A Parallel Monte Carlo Search Algorithm for the Conformational Analysis of Proteins

by

Daniel R. Ripoll and Stephen J. Thomas

National Research Council of Canada, Biotechnology Research Institute
6100 Royalmount Ave., Montréal, Quebec, H4P 2R2, Canada

Abstract

In recent years several approaches have been proposed to overcome the multiple minima problem associated with non–linear optimization techniques used in the analysis of molecular conformations. One such technique based on a parallel Monte Carlo search algorithm is analysed. Experiments on the Intel iPSC/2 confirm that the attainable parallelism is limited by the underlying acceptance rate in the Monte Carlo search. It is proposed that optimal performance can be achieved in combination with vector processing. Tests on both the IBM 3090 and Intel iPSC/2–VX indicate that vector performance is related to molecule size and vector pipeline latency.

1 Introduction

In order to study the folding of large proteins various molecular potential energy functions have been proposed (cf. AMBER [1], CHARMM [2], MM2 [3], ECEPP [4]). A potential energy function E may include atomic interaction terms attributed to Van der Waals, electrostatic, bending and torsional forces. The potential energy function of a molecule defines a conformational hypersurface in which local minima represent energetically stable three dimensional orientations of the molecule. Molecular biologists believe that the minimum energy conformation of a molecule corresponds to its native structure [5]–[7]. An open problem in computational chemistry is the prediction of the minimum global energy conformation through the use of such empirical models [8], [9]. The chemical properties of proteins are determined largely by their three dimensional structure. It is therefore important to develop methods which can predict the native conformation and also provide information on conformations which might represent higher energy transition states in biochemical reactions such as enzyme catalysis.

The Electrostatically Driven Monte Carlo method is an attempt to overcome the *multiple–minima* problem in conformational analysis. It performs an exhaustive search of the conformational hypersurface of relatively small polypeptide molecules for the minimum global energy conformation. The *multiple–minima* problem is overcome by generating new conformations by one of two different techniques. The first uses electrostatic predictions, while the second is based on random sampling. Once a conformation has been energy minimized a Metropolis [10] criterion selects a new local minimum energy conformation. A parallel algorithm implementing the EDMC method was originally developed by D. R. Ripoll and H. Scheraga [11], [12]. It is based on the ECEPP/2 molecular potential energy function developed by Scheraga and his co–workers (at the Baker Labs of Cornell University). The ECEPP/2 [13], [14] potential is based on a rigid geometry model of molecules where the number of degrees of freedom for any atom is restricted i.e. only torsional and rotational forces about bonds are allowed. Bond stretching and bending are not considered in this model.

The algorithm was first implemented on an IBM 3090 supercomputer at the Cornell National Supercomputing Facility (CNSF). This is a MIMD parallel machine consisting of 6 CPU's with pipelined vector processing capability and 1 gigabyte of shared virtual memory. The parallelization effort was directed towards the minimization step of the algorithm. It was considered the most computationally intensive and hence determined the granularity of the parallel computation. The resulting parallel algorithm called for the complete energy minimization of a starting conformation of the molecule on each processor. Once a suitable minimum energy conformation has been determined a new set of conformations is generated and the searching process continues. The authors have determined that certain limitations existed with this parallel implementation which could hinder performance. Briefly, when a local energy minimum is found a master processor is required to wait for other minimizations in progress to complete before loading slave processors with new starting conformations generated from the newly accepted conformation. Clearly, if the relative difference in times to energy–minimize these conformations is large then a significant amount of CPU time could be wasted. These problems have been corrected in a new implementation of the algorithm on the distributed memory, message–passing architecture of the Intel iPSC/2. Parallel processes carrying out energy minimization of conformations are now synchronized via message–passing and eventually told to halt and wait for a new conformation once one has been accepted during a given iteration. The new dis-

CH2916-5/90/0000/0094/$01.00 © IEEE

tributed memory version of the algorithm has been successfully implemented and tested by the authors on an Intel iPSC/2-SX parallel computer at the National Research Council of Canada (NRC) and on an iPSC/2-VX at the Cornell Theory Center.

Sub-programs which compute terms in the potential energy E and gradient vector ∇E consist of certain well defined and highly vectorizable loops. Several molecular modelling packages based on ECEPP/2 and associated energy minimization programs have already demonstrated the possible performance gains possible with vector processing. The iPSC/2-VX computer allows us to study pipelined vector arithmetic in combination with a distributed memory architecture. Our experiments on both Intel machines indicate a linear speed-up. A serial vectorized version of the code has also been tested on an IBM 3090 at the National Research Council of Canada and these results are reported in this paper. If we use the number of function evaluations of the potential energy E and its gradient ∇E as a basis for comparison, then preliminary results with the Intel vector facility are somewhat pessimistic. Our studies indicate that the inherent parallelism available in such a Monte Carlo search is limited, thus confirming that adding processors beyond a certain point only provides redundant information to the conformational search in progress. The underlying acceptance rate in the conformational search effectively limits the attainable parallelism by placing an upper bound on the number of processors which can be utilized in the search.

2 Molecular Modelling

Molecular biologists have increasingly turned to supercomputing techniques to model and visualize molecular structure. The motivation being that the three dimensional structure of a molecule, to a large extent, determines its chemical properties. Determination of the native conformation of protein molecules is essential in molecular biology. Knowledge of a molecular structure can aid in designing drugs which bind tightly to a molecule and inhibit a particular chemical reaction, e.g. an inhibitor binding to an enzyme. Proteins are characterized by long polypeptide chains of amino acid residues. A moderately sized protein may contain well in excess of 500 atoms.

The native $3D$ structure of a protein corresponds to the conformation with the lowest free energy. In computer simulation of protein folding, one attempts to find the conformation which yields the global minimum of an empirical potential energy function. This empirical potential energy function, which is a function of the conformation of a molecule, is parameterized against experimental data. In general, the potential energy function has a highly nonlinear functional form depending on a large number of variables and parameters. As a consequence, the resulting energy hypersurface will have a large number of local minima.

$$
\begin{aligned}
E \;=\; & \sum_{\text{bonds}} K_R(R-R_0)^2 + \sum_{\text{angles}} K_\theta(\theta-\theta_0)^2 \\
& + \sum_{\text{dihedrals}} \frac{V_n}{2}[1+cos(n\phi-\gamma)] \\
& + \sum_{i<j} \Big[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\varepsilon R_{ij}}\Big] + \sum_{\text{H-bonds}} \Big[\frac{C_{ij}}{R_{ij}^{12}} - \frac{D_{ij}}{R_{ij}^{10}}\Big].
\end{aligned}
$$

The terms included above represent bond stretching, bond angle bending, dihedral angle torsions, Van der Waals forces, electrostatic forces and hydrogen bonding. The ECEPP/2 [13], [14] (Empirical Conformational Energy Program for Peptides) potential energy function was the basis for our studies. In ECEPP/2 the geometry of the amino acid residues is assumed to be rigid i.e. the bond angles and bond lengths along the polypeptide chains are fixed at their equilibrium values. The complete set of backbone and atomic side-chain dihedral angles are considered to be independent variables.

The problem of locating the lowest value from among all the minima of such a multi-variable function is known as the *multiple-minima* problem [8], [9]. Numerous approaches have been proposed to overcome this problem [15]-[24]. These include, but are not limited to, energy minimization, molecular dynamics and integration by Monte Carlo techniques. All of which present a heavy computational burden to existing supercomputers, even for small molecular systems. The EDMC method consists of an algorithm which searches for the global minimum or a set of very low-energy minima of the potential function E. Specific details and the scientific basis for the method can be found elsewhere [11], [12]. In this paper we shall examine some of the basic steps taken by the algorithm during the Monte Carlo search process.

Before presenting any details of the EDMC algorithm we would like to briefly mention some of the computational constraints imposed by typical molecular modelling problems. It is clear from the basic form of a potential energy function for a given molecular system that a function or gradient vector computation will be dominated by the $O(n^2)$ pairwise interactions due to the Van der Waals and electrostatic terms. In practice it has been found that computation of the contributions due to these terms can easily account for 99% of the total CPU time required [29]. Consider a protein consisting of 3000 atoms represented in cartesian co-ordinates. The gradient of the potential energy function, a vector representing the forces acting on individual atoms, will consist of 9000 elements. The Hessian or its local approximation are not usually computed due to the large number of elements involved and the resulting storage requirements. Sparsity might be exploited if energy cut-offs are employed, however, the resulting matrix will have no discernable sparsity pattern because of the complex folding patterns exhibited by proteins. These limitations rule out the faster converging Newton or quasi-Newton optimization algorithms. Gradient-based optimization algorithms take longer to converge result-

ing in a larger number of gradient vector computations in the search for a local minimum. Working within a rigid geometry framework the dihedral angles ϕ are the only variables. The number of dihedral angles which can be varied within a given amino acid residue varies between 3 and 10. The result is a slightly less computationally intensive problem, however, a conversion back to cartesian co–ordinates is usually required for the gradient vector computation.

3 The EDMC Method

The EDMC method is an iterative procedure for systematically searching the conformational hypersurface of relatively small polypeptide molecules. It has been found to be computationally tractible on existing supercomputers for proteins consisting of up to 20 amino acid residues. The algorithm starts from an initial energy minimized conformation, which usually corresponds to the unfolded state of a protein. A search for the global minimum energy conformation proceeds along a conformational *pathway* corresponding to the conformations accepted by the algorithm over an unbounded number of iterations. In practice, however, a finite number of iterations is specified. The strategy used to produce new conformations from accepted ones is fairly complex and is based upon a combination of thermal effects and internal electrostatic interactions.

At each iteration a set of conformations is generated by one of two different techniques. The first uses electrostatic predictions, while the second is based on random sampling. An iteration starts with an electrostatic analysis of the current conformation (the initial energy minimized conformation or the accepted conformation from the previous iteration) to determine the alignment of the permanent dipoles with the local electric field produced by the whole molecule. This analysis leads to a set of possible conformational changes required to improve the local alignments. These predictions constitute the main source of information to generate new conformations in a subsequent search for states of lower energy. It may occur that none of these conformations leads to an acceptable one (when compared with the current minimum energy conformation), therefore a second group of conformations is generated by using a random sampling technique in combination with the electrostatic predictions. A conformation generated by either of these two procedures is subjected to energy minimization via unconstrained optimization techniques. With the backbone and dihedral angles of the molecule serving as variables, the potential energy and geometry are computed using the ECEPP/2 [13], [14] algorithm. Energy minimization is carried out with the Secant Unconstrained Minimization Solver (SUMSL) [25].

The potential energy corresponding to a local minima forms the basis for either acceptance or rejection of a given conformation as the new minimum energy conformation. A conformation must fulfill two conditions to be accepted:

1. The energy E of the new local minimum must satisfy the Metropolis [10] criterion when compared with the current minimum energy conformation i.e. if the energy difference $\Delta E = E_{new} - E_{old} < 0$ or (when $\Delta E > 0$) if $e^{-\Delta E/RT}$ is greater than a randomly generated number between 0 and 1.

2. When the energy of the new conformation satifies condition 1 the long–term behaviour or the search is examined to ensure that the search is not trapped in a local minimum.

The second condition above is extremely important if the search is to progress towards the global energy minimum. The energy of the new conformation is compared with a set of previously accepted minima; if the number of repetitions of the same local minimum exceeds a prescribed limit (usually 15 or 20), then the conformation is rejected. When the energy of the new conformation passes both tests successfully the conformation is accepted, replacing the current one, and a new iteration begins.

It may occur within a given iteration that neither the set of electrostatic predictions nor a significantly large number of randomly generated conformations will produce an acceptable conformation. The algorithm then assumes that the current local minimum is extremely stable and that the acceptance criteria must therefore be modified. When these circumstances arise, thermodynamic arguments are invoked in order to search a different region of the conformational hypersurface via what Ripoll and Scheraga refer to as a backtrack movement (of the first kind). The search within the current region of the hypersurface is abandoned and a new one initiated from a higher energy conformation. A conformation is selected probabilistically from among the high–energy ones generated within the current iteration, but rejected by the Metropolis criterion (within the current iteration) and whose energies are within certain limiting or cutoff values. The above procedure is equivalent to a perturbation due to thermal effects. A backtrack movement (of the second kind) is invoked when a conformation satisfies the Metropolis criterion, but is then rejected because it fails the second criterion. Once again the search is temporarily trapped. A type 2 backtrack movement is a random perturbation of at least two dihedral angles (chosen randomly) followed by energy minimization. Here again energy limiting or cutoff values are applied, restricting the conformation to lie within a specified energy interval. The backtrack process is repeated until a conformation satisfying the energy requirements is found.

It has been found in practice [11], [12], [26] [27] that the algorithm can compute very low energy minima of relatively small polypeptide molecules. An exhaustive search of the conformational hypersurface is not without a severe computational burden, even for CRAY class supercomputers. Often these computations are part of a protein engineering study where the amino acid sequence of

a peptide molecule is altered. Under these circumstances faster turnaround time would be desirable. In order to emphasize the need for parallel supercomputer performance levels in the 100 Gigaflop range or higher consider the following example of a conformational search:

- 20 amino acid residue protein.
- 115 dihedral angles, with $3^{115} \approx 10^{55}$ possible local minima.
- 36,000 random starting conformations generated.
- 220 CPU hours on an IBM 3090 computer.

4 The Parallel Algorithm

Gradient based optimization methodologies are known to be very demanding with respect to computing time. The EDMC method is certainly no exception to this rule, with a typical conformational search lasting anywhere from ten to a few hundred CPU hours on a machine such as the IBM 3090. To reduce these time scales by an order of magnitude or more (short of a breakthrough in the basic theory of protein folding) it appeared that parallelism, possibly combined with vectorization, was the only alternative. A wide variety of algorithm design choices are then possible. Certainly the *granularity* of the sub–problems to be solved in parallel plays a central role in choosing an architecture and parallel programming model. Another consideration is the random nature of a Monte Carlo search and the underlying acceptance rate of conformations.

A *fine–grained* approach might involve the energy minimization of one molecule at a time on a parallel machine. Atoms of the molecule would be distributed equally among the various processors so that components of the potential energy function and gradient vector could be computed in parallel. Due to the long range pairwise interactions represented in the potential energy (e.g. Van der Waals and electrostatic forces) a permutation algorithm (associated with the processor and memory interconnection topology) would be required to ensure that all $n(n-1)/2$ atomic pairings are formed. Many solutions to this problem have been proposed in the context of the n–body problem, for both the shared memory and distributed memory programming models. A distributed memory algorithm for molecular mechanics computations has already been proposed and implemented by Ostlund and Whiteside [28]. Their approach is based on a ring topology for processor and memory interconnection. The permutation they employ is known as the "tractor–tread" algorithm.

Achieving reasonable performance levels with pipelined vector processing invariably requires long vector lengths. We are therefore presented with conflicting design alternatives. To take full advantage of a parallel machine with vector processor nodes a large number of atoms should be processed on each node. Furthermore, algorithms for computing the potential function and gradient vector can take advantage of long chains of neighbouring atoms such as those found along the backbone of protein molecules (c.f. G. M. van Waveren [29]). A *coarse–grained* parallel algorithm consists of individual tasks executed concurrently, which contain a large number of instructions, but with relatively little communication overhead incurred throughout the computation. With vector processing available at each processor node a reasonable (if not obvious) choice is to perform the complete energy minimization of newly generated molecular conformations on separate processors [30]. Such an approach is well suited to a Monte Carlo search when it is expected that a large number of conformations must be energy–minimized before an acceptable one can be found.

We will examine a *coarse–grained* approach to parallelizing the EDMC conformational search. It is important to note, however, that the underlying acceptance rate of conformations will limit the attainable parallelism by placing an upper bound on the number of processors that can be used effectively. A performance analysis based on this approach should examine the acceptance rate of conformations under varying conditions. Both long term behaviour and localized behaviour along the conformational *pathway* would be of interest. We expect that the acceptance rate will vary across the regions of the energy hypersurface being searched. When temporarily trapped in a high energy region (defined by a set of local minima), that *appears* to be very stable, the algorithm will produce a large number conformations which are rejected before one of two types of backtrack movement restarts the search in a different region (where the acceptance rate should increase again). Ideally the parallel algorithm should be adaptive in the sense that the number of processors required during a given iteration should be varied according to the conformation acceptance rate in order to achieve a proper load balance.

The *coarse–grained* parallel EDMC algorithm itself is straightforward, relying on a simple replication of tasks at each node. A main program running on the *host* or *master* processor directs the conformational search. It generates new conformations and compares the energy of energy–minimized ones with the current local minimum in order to decide which to accept. Processor *nodes* acting as *slaves* receive new conformations from the *master* and perform an energy minimization to determine a local energy minimum with corresponding energy value. There is no synchronization (and therefore no communication) required between individual *nodes*. The *slaves* must report energy values back to the *master* processor. If a conformation is accepted the *master* must obtain the corresponding dihedral angles from the *slave* processor. The newly accepted conformation then becomes the basis for the next iteration.

Implementation of this parallel algorithm at the CSNF by Ripoll and Scheraga demonstrated the possible performance gains attainable by such a *coarse–grained* ap-

proach. The authors propose, however, that there are certain limitations in the original IBM 3090/600E implementation associated with processor synchronization at the end of an iteration. Conformations are energy–minimized in parallel until one is accepted. Once a suitable conformation is found the number of processors available to energy–minimize conformations in a new iteration is limited by the speed at which minimizations in progress from the previous iteration can complete. If the relative difference in times between minimizations is large then a significant amount of the available CPU time could be wasted. For example, such a situation would occur if a starting conformation to be energy–minimized on one of the processors is very close to a local energy minimum.

A number of improvements to the original parallel algorithm have been added by the authors to an implementation on the distributed–memory, message–passing Intel iPSC/2. Processes executing on *slave* processors, which perform energy minimization, periodically check if a conformation has been accepted by the *master* processor. After having accepted a conformation, the *master* processor will halt all *slave* processors and subsequently re–load them with new starting conformations. The added cost of such a processor synchronization is warranted as it ensures that all processors are available at the beginning of each iteration of the Monte Carlo search.

The code associated with atomic pairwise interactions in the potential energy function and gradient vector is isolated in one sub–program. It has been organized to take advantage of vectorizing compilers for both the IBM 3090 and Intel iPSC/2–VX machines. In the next section we report on the performance achieved by a serial vectorized implementation of the EDMC algorithm on an IBM 3090 at the NRC. We also provide results from our experiments with a 16 node Intel iPSC/2–SX at the NRC and a vectorized version of this code running on a 32 node Intel iPSC/2–VX at the Cornell Theory Center.

5 Performance Analysis

In this section we shall present performance results obtained from experiments with the EDMC method at the NRC and the Cornell Theory Center. On the hypercube machines our results are broken down into an analysis of parallel speed–up, efficiency and processor utilization with respect to the number of potential energy function and gradient evaluations for a fixed problem size. Our test molecule is a polypeptide chain of 10 glycine residues. A glycine amino acid consists of 7 atoms and has 3 dihedral angles which can be varied. We recall that the basic unit of computation performed in parallel is an energy minimization of a molecular conformation. In order to obtain meaningful results, the total number of potential energy function and gradient vector evaluations on all processors is used as a basis for comparison. We have therefore normalized the various performance measurements accordingly. An analysis of the possible performance gains attainable via vector processing is provided for both the IBM 3090 and Intel iPSC/2–VX machines. Variable length chains of glycine residues are used in experiments to determine if the computation time of the dominant pairwise interaction terms in the potential function and gradient vector can be improved through vectorization.

5.1 Parallel Performance Analysis

A number of definitions are in order before we present our experimental results. We shall denote the total number of potential energy function (or simply function) and gradient vector evaluations on P processors by f_P. The total time spent in energy minimizing conformations of size N on P processors will be denoted $T(N,P)$. In order to simplify the analysis we shall ignore the computation time of the *master* processor as this proves to be insignificant compared with the minimization time.

Speed–up

Parallel speed–up is usually defined as the ratio of the sequential to parallel computation times. We shall modify this definition slightly to account for possibly different minimization times on each processor.

$$Sp(N,P) = \frac{T(N,1)}{T(N,P)} \times \frac{f_P}{f_1}$$

We have chosen a fairly simple form of replicated parallelism where the only communication overhead incurred is between the *master* and *slave* processors. It is therefore expected that we would achieve a linear speed–up (normalized), as adding processors effectively allows us to perform more function and gradient evaluations in parallel.

Cost

Given P processors and a molecular conformation of size N, the cost of P parallel minimizations is

$$C(N,P) = T(N,P) \times P$$

Efficiency

The efficiency of the parallel EDMC algorithm can be defined as,

$$E(N,P) = \frac{Sp(N,P)}{P} = \frac{T(N,1)}{C(N,P)} \times \frac{f_P}{f_1}$$

We expect that the efficiency of the parallel EDMC algorithm should remain high, at least for small numbers of processors. The reason for this is that by adding "minimizers" more information is made available to the Monte Carlo search, allowing it to progress that much faster.

Clearly, we may not add processors indefinitely as eventually only redundant information is provided for the search in progress. It is for this reason that we are also interested in processor utilization.

Processor Utilization

As noted earlier the underlying acceptance rate of conformations in the Monte Carlo search will limit the attainable parallelism in our *coarse-grained* approach. One possible measure of the parallelism achieved in such a computation is the processor utilization. Given the total number of energy-minimized conformations in the Monte Carlo search, N_c, we can define processor utilization as,

$$U(N_c, P) = \frac{N_c}{P}$$

We expect, however, that processor utilization will vary greatly from one iteration to the next along the conformational *pathway*. For example, it may increase if a search becomes temporarily trapped in a high energy region of the conformational hypersurface. Under such conditions a large number of conformations may be generated before a type 1 or type 2 backtrack movement redirects the search into a different region. This type of behaviour is in fact observed and is reported along with our other experimental results.

5.2 Parallel Performance Results

Our experiments on the Intel iPSC/2 and iPSC/2-SX machines consisted of a conformational search of a 10 residue polyglycine molecule. A high-energy, randomly generated conformation was chosen as a starting point. The search was allowed to continue through 20 iterations of the algorithm. We observe almost perfect linear speed-up on both the hypercube computers. The speed-up curve for the iPSC/2-SX is given in Figure 1. Results for the iPSC/2 at the Cornell Theory Center were practically identical. Linear speed-up (measured from 1 to 16 processors) was expected as there is no communication between node (*slave*) processors. Communication between the *master* and *slave* processors takes the form of synchronization messages or an exchange of energy values and conformations defined by dihedral angles. As observed in Figure 2, efficiency remains close to 100% and our conclusion is that by adding processors we are simply able to compute more function and gradient vector evaluations in a given amount of time.

Unfortunately we can not add processors *ad infinitum* as demonstrated by the observed processor utilization. Curves for both machines are plotted in Figure 3. We note that $U(N_c, P)$ drops off rapidly on the iPSC/2-SX and remains almost constant after 4 processors are in use. The observed behaviour reflects the high conformation acceptance rate for a random starting conformation, from which local minima are more readily accessable. A marked increase in $U(N_c, P)$ at 16 processors indicates certain differences between the iPSC/2 and iPSC/2-SX architectures (primarily the Weitek floating point co-processor) and also points out how a large number of conformations are generated when the search becomes trapped in a high energy region. Due to differences in the floating point hardware of the Intel 80387 and Weitek 1167 co-processors, different conformational *pathways* towards the global energy minimum have been taken on both machines. The search on the iPSC/2 encountered a stable set of conformations in a high energy region of the conformational hypersurface, resulting in a backtrack movement. Differences in floating point hardware between vector processor pipelines also can lead to such "anomalies". In practice it is found that, even if completely different conformational *pathways* are followed (with different search times) on different machines, they still arrive at the same set of stable low energy minima or the global minimum.

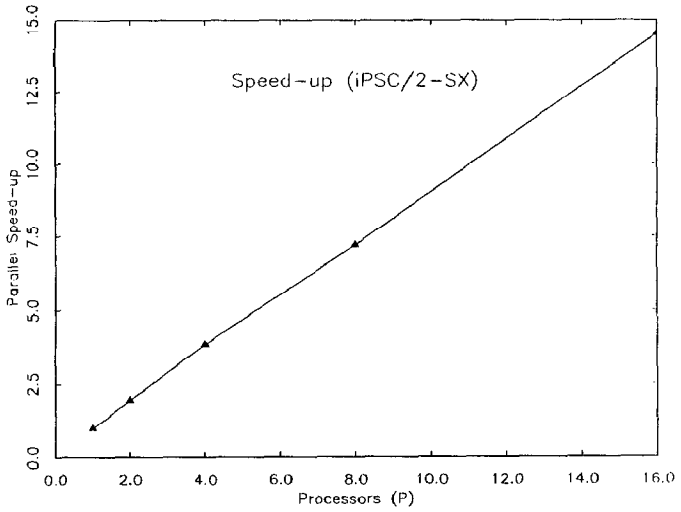

Figure 1: Speed-up Curve for iPSC/2-SX

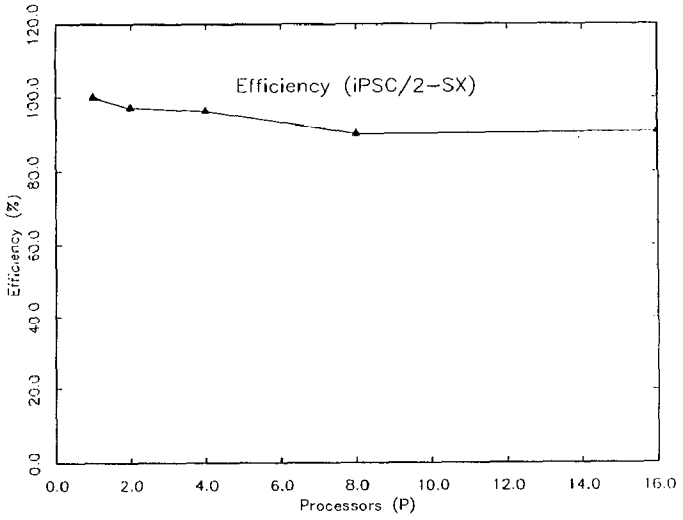

Figure 2: Efficiency Curve for iPSC/2-SX

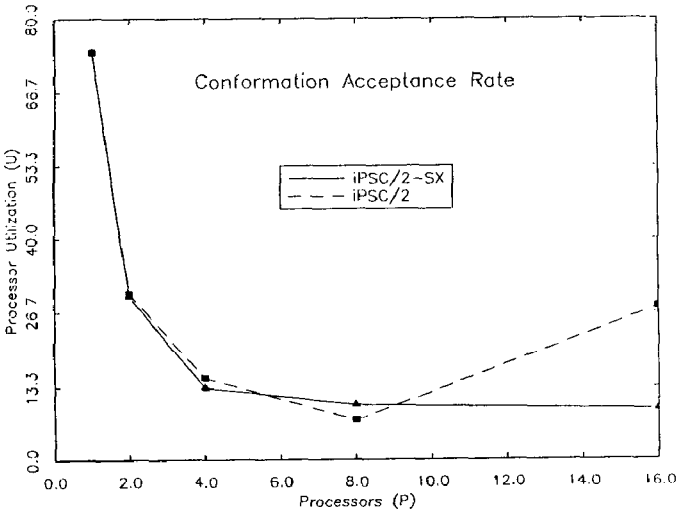

Figure 3: Number of Conformations per Processor

5.3 Vector Performance Analysis

As indicated throughout our presentation, the performance of the EDMC algorithm depends almost entirely upon the speed at which energy minimizations can be completed. Computation of the $O(n^2)$ atomic pairwise interactions dominate both the function and gradient vector evaluations. van Waveren [29] provides a detailed description and analysis of how molecular dynamics code can be organized to take full advantage of vectorization techniques. We will not present a detailed analysis of such techniques here and the reader is referred to [29]. We will instead briefly describe how pairwise interactions are computed for a chain of amino acid residues. Our interest is whether or not the number of function and gradient vector evaluations per second can be increased significantly through vectorization.

Vector lengths will depend upon the number of amino acid residues along the backbone of a protein molecule and the number of atoms in each residue. Computations are organized according to the number of atoms and bonds involved in the various terms of E and ∇E. For example, a 1–4 term refers to an atom pair interaction involving one degree of rotational freedom, usually between atoms k and $k+3$ along a chain of bonds. A 1–5 term refers to a long–range force interaction. Consider the i–th residue of a molecule consisting of N_r residues. The computation of pairwise interactions between atoms in residue i and $i+1$ must distinguish between 1–4 and 1–5 terms, therefore these computations have to be carried out separately. Interactions between atoms of the i–th and the j–th residues ($j \geq i+2$) are considered (with a few exceptions) to be of type 1–5. These computations can be programmed to take full advantage of vectorizing compilers. The performance of a vector pipeline is related to the time $T_v(N_v)$ required to process a vector of length N_v,

$$T_v(N_v) = \alpha + N_v\beta$$

where α is the pipeline latency and β is the time required to pass through one stage in the pipeline. If the vector length N_v from residue $i+2$ to N_r is long enough, on average, to overcome the pipeline latency α, then a performance improvement in the form of a speed–up is expected.

5.4 Vector Performance Results

Polyglycine was used once again for tests on the IBM 3090 and iPSC/2–VX computers. We used 16 nodes for testing on the iPSC/2–VX at the Cornell Theory Center. The number of residues was varied in order to determine if vectorization leads to an improvement over the original scalar code. Scalar and vector timings for the potential function E and gradient vector ∇E evaluations were tabulated. The computation time (in seconds) and number of evaluations for the IBM 3090 computer are reported in Figure 4. The number of function and gradient vector evaluations per second on the IBM 3090 are plotted in Figures 5 and 6. Results for the iPSC/2–VX are also provided and are plotted in Figures 7 and 8. For the IBM 3090 we observe that the vector performance is slightly better than scalar after 10 residues. At 40 residues the vector computation is approximately double the speed of the equivalent scalar computation. Unfortunately we do not observe similar behaviour on the iPSC/2–VX computer. We do notice a sharp decrease in the number of evaluations computed serially from 10 to 60 residues. The decline in vector performance is more gradual, however, the two curves do not intersect. It appears that the vector pipeline latency on the iPSC/2–VX has not been overcome for our test cases.

No. Res.	Func. Eval.	Func. Time	Grad. Eval.	Grad Time	Total Time	Min. Time	Vector Time
5							
	3812	18	2878	23	52	51	
	2735	15	2043	19	43	42	12
10							
	21240	296	15637	353	735	733	
	9743	123	7299	158	321	320	119
20							
	12649	608	9218	726	1448	1446	
	25889	828	19446	1158	2216	2214	417
30							
	2292	241	1829	317	603	601	
	4291	265	3387	406	750	749	368
40							
	7514	1354	6141	1832	3432	3430	
	7377	735	6035	1194	2162	2160	1100

Figure 4: Vector vs. Scalar Performance

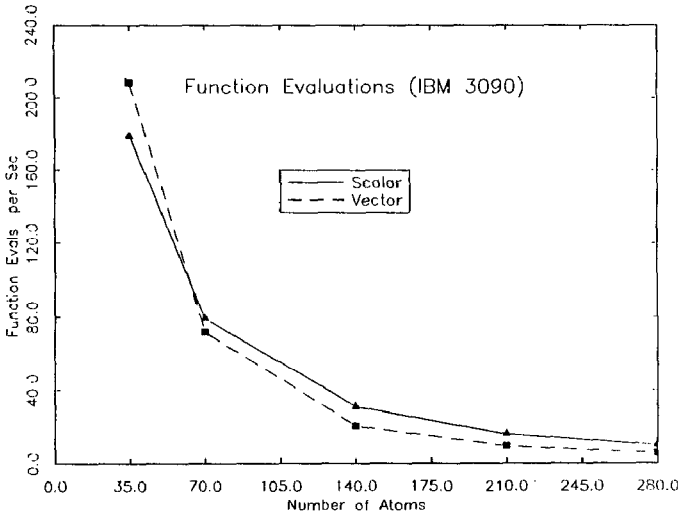

Figure 5: Function Evals. per Sec – IBM 3090

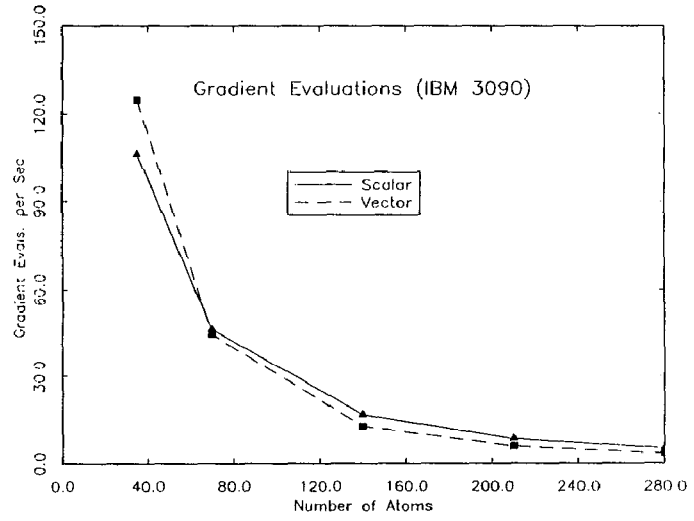

Figure 6: Gradient Vector Evals. per Sec. – IBM 3090

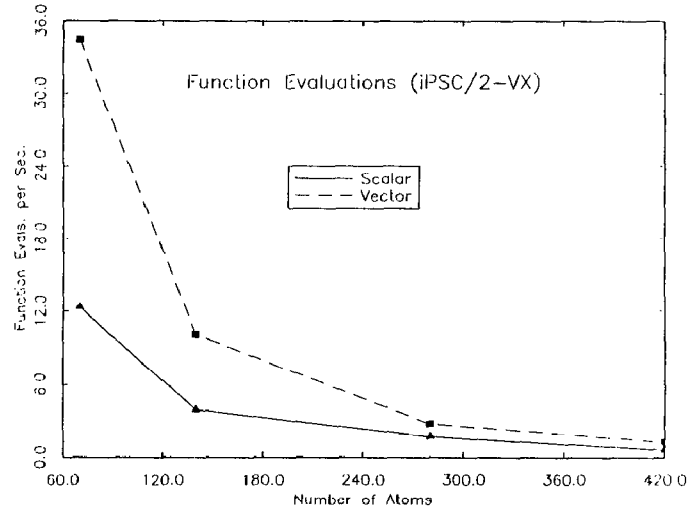

Figure 7: Function Evals. per Sec. – iPSC/2-VX

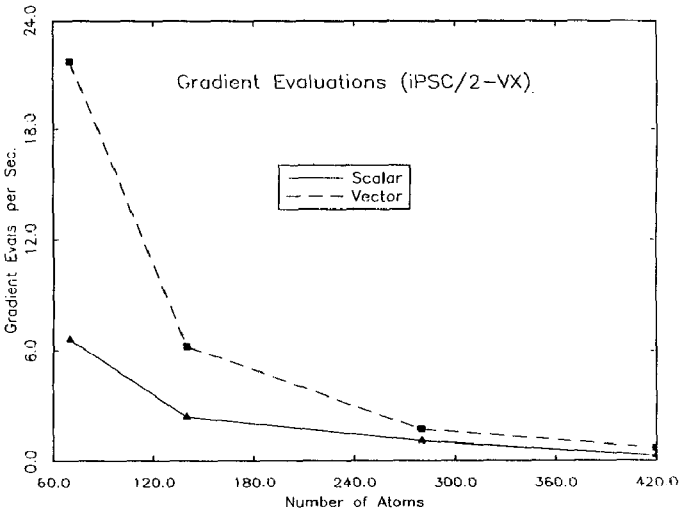

Figure 8: Gradient Vector Evals. per Sec. - iPSC/2-VX

6 Conclusion

In recent years several techniques have been proposed to overcome the multiple minima problem in conformational analysis associated with non-linear optimization. The EDMC method has proven to be an effective computational tool for searching the potential energy hypersurface of polypeptide molecules consisting of up to 20 amino acid residues. It has been found that such a Monte Carlo search combined with gradient-based energy minimization of molecular conformations results in the need for 100 Gigaflop or higher performance levels.

The parallel EDMC algorithm has been designed to exploit currently available supercomputing technology. Our implementation on the iPSC/2 appears to represent an improvement over the original version for the IBM 3090. A performance analysis indicates that the attainable parallelism is limited by the underlying acceptance rate of minimized conformational energies in the Monte Carlo search. We have observed that this rate may vary greatly depending upon the particular region of the potential energy hypersurface being searched. These results indicate that our *coarse-grained* approach is suitable for architectures such as the CRAY-XMP, particularly if vectorization techniques can be exploited. A series of tests on the Intel iPSC/2-VX computer have shown, however, that even the easily vectorized parts of the computation may not overcome a large vector pipeline latency. Comparisons using a serial version of the code on an IBM 3090/200E at the NRC showed that vector performance was slightly better than scalar for molecules composed of 70 atoms. Tests on molecules containing 280 atoms indicated that the vector speed was approximately double that of the scalar. Finally, we feel that further testing with larger amino acid residues on machines with differing vector pipeline latencies is warranted.

7 Acknowledgements

We would like to thank Dr. Enrico Purisima of the Biotechnology Research Institute for reviewing our manuscript and providing many helpful comments. We are also very grateful to the Cornell Theory Center for providing us with access to their Advanced Computing Facility.

References

[1] P. K. Weiner & P. A. Kollman, *J. Comp. Chem.* **2**, 287–303 (1981).

[2] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan & M. Karplus, *J. Comp. Chem.* **4**, 187–217 (1983).

[3] U. Burkert & N. L. Allinger, *Molecular Mechanics*, ACS Monograph A77, American Chemical Society, Washington D.C., 1982.

[4] F. A. Momany, R. F. McGuire, A. W. Burgess & H. A. Scheraga, *J. Phys. Chem.* **79**, 2361–2381 (1975).

[5] G. Némethy & H. A. Scheraga, *Biopolymers* **3**, 155–184 (1965).

[6] K. D. Gibson & H. A. Scheraga in "*Structure & Expression: Vol.I, From Proteins to Ribosomes*", Eds. M. H. Sarma & R. H. Sarma, Adenine Press, Guilderland, N.Y., 67–94 (1988).

[7] G. Némethy & H. A. Scheraga, *Q. Rev. Biophys.* **10**, 239–352 (1975).

[8] H. A. Scheraga in "*Fourth International Symposium on Biological and Artificial Intelligence Systems*", Eds. E. Clementi and S. Chin, ESCOM Science Publ., Leiden, 1–14 (1988).

[9] H. A. Scheraga in "*Computer-Assisted Modelling of Receptor-Ligand Interactions: Theoretical Aspects and Applications to Drug Design*", R. Rein and A. Golombek; Alan R. Liss, Inc., New York, 3-18 (1989).

[10] N. Metropolis, A. W. Rosenbluth, M.N. Rosenbluth, A. H. Teller & E. Teller, *J. Chem. Phys.* **21**, 1087–1092 (1953).

[11] D. Ripoll & H. A. Scheraga, *Biopolymers* **27**, 1283–1303 (1988).

[12] D. Ripoll & H. A. Scheraga, *J. Prot. Chem.* **8**, 263–287 (1989).

[13] G. Némethy, M. S. Pottle & H. A. Scheraga, *J. Phys. Chem.* **87**, 1883–1887 (1983).

[14] M. J. Sippl, G. Némethy, & H. A. Scheraga, *J. Phys. Chem.* **88**, 6231–6233 (1984).

[15] I. Simon, G. Némethy & H. A. Scheraga, *Macromolecules* **11**, 797–804 (1978).

[16] M. R. Pincus, R. D. Klausner, & H. A. Scheraga, *Proc. Natl. Acad. Sci.* **79**, 5107–5110 (1982).

[17] G. M. Crippen, *Biopolymers* **21**, 1933–1943 (1982).

[18] M. Cotrait, *Int. J. Peptide Protein Res.* **23**, 355–360 (1984).

[19] M. Vásquez & H. A. Scheraga, *Biopolymers* **24**, 1437–1447 (1985).

[20] M. Billeter, T. F. Havel & K. Wüthrich, *J. Comp. Chem.* **8**, 132–141 (1987).

[21] E. O. Purisima & H. A. Scheraga, *J. Mol. Biol.* **196**, 697–709 (1987).

[22] L. Piela & H. A. Scheraga, *Biopolymers* **26**, S33–S58 (1987).

[23] Z. Li & H. A. Scheraga, *Proc. Natl. Acad. Sci.* **84**, 6611–6616 (1987).

[24] Z. Li & H. A. Scheraga, *J. Molec. Str. (Theochem)* **179**, 333–352 (1988).

[25] D. M. Gay, *ACM Trans. Math. Software* **9**, 503–524 (1983).

[26] D. Ripoll & H. A. Scheraga, *Biopolymers*, in press (1990).

[27] D. Ripoll, M. J. Vásquez & H. A. Scheraga, in preparation (1990).

[28] Ostlund & Whiteside, *Ann. New York Acad. Sci.*, **439**, 195–208 (1986).

[29] G. M. van Waveren, "*Algorithms and Applications on Vector and Parallel Computers*", Eds. H. J. J. te Riele, Th. J. Dekker and H. A. van der Vorst, Elsevier (North–Holland), Amsterdam, 405–428 (1987).

[30] A. R. Larrabee & R. G. Babb, "*Hypercube Multiprocessors*", Ed. M. T. Heath, SIAM, 464–472 (1987).