Fast Parallel Bio-Molecular Logic Computing Algorithms: Protein Folding

Yu-Ying Shih and Michael (Shan-Hui) Ho

***Abstract*—Any protein is a linear chain of amino acid, and under specific conditions it folds into a unique native three-dimensional structure. Based upon thermodynamical hypothesis, the native structure of a protein always achieves the global minimum of free energy. The hydrophobic-hydrophilic model [1] for predicting the protein native structure is perhaps one of the most successful and best-studied models. The protein folding problem [2] in the hydrophobic-hydrophilic (HP) model has been proved as an NP-complete problem. This paper is the first one in which we demonstrate that bio-molecular operations with basic logical operations can be used to solve the three dimensional protein folding problem in the HP model. In order to achieve the goal, intelligent bioinformatics logic algorithms are proposed.**

I. Introduction

Proteins are very complex molecules that consist of a chain of amino acids. The importance of protein folding has been recognized for many years. The misfolded protein can result in diseases such as Alziemers and Cystic Fibrosis. Proper protein folding is key to producing recombinant proteins for structure determination. By understanding how proteins achieve their native 2D and 3D structure, we can also help to develop treatments for diseases.

There is still much work to be done in simulating proteins with many chains. Computations of this kind still remain infeasible with current processing technologies. The Hydrophobic-Hydrophilic (HP) lattice based protein model simplifies the problem while maintaining behavioral relevance. Until recently the 3D HP model has been largely ignored in favor of the 2D HP model due to the simplicity of the search space and easier visualization. Any simplified HP model has been proven to be NP-complete [1] thus are perfect for evaluating and improving heuristic based algorithms that will assist future development of expanded protein folding problems. Much of the previous work in this field focuses on developing and implementing solutions to 2D protein folding without considering 3D implementations. We propose a newly developed and very intelligent parallel DNA algorithm for 3D protein structure prediction using the HP lattice model on **optimal bioinformatics logic computing**

Manuscript received July 5, 2008.

Yu-Ying Shih is with Department of Graduate Institute of Industrial and Business Management, National Taipei University of Technology, Taipei, Taiwan. (phone: 886-3-3507001 ext. 3408; fax: 886-3-3593875; e-mail: amy_shyy@yahoo.com.tw).

Michael (Shan-Hui) Ho is with the Department of Information Management, Ming Chuan University, Taoyuan, Taiwan. (phone: 886-3-3507001 ext. 3408; fax: 886-3-3593875; e-mail: mhoincerritos@yahoo.com).

II. DNA Manipulations

A. Biological Operations of Bio-molecular computing

A (test) tube is a set of molecules of DNA (a multi-set of finite strings over the alphabet {*A*, *C*, *G*, *T*}). Given a tube, one can perform the following operations:

1. *Extract*: Given a tube T and a short single strand of DNA, "s", produce two tubes + (T, s) and – (T, s), where + (T, s) is all of the molecules of DNA in T which contain the strand "s" as a sub-strand and – (T, s) is all of the molecules of DNA in T which do not contain the short strand "s".
2. *Merge*: Given tubes T_1 and T_2, yield $\cup$ (T_1, T_2), where $\cup$ (T_1, T_2) = $T_1 \cup T_2$. This operation is to pour two tubes into one, with no change of the individual strands.
3. *Amplify*: Given a tube *T*, the operation, Amplify (*T*, T_1, T_2), will produce two new tubes T_1 and T_2 so that T_1 and T_2 totally a copy of *T* (T_1 and T_2 are identical) and *T* becomes an empty tube.
4. *Append*: Given a tube *T* and a short strand of DNA, *"s"*, the operation will append the short strand, *"s"*, onto the end of every strand in the tube *T*. It is denoted by append (*T*, *s*).
5. *Append-head*: Given a tube *T* and a short strand of DNA, *"s"*, the operation will append the short strand, *"s"*, onto the head of every strand in the tube *T*. It is denoted by append-head (*T*, s).
6. *Detect*: Given a tube *T*, say 'yes' if *T* includes at least one DNA molecule, and say 'no' if it contains none. It is denoted by detect (*T*).
7. *Discard*: Given a tube *T*, the operation will discard the tube *T*. It is denoted by discard (*T*).
8. *Read*: Given a tube *T*, the operation is used to describe a single molecule, which is contained in the tube *T*. Even if *T* contains many different molecules each encoding a different set of bases, the operation can give an explicit description of exactly one of them. It is denoted by read (*T*).

B. Optimal Bioinformatics Logic Computing

In this article, we propose a newly developed set of bio-molecular parallel adder, parallel subtractor, parallel multiplier, and parallel divider with basic logic operations and compose them as a part of **Optimal bioinformatics logic computing.** The parallel adder is shown in [11], and parallel subtractor, parallel multiplier, and parallel divider are in [12], [13] for solving NP problems.

These basic bio-molecular logic operations (bio-molecular logic gates) used in this project are only **OR**, **AND**, **XOR.** We use these three bio-molecular logic gates to construct bio-molecular parallel adder, parallel subtractor, parallel

multiplier, parallel divider, parallel assignment, and parallel comparator. All operations of **Optimal Bioinformatics Logic Computing** are shown in Figure 1.

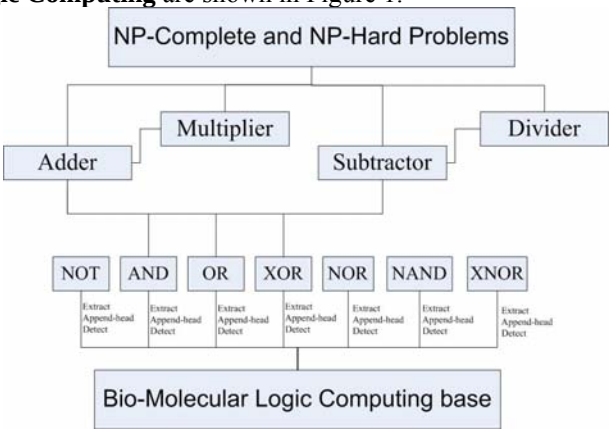

Fig.1. Bio-molecular logic computing model

III. THREE DIMENSIONAL PROTEIN STRUCTURE USING THE HP LATTICE MODEL

A. *The HP model*

The hydrophobic-hydrophilic model was introduced by Dill [1] and has been intensively studied in [8, 9]. It models a protein as a linear chain of amino acid residues. Each amino acid can be either of two types: H (hydrophobic, i.e., nonpolar) or P (hydrophilic, i.e., polar). For simplicity, we denote H by "l" (black) and P by "0" (white).

Conformations of proteins are embedded in either a two dimensional or three-dimensional square lattice. In the HP model a protein, i.e. an amino acid sequence, is abstracted as a string describing the hydrophobicity of each amino acid in the sequence. Throughout this proposal we will use S to denote the abstraction of an amino acid sequence of length n, where S[i], for i = 1; 2; : : : ; n, is 1 if the ith amino acid in the sequence is hydrophobic and 0 if it is hydrophilic. We will use the term "hydrophobic amino acid" to refer to a 1 at some position in S, and say that the parity of the 1 is even if its position in S is even, and odd if its position in S is odd.

For example, a folding of a protein in the HP model is an embedding of its abstraction S in the 3D square lattice such that adjacent characters in S occupy adjacent grid points in the lattice, and no grid point in the lattice is occupied by more than one character. We say that two 1's in S form a non-local 1-1 bond if they occupy adjacent grid points in the lattice but are not adjacent in S.

The conformation of any protein can be topologically embedded into a lattice structure [1, 2, 4-10]. The potential energy of each topologically embedded conformation is measured by the size of its hydrophobic core. The free energy of a folding of S is the number of non-local 1-1 bonds in the folding multiplied by some constant $\varepsilon < 0$. The free energy function models the belief that the driving force of protein structure formation is interactions between hydrophobic amino acids.

In the two-dimensional lattice, every lattice site has four neighbors, and the number of bond orientations within the chain for every internal residue is z -1 = 3. In the three dimensional lattice, every lattice site has six neighbors, and the number of bond orientations within the chain for every internal residue is z -1 = 5. The value z is called the lattice coordination number. The conformation space is the set of all possible internal conformations of a molecule, due to all the different bond orientations. The sequence space is the set of all possible sequences of H and P residues. It is convenient to distinguish between pairs of amino acids that are connected neighbors in the chain, i.e., occurring in positions j' and j + 1 along the chain, and topological neighbors that are adjacent in space (in cent act) but not adjacent in the chain.

The potential energy of each topologically embedded conformation is measured by the size of its hydrophobic core. For this, the twenty naturally occurring amino acids are classified of being either hydrophobic (**H**) or hydrophilic (**P**) [1, 4-6]. From [1, 2, 4-10], two hydrophobic residues of a conformation have *loose contact* if they are adjacent in the lattice but not connected by its amino acid sequence. From [1, 4-6], the *potential energy* of a conformation Φ corresponds to the number of all pairs of hydrophobic residues which have loose contact

$$V(\Phi) = -|\{(S[i], S[j]) \mid S[i] \text{ and } S[j] \text{ are hydrophobic and they have loose contact in } \Phi\}|. \quad (1)$$

From $V(\Phi$, the HP model is based on the assumption that the global minimum energy conformation corresponds to the conformation with the largest hydrophobic core.) The model assumes that the free energy is computed between the topological neighbors in a conformation. Every H-H contact between topological neighbors has a contact free energy of $V(\Phi < 0$. Any other pairwise combination of H and P has free energy equal to 0. This is a very simple model of the free energy of protein conformations.) Hence, following the thermodynamical hypothesis, the native structure of a given sequence is the conformation that achieves minimum free energy.

B. *Mutation on the Absolute or Relative Encoding in the HP Model*

In the HP model, the structures can be represented by Cartesian coordinates, internal coordinates or distance geometry. We concentrate here on internal coordinates, which can be defined as absolute or relative. Under the absolute encoding, the structures are represented by a list of absolute moves. In a 3D square lattice, for example, a structure s is encoded as a string s = {Up, Down, Left, Right, Forward, Backward}. When using a relative coordinates, each move is interpreted in terms of the previous one, like in LOGO turtle graphics; a structure s is encoded as a string s = {Forward, TurnLeft, TurnRight }. Designing with black the hydrophobic residues and white the polar ones, the structure of Figure 2 is coded either as s = RRDFRRRLUL (absolute encoding) or s = RRLFRRRLLL (relative encoding), with 2 non-local H-H contacts.

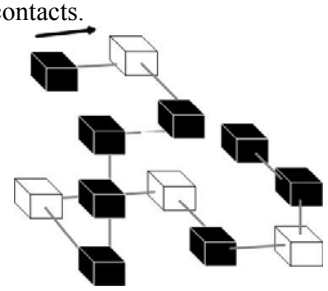

Fig.2. Structure-path on a 3D lattice

Any absolute direction on a square or cubic lattice, individuals are coded with a sequence in $\{U, D, L, R, F, B\}^{n-1}$ which correspond to up, down, left, right, forward and backward moves in a cubic for a length n protein. To avoid a reduction in diversity, the standard form of relative offsets represents a conformation as a sequence in $\{F, L, R, U, D\}^{n-2}$ which correspond to forward, left, right, up and down relative moves. This has the advantage of guaranteeing that all solutions are l-step self–avoiding (since there is no "back" move).

IV. Three Dimensional Protein Structure Prediction using the HP Lattice Model BASED UPon Optimal Bioinformatics Logic Computing

The following method is used for finding 3D protein structure using the HP lattice model based upon optimal bioinformatics logic computing.

Step 1. Given a protein with the amino acid sequence S of length n with $S = s_1 \ldots s_n$, create procedure **Delegate** to generate a DNA strand for representing that protein.

Step 2. Create procedure **Building3DHPLattice** to construct DNA strands for representing the origin (0, 0, 0) of the coordinates in the 3D HP lattice model. In this step, mutation is accomplished by absolute encoding.

Step 3. Given a protein with the amino acid sequence S of length n with $S = s_1 \ldots s_n$, create procedure **CreateAllOfPossibleConfirmationSpaces** to generate DNA strands for representing all of possible confirmation spaces for that protein in the 3D HP lattice model.

Step 4. Create procedure **ComputeTheNumberOfHH Contacts** for all of possible confirmation spaces fo r a given protein with the amino acid sequence S of length n with $S = s_1 \ldots s_n$ in the 3D HP lattice mod el. In order to compute the number of H-H contacts , there is a need to call a subprocedure, **FindHHCo ntact**, to find any valid H-H contact.

Step 5. To compute the global minimum energy from conformation space is equivalent to find the maximum number of H-H contacts. Because the more number of H-H contacts a protein has , the lower free energy that protein has. Procedure **FindTheMaxNumberOfHHContacts** is created in the 3D HP lattice model for finding the maximum number of H-H contacts.

A. Construction of a Delegate for a Protein in the 3D HP Lattice Model

Assume that the length for three integers, *a, b* and *c*, is *n* bits. Also suppose that three integers, *a, b* and *c*, are applied to, respectively, represent the *X* coordinate, *Y* coordinate and the *Z* coordinate of a lattice site in a 3D HP lattice. The conformation of a protein with the amino acid sequence $\mathbf{a} = a_1 \ldots a_n$, is represented by *n* lattice sites in a 3D HP lattice, and each lattice site is represented by a 3D HP coordinate, (*a, b, c*).

For every bit s_k for $1 \le k \le n$, two *distinct* 15 base value sequences were designed. One represents the value "0" for s_k and the other represents the value "1" for s_k. Figure 3 is shown to construct a DNA strand for representing a protein with the amino acid sequence $\mathbf{a} = a_1 \ldots a_L$ in the hydrophobic-hydrophilic model.

Procedure Delegate(T_0)
(1) **For** $k = 1$ **to** n
(1a) Append(T_0, s_k).
EndFor
EndProcedure

Fig.3. Procedure Delegate(T_0)

B. Building 3D HP Lattice Model

The origin of the coordinates in a three-dimensional lattice, (0, 0, 0), is represented as $(d_{1, n, n}{}^0\, d_{1, n-1, n-1}{}^0 \ldots d_{1, 1, 1}{}^0, b_{1, n, n}{}^0\, b_{1, n-1, n-1}{}^0 \ldots b_{1, 1, 1}{}^0, c_{1, n, n}{}^0\, c_{1, n-1, n-1}{}^0 \ldots c_{1, 1, 1}{}^0)$. Assume that two binary variables $w_{k,1}$, $w_{k,2}$ and $w_{k,3}$ for $1 \le k \le L$ are used to represent direction of moving from the $(k-1)^{th}$ lattice site to the k^{th} lattice site in a three-dimensional lattice. For convenience, assume that $w_{k,1}{}^1$ denotes the fact that the value of $w_{k,1}$ is 1 and $w_{k,1}{}^0$ denotes the fact that the value of $w_{k,1}$ is 0, suppose $w_{k,2}{}^1$ denotes the fact that the value of $w_{k,2}$ is 1 and $w_{k,2}{}^0$ denotes the fact that the value of $w_{k,2}$ is 0, and suppose $w_{k,3}{}^1$ denotes the fact that the value of $w_{k,3}$ is 1 and $w_{k,3}{}^0$ denotes the fact that the value of $w_{k,3}$ is 0.

Figure 4, **Building3DHPLattice**(T_1), is applied to construct the origin of the coordinates in a three-dimensional lattice and the *first* moving from the origin of the coordinates to the *second* lattice site, and then third lattice site.

Procedure Building3DHPLattice(T_1).
(0) **For** $q = 1$ **to** n
(0a) Append-head(T_1, $b_{1,q}{}^0$).
EndFor
(1) **For** $q = 1$ **to** n
(1a) Append-head(T_1, $d_{1,q}{}^0$).
EndFor
(2) **For** $q = 1$ **to** n
(1a) Append-head(T_1, $c_{1,q}{}^0$).
EndFor
(3) **For** $k = 2$ **to** 3
(3a) Amplify(T_1, T_3, T_4).
(3b) **ParallelAdder**(T_3, k, b).
(3c) **ParallelAssign**(T_3, k, d).
(3d) Append-head(T_3, $w_{k,1}{}^0$) and Append-head(T_3, $w_{k,2}{}^0$) and Append-head(T_3, $w_{k,3}{}^0$).
(3e) Amplify(T_4, T_5, T_6).
(3f) **ParallelAdder**(T_5, k, d).
(3g) **ParallelAssign**(T_5, k, b).
(3h) Append-head(T_5, $w_{k,1}{}^0$) and Append-head(T_5, $w_{k,2}{}^1$) and Append-head(T_5, $w_{k,3}{}^0$).
(3i) Amplify(T_6, T_7, T_8).
(3j) **ParallelSubtractor**(T_7, k, b).
(3k) **ParallelAssign**(T_7, k, d).
(3l) Append-head(T_7, $w_{k,1}{}^1$) and Append-head(T_7, $w_{k,2}{}^0$) and Append-head(T_7, $w_{k,3}{}^0$).
(3m) **ParallelSubtractor**(T_8, k, d).
(3n) **ParallelAssign**(T_8, k, b).
(3o) Append-head(T_8, $w_{k,1}{}^1$) and Append-head(T_8, $w_{k,2}{}^1$) and Append-head(T_8, $w_{k,3}{}^0$).
(3p) $T_1 = \cup(T_3, T_5, T_7, T_8)$.
EndFor
(4) Append-head(T_1, s_2).
EndProcedure

Fig.4. Procedure Building3DHPLattice(T_1)

C. Construction of All of Possible Conformation Space for a Protein in the 3D HP Lattice Model

Figure 5 shows that the construction of the conformation space of a protein with an amino acid sequence $\mathbf{a} = a_1 \ldots a_n$ in the 3D HP lattice model.

Procedure CreateAllOfPossibleConfirmationSpaces(T_1).
(1) **For** $k = 3$ **to** n
(1a) Amplify(T_1, T_3, T_4).
(1a0) $T_3^{bad} = +(T_3, w_{(k-1),\,3}^{\,1})$ and $T_3^{off} = -(T_3, w_{(k-1),\,3}^{\,1})$.
(1a1) $T_3^{bad} = +(T_3, w_{(k-1),\,2}^{\,1})$ and $T_3^{off} = -(T_3, w_{(k-1),\,2}^{\,1})$.
(1a2) $T_3^{bad} = +(T_3^{bad}, w_{(k-1),\,1}^{\,0})$ and $T_3^{on} = -(T_3^{bad}, w_{(k-1),\,1}^{\,0})$.
(1a3) Discard(T_3^{bad}) and $T_3 = \cup(T_3^{off}, T_3^{on})$.
(1b) **ParallelAdder**(T_3, k, b).
(1c) **ParallelAssignment**(T_3, k, d).
(1d) Amplify(T_4, T_5, T_6).
(1d0) $T_5^{bad} = +(T_5, w_{(k-1),\,3}^{\,1})$ and $T_5^{off} = -(T_5, w_{(k-1),\,3}^{\,1})$.
(1d1) $T_5^{bad} = +(T_5, w_{(k-1),\,2}^{\,1})$ and $T_5^{off} = -(T_5, w_{(k-1),\,2}^{\,1})$.
(1d2) $T_5^{bad} = +(T_5^{bad}, w_{(k-1),\,1}^{\,1})$ and $T_5^{on} = -(T_5^{bad}, w_{(k-1),\,1}^{\,1})$.
(1d3) Discard(T_5^{bad}) and $T_5 = \cup(T_5^{off}, T_5^{on})$.
(1e) **ParallelAdder**(T_5, k, d).
(1f) **ParallelAssignment**(T_5, k, b).
(1a) Amplify(T_1, T_3, T_4).
(1a0) $T_3^{bad} = +(T_3, w_{(k-1),\,3}^{\,1})$ and $T_3^{off} = -(T_3, w_{(k-1),\,3}^{\,1})$.
(1a1) $T_3^{bad} = +(T_3, w_{(k-1),\,2}^{\,1})$ and $T_3^{off} = -(T_3, w_{(k-1),\,2}^{\,1})$.
(1a2) $T_3^{bad} = +(T_3^{bad}, w_{(k-1),\,1}^{\,0})$ and $T_3^{on} = -(T_3^{bad}, w_{(k-1),\,1}^{\,0})$.
(1a3) Discard(T_3^{bad}) and $T_3 = \cup(T_3^{off}, T_3^{on})$.
(1b) **ParallelAdder**(T_3, k, b).
(1c) **ParallelAssignment**(T_3, k, c).
(1d) Amplify(T_4, T_5, T_6).
(1d0) $T_5^{bad} = +(T_5, w_{(k-1),\,3}^{\,1})$ and $T_5^{off} = -(T_5, w_{(k-1),\,3}^{\,1})$.
(1d1) $T_5^{bad} = +(T_5, w_{(k-1),\,2}^{\,1})$ and $T_5^{off} = -(T_5, w_{(k-1),\,2}^{\,1})$.
(1d2) $T_5^{bad} = +(T_5^{bad}, w_{(k-1),\,1}^{\,1})$ and $T_5^{on} = -(T_5^{bad}, w_{(k-1),\,1}^{\,1})$.
(1d3) Discard(T_5^{bad}) and $T_5 = \cup(T_5^{off}, T_5^{on})$.
(1e) **ParallelAdder**(T_5, k, c).
(1f) **ParallelAssign**(T_5, k, b).
(1g) Amplify(T_6, T_7, T_8).
(1g0) $T_7^{bad} = +(T_7, w_{(k-1),\,3}^{\,0})$ and $T_7^{off} = -(T_7, w_{(k-1),\,3}^{\,0})$.
(1g1) $T_7^{bad} = +(T_7, w_{(k-1),\,2}^{\,0})$ and $T_7^{off} = -(T_7, w_{(k-1),\,2}^{\,0})$.
(1g2) $T_7^{bad} = +(T_7^{bad}, w_{(k-1),\,1}^{\,0})$ and $T_7^{on} = -(T_7^{bad}, w_{(k-1),\,1}^{\,0})$.
(1g3) Discard(T_7^{bad}) and $T_7 = (T_7^{off}, T_7^{on})$.
(1h) **ParallelSubtractor**(T_7, k, b).
(1i) **ParallelAssignment**(T_7, k, d).
(1i0) $T_8^{bad} = +(T_8, w_{(k-1),\,3}^{\,0})$ and $T_8^{off} = -(T_8, w_{(k-1),\,3}^{\,0})$.
(1i1) $T_8^{bad} = +(T_8, w_{(k-1),\,2}^{\,0})$ and $T_8^{off} = -(T_8, w_{(k-1),\,2}^{\,0})$.
(1i2) $T_8^{bad} = +(T_8^{bad}, w_{(k-1),\,1}^{\,1})$ and $T_8^{on} = -(T_8^{bad}, w_{(k-1),\,1}^{\,1})$.
(1i3) Discard(T_8^{bad}) and $T_8 = (T_8^{off}, T_8^{on})$.
(1j) **ParallelSubtractor**(T_8, k, d).
(1k) **ParallelAssignment**(T_8, k, b).
(1k1) Append-head(T_3, $w_{k,\,3}^{\,0}$), Append-head(T_3, $w_{k,\,2}^{\,0}$) and Append-head(T_3, $w_{k,\,1}^{\,0}$).
(1k2) Append-head(T_5, $w_{k,\,3}^{\,0}$), Append-head(T_5, $w_{k,\,2}^{\,0}$) and Append-head(T_5, $w_{k,\,1}^{\,1}$).
(1k3) Append-head(T_7, $w_{k,\,3}^{\,0}$), Append-head(T_7, $w_{k,\,2}^{\,1}$) and Append-head(T_7, $w_{k,\,1}^{\,0}$).
(1k4) Append-head(T_8, $w_{k,\,3}^{\,0}$), Append-head(T_8, $w_{k,\,2}^{\,1}$) and Append-head(T_8, $w_{k,\,1}^{\,1}$).
(1l) $T_1 = \cup(T_3, T_5, T_7, T_8)$.
(1m) Append-head(T_1, s_k).
EndFor
EndProcedure

Fig.5. Procedure CreateAllOfPossibleConfirmationSpaces(T_1)

D. Construction of Parallel Assignment Operators

For any a lattice site (d, b) in a three-dimensional lattice, its neighbors are, respectively, (d, b + 1, c), (d + 1, b, c), (d, b, c+ 1), (d, b − 1, c), (d − 1, b, c) and (d, b, c-1). This implies that for moving of six different directions either the value of the X coordinate or the value of the Y coordinate or the value of the Z coordinate are preserved. Figure 6, will be used to perform parallel assignment operators for preserving those coordinate values for moving of four different directions. A tube T_9 is the first parameter in the algorithm, **ParallelAssignment**(T_9, k, Ω), and it is regarded as the input tube of the algorithm. The second parameter, k, is applied to represent the kth lattice site in L lattice sites that forms conformation of a protein with the amino acid sequence $\mathbf{a} = a_1 \ldots a_n$. The third parameter, Ω, is used to represent the value of the X coordinate or the value of the Y coordinate or the value of the Z coordinate that is preserved for moving of six different directions.

Procedure ParallelAssignment(T_9, k,$\mathbf{\Omega}$)
(1) **For** $q = 1$ **to** n
(1a) $T_{10} = +(T_9, \mathbf{\Omega}_{(k-1),\,q}^{\,1})$ and $T_{11} = -(T_9, \mathbf{\Omega}_{(k-1),\,q}^{\,1})$.
(1b) **If** (Detect(T_{10}) = = "*yes*") **then**
(1c) Append-head(T_{10}, $\mathbf{\Omega}_{k,\,q}^{\,1}$).
EndIf
(1d) **If** (Detect(T_{11}) = = "*yes*") **then**
(1e) Append-head(T_{11}, $\mathbf{\Omega}_{k,\,q}^{\,0}$).
EndIf
(1f) $T_9 = \cup(T_{10}, T_{11})$.
EndFor
EndProcedure

Fig.6. Procedure ParallelAssignment(T_9, k,$\mathbf{\Omega}$)

E. Compute the Number of H-H Contacts for All of Possible Confirmation Spaces in the 3D HP Lattice Model

Assume that a binary number, $\Phi_{k,\,p,\,j}$, for $1 \le k \le L$, $1 \le j \le L$ and $0 \le p \le 3$ is used to represent the coordinate value of the k^{th} lattice site in L lattice sites is adjacent to the coordinate value of the j^{th} lattice site in L lattice. Also suppose that $\Phi_{k,\,p,\,j}^{\,1}$ denotes the fact that the value of $\Phi_{k,\,p,\,j}$ is 1 and $\Phi_{k,\,p,\,j}^{\,0}$ denotes the fact that the value of $\Phi_{k,\,p,\,j}$ is 0. Also assume that $\Phi_{k,\,0,\,j}^{\,1}$, $\Phi_{k,\,1,\,j}^{\,1}$, $\Phi_{k,\,2,\,j}^{\,1}$ and $\Phi_{k,\,3,\,j}^{\,1}$ denote, subsequently, up neighbor, right neighbor, down neighbor and left neighbor for the k^{th} lattice site and the j^{th} lattice site.

Figure 7, **ComputeTheNumberOfHHContacts**(T_0, T_1), is employed to figure out the global minimum energy of any amino acid sequence $\mathbf{a} = a_1 \ldots a_n$ in the hydrophobic-hydrophilic model in a three-dimensional lattice. A tube T_0 is the first parameter in the algorithm, **ComputeTheNumberOfHHContacts**(T_0, T_1), and it is regarded as the first input tube of the algorithm. The first input tube T_0 contains the checked amino acid sequence $\mathbf{a} = a_1 \ldots a_n$. A tube T_1 is the second parameter in the algorithm, **ComputeTheNumberOfHHContacts**(T_0, T_1), and it is regarded as the second input tube of the algorithm. The second input tube T_1 consists of all of possible conformations formed by the checked amino acid sequence $\mathbf{a} = a_1 \ldots a_n$.

Procedure ComputeTheNumberOfHHContacts(T_0, T_1).
(1) **For** $k = 1$ **to** n
(1a) $T_0^{on} = +(T_0, s_k^{\,1})$ and $T_0^{off} = -(T_0, s_k^{\,1})$.
(1b) **If** (Detect(T_0^{on}) = = "*yes*") **Then**
(1c) $T_0 = \cup(T_0, T_0^{on})$.

(1d) **ComputeLooseContact**(T_0, T_1, k).
Else
(1e) $T_0 = \cup(T_0, T_0^{off})$.
(1f) **For** $p = 0$ **to** 3
(1g) **For** $j = 1$ **to** L
(1h) Append-head(T_1, $\Phi_{k,p,j}{}^0$).
EndFor
EndFor
EndIf
EndFor
EndProcedure

Fig.7. Procedure ComputeTheNumberOfHHContacts(T_0, T_1)

Figure 8, **ComputeLooseContact** (T_0, T_1, k), is applied to determine the number of loose contact for any amino acid sequence **a** = $a_1 \ldots a_n$ in the 3D HP lattice model.

Procedure ComputeLooseContact (T_0, T_1, k).
(0) $T_{33} = +(T_1, w_{(k+1),3}{}^0)$ and $T_{34} = -(T_1, w_{(k+1),3}{}^0)$.
(1) $T_{35} = +(T_1, w_{(k+1),2}{}^0)$ and $T_{36} = -(T_1, w_{(k+1),2}{}^0)$.
(2) $T_{37} = +(T_{35}, w_{(k+1),1}{}^0)$ and $T_{38} = -(T_{35}, w_{(k+1),1}{}^0)$.
(3-0) $T_{38} = \cup(T_{34}, T_{38})$.
(3) $T_{39} = \cup(T_{36}, T_{38})$.
(4) **FindLooseContact**(T_0, T_{37}, T_{39}, k, 0).
(5) $T_1 = \cup(T_{37}, T_{39})$.
(6-0) $T_{33} = +(T_1, w_{(k+1),3}{}^0)$ and $T_{34} = -(T_1, w_{(k+1),3}{}^0)$.
(6) $T_{35} = +(T_1, w_{(k+1),2}{}^0)$ and $T_{36} = -(T_1, w_{(k+1),2}{}^0)$.
(7) $T_{37} = +(T_{35}, w_{(k+1),1}{}^1)$ and $T_{38} = -(T_{35}, w_{(k+1),1}{}^1)$.
(8-0) $T_{38} = \cup(T_{34}, T_{38})$.
(8) $T_{39} = \cup(T_{36}, T_{38})$.
(9) **FindLooseContact**(T_0, T_{37}, T_{39}, k, 1).
(10) $T_1 = \cup(T_{37}, T_{39})$.
(11-0) $T_{33} = +(T_1, w_{(k+1),3}{}^0)$ and $T_{34} = -(T_1, w_{(k+1),3}{}^0)$.
(11) $T_{35} = +(T_1, w_{(k+1),2}{}^1)$ and $T_{36} = -(T_1, w_{(k+1),2}{}^1)$.
(12) $T_{37} = +(T_{35}, w_{(k+1),1}{}^0)$ and $T_{38} = -(T_{35}, w_{(k+1),1}{}^0)$.
(13-0) $T_{38} = \cup(T_{34}, T_{38})$.
(13) $T_{39} = \cup(T_{36}, T_{38})$.
(14) **FindLooseContact**(T_0, T_{37}, T_{39}, k, 2).
(15) $T_1 = \cup(T_{37}, T_{39})$.
(16-0) $T_{33} = +(T_1, w_{(k+1),3}{}^0)$ and $T_{34} = -(T_1, w_{(k+1),3}{}^0)$.
(16) $T_{35} = +(T_1, w_{(k+1),2}{}^1)$ and $T_{36} = -(T_1, w_{(k+1),2}{}^1)$.
(17) $T_{37} = +(T_{35}, w_{(k+1),1}{}^1)$ and $T_{38} = -(T_{35}, w_{(k+1),1}{}^1)$.
(18-0) $T_{38} = \cup(T_{34}, T_{38})$.
(18) $T_{39} = \cup(T_{36}, T_{38})$.
(19) **FindLooseContact**(T_0, T_{37}, T_{39}, k, 3).
(20) $T_1 = \cup(T_{37}, T_{39})$.
EndProcedure

Fig.8. Procedure ComputeLooseContact (T_0, T_1, k)

Figure 9, **FindLooseContact**(T_0, T_{37}, T_{39}, k, p), is used to judge whether a conformation in tube T_{37} and any conformation in tube T_{39} satisfy the condition of loose contact.

Procedure FindLooseContact(T_0, T_{37}, T_{39}, k, p)
(0) **For** $j = 1$ **to** $k + 1$
(0a) Append-head(T_{37}, $\Phi_{k,p,j}{}^0$).
EndFor
(1) **For** $j = k + 2$ **to** L
(1a) **CompareCoordinateValue**(T_{37}, T_{39}, $T_{39}^{=}$, $T_{39}^{\neq}$, k, j, d).
(1b) **If** (Detect($T_{39}^{=}$) = = "*yes*") **Then**
(1c) $T_{39} = \cup(T_{39}, T_{39}^{=})$.
(1d) **CompareCoordinateValue**(T_{37}, T_{39}, $T_{39}^{=}$, $T_{39}^{\neq}$, k, j, b).
(1e) **If** (Detect($T_{39}^{=}$) = = "*yes*") **Then**
(1f) $T_0 = +(T_0, s_j{}^1)$ and $T_0^{off} = -(T_0, s_j{}^1)$.
(1g) **If** (Detect(T_0) = = "*yes*") **Then**

Fig.9. Procedure FindLooseContact(T_0, T_{37}, T_{39}, k, p)

Figure 10, **CompareCoordinateValue**(T_{37}, T_{39}, $T_{39}^{=}$, $T_{39}^{\neq}$, k, j, δ), is applied to judge whether coordinate values in a conformation in tube T_{37} and coordinate values in any conformation in tube T_{39} are the same.

Procedure CompareCoordinateValue(T_{37}, T_{39}, $T_{39}^{=}$, $T_{39}^{\neq}$, k, j, δ)
(1) **For** $q = 1$ **to** n
(1a) $T_{37}^{on} = +(T_{37}, \delta_{(k+1),q}{}^1)$ and $T_{37}^{off} = -(T_{37}, \delta_{(k+1),q}{}^1)$.
(1b) $T_{39}^{on} = +(T_{39}, \delta_{j,q}{}^1)$ and $T_{39}^{off} = -(T_{39}, \delta_{j,q}{}^1)$.
(1c) **If** (Detect(T_{37}^{on}) = = "*yes*") **Then**
(1d) $T_{39}^{=} = \cup(T_{39}^{=}, T_{39}^{on})$, $T_{39}^{\neq} = \cup(T_{39}^{\neq}, T_{39}^{off})$ and $T_{37} = \cup(T_{37}, T_{37}^{on})$.
Else
(1e) $T_{39}^{=} = \cup(T_{39}^{=}, T_{39}^{off})$, $T_{39}^{\neq} = \cup(T_{39}^{\neq}, T_{39}^{on})$ and $T_{37} = \cup(T_{37}, T_{37}^{off})$.
EndIf
(1f) **If** (Detect($T_{39}^{=}$) = = "*yes*") **Then**
(1g) $T_{39} = \cup(T_{39}, T_{39}^{=})$.
Else
(1h) Terminate the execution of the loop.
EndIf
EndFor
(2) **If** (Detect(T_{39}) = = "*yes*") **Then**
(2a) $T_{39}^{=} = \cup(T_{39}, T_{39}^{=})$.
EndIf
EndProcedure

Fig.10. Procedure CompareCoordinateValue(T_{37}, T_{39}, $T_{39}^{=}$, $T_{39}^{\neq}$, k, j, δ)

F. Find the Maximum Number of H-H Contacts in the 3D HP Lattice Model

Figure 11, **FindTheMaxNumberOfHHContacts**(T_1), is applied to find conformation of a protein with global minimum energy in T_1.

Procedure FindTheMaxNumberOfHHContacts(T_1)
(1) $T_{1,0,0} = \cup(T_{1,0,0}, T_1)$.
(2) **For** $k = 1$ **to** L
(3) **For** $p = 0$ **to** 3
(4) **For** $j = 0$ **to** $L - 1$
(5) **For** $q = (k - 1) * (4 * L) + (p * L) + (p * L)$ **downto** 0
(5a) $T_{k,p,q+1}{}^{on} = +(T_{k,p,q}, \Phi_{k,p,j+1}{}^1)$ and $T_{k,p,q+1}{}^{off} = -(T_{k,p,q}, \Phi_{k,p,j+1}{}^1)$.
(5b) $T_{k,p,q+1} = \cup(T_{k,p,q+1}, T_{k,p,q+1}{}^{on})$.
(5c) $T_{k,p,q} = \cup(T_{k,p,q}, T_{k,p,q+1}{}^{off})$.
EndFor
EndFor
EndFor
EndFor
(6) **For** $k = L$ **downto** 1
(7) **For** $p = 3$ **downto** 0
(8) **For** $j = L$ **downto** 1
(8a) **If** (Detect($T_{k,p,j}$) = = "*yes*") **Then**

(8b) Read($T_{k,p,j}$).
(8c) Terminate the algorithm.
EndIf
EndFor
EndFor
EndFor
EndProcedure

Fig.11. Procedure FindTheMaxNumberOfHHContacts(T_1)

G. Intelligent Parallel DNA algorithm for 3D Protein Structure Prediction using the HP Lattice Model on Optimal Bioinformatics Logic Computing

In this subsections, *intelligent* DNA algorithms are proposed to predicate the conformation of any given a protein with the amino acid sequence **a** = $a_1 \ldots a_n$ in the 3D HP lattice model. The algorithm is shown in Figure 12.

Algorithm 1: Finding conformation of a protein with the amino acid sequence **a** = $a_1 \ldots a_n$ and the maximum number of H-H contacts in the 3D HP lattice model.

(1) **Delegate**(T_0).
(2) **Building3DHPLattice**(T_1).
(3) **CreateAllOfPossibleConfirmationSpaces** (T_1).
(4) **ComputeTheNumberOfHHContacts** (T_0, T_1).
(5) **FindTheMaxNumberOfHHContacts** (T_1).
EndAlgorithm

Fig.12. Parallel DNA algorithm for 3D Protein Structure Prediction using the HP Lattice Model on Optimal Bioinformatics Logic Computing

V. CONCLUSIONS

Some famous problems in biology science contain sequence comparison, multiple alignments, finding signals in DNA, gene prediction, genome rearrangements, computational proteomics, and protein folding. Those famous problems have been proved to be NP-Complete problems or NP-Hard problems. Based on bio-molecular computational operations in Adleman's experiments, the logic truth tables are used to implement and optimize logic bio-circuit operations to construct most basic DNA logic circuits such as parallel adder, parallel subtractor, parallel multiplier, and parallel divider that can be found in [11-13]. For protein folding, our proposed bio-molecular logic computing algorithm is the first algorithm to demonstrate theoretically how basic biological operations can be used to solve the problem of protein folding with polynomial-time biological operations in the HP lattice model. This implies that it is another feasible selection for using biological operations to solve famous problems in biology science.

REFERENCES

[1] K.A. Dill. "Theory for the Folding and Stability of Globular Proteins". *Biochemistry* 24: pp. 1501-1506, 1985.

[2] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni and M. Yannakakis. "On the Complexity of Protein Folding". *Journal of Computational Biology*, 5(3): pp. 409-422, 1998.

[3] L. M. Adleman. "Molecular Computation of Solutions to Combinatorial Problems". *Science*, Volume 266, Number 11, pp. 1021-1024, 1994.

[4] Karl-Heinz Zimmermann. *An Introduction to Protein Informatics*. Kluwer Academic Publisher, 2003, ISBN 1-4020-7578-2.

[5] K.A. Dill. "Dominant Forces in Protein Folding". *Biochemistry* 29, pp. 7133-7155, 1990.

[6] K.A. Dill, S. Bromberg, K. Yue, K.M. Fiebig, D.P. Yee, P.D. Thomas and H.S. Chan. "Principles of Protein Folding: a Perspective from Simple Exact Models". *Protein Science* 4, pp. 561-602, 1995.

[7] R. Unger and J. Moult. "Finding the lowest free energy conformation of a protein is an NP-hard problem proof and implications". *Bulletin of Mathematical Biology* 55, pp. 1183-1198, 1993.

[8] W. E. Hart and S. Istrail. "Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal". *Journal of Computational Biology* 3, pp. 53-96, 1996.

[9] W. E. Hart and S. Istrail. ""Lattice and off-lattice side chain models of protein folding: linear *time structure prediction better than 86% of optimal"*. *Journal of Computational Biology* 4, pp. 241-259, 1997.

[10] A. Nayak, A. Sinclair and U. Zwick. "Spatial codes and the hardness of string folding problems". *Symposium on Discrete Algorithms archive Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms table of contents*, San Francisco, California, United States, pp. 639 - 648, 1998.

[11] Sientang Tsai, Michael(Shan-Hui) Ho, and Weng-Long Chang. "Constructing Bio-molecular Parallel Adder with Basic Logic Operations in the Adleman-Liption Model ". *The 10th International Conference on Convergence Information Technology*, 2007, 21-23 Nov. 2007, pp 925 – 930.

[12] Yu-Ying Shi, Michael(Shan-Hui) Ho, Yu-Jen Wang, and Chun-Yu Huang, "Constructing bio-molecular parallel adder and multiplier with basic logic operations in the Adleman-Liption model (Submitted for publication)," in *ICSC &ISIS*, 2008, submitted for publication.

[13] Yu-Ying Shi, Michael(Shan-Hui) Ho, Chun-Yu Huang, and Yu-Jen Wang, "Constructing bio-molecular parallel subtractor and divider with basic logic operations in the Adleman-Liption model (Submitted for publication)," in *ICSC &ISIS*, 2008, submitted for publication.