An Efficient Genetic Algorithm for Predicting Protein Tertiary Structures in the 2D HP Model

Thang N. Bui and Gnanasekaran Sundarraj
Computer Science Program
The Pennsylvania State University at Harrisburg
Middletown, PA 17057
{tbui, gxs241}@psu.edu

ABSTRACT

Given the amino acid sequence of a protein, predicting its tertiary structure is known as the protein folding problem. This problem has been widely studied under the HP model in which each amino acid is classified, based on its hydrophobicity, as an H (hydrophobic or non-polar) or a P (hydrophilic or polar). Conformation of a protein in the HP model is embedded as a self-avoiding walk in either a two-dimensional or a three-dimensional lattice. The protein folding problem in the HP model is to find a lowest energy conformation. This problem is known to be NP-hard in both two-dimensional and three-dimensional square lattices. In this paper, we present an efficient genetic algorithm for the protein folding problem under the HP model in the two-dimensional square lattice. A special feature of this algorithm is its usage of secondary structures, that the algorithm evolves, as building blocks for the conformation. Experimental results on benchmark sequences show that the algorithm performs very well against existing evolutionary algorithms and Monte Carlo algorithms.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods* ; J.3 [**Life and Medical Sciences**]: Biology and genetics

General Terms

Design, Algorithms

Keywords

Genetic Algorithm, Protein Folding Problem, 2D HP Model

1. INTRODUCTION

Amino acids are the building blocks of proteins. When two amino acids bind together in a protein sequence, a water molecule is released, and the joined amino acids are known as amino acid *residues.* The bond between two amino acid residues is referred to as a *peptide* bond. Proteins are polypeptide chains of amino acid residues. There are 20 different amino acids that make up most of the known proteins. The linear sequence of amino acids that makes up a protein is called the *primary structure* of the protein. At the proper solvent composition and temperature, a protein folds from its primary structure into a specific three-dimensional shape called its *tertiary structure.* The tertiary structure of a protein, also called its *native* state, determines the protein's biological functions. The native state of a protein generally corresponds to the lowest free energy state for the protein. The protein folding problem is the problem of determining the native state of a protein given its primary structure. The sequence of structural changes that the protein undergoes during the folding process is referred to as the *folding pathway.* In general, determination of the folding pathway is also considered part of the protein folding problem. In this paper, we consider only the problem of determining the tertiary structures and not the folding pathways.

The protein folding problem has been widely studied under the HP model [5, 12] in which conformation of a protein is embedded as a self-avoiding walk in either a two-dimensional (2D) or a three-dimensional (3D) lattice. There are many variations of lattices used for the study of this problem in the HP model [9]. In this paper, we consider only square lattices where each lattice site is arranged orthogonally to its neighbors. In the remainder of the paper, we use the term *2D HP model* to refer to the HP model in the 2D square lattice. Similarly, we refer to the HP model in the 3D square lattice as simply the *3D HP model.* The objective of the protein folding problem is to determine a conformation of minimum energy.

The protein folding problem in the 2D HP model has been proved to be NP-hard [4] as is the case with the 3D HP model [2]. There exist a number of approximation algorithms for both the 2D and the 3D HP models [6, 14, 15]. Aside from these approximation algorithms, there are also heuristic algorithms for the 2D and the 3D HP models. They generally fall into two categories: evolutionary algorithms [8, 10, 11, 16, 21, 22, 23] and Monte Carlo algorithms [1, 7, 13, 18, 23]. Even though this problem has been studied for a long time, there is no algorithm that works better on all benchmark sequences in terms of solution quality and running time. Many different techniques have been tried in these algorithms, and each technique seems to work for a specific sequence, but fails in other sequences. In this paper, we present a genetic algorithm (GA) for the protein folding problem in the 2D HP model that performs very well against the existing evolutionary and Monte Carlo algorithms on a common set of benchmark sequences.

The rest of the paper is organized as follows. Section 2 gives the preliminaries and the formal definition of the protein folding problem in the 2D HP model and Section 3 describes our algorithm in detail. The experimental results comparing our algorithm against other known algorithms are given in Section 4 and the conclusion is given in Section 5.

2. PRELIMINARIES

Proteins play a variety of very important roles in a biological system. For examples, the alpha-keratin protein is a key structural component in biological materials such as hair and fingernails. Some proteins such as actin and myosin make muscular movement possible. Enzymes are proteins that help in the digestion of food. Hemoglobin is a protein that helps carry oxygen in blood. Antibodies are proteins that form an important part of our immune system. Other proteins help in the control of brain signals and copying genes during cell division.

Each protein starts out as a sequence of amino acids, the primary structure. The protein then folds itself into a three dimensional shape, the tertiary structure. This folding process happens very fast, in less than one second for most proteins, and occurs immediately after the primary structure of the protein is created. A protein is not active until it has completed the folding and settled in its final shape (the native state). A protein's function depends mainly on its tertiary structure which in turn depends on its primary structure. Mistakes in the folding process create proteins with abnormal shapes which are the causes of diseases such as cystic fibrosis, Alzheimer's, and "mad cow" [3]. It is believed that if we could predict the tertiary structures of proteins from their sequences, we would be able to treat these diseases better. The knowledge of protein tertiary structures also has other applications such as in the structure-based drug design area.

A protein's amino acid sequence can be readily determined through experimental techniques such as Mass Spectroscopy. However, predicting the protein's tertiary structure from its amino acid sequence remains one of the most difficult problems in structural biology. X-ray Crystallography and Nuclear Magnetic Resonance Spectroscopy are the two laboratory structure determination techniques that are in use to experimentally determine the tertiary structures of proteins. Even though these techniques have been extremely powerful in solving macromolecular structures, they are laborious, time consuming, and are limited to very small proteins because of the fundamental resolution limit of these techniques. It is estimated that there are 80,000 to 100,000 proteins produced in a human cell. However, only a small fraction of these have experimentally determined structures. So computational techniques are also being studied in parallel to expedite this process [17]. But success has been limited in this front also. Exhaustive search of a protein's conformational space using computational techniques is not feasible even for small protein sequences because of the exponential number of possible solutions. This led to several simplified models for the protein folding problem which allow efficient sampling of the conformational space. These models often help understand the physics governing the protein folding process.

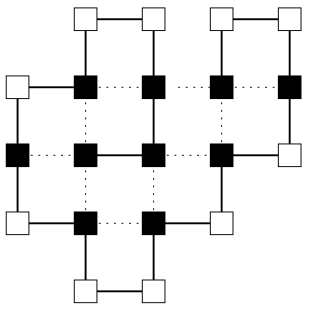
Figure 1: An optimal conformation for the sequence HHHPPHPHPHPPHPHPHPPH in the 2D square lattice. The black squares denote the hydrophobic (H) amino acid residues and the white squares denote the hydrophilic (P) residues. Dotted lines denote H-H contacts.

2.1 HP Model

The HP model is based on the observation that the hydrophobic interaction between the amino acid residues is the driving force for the protein folding and for the development of native state in proteins [12, 19]. In this model, each amino acid is classified based on its hydrophobicity as an H (hydrophobic or non-polar) or a P (hydrophilic or polar). Conformation of a protein in the HP model is then represented as a self-avoiding walk in a 2D or a 3D lattice where each lattice site is occupied by one amino acid residue, connected to its sequence neighbor(s) on adjacent lattice site(s). In the 2D square lattice model, each residue has at most 4 lattice neighbors. Similarly, there are at most 6 lattice neighbors for each residue in the 3D model. Note that each residue has at most two sequence neighbors in both the 2D and the 3D models. An H-H *contact* is a pair of H's that are adjacent in the conformation, but not in the sequence. In the HP model, each H-H contact in the conformation is assigned an energy value of -1. Since the native state of a protein generally corresponds to the lowest free energy state for the protein, the optimal conformation in the HP model is the one that has the maximum number of H-H contacts, which gives the lowest energy value. Figure 1 shows an optimal conformation for the sequence HHHPPHPHPHPPHPHPHPPH in the 2D square lattice. The conformation has 10 H-H contacts, denoted by the dotted lines, and hence an energy value of -10.

For the 2D HP model, there exists a 1/4-approximation algorithm, i.e., an algorithm that guarantees to return a solution whose number of H-H contacts is at least 1/4 that of the optimal solution [6]. The approximation factor was later improved to 1/3 [14]. We have a 3/8-approximation algorithm for the 3D HP model [6]. This factor was later improved slightly to 0.37501 [15].

2.2 Problem Definition

Let $A = a_1a_2a_3 \dots a_n$, where $a_i \in \{\text{H,P}\}$, $1 \leq i \leq n$, be a string representing an amino acid residue sequence in the HP model. We denote a conformation of A by a sequence of fold directions starting from the lattice site occupied by the first amino acid residue a_1. There are two common types of encodings that are in use to denote a conformation in the 2D square lattice. In the first encoding scheme, the fold directions are relative to the lattice, and in the second one, they are relative to the conformation itself. The first scheme uses the four symbols u, d, l, and r to denote the fold directions up, down, left, and right, respectively. The symbols s, l, and r are used to denote the fold directions straight, left, and right in the second encoding scheme. Using the first scheme, the conformation given in Figure 1 can be denoted as *ruuldlddrdrururuuld*. If the input sequence A is of length n, a conformation of A will be of length $n-1$ in this scheme. In the second encoding scheme, the first fold direction is fixed as s, denoting straight, and the rest follows from this. The conformation given in Figure 1 can be denoted as *slsllrlslrllrlrlsll* using the second scheme. In this scheme, the first fold direction is often not explicitly mentioned. So a conformation for a sequence of length n can be represented using $n-2$ symbols.

In this paper, we use the first encoding scheme where the fold directions are relative to the 2D square lattice. We call a conformation *valid*, if each lattice site is occupied by at most one amino acid residue, and each residue is connected to its sequence neighbor(s) on adjacent lattice site(s). Otherwise the conformation is invalid. We are now ready to define the protein folding problem in the 2D HP model formally.

Input: A string $A = a_1a_2a_3 \dots a_n$, where $a_i \in \{\text{H,P}\}$, $1 \leq i \leq n$, representing an amino acid residue sequence.

Output: A valid 2D square-lattice conformation of A that has the maximum number of H-H contacts.

3. ALGORITHM

In this section, we present the motivation and the details of our algorithm for the protein folding problem in the 2D HP model.

The main idea behind our algorithm is the use of *secondary structures* in exploring the conformation space of a protein. The actual conformations of proteins are centered around polypeptide backbones that are in some particular shape. These backbone conformations are known as the secondary structures, which are generally classified into the following three types: the α-helix, the β-strand or β-sheet, and the turns or loops that connect the helices and strands. The secondary structures and their side chains are then packed tightly to form the three-dimensional tertiary structure of the proteins. Extending this concept to the HP model, a subsequence of hydrophobic residues is constrained to a specific conformation in the process of determining the optimal conformation for the input sequence. That is, a secondary structure here is just a conformation of a subsequence consisting only of hydrophobic residues. This idea has already been explored before in [13]. By using specific secondary structures, best-known conformations were achieved in [13] for some of the benchmark sequences. However, no reasons were given for choosing one secondary structure over another for a hydrophobic subsequence. Since there are too many secondary structures even for a small subsequence of hydrophobic residues, it is impractical to try them all. We use a genetic algorithm to systematically evolve the secondary structures which are then used as building blocks to evolve the best conformation for the given input sequence.

Our algorithm consists of two parts: the Secondary Structure Genetic Algorithm (SSGA) and the Protein Folding Genetic Algorithm (PFGA). We first scan the input amino acid sequence to determine the length k of the longest subsequence of hydrophobic residues. We next run SSGA to create a library of secondary structures for the hydrophobic sequence of length k. Finally, we run PFGA which uses the secondary structures from the library as building blocks.

In what follows, we first describe PFGA assuming that the library of secondary structures has already been created. We then describe SSGA which has a similar structure to that of PFGA.

3.1 Protein Folding Genetic Algorithm (PFGA)

We use a steady state GA combined with local optimization schemes for folding the protein sequence. The algorithm is given in Figure 2. The details of the algorithm are given in the following sections.

```
PFGA($A$)
  generate initial population $P$
  **while** stopping criteria not met
    select two parents $p_1, p_2$
    $o_1, o_2 \leftarrow$ crossover($p_1, p_2$)
    mutate offspring $o_1, o_2$
    adjust offspring $o_1, o_2$
    local optimize offspring $o_1, o_2$
    replace $(P, p_1, p_2, o_1, o_2)$
  **end-while**
  **return** the best member of $P$
```

Figure 2: The PFGA algorithm

3.1.1 Initial Population

If the input amino acid sequence is of length n, then each individual in the population is a string of length $n-1$ over the alphabet $\Sigma = \{u, d, l, r\}$, and denotes a valid conformation in the 2D square lattice. This is the first encoding scheme described in Section 2.2.

Let $A = a_1 \dots a_n$ be the input sequence. Let k be the length of the longest hydrophobic subsequence in A. Let $C = c_1 \dots c_{n-1}$ be the conformation of an individual. Figure 3 shows the algorithm for creating an individual in the initial population. For $i = 1$ to $n-1$, we consider the successive pairs of residues (a_i, a_{i+1}) in A, and construct the fold direction c_i based on the placement of a_{i+1} in the lattice with respect to that of a_i. If a_i is the beginning of a hydrophobic subsequence of length k, then we randomly decide whether to use a secondary structure for this hydrophobic subsequence or not. If we do, a secondary structure, S, is randomly selected from the library and placed in the conformation as $c_i \dots c_{i+k-2}$. Now, if $c_1 \dots c_{i+k-2}$ becomes an invalid conformation, we try the 90°, 180°, and 270° rotations of S, one at a time, until it gives a valid conformation. If it still doesn't give a valid conformation, we recreate the individual. Once we get a valid conformation, we then move to the pair (a_{i+k-1}, a_{i+k}). On the other hand, if we do not want to use a secondary structure, or if a_i is not the beginning of a hydrophobic subsequence of length k, then we select an available fold direction for (a_i, a_{i+1}) at random and place it in the conformation as c_i. If there is no available

```
CREATE_INDIVIDUAL($A$)
  $i \leftarrow 1$
  while $i < n$
    if $a_i \dots a_{i+k-1}$ is a hydrophobic subsequence then
      randomly decide whether to use a secondary
        structure from the library or not
      if yes then
        randomly select a secondary structure, $S$, from
          the library and place it as $c_i \dots c_{i+k-2}$
        if $c_1 \dots c_{i+k-2}$ is a valid conformation then
          $i \leftarrow i + k - 2$
        else
          try $90^\circ$, $180^\circ$, and $270^\circ$ rotations of $S$, one
            at a time, until $c_1 \dots c_{i+k-2}$ is valid
          if $c_1 \dots c_{i+k-2}$ is still invalid then
            $i \leftarrow 0$
          end-if
        end-if
      else
        randomly select one of the available fold
          directions for $(a_i, a_{i+1})$ and place it as $c_i$
        if no fold direction is available then
          $i \leftarrow 0$
        end-if
      end-if
    else
      randomly select one of the available fold
        directions for $(a_i, a_{i+1})$ and place it as $c_i$
      if no fold direction is available then
        $i \leftarrow 0$
      end-if
    end-if
    $i \leftarrow i + 1$
  end-while
  return $C$
```

Figure 3: Algorithm to create an individual

fold direction, we recreate the individual. When a secondary structure is used in an individual, it is marked so that no crossover cutpoint or mutation occurs inside that secondary structure. Note that the algorithm always creates a valid individual. We create 1000 individuals in this manner.

The fitness of each individual is then calculated, and it is simply the number of H-H contacts in the conformation for the input sequence A. The initial population is formed by selecting the 500 best-fit individuals from the original set of 1000.

3.1.2 Crossover

We use tournament selection to select the two parents from the population as follows. Two pools of individuals are selected at random from the population so that the size of each pool is 10% of the population size. The best member of each pool is selected as a parent. We use one-point crossover on these two parents to create two offspring.

3.1.3 Mutation

Mutation is applied to both offspring, and we use uniform mutation. If the offspring does not contain a secondary structure, then each fold direction in the encoding is mutated with a probability that varies with time from 0.2 at the beginning to 0.1 at the end of the algorithm. A fold direction is mutated by replacing it with a fold direction selected at random from Σ. The mutation probability is varied with time so that more exploration is done in the initial generations and more exploitation is done in the later ones.

If the offspring contains a secondary structure, no mutation is performed inside this structure. We treat this as a single unit, and it is mutated with a time varying probability as above. A secondary structure is mutated by replacing it with some other structure selected at random from the library.

3.1.4 Offspring Adjustment

It is most likely that after crossover and mutation, the offspring produced will have invalid conformations. We adjust the offspring, as explained below, so that their conformations become valid. Let A be the input sequence of amino acid residues. Let C be an offspring to be adjusted.

Offspring with no secondary structure. Starting from the first amino acid residue in the input sequence A, and for each subsequent residue, we first place the residue in the 2D lattice as per its current fold direction dictated by the offspring C. If this is infeasible, i.e., if the lattice site is already occupied, we place the residue at one of the randomly chosen, available adjacent lattice sites, and modify the fold direction in the offspring accordingly. If there is no available adjacent lattice site, then the offspring is discarded.

Offspring with one secondary structure. Let $A = a_1 \dots a_p \dots a_q \dots a_n$ be the input sequence, where $a_p \dots a_q$ is the longest hydrophobic subsequence. Let A_1, A_2, and A_3 denote the three subsequences $a_1 \dots a_p$, $a_p \dots a_q$, and $a_q \dots a_n$, respectively. Let α, β, and γ denote the conformations for A_1, A_2, and A_3, respectively, in the offspring C. Here β denotes the secondary structure that represents the fold directions for the hydrophobic subsequence A_2.

Let A_1^R and α^R denote the reverse of A_1 and α, respectively. Let $\bar{\alpha}$ denote the conformation obtained from α^R by taking the opposite of each fold direction in α^R. The opposite of l is r, and vice versa. Similarly, the opposite of u is d, and vice versa. For example, if $\alpha = ldrru$, then $\bar{\alpha} = dllur$.

Now the offspring C can be adjusted as follows. First, the hydrophobic residues in A_2 are placed at the lattice sites as per the fold directions in β. Next, the residues in A_1^R, starting from a_p and ending in a_1, are placed in the lattice sites as dictated by $\bar{\alpha}$. For any residue, if its lattice site is already occupied, we randomly choose one of the available adjacent lattice sites for the residue. If there is no available adjacent lattice site, the offspring is discarded. We then place the residues a_q through a_n in A_3 based on γ in a similar manner.

Finally, the adjusted offspring C can be obtained by reading off the fold directions from the just constructed embedding starting from a_1.

Offspring with more than one secondary structure. Here we explain the adjustment operation for an offspring with only two secondary structures. However, this process can be extended to include any number of secondary structures.

Let $A = a_1 \dots a_p \dots a_q \dots a_r \dots a_s \dots a_n$ be the input sequence. Let A_1, A_2, A_3, A_4, and A_5 denote the subsequences $a_1 \dots a_p$, $a_p \dots a_q$, $a_q \dots a_r$, $a_r \dots a_s$, and $a_s \dots a_n$, respectively, in A where A_2 and A_4 are the longest hydrophobic subsequences. Let $\alpha, \beta, \gamma, \delta$, and η be the conformations for A_1, A_2, A_3, A_4, and A_5, respectively, in the offspring C. Here β and δ denote the secondary structures that represent the fold directions for the hydrophobic subsequences A_2 and A_4, respectively.

We first place the residues in the subsequences A_1, A_2, and A_3 as before. Then the hydrophobic residues in A_4 are placed in the lattice sites as per the fold directions in δ. For any residue in A_4, if its lattice site is occupied, we try the 90°, 180°, and 270° rotations of the secondary structure δ, one at a time, until all the residues in A_4 can be placed in the lattice sites. If none of the rotated secondary structures works, then the offspring is discarded. Once all the residues in A_4 are placed, we then place the residues in A_5. The adjusted offspring C can then be obtained by reading off the fold directions from the finished embedding starting from a_1.

3.1.5 Local Optimization

We use two local optimization schemes, explained below, to improve the fitness of the offspring before introducing one of them into the population. Neither of these schemes requires an empty lattice site to perform the local optimization. For each offspring, Scheme 1 is applied followed by Scheme 2.

Scheme 1. In this scheme, we consider a 4-point configuration in the offspring consisting of a lattice square, where each corner of the square is occupied by an H or a P as shown in Figure 4(a). The sequence is then refolded so that the conformation remains the same except for the fold directions around the four lattice sites W, X, Y, and Z. The new fold directions around these lattice sites are as shown in Figure 4(b). Once the sequence is refolded, we calculate the number of H-H contacts. If the new conformation is better than the current one, we replace the current conformation with the new one. We perform this refolding operation on each 4-point configuration found in the offspring.

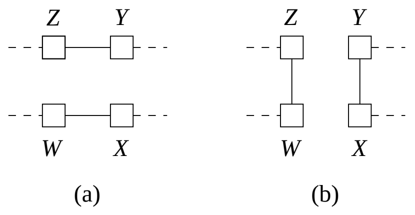

Figure 4: Local Optimization Scheme 1

It should be noted that this scheme will sometimes break the conformation into two parts, and create a loop in one of them. In order to break the loop and to join the two parts, we select a lattice square as shown in Figure 5(a) where the lattice sites S, T, U, and V are the four corners of the lattice square. The sequence is then refolded as shown in Figure 5(b) so that the conformation remains the same except for the fold directions around the four lattice sites S, T, U, and V. If there are more than one such place to break the loop, then we select the one that gives the highest fitness. If there is no way to break the loop without decreasing the fitness of the offspring, we ignore the 4-point configuration that creates this loop, and search for the next 4-point configuration. A variation of this scheme has already been explored in [18].

Scheme 2. This scheme considers 6-point configurations

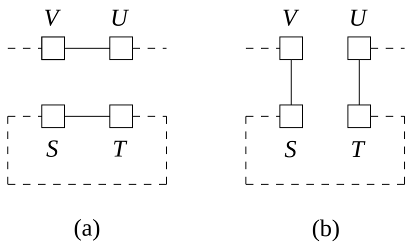

Figure 5: Local Optimization Scheme 1 - loop break

in the offspring consisting of two adjacent lattice squares. One such configuration is shown in Figure 6(a). The sequence in this case is refolded so the conformation around the six lattice sites U, V, W, X, Y, and Z is as shown in Figure 6(b). We then compute the number of H-H contacts in the new conformation. If it is more than the current conformation's, the new conformation replaces the current one. This refolding process is done for each 6-point configuration found in the offspring.

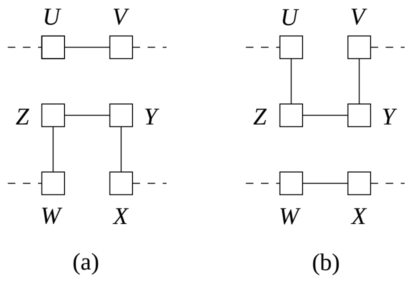

Figure 6: Local Optimization Scheme 2

3.1.6 Replacement

Let o be the most fit of the two offspring, and p be the least fit of the two parents. If o is better than p, then we replace p with o in the population. Otherwise, let w be the least fit member of the population. If o is better than w, then we replace w with o in the population. If no replacement can be made, we discard o.

3.1.7 Stopping Criteria

The algorithm runs for a maximum of 100,000 iterations or until there is no improvement in the last 10,000 iterations. The best member of the population is then returned.

3.2 Secondary Structure Genetic Algorithm (SSGA)

The secondary structure library is generated by SSGA, which is similar to the main genetic algorithm, PFGA, described in the previous section. The input to SSGA is the longest hydrophobic subsequence in the input sequence of amino acid residues, i.e., a sequence of H's only. Here the fitness of a conformation is no longer just the number of H-H contacts as in the case of the PFGA. SSGA also differs from PFGA in that individuals in its population obviously cannot have any secondary structures. In addition, we also put some constraints on the conformations evolved by SSGA. The rest of SSGA is identical to that of PFGA. We first de-

Table 1: Benchmark HP sequences

Seq. No.	Sequence	Length	C_{max}
1	$(HP)_2PH(HP)_2(PH)_2HP(PH)_2$	20	9
2	$H_2P_2(HP_2)_6H_2$	24	9
3	$P_2HP_2(H_2P_4)_3H_2$	25	8
4	$P(P_2H_2)_2P_5H_5(H_2P_2)_2P_2H(HP_2)_2$	36	14
5	$P_2H(P_2H_2)_2P_5H_{10}P_6(H_2P_2)_2HP_2H_5$	48	23
6	$H_2(PH)_3PH_4PH(P_3H)_2P_4(HP_3)_2HPH_4(PH)_3PH_2$	50	21
7	$P(PH_3)_2H_5P_3H_{10}PHP_3H_{12}P_4H_6PH_2PHP$	60	36
8	$H_{12}(PH)_2((P_2H_2)_2P_2H)_3(PH)_2H_{11}$	64	42
9	$H_4P_4H_{12}P_6(H_{12}P_3)_3HP_2(H_2P_2)_2HPH$	85	53
10	$P(P_2H_2)_2H_2P_2H_3(PH_2)_3H_2P_8(H_6P_2)_2P_7H(PH_2)_2H_9P_2H(H_2P)_2HP(PH)_2H_2P_6H_3$	100	50
11	$P_5(PH)_2HP_5H_3PH_5PH_2P_2(P_2H_2)_2(PH_5)_2H_5(PH_2)_2H_5P_{11}H_7P(PH)_2H_2P_5(PH)_2H$	100	48

scribe the constraints and then the rules for evaluating the fitness.

Constraint 1. Since the conformations for the secondary structures in the native state of a protein have certain degree of symmetry, we require that the secondary structures of the hydrophobic subsequences to be symmetric to either one of the two lattice axes.

Constraint 2. Since each secondary structure will be extended from both of its end lattice sites, the end lattice sites should have at least one unoccupied lattice neighbor.

In creating the initial population as well as when an offspring is created, an individual is eliminated if it does not satisfy the above two constraints.

Let C be an individual. The fitness, $f(C)$, of C is defined as

$$f(C) = f_1(C) + f_2(C),$$

where f_1 and f_2 are defined by Rules 1 and 2, respectively.

Rule 1. Let B be the longest hydrophobic subsequence in the input sequence A. Since B is more likely to form the core in the conformation of A, we give a higher fitness to the conformation of B that has the highest number of *total* H-H contacts. The total H-H contacts is the total number of H-H contacts the conformation will have if it is surrounded by hydrophobic residues. That is, the fitness is not based on just the number of H-H contacts within the conformation of B itself. Figure 7 shows two different conformations for a hydrophobic subsequence of twelve residues. The one in Figure 7(a) has 6 H-H contacts, whereas the one in Figure 7(b) has only 5 H-H contacts. However, the conformation in Figure 7(b) will have 21 total H-H contacts when surrounded by hydrophobic residues, whereas the one in Figure 7(a) will have only 20 total H-H contacts. So the conformation in Figure 7(b) is given a higher fitness than the one in Figure 7(a). The fitness $f_1(C)$, assigned for C, is just the number of total H-H contacts.

Rule 2. In order to achieve the maximum number of H-H contacts, the hydrophobic core is normally compact, leaving no unoccupied lattice sites inside the subsequence conformation. Thus, we assign higher fitness for the conformation that is more compact. Let s be the number of lattice squares that the conformation occupies. The fitness $f_2(C)$, assigned for C, is just $-s$.

The total fitness $f(C)$ of individual C is then the sum of $f_1(C)$ and $f_2(C)$. The rest of SSGA is the same as that of PFGA.

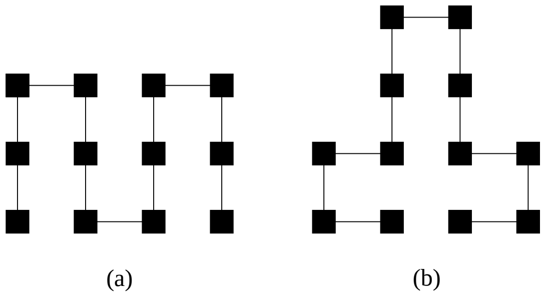

Figure 7: Two conformations for the hydrophobic subsequence HHHHHHHHHHHH.

4. EXPERIMENTAL RESULTS

In this section, we describe the results from our algorithm, PFGA, on various benchmark sequences, and compare them against the results from the following algorithms: an Ant Colony Optimization (ACO) algorithm [21], an Evolutionary Monte Carlo (EMC) algorithm [13], a Genetic Algorithm combined with Tabu Search (GTS) [8], a Metropolis Monte Carlo (MMC) algorithm [23], and a Genetic Algorithm (GA) [23]. Our algorithm was implemented in C++, and run on a PC with a Pentium IV 3.2GHz processor and 1GB of RAM under Windows XP Pro.

The details of the benchmark sequences used in our test are shown in Table 1. H_i, P_i, and $(\ldots)_i$ represent repetitions of the respective residue(s) i times. C_{max} represents the best-known maximum number of H-H contacts found for the corresponding sequence. Sequences 1 through 8 were introduced in [23]. Sequence 9 was introduced in [10] and sequences 10 and 11 were introduced in [18]. These sequences have been used as benchmark by various algorithms for the 2D HP model.

The C_{max} values, listed in Table 1, were obtained by a PERM (Pruned Enriched Rosenbluth Method) algorithm [7]. PERM is a biased chain growth algorithm based on the Rosenbluth-Rosenbluth (RR) method [20]. Several variations of PERM have been studied, and the one in [7] seems to perform very well compared to other PERM variations. Since PERM uses exhaustive enumeration at several stages, it is not a feasible algorithm to use in practice for longer sequences. Thus, even though the C_{max} values were obtained by PERM, we did not include PERM in the comparison.

Table 2: Comparison of maximum number of H-H contacts obtained by different algorithms on the benchmark sequences.

Seq. No.	C_{max}	PFGA Best	ACO Best	EMC Best	GTS Best	MMC Best	GA Best
1	**9**	**9**	**9**	**9**	**9**	**9**	**9**
2	**9**	**9**	**9**	**9**	**9**	**9**	**9**
3	**8**	**8**	**8**	**8**	**8**	**8**	**8**
4	**14**	**14**	**14**	**14**	**14**	13	12
5	**23**	**23**	**23**	**23**	**23**	20	22
6	**21**	**21**	**21**	**21**	**21**	**21**	**21**
7	**36**	**36**	**36**	35	35	33	34
8	**42**	**42**	**42**	39	39	35	37
9	**53**	**53**	51	-	-	-	-
10	**50**	49	47	-	-	-	-
11	**48**	**48**	47	-	-	-	-

Table 2 gives the maximum number of H-H contacts obtained by various algorithms on the benchmark sequences. For all the sequences, the maximum number of H-H contacts obtained by the PFGA were out of 500 independent runs. Similarly the maximum obtained by EMC, MMC, and GA were out of 5 independent runs. For GTS, no such information is available. The maximum obtained by ACO for sequences 1 through 6 were out of 500 runs, the maximum for sequences 7 and 8 were out of 300 runs, and the maximum for sequences 9 through 11 were out of 100 runs. Experimental results for the last three sequences were not available for EMC, GTS, MMC, and GA. It is clear from the table that PFGA performs very well against existing algorithms.

Table 3: Performance of PFGA on the benchmark sequences.

Seq. No.	C_{max}	Number of H-H contacts returned by PFGA: Best	Average	Standard deviation	Average running time t_{avg} (seconds)
1	**9**	**9**	8.39	0.63	3.23
2	**9**	**9**	7.93	0.70	4.48
3	**8**	**8**	6.51	0.93	4.77
4	**14**	**14**	11.44	1.03	7.98
5	**23**	**23**	18.51	1.49	12.85
6	**21**	**21**	17.51	1.33	12.78
7	**36**	**36**	31.62	1.28	19.88
8	**42**	**42**	37.24	1.91	23.82
9	**53**	**53**	47.40	1.89	44.05
10	**50**	49	43.38	1.96	56.71
11	**48**	**48**	42.10	1.92	58.93

Table 3 gives the performance of our algorithm on the benchmark sequences. For each sequence, the table gives the maximum number of H-H contacts obtained, average value out of 500 runs, standard deviation, and average running time, t_{avg}. The running times reported in Table 3 do not include the time taken by SSGA. The reasons for this are (i) the time taken by SSGA is insignificant compared to that of PFGA and (ii) the results produced by SSGA can be reused again for different input sequences. We were not able to compare the running times of our algorithm with others due to the following reasons: (i) a variety of methods for reporting running times were used in previous papers and (ii) insufficient information to allow for the adjustment of running times from different system configurations used to test previous algorithms. Finally, we give the conformations obtained by our algorithm for the benchmark sequences in Table 4.

5. CONCLUSION

In this paper, we gave an efficient genetic algorithm for the protein folding problem in the 2D HP model. The algorithm combines the concept of secondary structures with a genetic algorithm. Experimental results on the benchmark sequences show that it outperforms existing evolutionary and Monte Carlo algorithms. We would like to improve our algorithm further to be able to obtain the best-known values for all the benchmark sequences. Also we would like to extend this concept of systematic evolution of secondary structures to the protein folding problem in the 3D HP model.

6. ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable comments.

7. REFERENCES

[1] Bastolla, U., H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler, "Testing a New Monte Carlo Algorithm for Protein Folding," *Proteins*, 32(1), July 1998, pp. 52–66.

[2] Berger, B. and T. Leighton, "Protein Folding in the Hydrophobic-Hydrophilic (HP) Model is NP-Complete," *Proceedings of the Second Annual International Conference on Computational Molecular Biology*, New York, NY, USA, March 1998, ACM Press, pp. 30–39.

[3] Cohen, F. E. and J. W. Kelly, "Therapeutic Approaches to Protein-misfolding Diseases," *Nature*, 426, December 2003, pp. 905–909.

[4] Crescenzi, P., D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis, "On the Complexity of Protein Folding (Extended Abstract)," *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, Dallas, TX, USA, May 1998, ACM Press, pp. 597–603.

[5] Dill, K. A., "Theory for the Folding and Stability of Globular Proteins," *Biochemistry*, 24(6), March 1985, pp. 1501–1509.

[6] Hart, W. E. and S. Istrail, "Fast Protein Folding in the Hydrophobic-hydrophilic Model Within Three-eights of Optimal (Extended Abstract)," *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, Las Vegas, NV, USA, June 1995, ACM Press, pp. 157–168.

[7] Hsu, H., V. Mehra, W. Nadler, and P. Grassberger, "Growth Algorithms for Lattice Heteropolymers at Low Temperatures," *Journal of Chemical Physics*, 118(1), January 2003, pp. 444–451.

[8] Jiang, T., Q. Cui, G. Shi, and S. Ma, "Protein Folding Simulations of the Hydrophobic-Hydrophilic Model by Combining Tabu Search with Genetic Algorithms," *Journal of Chemical Physics*, 119(8), August 2003, pp. 4592–4595.

Table 4: Conformations obtained by PFGA for the benchmark sequences

Seq. No.	Conformation
1	*dlluruluurdrdrurddl*
2	*rdruruluuuruluIdldrdddl*
3	*lurulurulldlulddrdddruuu*
4	*ruluruluuuldldrdldrddluldluurulurul*
5	*ururdrurdrrrdldluldlldrrrrrddlluldluldluluruurr*
6	*rdlluluurdrruluuldllddlldrrdrddldrrdruulurrdrddlu*
7	*rrrrrddlullldlddrururddldldrdrruluruurddrdldrruuuluuurddrdd*
8	*uuuurdddddlululurulurulururdrurdrurdrdldrdldrdldldluuuuuurdddd*
9	*rulllurrrrurrdldldrddrrruullurruluurdruulluldd lldllluurdrurrrullllurrurdrurdrurdrd*
10	*rdddrdlluuuldddldrdlluullddd lluruuurruuurdddruuuurulllldldrrddluldldrdldluuulullulurrdrruuurddddl*
11	*rrurruruurdruullldrdldlulddluuuluruurdddruurruuldlulurrurdrdddrrrdrrrullulllluuuluurdrdddrrrrulluld*

[9] Kolinski, A. and J. Skolnick, *Lattice Models of Protein Folding, Dynamics and Thermodynamics*, Chapman & Hall, 1996.

[10] König, R. and T. Dandekar, "Improving Genetic Algorithms for Protein Folding Simulations by Systematic Crossover," *BioSystems*, 50(1), April 1999, pp. 17–25.

[11] Krasnogor, N., W. E. Hart, J. Smith, and D. A. Pelta, "Protein Structure Prediction with Evolutionary Algorithms," *Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, FL, USA, July 1999, Morgan Kaufmann, pp. 1596–1601.

[12] Lau, K. F. and K. A. Dill, "A Lattice Statistical Mechanics Model of the Conformational and Sequence Spaces of Proteins," *Macromolecules*, 22(10), October 1989, pp. 3986–3997.

[13] Liang, F. and W. H. Wong, "Evolutionary Monte Carlo for Protein Folding Simulations," *Journal of Chemical Physics*, 115(7), August 2001, pp. 3374–3380.

[14] Newman, A., "A New Algorithm for Protein Folding in the HP Model," *Proceedings of the thirteenth annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, USA, January 2002, SIAM, pp. 876–884.

[15] Newman, A. and M. Ruhl, "Combinatorial Problems on Strings with Applications to Protein Folding," *Proceedings of the Sixth Latin American Symposium on Theoretical Informatics*, Buenos Aires, Argentina, April 2004, Springer Verlag, pp. 369–378.

[16] Patton, A. L., W. F. Punch III, and E. D. Goodman, "A Standard GA Approach to Native Protein Conformation Prediction," *Proceedings of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, USA, July 1995, Morgan Kaufmann, pp. 574–581.

[17] Pedersen, J. and J. Moukt, "Protein Folding Simulations with Genetic Algorithms and a Detailed Molecular Description," *Journal of Molecular Biology*, 269(2), June 1997, pp. 240–259.

[18] Ramakrishnan, R., B. Ramachandran, and J. F. Penky, "A Dynamic Monte Carlo Algorithm for Exploration of Dense Conformational Spaces in Heteropolymers," *Journal of Chemical Physics*, 106(6), February 1997, pp. 2418–2424.

[19] Richards, F. M., "Areas, Volumes, Packing, and Protein Structure," *Annual Review of Biophysics and Bioengineering*, 6, 1977, pp. 151–176.

[20] Rosenbluth, M. N. and A. W. Rosenbluth, "Monte Carlo Calculation of the Average Extension of Molecular Chains," *Journal of Chemical Physics*, 23(2), February 1955, pp. 356–359.

[21] Shmygelska, A. and H. H. Hoos, "An Improved Ant Colony Optimisation Algorithm for the 2D HP Protein Folding Problem," *Proceedings of the Sixteenth Conference of the Canadian Society for Computational Studies of Intelligence*, Halifax, Canada, June 2003, Springer Verlag, pp. 400–417.

[22] Unger, R. and J. Moult, "A Genetic Algorithm for 3D Protein Folding Simulations," *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana, IL, USA, July 1993, Morgan Kaufmann, pp. 581–588.

[23] Unger, R. and J. Moult, "Genetic Algorithms for Protein Folding Simulations," *Journal of Molecular Biology*, 231(1), May 1993, pp. 75–81.