, and concentrate on regions with average low function values where a global minimizer is most likely to be located.

Different methods can be employed to trace the solutions. For example, a simple method is to apply a random search procedure to the transformed functions successively to locate their low local minimizers. Another possible method is to apply local optimization procedures to each transformed function and trace a set of local minimizers.

Our approach is called continuation-based, because the transformation can actually be viewed as a special continuation process by the theory described in [7]. Following this theory, our new approach can be studied in a general numerical continuation setting, and algorithms can be developed by employing standard advanced numerical methods. We will discuss these issues later in this paper.

3 Transformation

We first introduce the transformation.

Definition 1 *Given a nonlinear function f, the transformation $< f >_\lambda$ for f is defined such that for all x,*

$$< f >_\lambda (x) \quad = \quad \mathcal{C}_\lambda \int f(x')\; e^{-\|x-x'\|^2/\lambda^2}\; dx', \qquad (2)$$

or equivalently,

$$< f >_\lambda (x) \quad = \quad \mathcal{C}_\lambda \int f(x-x')\; e^{-\|x'\|^2/\lambda^2}\; dx', \qquad (3)$$

where λ is a positive number and $\mathcal{C}_\lambda$ is a normalization constant such that

$$\mathcal{C}_\lambda \int e^{-\|x\|^2/\lambda^2}\ dx = 1. \tag{4}$$

To understand this transformation, consider that given a random function $g(x')$ and a probability distribution function $p(x')$ for the random variable x', the expectation of the function g with respect to p is

$$<g>_p = \int g(x')\ p(x')\ dx'. \tag{5}$$

In light of (5), the defined transformation (2) yields a function value for $< f >_\lambda$ at any x equal to the expectation for f sampled by a Gaussian distribution function centered at x.

For example, consider the following nonlinear function:

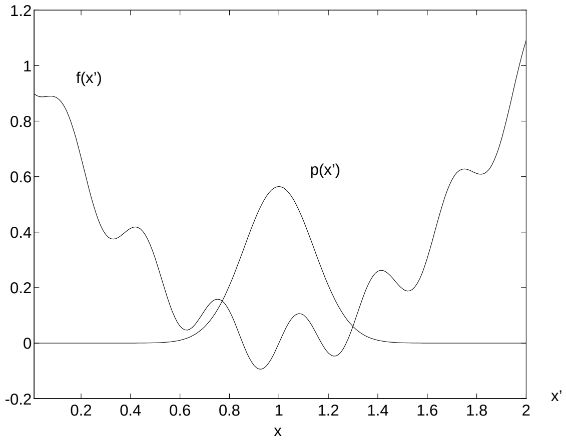
$$f(x) = (x-1)^2 + 0.1 sin20(x-1) \tag{6}$$

which is a quadratic function augmented with a "noise" function. The transformation for this function can be computed:

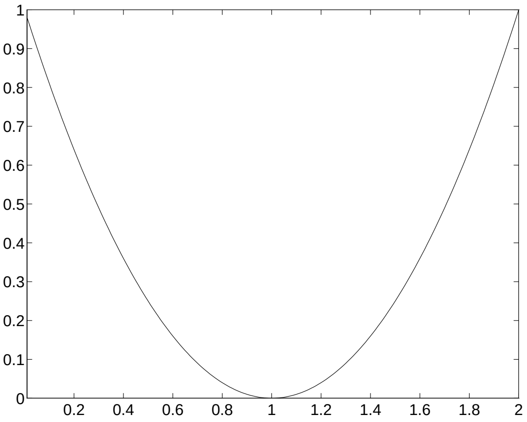
$$<f>_\lambda (x) = (x-1)^2 + \frac{\lambda^2}{2} + 0.1e^{-(20\lambda)^2/4} sin20(x-1). \tag{7}$$

The function value $< f >_\lambda (x)$ for fixed x is equal to the integration with respect to the product of two functions, the original function $f(x')$ and the Gaussian distribution function $p(x') = \mathcal{C}_\lambda e^{-\|x-x'\|^2/\lambda^2}$ (Figure 1 (a)), where λ determines the size of the dominant region of the Gaussian. Since the most significant part of the integration is that within the dominant region of the Gaussian, $< f >_\lambda (x)$ can be viewed as the average value for the original function f within a small λ-neighborhood around x. If λ is equal to zero the transformed function is exactly the original function. Otherwise, original function variations in small regions are averaged out, and the transformed function will become "smoother" (Figure 1 (b)).

Figure 2 shows how the function $< f >_\lambda$ in (7) behaves with increasing λ. Observe that when $\lambda = 0.0$ the function is the original function; when we increase λ to 0.1, the function becomes "smoother"; when λ is increased further to 0.2, the function becomes entirely "smooth". As we will show in the following sections, what we observed here is a general property of the transformation, i.e., for any function f, the larger of λ, the "smoother" the transformed function.

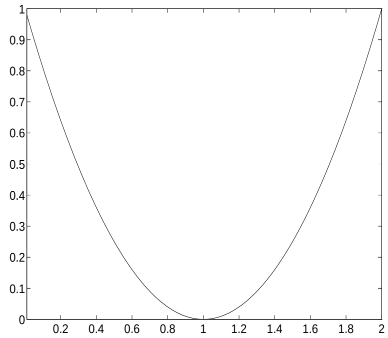


(a)

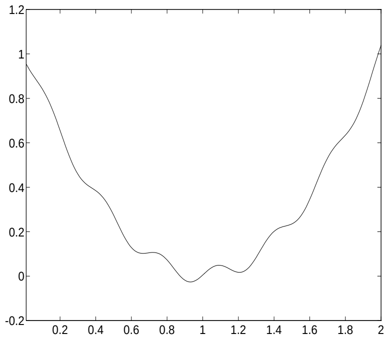


(b)

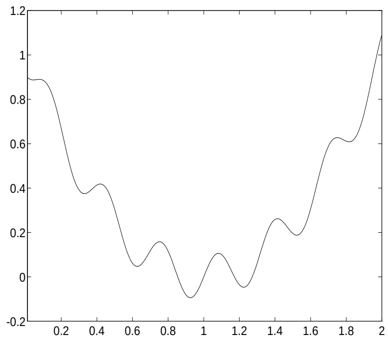
Figure 1: A 1-dimensional transformation example.



(a) $\lambda = 0.2$



(b) $\lambda = 0.1$



(c) $\lambda = 0.0$

Figure 2: A class of gradually deformed functions.

4 Smoothness

Let $\hat{f}$ be the Fourier transformation for function f, and $\widehat{<f>}_\lambda$ the Fourier transformation for function $<f>_\lambda$. Recall that the transformation $<f>_\lambda$ for f is just a convolution of f and p, where p is the Gaussian distribution function

$$p(x) = \mathcal{C}_\lambda \; e^{-\|x\|^2/\lambda^2}. \tag{8}$$

Therefore the Fourier transformation for $<f>_\lambda$ is equal to the product of the Fourier transformations for f and p. The Fourier transformation for the Gaussian distribution function is

$$\hat{g}(\omega) = e^{-\frac{\lambda^2\|\omega\|^2}{4}}. \tag{9}$$

So, we have

$$\widehat{<f>}_\lambda(\omega) = e^{-\frac{\lambda^2\|\omega\|^2}{4}} \hat{f}(\omega). \tag{10}$$

We see from (10) that if $\lambda \to 0$, $\widehat{<f>}_\lambda$ converges to $\hat{f}$, and $<f>_\lambda$ converges to f.

Also by (10), for fixed λ, if ω is large $\widehat{<f>}_\lambda(\omega)$ will be very small. This implies that high frequency components of the original function become very small after the transformation. This is why the transformed function is "smoother". In addition, for larger λ values, wider ranges of high frequency components of the original function practically vanish after the transformation. Therefore, the transformed function becomes increasingly smooth as λ increases. We state these properties formally in the following theorem.

Theorem 1 *Let f, $\hat{f}$, $<f>_\lambda$ and $\widehat{<f>}_\lambda$ all be given and well defined. Then $\forall \varepsilon > 0$, $\exists \delta \propto 1/\lambda$ for fixed λ, such that $\forall \omega$ with $\|\omega\| > \delta$,*

$$\frac{|\widehat{<f>}_\lambda(\omega)|}{|\hat{f}(\omega)|} < \varepsilon. \tag{11}$$

Proof: See [7]. □

From this theorem we learn that the relative size of $\widehat{<f>}_\lambda(\omega)$ can be made arbitrarily small for all ω with $\|\omega\|$ greater than a small value δ. Since δ is inversely proportional to λ, high frequency components are removed when λ is large.

5 Numerical Properties

The definition of the transformation (2) involves high dimensional integration which cannot be computed in general (except perhaps by the Monte Carlo method which is not appropriate for our purposes because it is too expensive). So the transformation may not be applicable to arbitrary functions, at least numerically. However, this transformation does apply to a large class of nonlinear partially separable functions, and especially to typical molecular conformation and protein folding energy functions.

Consider a large class of nonlinear partially separable functions, called generalized multilinear functions,

$$f = \sum_i \prod_j g_j^i, \tag{12}$$

where g_j^i's are one dimensional nonlinear functions. It is easy to verify that

$$< f >_\lambda = \sum_i \prod_j < g_j^i >_\lambda. \tag{13}$$

Since transformation $< g_j^i >_\lambda$, for all i and j, involves only one dimensional integration, the transformation for a generalized multilinear function can be numerically computed.

In particular, let us consider a typical n-atom molecular conformation energy function,

$$f(x) \;=\; \sum_{i=1,j>i}^{n} h_{ij}(\|x_i - x_j\|) \tag{14}$$

where $x = \{x_i \in \mathbf{R}^3,\ i = 1, \ldots, n\}$ and h_{ij} is the pairwise energy function determined by $\|x_i - x_j\|$, the distance between atoms i and j. Because of the partial separability of this type of function, the transformation for f is equal to the sum of the transformations for the pairwise functions h_{ij}. However the computation for the pairwise transformation still cannot be conducted directly, because there is still more than one variable. Nevertheless, the following theorem provides a feasible way to compute the molecular energy transformation:

Theorem 2 *Let f be defined as in (14). Then the transformation (2) for f can be computed using the formula*

$$< f >_\lambda (x) = \sum_{i=1,j>i}^{n} < h_{ij} >_{\sqrt{2}\lambda} (\|r_{ij}\|) \tag{15}$$

where $r_{ij} = x_i - x_j$ *and*

$$< h_{ij} >_{\sqrt{2}\lambda} (\|r_{ij}\|) = c_{\sqrt{2}\lambda} \int h_{ij}(\|r'_{ij}\|) e^{-\|r_{ij}-r'_{ij}\|^2/2\lambda^2} dr'_{ij}. \qquad (16)$$

Proof: See [7]. □

Note that $< h_{ij} >_{\sqrt{2}\lambda} (\|r_{ij}\|)$ can be computed with a standard numerical integration technique; therefore, the transformation $< f >_\lambda (x)$ can be computed in this fashion.

6 Minimization

In summary, we have introduced a parametrized integral transformation to transform the object function of a global optimization problem. Statistically, the transformation averages the function values, and provides coarse estimates for the function variation. Geometrically, the transformation deforms the function into a class of "smoother" functions with small high frequency components removed in the transformed functions. Physically, the transformation allows a physical system to have small perturbations, and the transformed function reflects the average behavior of the system dynamics. Finally, the transformation can exploit partial separability, and is particularly suitable for molecular conformation and protein folding energy functions.

With this transformation, a general global minimization procedure can immediately be constructed as illustrated in Figure 3. That is, given a global minimization problem with a nonlinear objective function f, we first transform the function into a class of new functions $< f >_{\lambda_1}$, $< f >_{\lambda_2}$, ..., $< f >_{\lambda_m}$ for $\lambda_1 > \lambda_2 > \ldots > \lambda_m = 0$ with $< f >_{\lambda_m}$ corresponding to f. We then apply local optimization procedures to the transformed functions successively, to trace their solutions back to the original function. Since the transformed function with a larger λ value is "smoother" with possibly fewer local minimizers, we can start by minimizing $< f >_{\lambda_1}$, and next, take its solution as the initial point and minimize $< f >_{\lambda_2}$, and so on and so forth. Since a transformed function is also a coarse approximate to the original function, its solution should also be a rough estimate for the solution of the original function. So by minimizing the transformed functions successively, 1 **Choose**

$$\{\lambda_i : \ i = 1, \ldots, m, \ \lambda_1 > \ldots > \lambda_m = 0\}$$

2 **For** $i = 1, \ldots, m$

$$\mathbf{min}_{x \in \mathbf{S}} \ <f>_{\lambda_i} (x)$$

Figure 3: A global minimization procedure.

the whole process is concentrated in regions where the solution of the original function is most likely to be located.

7 Tracing Solutions

The continuation-based global minimization approach contains two major components:

1. Application and computation of the transformation (2),
2. A solution tracing procedure.

Clearly, different algorithms can be implemented if different solution tracing procedures are employed. An efficient solution tracing method is crucial for the algorithm to be numerically effective and efficient.

In principle, tracing solutions means tracing global minimizers: the solution for a global minimization problem is sought for each transformed function. However, in a broader sense, the solutions can actually be either global or local, as long as they form a "path" that can lead to a global minimizer for the original objective function. Under some circumstances, such a "path" exists as a smooth curve, and then tracing solutions simply implies following a smooth solution curve determined by a set of transformed functions.

A random search procedure is an example of a simple solution tracing method, e.g., the simulated annealing random search [1]. This method is easy to implement, and especially robust in the sense that the random search procedure can be designed to converge asymptotically to a global minimizer. However, convergence depends on how thoroughly the search can be conducted. Usually, an unaffordable amount of computation is required even for small problems. Another problem with this method is that the randomness introduces uncertainty.

A more deterministic and efficient alternative is to use a local minimization procedure. This method applies local minimization to the transformed functions successively, and returns a local minimizer as the candidate for the solution to the given problem. The method is relatively inexpensive, and clearly more feasible for large scale problems, e.g., the protein problems. In particular, it can take advantage of well-developed local optimization techniques [6].

The effectiveness of this method can be illustrated in the following simple experiment: Consider the function in (6), and suppose that we want to find its global minimizer. First we transform the function to obtain a class of new functions given in (7). Choose $\lambda_1 = 0.2$, $\lambda_2 = 0.1$ and $\lambda_3 = 0.0$. We then have three transformed functions as shown in Figure 2 (a), (b) and (c). The function in Figure 2 (c) is equivalent to the original function. Then we apply a local minimization procedure to the transformed functions from $<f>_{\lambda_1}$ to $<f>_{\lambda_3}$. Since $<f>_{\lambda_1}$ is "smooth" with only one local minimizer, the solution can immediately be found for it. Started from this solution, a local minimizer, being also a global minimizer, for $<f>_{\lambda_2}$ can be found subsequently. Continuing the process, the global minimizer for the original function can be located at the end.

The example shows that the local minimization skips small local minimizers at the first stages and goes directly to a region of interest, where a global minimizer is very likely to be found subsequently. In general, the method may not always be this fortunate. For example, the early transformed functions may still have more than one local minimizer; the chosen minimizer may not necessarily lead to a global minimizer for the function at the final stage.

To begin with the "right local minimizer", either a good initial point is provided based on the known knowledge of given problem, or a set of local minimizers can be selected and traced, and one of them may lead to a good solution.

8 Numerical Continuation

Our recent work [7] shows that the parametrized integral transform in (2) defines for f a homotopy on $[0, \lambda_0]$ for any $\lambda_0 < \infty$. Moreover, under appropriate assumptions, the transformed functions $\{< f >_\lambda: \ \lambda \in [0, \lambda_0]\}$ determine for any given local minimizer x_0 of $< f >_{\lambda_0}$ a continuous and differentiable curve $x(\lambda)$ so that for all $\lambda \in [0, \lambda_0]$, $x(\lambda)$ is a local minimizer of $< f >_\lambda$. In this case, the deterministic trace of the solution, e.g., using local minimization, is equivalent to following a solution curve $x(\lambda)$ (or a set of such curves). This forms the theoretical basis for our method as a special continuation approach to global optimization. Therefore, an initial value problem to determine the solution curve can be derived in a simple and computable form:

$$
\begin{aligned}
x' &= -\frac{\lambda}{2} < \nabla^2 f >_\lambda^{-1} (x) < \Delta g >_\lambda (x) && (17)\\
x_0 &= x(\lambda_0) && (18)
\end{aligned}
$$

where $\nabla^2 f$ is the Hessian of the function, and Δg the Laplace operation applied to the components of the gradient. This result opens another direction for the effective trace of the solution - solve the initial value problem using standard numerical IVP-methods, e.g., the predictor-corrector methods [2]. One simple example is to use an Euler-Newton method as shown in Figure 4. In this method, at each iteration, an Euler predictor is computed to start a Newton's local minimization procedure to find a solution on the curve. The process is continued, and the solution curve is followed to its end.

9 Parallelism

Different levels of parallelism can be exploited for continuation-based global optimization, e.g., parallel solution tracing, parallel function evaluation, and parallel linear algebra and optimization.

At the solution tracing level, parallelism can be exploited by using multiprocessors to generate multiple random searches, or trace a set of local minimizers in parallel. For the random search technique, increasing the number of processors is equivalent to increasing the number of trials. The more processors that are used, the higher the probability a solution can be found. For tracing multiple local minimizers, using multiprocessors simply reduces the total computation and increases the potential for finding the best

$\lambda = \lambda_0, \quad x = x_0$

Repeat

Compute $x' = -\frac{\lambda}{2} \; < \nabla^2 f >_\lambda^{-1} (x) \;\; < \Delta g >_\lambda (x)$
$\lambda = \lambda + h, \quad x = x + x' \; h$

Repeat

Compute $s = - < \nabla^2 f >_\lambda^{-1} (x) \;\; < g >_\lambda (x)$
$x = x + \alpha \; s$

End

End

Figure 4: Euler-Newton prediction and correction.

possible local minimizer. In either case, the parallelism is coarsely grained with little communication required among processors but intensive computation for each, which is good for massively parallel computation, especially on the machines with high communication to computation ratios.

Parallel function evaluation is important for both local and global optimization. For the continuation-based global optimization method, more than half of the total computation involves function evaluation, and each evaluation is costly, requiring numerical integration. However, for molecular conformation and protein folding, the energy functions to be minimized are partially separable with typically a small number of element functions. So for each element function, we can construct a function value look-up table. The function evaluation can then be conducted with cubic spline interpolation using the function values already calculated in the look-up tables. In this way, the total function evaluation cost can be reduced; moreover, the function value look-up tables, no matter how expensive they are, can be computed in parallel with perfect parallel efficiency. In this sense, we say that the function evaluation can be indirectly parallelized.

Finally, the continuation-based global optimization method is rich in linear algebra which is good for high performance computing. When the problem is large, say, the problem for a protein with ten thousand atoms, the parallelism at this level can also be exploited by parallelizing the major linear algebra operations, e.g., linear system solve and local minimization. This type of parallelism has been well studied and understood, and can be exploited using standard techniques.

10 Numerical Experience

The development of the continuation-based approach to global optimization has been accompanied with a series of computational works [3, 4, 5]. The algorithms have been implemented on parallel machines and tested with a set of molecular conformation problems. The results we obtained support the approach, and show that the algorithms perform much more effectively and efficiently than conventional global optimization methods. They are also very suitable for massively parallel computation. We illustrate in the following some of our numerical experience with two particular algorithms. Both methods are continuation-based, but differ in solution tracing strategies.

The first method, called the effective energy simulated annealing, uses a random search procedure, the simulated annealing method, to trace the solutions. Recall that in the simulated annealing method, a temperature parameter T is decreased from a positive number to zero as the iteration count increases. For each value of T, a number of random trials is applied to the given energy function. For the effective energy simulated annealing method, a function $\lambda = \alpha\, T$ first is defined, where α is a constant. For each value of T, a λ value is determined, which, in turn, defines a transformed function, called the effective energy function. A number of random trials is then conducted on this function to locate a solution. The parameter λ goes to zero as T decreases, and the transformed function changes to the original function. The process is equivalent to tracing the solutions for a set of transformed functions using the Monte Carlo search with a different temperature T for each different transformed function. Note that if α is set to zero, λ is equal to zero for all T. In this case all transformed functions are the same original function, and the algorithm is reduced to a standard simulated annealing procedure.

The effective energy simulated annealing algorithm has been implemented on a 32-node Intel iPSC/860 at Cornell. The machine is a parallel distributed memory system with a hypercube interconnection network. Each processor has 8 Mbytes of local memory, and achieves a theoretical peak performance of 40 Mflops. The parallelization of the algorithm is straightforward: Multiple processors are used at each iteration to generate multiple sequences of trials independently. Little communication is required among processors except for calculating the global acceptance rate at the end of each iteration. The load also is well balanced: the number of trials is the same each processor. For more implementation details, readers are referred to [3].

The algorithm is tested with a set of small sizes of Lennard-Jones microcluster conformation problems, which have been well studied, and widely used as model problems for molecular conformation. Typical results for these problems are shown in Figure 5, where three pictures for clusters of $n = 8, 12, 16$ atoms are given. The curves indicate the energy levels for the solutions obtained by the algorithm with different α values. We see when α is equal to zero, the algorithm corresponding to a standard simulated annealing procedure can only find solutions with very high energy levels. However, within the same amount of computation time, the effective energy simulated annealing algorithm with a proper choice of positive λ value can find solutions whose energy levels are already very close to the best known values (the bottom lines of the pictures). As a matter of fact, by applying a local minimization procedure started with these solutions, we obtained immediately the best known solutions for all the clusters. These results just show how effective the method with the transformation scheme can be for molecular conformation, compared with a conventional global optimization technique.

The parallel performance for the algorithm is illustrated in Figure 6, where two examples are given to show how rapidly the energy levels of the solutions found by the algorithm decrease with increasing numbers of processors.

The second algorithm we want to discuss is the deterministic local tracing algorithm, which uses local minimization as a solution tracing procedure. The algorithm first requires the objective function to be transformed into a class of new functions $<f>_{\lambda_1}$, $<f>_{\lambda_2}$, ..., $<f>_{\lambda_m}$ for a set of parameter values $\lambda_1 > \lambda_2 > \ldots > \lambda_m = 0$, with $<f>_{\lambda_m}$ corresponding to f. A set of starting points are sampled randomly so that a group of local minimizers for $<f>_{\lambda_1}$ are obtained at the beginning. Then local minimization is applied to the remaining transformed functions successively to trace the changes of these local minimizers, and the one with the lowest function value is selected at the last stage as a candidate for the solution to the given problem.

The deterministic local tracing algorithm has been implemented on a 64-node IBM SP1 at Cornell. The SP1 is a parallel distributed memory system with a high performance switch installed for better interprocessor communication. Each processor is an IBM RS/6000 with 128 Mbytes of memory and a peak performance of 125 Mflops. In this implementation, multiprocessors are used to trace multiple local minimizers in parallel with one local minimizer for each processor. Little communication is required. Each processor carries a sequence of local minimizations. Basically, the more processors used, the more local minimizers traced, and hence the higher the probability of obtaining a good solution. Also, the larger the problem sizes, the more intensive the computation for each processor. Since the problem sizes of practical interest tend to be very large, the machines with high communication to computation ratios, such as the IBM SP1, can be very suitable for the algorithm to achieve good performance in practice.

The algorithm has been tested with a set of "perturbed Lennard-Jones microcluster conformation problems". Such a problem is obtained by adding in each pairwise Lennard-Jones potential function a periodically varying term, $\rho \sin(\omega r)/r$, where ρ and ω are constants, and r is the distance between given pair of atoms. The functions with properly adjusted ρ and ω can generate a set of even more complicated global optimization test problems. The perturbed functions reduce to pure Lennard-Jones problems when ρ is

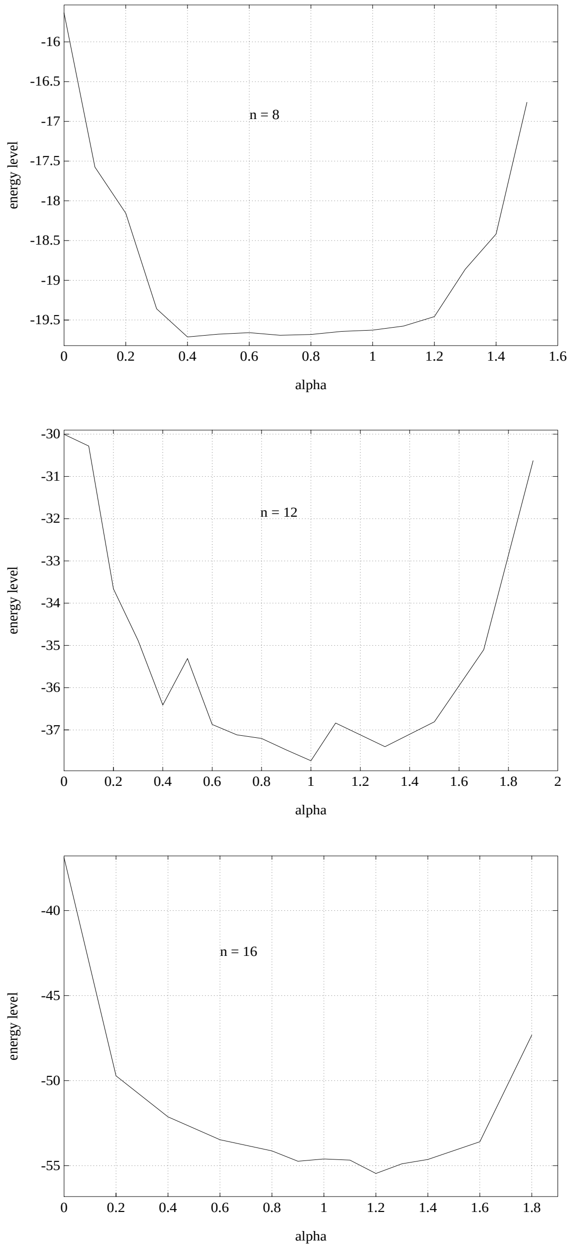
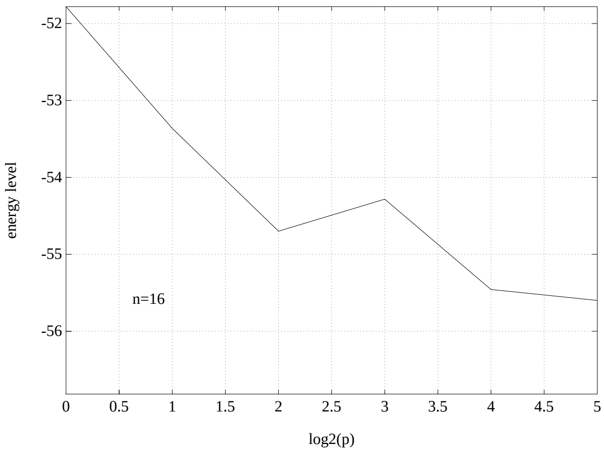
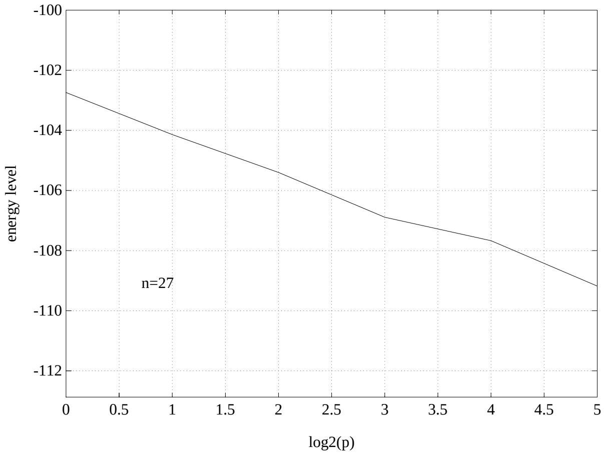


Figure 5: Typical numerical results obtained by the effective energy simulated annealing algorithm.



(a)



(b)

Figure 6: The parallel performance of the effective energy simulated annealing algorithm.

Deterministic Local Tracing						
p	$n = 16$		$n = 20$		$n = 24$	
	$m = 1$	$m = 40$	$m = 1$	$m = 40$	$m = 1$	$m = 40$
1	-4.2805e1	-5.7933e1	-5.2270e1	-7.6255e1	-1.0112e2	-1.0312e2
2	-5.5878e1	-5.6551e1	-7.4508e1	-8.0626e1	-1.0129e2	-1.0048e2
4	-5.8068e1	-6.0420e1	-7.6577e1	-7.9048e1	-1.0555e2	-1.0419e2
8	-5.8068e1	-6.1350e1	-7.7593e1	-7.9561e1	-1.0250e2	-1.0419e2
16	-5.8068e1	-6.1350e1	-8.0518e1	-8.3793e1	-1.0411e2	-1.0604e2
32	-6.1350e1	-6.1350e1	-8.3664e1	-8.3793e1	-1.0463e2	-1.0604e2

Table 1: Energy values obtained by the deterministic local tracing method for the perturbed Lennard-Jones problems.

set to zero. In this test, ρ is set to 1, and ω to 10.

Table 1 lists the results for some example problems (n=16,20,24), obtained by the algorithm using different numbers of processors (p). The data in the table are the energy values for the solutions obtained by the algorithm. To transform the function, a set of values $\{\lambda_i : \ i = 1, \ldots, m\}$ are used with $\lambda_i = (i-1)h$, $h = 0.01$. So, $m = 1$ simply implies that no transformation is used, and the algorithm is just a local minimization sampling procedure. The comparison between the two cases, $m = 1$ and $m = 40$, shows that with transformation, the algorithm performs much more effectively than directly doing local minimization on the given function. In the table, we can also see that as the number of processors increases, the energy values for the solutions obtained by the algorithm decreases rapidly.

11 Software Development

Based on this work, we are currently developing a parallel continuation-based global optimization software system, called **Cglop** (Figure 7), for molecular conformation and protein folding. An initial version of the system has just been completed (see [5] for more details).

The system transforms the objective function into a sequence of gradually deformed functions. There are three subsystems corresponding to three different solution tracing procedures, namely, the global simulated annealing random search (**GLOBAL**), the Newton's local minimization method (**LOCAL**), and the Euler-Newton predictor-corrector method (**PC**). As we have discussed in this paper, the random search method is more robust but also costly. The deterministic local tracing is efficient, but may not guarantee a global minimizer. The predictor-corrector method provides a more accurate way to trace the solution. Overall, each of these methods has advantages and disadvantages, but the combination of them provides a robust set of numerical tools for both effective and efficient trace of the solutions. The system also provides transformation routines (**TRANSFORMATION**) to both transform user-supplied functions (**USER FUNCTIONS**) using numerical integration (**INTEGRAL**) and construct corresponding function values look-up tables. The function evaluations in the solution tracing process are conducted by cubic spline (**SPLINE**) using the function values in the look-up tables.

The system is written in C and developed on the IBM SP1 with PVM used for parallel message passing extensions. It is easy to port to a variety of parallel architectures including a cluster of local workstations. The system is meant to be used as a computational platform for basic interdisciplinary studies on molecular conformation and protein folding.

Acknowledgements

This research was supported partially by the Cornell Theory Center, which receives funding from members of its Corporate Research Institute, the National Science Foundation (NSF), the Advanced Research Projects Agency (ARPA), the National Institutes of Health (NIH), New York State, and IBM Corporation.

References

[1] Emile Aarts, and Jan Korst [1989]. *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons, New York, NY.

[2] Eugene L. Allgower and Kurt Georg [1990]. *Numerical Continuation Methods.* Springer-Verlag, New York, NY.

[3] Thomas F. Coleman, David Shalloway and Zhijun Wu [1993]. *Isotropic Effective Energy Simulated Annealing Searches for Low Energy Molecular Cluster States.* Computational Optimization and Applications, 2, 145-170, 1993.

[4] Thomas F. Coleman, David Shalloway and Zhijun Wu [1994]. *A Parallel Build-Up Algorithm for Global Energy Minimizations of Molecular Clusters Using Effective Energy Simulated Annealing.* Journal of Global Optimization, 4, 171-185, 1994.

[5] Thomas F. Coleman and Zhijun Wu [1994]. *Cglop – A Parallel Continuation-Based Global Optimization Package for Molecular Conformation.* Advanced Computing Research Institute, Cornell University, Ithaca, NY, to be submitted to ACM Transactions on Mathematical Software.

[6] J. E. Dennis, Jr. and R. B. Schnabel [1983]. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, NJ.

[7] Zhijun Wu [1993]. *The Effective Energy Transformation Scheme as a General Continuation Approach to Global Optimization with Application to Molecular Conformation.* Technical Report CTC93TR143, Advanced Computing Research Institute, Cornell University, Ithaca, NY, submitted to SIAM Journal on Optimization.

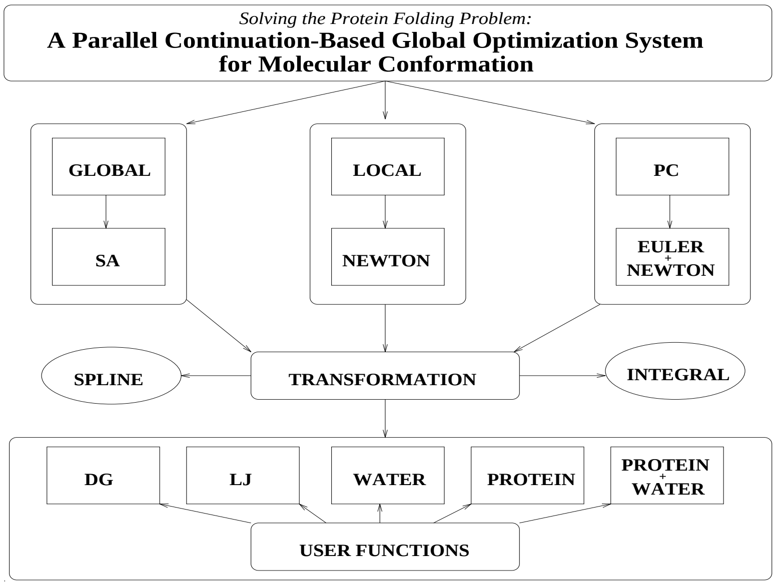


Figure 7: The **Cglop** system structure.