Journal of Global Optimization, 11, 55–68 (1997)

Protein Conformation of a Lattice Model Using Tabu Search*

P.M. PARDALOS, XIN LIU
Center for Applied Optimization, Department of Industrial and Systems Engineering, 303 Weil Hall, University of Florida, Gainesville, FL 32611. Email: {pardalos,liu}@ufl.edu

G.L. XUE
Department of Computer Science and Electrical Engineering, College of Engineering and Mathematics, The University of Vermont, Burlington, VT 05405. Email: xue@cs.uvm.edu

Abstract. We apply tabu search techniques to the problem of determining the optimal configuration of a chain of protein sequences on a cubic lattice. The problem under study is difficult to solve because of the large number of possible conformations and enormous amount of computations required. Tabu search is an iterative heuristic procedure which has been shown to be a remarkably effective method for solving combinatorial optimization problems. In this paper, an algorithm is designed for the cubic lattice model using tabu search. The algorithm has been tested on a chain of 27 monomers. Computational results show that our method outperforms previously reported approaches for the same model.

Keywords: Protein conformation, lattice model, global and combinatorial optimization, tabu search, computational results.

1. Introduction

The determination of the three-dimensional structure of a protein from a given sequence of amino acids is one of the most challenging unsolved problems in the science of molecnlar biology (see e.g. [22, 28, 2]). There have been many computer models designed to solve the protein folding problem. It is essential for these models to simulate the mechanism of protein folding and to search the native states of a chain of protein sequences. The basic difficulty in solving these models is the existence of multiple local minimizers. All computer models, though employing different types of energy minimization, can be expressed as the global optimization of a non-convex potential energy function. Recently, there have been various approaches [3, 16, 27] used to solve these models arising from protein folding. A survey of these approaches can be found in Pardalos, Shalloway, and Xue [19, 20].

Recently, lattice models have been used by many researchers to describe the protein folding mechanism [23, 13, 17]. This is motivated from two aspects of research interests. On one side, scientists (with practical insight) are hoping to use some lattice structures to obtain initial solutions of protein conformations, with the as-

*This research was supported in part by the National Science Foundation grants BIR-95-05919 and ASC-94-09285.

sumption that an optimal or near-optimal native state can be obtained by relaxing the monomers around the lattice structure [17]. On the other side, not understanding the mechanism of protein folding, scientists grasp protein conformations by using lattice models. It does not mean that these particular lattice models can describe the protein folding problem in detail, but they can actually address the keen mechanism of protein conformation. Among these models, the lattice model of a chain of monomers freely joined with a unit bond length on a simple cubic lattice [24] (see Fig. 1) has been studied in a series of papers [24, 23, 25, 26]. In this paper, we study this model using tabu search and test our approach with a chain of 27 monomers.

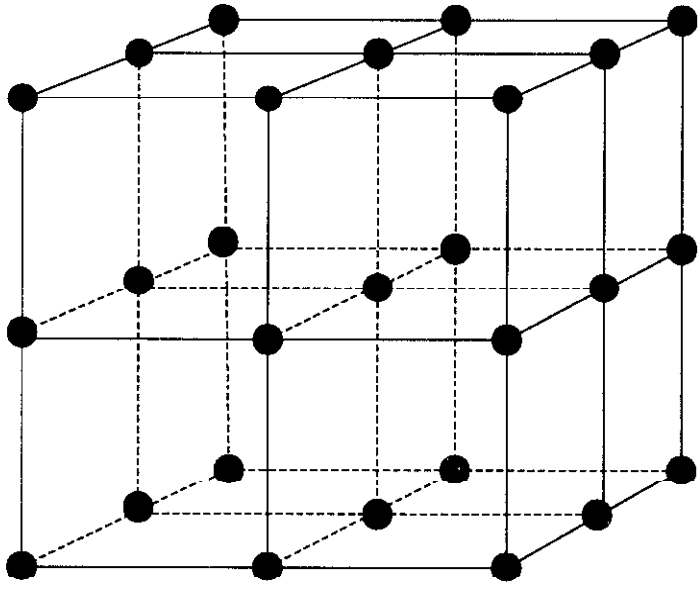

Figure 1. Lattice model of 27 beads.

First, we introduce some definitions of the lattice model. Given two lattice sites m and n, $D(m,n)$ is used to express the distance between these two sites. We say $D(m,n) = 1$, if two sites m and n are neighbors in the lattice (and $D(m,n) = 0$, otherwise). For a protein chain with n monomers, we ranked it with the set $\mathcal{N} = \{1,2,\ldots,n\}$ according to the order of the protein chain. A permutation $p = \{v_1, v_2, \ldots, v_n\} \in \Pi_{\mathcal{N}}$ is used to describe the lattice sites where each monomer is positioned, and $\Pi_{\mathcal{N}}$ is the set of all permutation of $\mathcal{N}$. Therefore, the space of all possible conformations of the chain corresponds to a subset of all permutations $\Pi_{\mathcal{N}}$. The energy of the protein chain is the following [23] :

$$E_n = \sum_{i,j=1}^{n} B_{i,j}\, D(v_i, v_j) \tag{1}$$

where $D(v_i, v_j)$ is the distance between two lattice sites v_i and v_j at which two monomers i and j are positioned, and the interaction energies $B_{i,j}$ are obtained from a Gaussian distribution:

$$P(B_{i,j}) = (2\pi\, B^2)^{-1/2}\, exp(-B_{i,j}^2/2\, B^2) \tag{2}$$

where B is the standard variance. Different chain sequences have different sets of $B_{i,j}$ generated according to $P(B_{i,j})$ given above. We only allow one monomer of the chain positioned at each lattice site. This is called self avoiding. We say that the conformation is compact if $D(v_i, v_{i+1}) = 1$ for any two successive monomers in the chain. A compact self-avoiding conformation is actually a Hamiltonian path. Hence, to find a native state of the compact self avoiding conformation of the chain we must search for a permutation which minimizes the energy of the chain. This can be described as the following global optimization problem:

$$\min_{p \in \Pi_{\mathcal{N}}} \sum_{i,j=1}^{n} B_{i,j}\, D(v_i, v_j), \tag{3}$$

where the minimization is understood over a subset of permutations. Two monomers interact if they are not successive in sequence and are at unit distance from each other [24].

There are several methods to solve this model [23, 25, 24]. In this paper, we design a new algorithm using tabu search.

Tabu search, developed by Glover [7, 8], is an adaptive iterative search procedure that has been found to be remarkably effective for a spectrum of combinatorial and continuous optimization problems [1, 9, 4]. Previous studies show that for many relatively large size problems, tabu search is better than simulated annealing both in the time required and in the quality of solutions found. Tabu search makes some modifications during the local search process. Starting from a randomly selected solution, tabu search accepts a best solution from candidate solutions in the neighborhood of the current solution. This best solution becomes the next current feasible solution, and the iterative process restart from it. During each iterative process, tabu search intelligently learns information from previous iterative processes, and the procedure moves step by step until an expected minimum (or near minimum) is found. To avoid searching cyclically to recently visited solutions and continue to search without becoming entangled, the management of all modifications is made through tabu condition and the aspiration level function [7, 8].

In this paper, we first discuss the general principle of heuristics for global optimization problems, and analyze two heuristic strategies: simulated annealing and

tabu search. In section 3, we consider a different approach to design the new algorithm for the lattice model of protein folding. We explain how to use tabu search for the lattice model, and in section 4 present our computational results. Finally, in section 5 we make some conclusions.

2. Heuristic Procedures

We consider the general form of global optimization problem:

$$\text{Minimize } f(x) : x \in X \subseteq \Re^n, \tag{4}$$

where f is the real-value objective function : $X \longrightarrow \Re$, and X is the set of feasible solutions. We must compute a feasible solution $x^* \in X$ for which the value $f(x^*)$ is the global minimum. For each solution $x \in X$, we are referred to the neighborhood of x, denoted $N(x)$, as the set which consists of all feasible solutions that can be obtained by applying a modification of x, generally called a *move*. Specifically, we consider a set, denoted M_x, of pre-designed moves acceptable at solution x, from which we can obtain a new solution $x' = x \bigoplus m$, where $m \in M_x$. Therefore, mathematically we have $N(x) = \{x' \mid x' = x \bigoplus m, \exists m \in M_x\}$.

Throughout the following sections a solution x and the related *move* determine each other. We say a solutiou or move $x \in N(x)$ is "improving" or "non-improving" if $f(x') - f(x) < 0$ or $f(x') - f(x) \geq 0$, respectively.

To design an optimization algorithm we consider the construction of an initial solution, the local search process, a stopping criterion, and complexity (required computational time). Much work has been done on these aspects over the past several years [10, 11, 21]. For most cases, difficulties occur because the computational time increases exponentially as the size of the problem increases, and it is not always necessary or possible to use exact algorithm searching for the best solution in reasonable time, especially for most problems arising from industry. Because of these reasons, various heuristic (some local) search techniques have been developed by which we cau obtain acceptable sub-optimal solutious in limited computational time. The most used among these heuristics include: genetic algorithms [6], simulated annealing [12], tabu search [7, 8] and grasp [5]. A terse review on heuristic techniques can be found in Feo and Resende [5].

A fundamental requirement for computational techniques to solve global optimization problems is their ability to escape from local traps and continue to search for an optimal or near-optimal solution, whatever the initial solution is. For a given starting solution x, traditional methods use an improving solution $x' \in N(x)$ to produce a new solution and return the best solution which they first encounter. In such cases, the best solution found may not be a global minimum and is very sensitive to the initial solution. Furthermore, these methods are unable to continue the search when they reach a local minimizer.

Contrary to conventional approaches, given a solution x, some non-improving solution $x' \in N(x)$ (i. e. $f(x') > f(x)$) should be acceptable iu order to have the chance for the heuristic to compute an optimal solution. Simulated annealing uses a straightforward technique. Besides accepting an improving solution, simulated annealing allows non-improving solutions. For a giveu solution x in X, simulated annealing generates a single solutiou $x' \in N(x)$, and if the solution is non-improving accepts it with certain probability (using a probability distribution that depends on $f(x)$ and $f(x')$ with the control of a temperature parameter [12]). Simulated annealing is actually a stochastic iterative approach.

Tabu search shares similar ideas with the simulated annealing. However, the key difference is that tabu search is a local search heuristic but simulated annealing is not. Although tabu search also accepts both improving and non-improving solutions, the tabu search emphasizes the importance of the local search iterative strategy, which works better than such blind search as simulated annealing does. By using specific prescribed conditions which are known as " tabu conditions ", a certain move is considered to be acceptable when it satisfies such a condition; otherwise, it is considered to be tabu during the " tabu period ", the predefined or dynamically managed number of iterations. The set of tabu moves is called " tabu list " which forbids cyclic search of recently visited solutions. The tabu list size plays a role in producing good solutions. If the tabu list size is too small, the search procedure will be trapped among recently visited solutions; If the tabu list size is too big, the search procedure will be prevented from finding interesting solutions not yet encountered. There is no single standard approach to defiue a tabu list size for all problems. For different applications or stages of the search procedure, tabu list sizes can be fixed, dynamically managed, or both during the entire procedure.

To continue the search for a global optimal or near optimal solution, the aspiration level function is defined. If a tabu move satisfies the aspiration level function, it becomes acceptable even if it is non-improving compared to the current best solution. Tabu search procedure is an intelligent search procedure, because at each iterative step it learns from previous information of the iterations, searches all acceptable moves (candidate solutions or sample solutions) in a certain neighborhood structure, and then finds a best solution (even if it is a non-improving). This best solution becomes the starting solution for the next iterative stage until it satisfies some stopping criteria.

We can describe the framework of tabu search by the following pseudo-code:

begin
  choose an initial solution $s \in X$.
    $Bestsolution := s$
    $Bestcost := f(s)$
    $Tabulist := \emptyset$
  while stopping criterion is not satisfied **do**:

```
    Generate a candidate solution set V ⊆ N(s)
    Choose the best s' in V
    Put the move that leads s to s' in Tabulist
    s := s'
    if f(s) < Bestcost then
        Bestsolution :− s
        Bestcost := f(s)
    end then
  end
  return Bestsolution
end
```

Observe that, for a given solution s, the candidate solution (or sample solution) set $V \subseteq N(s)$ should include $s' \in N(s)$ which does not belong to the tabu list or which satisfies the aspiration level function. General possible stopping criteria considered are the following:

1. The best (known) optimal solution is found.
2. The number of iterations exceeds the maximum number of iterations (we have initially specified).

In the next section, we will give a new definition of a neighborhood and describe how the tabn heuristic will be applied to the lattice model.

3. A Tabu Search Based Algorithm for Protein Conformation

The principal difficulty of the protein conformation of the lattice model discussed above, like most other global combinatorial optimization problems, is the existence of multiple optimal. For a chain of n monomers with coordinate number z, there are z^n possible conformations, while the number of all compact conformations is of order of $(z/e)^n$, see e. g. [23]. We have discussed general principles of how to design a global search procedure, but there are still many techniques which we must use especially when we try to solve combinatorial optimization problems because tabu search is only a local search iterative strategy. Among these techniques, the tactical definition of a " move " and " neighborhood " are very important [15, 14]. Different definitions of neighborhood and move have a different impact on the effectiveness of a certain algorithm. There have been two types of moves to solve the lattice model previously discussed [23, 25]. Multiple occupancy uses a penalty function to prevent one or two monomers from being located at the same lattice site[23]. A self-avoid move is selected at random while it conserves the unit bond length and does not produce more than one monomer in the same lattice site [23].

Next, we consider a different definition of move and search and a relatively complex neighborhood structure. The conformation space includes all possible compact

self-avoid conformations. As we know, a compact self-avoiding conformation of the lattice model is actually a Hamiltonian path. We consider a slightly general case. Let us think of the lattice as a graph $G = (V, E)$, where V is the set of all lattice sites and E is the set of all edges which connect any two lattice neighbor sites. $\forall v \in V$, $d(v)$ express the degree of v and $N(v)$ is the neighborhood of v. In the case discussed we have $d(v) \geq 3$, for any $v \subset V$.

Given a chain of n monomers with a ranked set $\mathcal{N} = \{1, 2, \ldots, n\}$, we use the vertices $v_1, v_2, \ldots, v_n$ to describe the lattice sites in which monomers are positioned. Let us consider the permutation $p = \{v_1, v_2, \ , v_n\}$. Then a Hamiltonian path can be referred as a permutation $p = \{v_1, v_2, \ldots, v_n\}$ which satisfies the condition that for any two successive monomers in the chain, their lattice sites are neighborhoods. For a permutation, or in other words, a Hamiltonian path $p = \{v_1, v_2, \ldots, v_n\}$, $\forall k, k' \in Z^+$ with $k < k'$, if $D(v_k, v_{k'}) = 1$ and $D(v_{k+1}, v_{k'+1}) = 1$, we can join edges of $(v_k, v_{k'})$ and $(v_{k+1}, v_{k'+1})$, and then delete edges of (v_k, v_{k+1}) and $(v_{k'}, v_{k'+1})$ in the Hamiltonian path. Therefore we have a new Hamiltonian path (see Fig. 2). Let us denote the above edge patch technique, also called move, as π; then we have the following:

$$\pi(j) = v_{k'-k+1} \quad j = k, k+1, \ldots, k'$$

and the new Hamiltonian path can be written as :

$$p_\pi = \{v_1, v_2, \ldots, v_k, \pi(k+1), \ldots, \pi(k'), v_{k'+1}, \ldots, v_n\}$$

In this way, we can define the neighborhood structure for a given chain:

$$\mathcal{N}(p) = \{\, \pi \mid \exists \pi \ . \quad p_\pi = p \bigoplus \pi \,\}$$

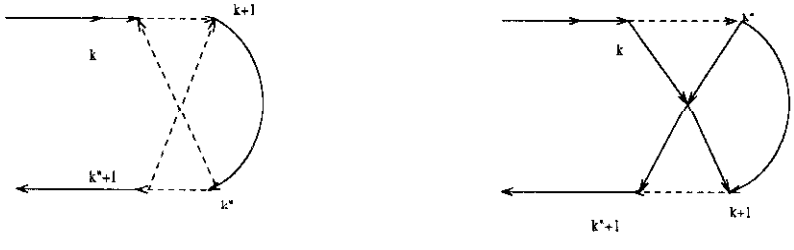

Figure 2. Edge patching for Hamiltonian paths.

Obviously, the above neighbors are defined only for a Hamiltonian path with fixed terminal vertices, so we continue our work on how to change the terminal vertices of a Hamiltonian path.

Because of $d(v) \geq 3$ for any $v \in V$, given a Hamiltonian path $p = \{v_1, v_2, \ldots, v_n\}$, there must be v_k, $v_{k'} \in V$ which satisfy $v_1 \in N(v_k) - \{v_{k-1}\}$ and $v_n \in N(v_{k'}) - \{v_{k'+1}\}$, respectively. Then we also can operate edge patching: deleting edge (v_{k-1}, v_k) or $(v_{k'}, v_{k'+1})$ and joining edge (v_1, v_k) or $(v_{v^n}, v_{k'})$ in Hamiltonian path (see Fig. 3). We can formally denote this kind of patching as π':

$$\pi'(j) = v_{k-j} \quad j = 1, \ldots, k-1$$

and

$$\pi'(j) = v_{n-k-1+j} \quad j = k+1, \ldots, n.$$

Thus, we have two new Hamiltonian paths:

$$p_{\pi'} = \{\pi'_1, \pi'_2, \ldots, \pi'_{k-1}, v_{k+1}, \ldots, v_n\}$$

and

$$p_{\pi'} = \{v_1, v_2, \ldots, v_{k'}, \pi'_{k'+1}, \ldots, \pi'_n\}.$$

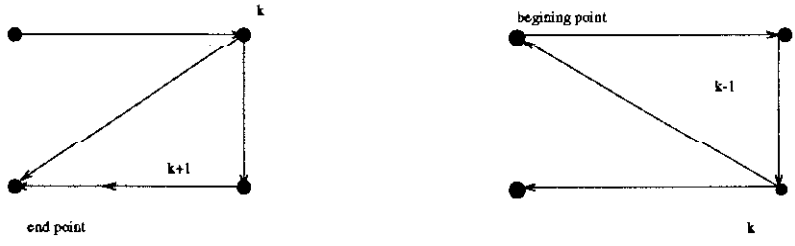

Figure 3. Edge patching for Hamiltonian paths with different terminal points.

Note that our edge patching is not constrained to the lattice model that we consider. In our implementation, instead of a single tabu list, we use two tabu lists:

1. $T1$ is a list of all tabu edge patches or moves for a path with fixed starting and terminal sites; $|T1|$ is the number of recently visited solutions, called the size of the tabu list $T1$.

2. $T2$ is a list of starting and terminal sites from which we can generate a Hamiltonian path, and $|T2|$ is the size of the tabu list $T2$.

Both $|T1|$ and $|T2|$ are managed dynamically during different calculating stages. The algorithm is described by the following pseudocode:

Input

B(i,j), D(i,j)

$|T1|$=size of $T1$ with fix terminal vertexes

$|T2|$=size of $T2$ for different terminal vertices
$MaxIterationNumber$=maximum number of iterations between two improvements of the function
$MaxIterationNumber1$=maximum number of iterations between two improvements of the function
$BestIterationNumber$= the number of iterations when the solution improved
$BestIterationNumber1$= the number of iterations when the solution improved when terminal points are fixed

Initialization
generate a Hamiltonian path s
$IterationNumber = 0$ (iteration counter)
$T2 := \emptyset$
$BestSolution = s$
While $(IterationNumber - BestIterationNumber) < MaxIterationNumber)$
$IterationNumber := IterationNumber + 1$
for a path $s0$ with fixed terminal vertices generated in neighbors of s
$T1 := \emptyset$
$IterationNumber1 = 0$ (iteration counter)
$BestSolution1 = s0$
while$(IterationNumber1 - BestIterationNumber1) < MaxIterationNumber1)$ **d**
$IterationNumber1 := IterationNumber1 + 1$
$s'' = s0$
let s' be the best solution so far in the neighbor of s''
npdate the tabu list $T1$
if $f(s') < f(BestSolution1)$ then
$BestSolution1 := s'$
$BestIterationNumber1 = IterationNumber1$
end then
$s'' := s'$
end **do**
return BestSolution1
end **for**
let $s00$ be the best solution so far found
update the tabu list $T2$
if$f(s00) < f(BestSolution)$ then
$BestSolution := s00$
$BestIterationNumber = IterationNumber$
end then
$s := s00$
end **While**
output $Bestsolution$
end

4. Computational Results

The algorithm was implemented in C and the computational results have been obtained on a SUN SPARC 10 workstation.

Table 1. Results for sequences 1-100

sequence	*TABU*	*SAM*	*sequence*	*TABU*	*SAM*
1	-75.25	-75.17	51	-73.68	-73.68
2	-81.02	-81.02	52	-75.92	-75.92
3	-76.69	-77.44	53	-77.00	-77.00
4	-79.67	-80.19	54	-75.18	-77.54
5	-80.68	-80.68	55	-76.68	-76.06
6	-77.99	-77.99	56	-75.59	-75.59
7	-74.65	-75.57	57	-79.44	-77.73
8	-76.83	-76.57	58	-75.03	-75.03
9	-76.79	-76.79	59	-76.82	-76.82
10	-71.34	-70.53	60	-77.26	-77.26
11	-73.74	-72.87	61	-80.20	-81.20
12	-77.92	-77.92	62	-80.83	-80.83
13	-74.71	-72.83	63	-74.32	-74.32
14	-76.42	-76.38	64	-76.03	-76.03
15	-77.18	-71.58	65	-76.67	-76.67
16	-78.40	-78.40	66	-72.96	-72.62
17	-78.63	-78.63	67	-80.93	-80.93
18	-76.02	-76.02	68	-79.70	-79.69
19	-72.56	-72.56	69	-79.45	-79.23
20	-72.03	-72.03	70	-74.09	-74.09
21	-80.30	-80.39	71	-74.26	-74.26
22	-73.77	-76.33	72	-76.16	-76.16
23	-75.61	-74.23	73	-76.18	-73.38
24	-70.38	-70.38	74	-71.79	-70.86
25	-77.03	-77.03	75	-75.64	-75.64
26	-73.33	-73.33	76	-81.75	-81.75
27	-77.15	-76.30	77	-74.62	-74.62
28	-82.05	-82.05	78	-75.95	-75.52
29	-77.82	-77.82	79	-72.42	-72.42
30	-77.84	-77.57	80	-74.95	-76.46
31	-78.07	-77.88	81	-75.78	-75.78
32	-74.24	-74.24	82	-71.48	-71.48
33	-75.57	-75.57	83	-73.94	-73.94
34	-78.17	-78.17	84	-75.83	-75.33
35	-80.11	-80.11	85	-80.20	-80.20
36	-74.39	-74.39	86	-73.43	-72.46
37	-76.84	-75.88	87	-79.09	-79.09
38	-80.90	-79.83	88	-74.91	-74.91
39	-72.68	-72.68	89	-75.27	-75.17
40	-75.05	-75.03	90	-76.70	-76.46
41	-82.30	-82.30	91	-72.85	-70.99
42	-75.31	-75.58	92	-77.34	-79.36
43	-77.75	-77.75	93	-76.55	-79.04
44	-78.91	-78.91	94	-78.07	-77.16
45	-81.18	-81.18	95	-75.65	-75.65
46	-75.21	-75.06	96	-79.19	-79.19
47	-78.04	-78.20	97	-68.94	-68.85
48	-73.55	-73.26	98	-76.80	-76.80
49	-74.08	-73.16	99	-76.91	-74.23
50	-73.99	-73.99	100	-75.24	-74.83

Table 2. Results for sequences 101-200

sequence	*TABU*	*SAM*	*sequence*	*TABU*	*SAM*
101	-75.33	-73.02	151	-73.12	-73.12
102	-76.80	-76.80	152	-71.52	-72.67
103	-79.36	-79.36	153	-74.49	-74.49
104	-79.15	-77.19	154	-81.49	-81.49
105	-75.41	-79.45	155	-71.29	-71.18
106	-73.68	76.20	156	75.88	75.65
107	-79.31	-79.31	157	-70.63	-70.63
108	-76.22	-76.99	158	-74.55	-74.27
109	-76.50	-76.18	159	-71.98	-72.09
110	-76.25	-76.25	160	-74.71	-74.01
111	-74.07	-74.07	161	-75.87	-75.09
112	-78.20	-78.21	162	-77.33	-77.33
113	-71.27	-71.27	163	-74.90	-78.88
114	-79.38	-79.38	164	-68.70	-68.70
115	-82.94	-81.13	165	-73.61	-73.61
116	-77.22	-77.22	166	-71.22	-71.22
117	-82.79	-81.54	167	-74.82	-74.82
118	-77.24	-79.92	168	-81.96	-77.87
119	-77.08	-75.25	169	-78.84	-78.71
120	-84.58	-84.58	170	-74.60	-74.60
121	-73.91	-73.91	171	-83.61	-83.26
122	-75.25	-75.25	172	-77.30	-77.30
123	-73.43	-72.58	173	-73.24	-71.36
124	-78.82	-78.82	174	-72.28	-70.84
125	-74.03	-74.03	175	-75.88	-75.88
126	-79.44	-79.44	176	-80.32	-80.32
127	-76.30	-74.86	177	-73.68	-72.68
128	-73.84	-77.84	178	-72.68	-72.76
129	-79.85	-79.85	179	-74.66	-74.66
130	-81.02	-78.54	180	-79.53	78.50
131	-80.26	-80.26	181	-76.04	-75.19
132	-76.99	-76.99	182	-72.69	-71.00
133	-76.03	-76.04	183	-77.21	-76.35
134	-74.31	-74.31	184	-84.11	-84.11
135	-77.98	-77.98	185	-78.08	-76.87
136	-74.36	-74.36	186	-72.67	-70.62
137	-72.62	-71.93	187	-78.78	-78.78
138	-75.82	-76.56	188	-78.51	-77.44
139	-79.19	-78.82	189	-70.67	-70.14
140	-81.96	-81.96	190	-77.71	-76.42
141	-72.65	-71.25	191	-73.64	-73.46
142	-79.74	-79.74	192	-76.40	-76.40
143	-72.03	-72.03	193	-76.11	-75.66
144	-74.29	-74.29	194	-72.48	-72.46
145	-77.45	-77.45	195	-77.77	-77.77
146	-74.34	-74.34	196	-76.02	-75.17
147	-81.75	-81.75	197	-78.06	-75.87
148	-75.70	-73.37	198	-75.02	-74.96
149	-72.40	-72.25	199	77.50	76.29
150	-75.57	-75.57	200	-79.69	-79.69

The tabu search applied to the lattice model is outlined in section 1. We tested our algorithm using the same original data in [24]. Among 200 protein sequences which we tested, only few sequences are worse than the results previously reported. For all other cases we obtained the same or better results.

All computational results are presented in Tables 1 & 2. In the headings of the tables, *sequence* is the number of protein chain, $TABU$ is the solution obtained by our algorithm, and SAM is the solution obtained by the method described in [23].

5. Concluding Remarks

In this paper we developed and implemented the tabu search heuristic applied to the lattice model outlined in the introduction section. We tested our algorithm using the same data given in [24]. Among 200 protein sequences which we tested, only few sequences are worse than the results previously reported. In all other cases we obtained the same or better results. As we discussed above, there is no standard way to define the size of the tabu list. This is an important computational issue regarding the quality of the best solution found. Other parameters such as the maximum number of iterations between two improving solutions and the number of iterations during which a move is tabu (i.e. tabu period) are also difficult to determine. The effectiveness of using tabu search heuristics depends greatly on the insight into the practical problem we want to solve and numerical experimentation. Therefore, extensive computational experimentation is needed to improve the efficiency of the algorithm presented in this paper. In our current implementation, we tried to mauage the tabu size dynamically and fixed, and observed the relationship between tabu list sizes and the best solution found. In our tabu algorithm, we tested only the short term memory. To fnrther improve the performance of the algorithm, the introduction of concepts snch as long term memory, diversification of search, and oscillation should be incorporated and implemented. This is part of our current investigation. Furthermore, larger classes of problems can be solved by implementing tabu search on a parallel computer environment [18].

Acknowledgements

The authors would like to thank Dr. A. Šali for providing all test data and Professor Fred Roberts for his support during the DIMACS special year on "Mathematical Support for Molecular Biology."

References

[1] Battiti, R. and Tecchiolli, G., *The Reactive Tabu Search,* ORSA J. COMPUT. 6 (1994), pp. 126–140.

[2] Chan, H. S., and K. A. Dill, *The Protein Folding Problem,* Physics Today (February 1993), pp. 24–32.

[3] Coleman, T., D. Shalloway and Z. Wu, *A Parallel Build-Up Algorithm for Global Energy Minimizations of Molecular Clusters Using Effective Energy Simulated Annealing,* J. Global Opt. 4 (1994), pp. 171–185.

[4] Cvijović, D. and J. Klinowski, *Taboo Search: An Approach to the Multiple Minima Problem,* Science 267 (1995), pp. 664–666.

[5] Feo, T. A., and M. G. C. Resende, *Greedy Randomized Adaptive Search Procedures,* J. Global Opt. 6 (1995), pp. 106–133.

[6] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, 1989.

[7] Glover, F., *Tabu Search, PART I,* ORSA J. COMPUT. 1 (1989), pp. 190–206.

[8] Glover, F., *Tabu Search, PART II,* ORSA J. COMPUT. 2 (1990), pp. 4–32.

[9] Glover, F., E. Taillard and D. de Werra, *A User's Guide to Tabu Search,* Annals of Operations Research 41 (1993), pp. 3–28.

[10] Horst, R. and P.M. Pardalos (Editors), *Handbook of Global Optimization,* Kluwer Academic Publishers (1995).

[11] Horst, R., P.M. Pardalos and N.V. Thoai, *Introduction to Global Optimization,* Kluwer Academic Publishers (1995).

[12] Kirkpatrick, S., C. D. Gelatt and Jr., M. P. Vecchi, *Optimization by Simulated Annealing,* Science, 220 (1983), pp. 671–680.

[13] Kolinski, A., and J. Skolnick, *Monte Carlo Simulations of Protein Folding. I. Lattice Model and Interaction Scheme,* PROTEINS 18 (1994), pp. 338–352.

[14] Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (Editors), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (Wiley-Interscience, Chichester, 1985).

[15] Lin, S., and B. W. Kernighan, *An Effective Heuristic Algorithm for the Traveling Salesman Problem,* Oper. Res. 21 (1973), pp. 498–516 .

[16] Maranas, C. D., and C. A. Floudas, *Global Minimum Potential Energy Conformations of Small Molecules,* J. Global Opt. 4 (1994), pp. 135–170.

[17] Northby, J. A., *Structure and Binding of Lennard-Jones Clusters:*$13 < n < 147$, J. Chem. Phys. 87 (1987), pp. 6166–6178.

[18] Pardalos, P. M., L. Pitsoulis, T. Mavridou and M.G.C. Resende, *Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search and GRASP,* In "Parallel Algorithms for Irregularly Structured Problems", Springer-Verlag, Lecture Notes in Computer Science Vol. 980 (1995) (Editors: A. Ferreira and J. Rolim), pp. 317–331.

[19] Pardalos, P. M., D. Shalloway and G. L. Xue (Editors), *Global Minimization of Nonconvex Potential Energy Functions: Molecular Conformation and Protein Folding,* DIMACS series Vol. 23, American Mathematical Society (1995)

[20] Pardalos, P. M., D. Shalloway and G. L. Xue, *Optimization Methods for Computing Global Minima of Nonconvex Potential Energy Functions,* J. Global Opt. 4 (1994), pp. 117–133.

[21] Papadimitriou, C. H., and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[22] Richards, F. M., *The Protein Folding Problem,* Scientific American (January 1991), pp. 54–63.

[23] Šali, A., E. Shakhnovich and M. Karplus, *Kinetics of Protein Folding: A Lattice Model Study of the Requirements for Folding to the Native State,* J. Mol. Biol. 234 (1994), pp. 1614–1636.

[24] Šali, A., E. Shakhnovich and M. Karplus, *Thermodynamics and Kinetics of Protein Folding from Lattice Monte Carlo Simulations,* In "Global Minimization of Nonconvex energy Functions: Molecular Conformation and Protein Folding" (Pardalos, P. M., D. Shalloway and G. L. Xue. Editors), DIMACS series Vol. 23, American Mathematical Society (1995).

[25] Shakhnovich, E. and A. Gutin, *Enumeration of All Compact Conformation of Copolymers With Random Sequence of Links,* J. Chem. Phys. 93 (1990), pp. 5967–5971.

[26] Shakhnovich, E., G. Farztdinov, A. M. Gutin and M. Karplus, *Protein Folding Bottleneck: A Lattice Monte Carlo Simulation,* Phys. Rev. Lett. 67 (1991), pp. 1665–1668.

[27] Xue, G., *Molecular Conformation on the CM-5 by Parallel Two-Level Simulated Annealing,* J. Global Opt. 4 (1994), pp. 187–208.

[28] Wolynes, P. G., J. N. Onuchic and D. Thirumalai, *Navigating the Folding Routes,* Science 267 (1995), pp. 1619–1620.