Protein structure prediction in the 3D HP model

Fatima Kanj[1], Nashat Mansour[1], Hassan Khachfe[2], Faisal Abu-Khzam[1]
[1] Department of Computer Science and Mathematics, Lebanese American University, Lebanon
[2] Department of Biology and Biomedical Sciences, Lebanese International University, Lebanon
Emails: malakr@hotmail.com; nmansour@lau.edu.lb; hassan.khachfe@liu.edu.lb; faisal.abukhzam@lau.edu.lb

***Abstract*—Proteins are initially linear chains of amino acids that fold, under the influence of several chemical and physical factors, into their 3-dimensional structures. Due to the importance of this problem and since laboratory techniques are not always feasible, computational methods for characterizing protein structures have been proposed. In this paper, we present a Particle Swarm Optimization (PSO) based algorithm for predicting protein structures in the 3D HP model. Starting from a small set of potential solutions, our algorithm efficiently explores the search space of candidate solutions and returns 3D protein structures with minimal energy. To test our algorithm, we use two sets of benchmark sequences of different lengths. It is found that the results of the PSO algorithm are better than those of previous algorithms.**

I. INTRODUCTION

Proteins perform many biological functions and represent the building blocks of organisms. Proteins fold, under the influence of several chemical and physical factors, into their 3-dimensional structures which determine their biological functions and properties. Due to the importance of this issue to the human life, scientists have developed laboratory techniques such as X-ray crystallography and nuclear magnetic resonance to determine the native structures of proteins. Although these methods are reliable, they are not always feasible. Hence, predicting the native structure of a protein, given its primary sequence, is an important and challenging task in computational biology.

The primary structure is a linear sequence of amino acids connected together via peptide bonds. Proteins fold due to hydrophobic effect, Vander Waals interactions, electrostatic forces, and Hydrogen bonding [1]. The secondary structures are three-dimensional structures characterized by a repeating bonding pattern. The most common structures are helices and strands. The proteins that include these secondary structures can further fold into the tertiary structure forming a bundle of secondary structures, turns and loops. Furthermore, the aggregation of tertiary structure regions of separate protein sequences forms the so called quaternary structures.

The main computational approaches of PSP are: Homology modeling, threading, and ab initio methods [2]. For the latter ones, the only needed input is the amino acid sequence whereas for the first two methods, data of previously predicted protein structures are used. Ab initio methods predict the 3D structure of proteins given their primary sequences without relying on protein databases. The underlying strategy is find the best possible structure based on a chosen energy function. Based on the laws of physics, the most stable structure is the one with the lowest possible energy [3]. The main challenge of these approaches is searching for the most stable structure in a huge search space. Some models, such as the Hydrophobic-Polar models have been developed and used in order to restrict the search to a smaller search space whereas other models use the detailed representation of proteins with all the corresponding atoms.

Hydrophobic-Polar (HP) models represent each amino with all of its atoms as one bead labeled as either hydrophobic (H) or polar (P) [4]. According to this model, beads lie on points defined by a lattice according to some chosen algorithm such that the most stable structure is the one with the hydrophobic amino acids lying in its core. The underlying concept is that hydrophobic amino acids tend to escape from having contact with the solvent and hence tend to move inside the structure whereas the polar ones remains on the outside. The main energy function used in this model is the total number of the hydrophobic interactions between the amino acids and the goal is to have a lattice with minimum energy, i.e., with maximum number of H-H contacts. HP models can be 2D or 3D. We focus on cubic 3D models.

The problem of predicting protein structures in the HP model is intractable [5]. Hence, heuristic and meta-heuristics algorithms have been reported for finding good sub-optimal solutions. In the early nineties, Unger and Moult [6,7] developed a genetic algorithm (GA) combined with the Monte Carlo method to fold proteins on a two dimensional lattice and they extended their work later to a 3D lattice. Later, a standard GA was developed and it outperformed that of Unger and Moult by reaching higher number of hydrophobic contacts with less number of energy evaluations [8]. Another genetic algorithm to fold proteins on a 3D lattice using a modified energy function was developed by Custódio et al. [9]. Recently, Johnson and Katikireddy [10] proposed a genetic algorithm with a backtracking method to resolve the collision problem. Also, Bui and Sundarraj [11] proposed a method which is a combination of two genetic algorithms. The first is a GA for the secondary structure evolution. The second GA uses the resulting secondary structures to find the most stable conformations of the protein.

Heuristic methods based on assumptions about the folding mechanism were proposed, such as: the hydrophobic zipper of Dill et al. [12], the constrained hydrophobic core construction algorithm of Yue and Dill [13], and the contact interactions method of Toma and Toma [14]. Also, a branch and bound algorithm was developed by Chen and Huang [15]. The algorithm evaluates the importance of every possible position of the hydrophobic amino acids and only those promising locations are preserved for more branching at every level. A number of methods based on the Monte Carlo (MC) algorithm have also been proposed: the pruned-enriched Rosenbluth method [16], an Monte Carlo based growth algorithm [17], the dynamic Monte Carlo algorithm [18], and the evolutionary Monte Carlo algorithm [19]. Further, a modified particle swarm optimization algorithm for the protein structure prediction problem in the 2D toy model was proposed by Zhang and Li [20]. An Ant Colony Optimization algorithm was proposed by Shmygelska and Hoos [21] for both 2D and 3D lattice models.

In this paper, we present a Particle Swarm Optimization (PSO) based algorithm for protein structure prediction in the cubic 3D hydrophobic polar (HP) model. PSO is a population-based evolutionary search strategy in which the underlying metaphor is cooperation rather than rivalry and competition. We evaluate our predicted structures using their energy values and the number of energy evaluations required. Our PSO based algorithm efficiently searches the search space of potential 3D solutions to find structures with compact hydrophobic cores and higher number of H-H contacts. Our algorithm produces comparable but slightly better results than those of published methods that are based on genetic algorithms.

This paper is organized as follows. Section 2 gives background information on PSO algorithms. Section 3 describes the proposed PSO algorithm for PSP. Section 4 presents empirical results. Section 5 concludes the paper.

II. Background on Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based evolutionary search strategy. The initial version of PSO was developed by Eberhart and Kennedy [22]. The underlying theory is that individuals maintain some levels of cognitive consistency through social learning, cooperation and communication with others; the same applies to swarms of birds or fish which move in the same direction towards the same destination by following each other.

The basic steps of the PSO algorithm are as follows:

a. Randomly initialize the *Swarm*, which is the population of candidate solutions called "*Particles*". At any time t, any *Particle i* represents a *Position* X_i in the search space.
b. Compute the objective functions of the particles which evaluate the *Positions* of the *Particles* in the search space.
c. Keep track of every *Particle*'s best *Position* which it has achieved so far. This position is henceforth referred to as *pBest* or P_i. Also, keep track of the best *Position* achieved so far by all *Particles* in the *Swarm*. This position is henceforth referred to *gBest* or P_g.
d. Update the *Velocity* of every *Particle*, at time, t, so that it moves to a *New_Position* closer to *pBest* and *gBest.*
e. Repeat steps b-d until a stopping criterion is satisfied.

To update the velocity and position of a particle, the velocities and positions of all of its components need to be updated. The velocity V_d and the position X_d for component d of particle i at time t+1 are given by:

$$V_d(t+1) = \omega * V_d(t) + c_1 * r_1 * (P_{i,d} - X_d(t)) + c_2 * r_2 * (P_{g,d} - X_d(t)) \quad (1)$$

Then, the position is shifted according to the following equation:

$$X_d(t+1) = X_d(t) + V_d(t+1) \quad (2)$$

where: $P_{i,d}$ is the position of the component d found in P_i (i.e. pBest) and $P_{g,d}$ is the position of the component d found in P_g (i.e. gBest).

The parameters used to update the velocity:

- ω is referred to as the inertia. It represents the weight given to the velocity of the component d.
- c_1 and c_2 are referred to "self confidence" and "swarm confidence" respectively. These parameters are to be multiplied by both the vectors from the current position $X_d(t)$ to pBest ($P_{i,d}$) and gBest ($P_{g,d}$).
- r_1 and r_2 are random real numbers between 0 and 1; they determine the influence of pBest (P_i) and gBest (P_g) respectively. It is claimed that the randomness generated by these two parameters allows the particles to fly through the search space and prevents fast convergence into local optima.

The update velocity formula takes into account three vectors for finding the new position. The first vector is the velocity of the particle's component d and it is scaled using the parameter ω. The second vector is represented by ($P_{i,d} - X_d(t)$, and is scaled with $c_1 * r_1$. The third vector is represented by ($P_{g,d} - X_d(t)$, and it is scaled with $c_2 * r_2$.

III. PSO for Protein Structure Prediction

In this section, we adapt the PSO algorithm for solving the PSP problem in the cubic HP model.

A. *Solution Representation*

A particle is a candidate solution represented by an array of length n-1, where n is the number of amino acids in the respective protein. Each element in the array represents the position X_d of the corresponding amino acid d with respect to the preceding one and its value can be one of six characters {b, f, u, d, l, r}. These characters represent the following six directions, respectively {backward, forward, up, down, left, right}. Initially, the swarm is populated with a set of N candidate solutions which are randomly generated. That is, each position X_d of amino acid d (d = 1, 2,…, n-1) is assigned a random value for the candidate solution/particle i (i = 0, 1, …., N-1). All the velocities are initially set to 0.

B. *Repair Algorithm*

A particle is invalid if it experiences collision. Collision occurs if two or more amino acids lie at the same point on the cubic 3D lattice. Invalid particles are not accepted in our proposed algorithm but are repaired using a backtracking repair function, which takes as input the invalid particle and returns as output the repaired one, if possible, or the same particle if it could not be repaired. The repair function detects a collision and tries to repair it locally by finding an alternative empty location for the amino acid which caused the collision. If none is available, then it searches for previous amino acids which locations can be modified. If modifications were performed for more than three amino acids or if none can be modified, then it is assumed that the particle cannot be repaired and the initial input particle is returned.

C. *Objective Function*

The objective function is the sum of the hydrophobic contacts between non adjacent amino acids multiplied by -1. Since we are using the cubic lattice, the maximum number of possible contacts per amino acid is four, except for the first and last amino acids, which might have up to five contacts. Each H-H contact is given the score of -1. This type of scoring is used since we need to minimize the objective function to make it similar to the energy function of real proteins. We, henceforth, use the terms energy and objective function interchangeably. The goal is to minimize the energy of the particles to obtain structures with the most compact hydrophobic core. For example, in Figure 3, the energy value of the displayed structure is -5. The hydrophobic contacts are displayed in dotted lines and there are five of them between the following pairs of hydrophobic amino acids: (3, 8), (3, 10), (4, 7), (5, 10) and (6, 9).

Evaluating the energy of a particle is simple. Every hydrophobic amino acid in the sequence is checked for any non-adjacent (not connected by a bond) hydrophobic amino acids in the six positions around it on the lattice, at a distance 1, and the number of these amino acids is accumulated.

D. *Position Update*

The position of the amino acids in each particle are updated using the procedure UPDATE_POSITION, which employs Equations 1 and 2. In this procedure, the direction of each amino acid (AA) i in a particle, except the first one, with a certain probability, RATE. The possible direction that an AA i can take is one of six: b, f, u, d, l and r with respect to AA (i-1). The new position of AA i is determined by a change in the x, y or z coordinate, which is then translated into one of the six directions. The choice of which coordinate to change is done randomly. To determine the new position of AA i, we first calculate its velocity with respect to the chosen coordinate/axis using Equation (1). Then, the position fo the amino acid is updated, using Equation 2, along the same axis. The value of the value of the position is converted to either 1 (if positive) or -1 (if negative), with respect to AA (i-1), since the distance between any two consecutive amino acids in the lattice is considered to be unity. This position value, thus, determines the updated direction in the lattice.

E. *Determining RATE and Selection Policy of Amino Acids*

The RATE parameter is used to determine the percentage of the amino acids of a particle for which the UPDATE_POSITION function is applied. In our adapted PSO algorithm, RATE is set to 0.1, meaning that only 10% of the directions of the particles are updated, in each iteration. This reduced RATE value is employed in order to accomplish a better neighborhood search. In updating the positions of the amino acids of a particle, the amino acids are selected in an ordered way, starting with the first (the left-most) amino acid. Henceforth, we refer to this policy as 'sequential'.

F. *Acceptance Criterion of New Particle*

In this adapted version of PSO, we use a greedy policy for accepting a new particle. That is, a new particle replaces the old particle if its energy value is lower or equal to the energy of the old particle.

IV. Empirical Results

In this section, we report our empirical results and compare them to those of published techniques: one by Patton et al. [8], which proposed a standard genetic algorithm for this problem and reported better results than those achieved by Unger and Moult [6]; the second is by Johnson et al. [10], which reports better results than those achieved by Patton et al. for the smaller sequences. We also experiment with some variants of our PSO algorithm.

A. *Empirical Procedure*

We use two sets of benchmark sequences used first by Unger and Moult [6]. These are amino acid sequences of Hs and Ps generated randomly: 10 sequences are of length 27 and 10 sequences of length 64.

We evaluate the results using the following metrics:

- Energy: It is the total number of non consecutive H-H contacts multiplied by -1.

- Number of Energy Evaluations: This is the number of times the energy function is computed to reach the final energy score for a specific sequence. This metric is used as an indicator of the efficiency of our algorithm.
- Relative Percentage of Energy Evaluations: This metric refers to the percentage of our number of energy evaluations with respect to the number of energy evaluations recorded by the published results.
- Time: This is the time needed to produce results, which is reported only to give an idea of the required execution time.

The parameters used in the PSO algorithm are set as follows:

- Inertia (ω): It is typically set between 0.4 and 0.9 [23]. For all of our versions, we set it to 0.5.
- Self Confidence (c_1) and Swarm Confidence (c_2) values are set to 2 [22].
- r_1 and r_2: are real random numbers which can range between 0 and 1 [24].
- Swarm Size: Typically, the swarm size used in PSO algorithms is fairly small. For many problems, a swarm size of 20 particles can be sufficient [23]. We use a swarm size of 5 for the smaller sequences and 10 for the longer sequences.
- Number of Iterations: We allow the algorithm to run for 10000 iterations for the 27-long sequences and 40000 iterations for the 64-long sequences. However, we record the results at the point beyond which no improvement takes place.

B. *Results*

Tables 1 and 2 present the results of the PSO algorithm and also include the previously published results, for proteins with lengths 27 and 64 amino acids, respectively. Based on these results, we make the following comments:

- For the 27-long sequences, the energies recorded by PSO for all sequences, except for sequence 273d.6, were equal to those of Johnson et al. [10] but with a fewer number of energy evaluations. For sequence 273d.6, PSO found a structure with an energy value of -12, which is lower than the lowest so far in the literature, to the best of our knowledge.
- For the 64-long sequences, PSO found lower energy values than Patton et al. [8] for 9 out of the 10 sequences, although for a higher number of energy evaluations. However, for finding the same energy values, PSO runs for a comparable number of energy evaluations.
- PSO incorporates a combination of appropriate choices that keep only good particles and searching for same quality or better ones in their small neighborhoods. This way, we are forcing the swarm to improve while exploring the search space with small jumps. PSO slightly outperforms the best published results so far for the given set of sequences with respect to the number of evaluations for the 27-long sequences and with respect to the energy values for the 64-long sequences. By using a very small set of candidate solutions in the swarm, PSO is capable of exploring the search space and finds slightly better structures with lower energy than genetic algorithms, which normally require a fairly large population. However, as shown in Table 2, the PSO algorithm performs a large number of energy evaluations.

V. CONCLUSION

We have presented a PSO based algorithm for solving the PSP problem in the 3D HP model. The goal is to minimize the energy by maximizing the number of H-H contacts. Our proposed PSO algorithm efficiently explores the search space of possible solutions and returns the 3D structure with low energy. The performance of our algorithm is evaluated by comparing it to previous algorithms using the same set of benchmark sequences. Our PSO algorithm produces better results than those published results by reaching the same energy values with fewer number of energy evaluations for the small sequences and by finding lower energy structures for the longer ones.

ACKNOWLEDGMENT

This work was partially supported by the Lebanese American University and the National Council for Scientific Research.

REFERENCES

[1] J. Setubal, and J. Meidanis, *Introduction to Computational Molecular Biology,* Boston: PWS Publishing Company, 1997.

[2] A.R. Sikder, and A.Y. Zomaya, "An overview of protein-folding techniques: issues and perspectives," *International Journal of Bioinformatics Research and Applications*, Vol. 1, No. 1, pp. 121-143, 2005.

[3] Anfinsen, C.B., "Principles that govern the folding of proteins," *Science*, pp. 181-187, 1973.

[4] K. A. Dill, "Theory for the folding and stability of globular proteins," *Biochemistry*, Vol. 24, No. 6, pp. 1501-1509, 1985.

[5] 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*, pp. 1183-1198, 1993.

[6] R. Unger, and J. Moult, "A genetic algorithm for 3D protein folding simulations," *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 581-588.

[7] R. Unger, and J. Moult, "Genetic algorithms for protein folding simulations," *Journal of Molecular Biology*, Vol. 231, pp. 75-81, 1993.

[8] A. L. Patton, W. F. Punch, and E. D. Goodman, "A standard GA approach to native protein conformation prediction," *Proceedings of the Sixth International Conference On Genetic Algorithms,* 1995.

[9] F. Custódio, H. Barbosa, and L. Dardenne, "Investigation of the three-dimensional lattice HP protein folding model using a genetic algorithm," *Genetics and Molecular Biology*, Vol. 27, No. 4, pp. 611-615, 2004.

[10] C. Johnson, and A. Katikireddy, “A genetic algorithm with backtracking for protein structure prediction,” *Proceedings of the 8th Annual Conference on Genetic And Evolutionary Computation,* Washington, USA, 2006.

[11] T.N. Bui, and G. Sundarraj, “An efficient genetic algorithm for predicting protein tertiary structures in the 2D HP model,” *Proceedings of the 7th Conference on Genetic and Evolutionary Computation*, 2005, pp. 385-392.

[12] K.A. Dill, K.M. Fiebig, and H.S. Chan, “Cooperativity in protein-folding kinetics,” *Proceedings of the National Academy of Sciences of the United States of America*, 90, 1993, pp. 1942-1946.

[13] K. Yue, and K.A. Dill, “Forces of tertiary structural organization in globular proteins,” *Proceedings of the National Academy of Sciences of the United States of America*, 92, 1995, pp. 146-150.

[14] L. Toma, and S. Toma, “Contact interactions method: A new algorithm for protein folding simulations,” *Protein Science*, Vol. 5, pp. 147-153, 1996.

[15] M. Chen, and W. Huang, “A branch and bound algorithm for the protein folding problem in the HP lattice model,” *Genomics, Proteomics & Bioinformatics*, Vol. 3, No. 4, 2005.

[16] U. Bastolla, H. Fravenkron, E. Gestner, P. Grassberger, and W. Nadler, “Testing a new Monte Carlo algorithm for the protein folding problem,” *Proteins*, Vol. 32, pp. 52-66. 1998.

[17] H.P. Hsu, V. Mehra, W. Nadler, and P. Grassberger, “Growth algorithm for lattice heteropolymers at low temperatures,” *Journal of Chemical Physics*, Vol. 118, pp. 444-51, 2003.

[18] R. Ramakrishnan, B. Ramachandran, and J.F. Pekny, “A dynamic Monte Carlo algorithm for exploration of dense conformational spaces in heteropolymers,” *Journal of Chemical Physics*, Vol. 106, No. 6, pp. 2418-2424, 1997.

[19] F. Liang, and W.H. Wong, “Evolutionary Monte Carlo for protein folding simulations,” *Journal of Chemical Physics*, Vol. 115, No. 7, pp. 3374-3380, 2001.

[20] X. Zhang, and T. Li, “Improved particle swarm optimization algorithm for 2D protein folding prediction. *Proceedings of the 1st International Conference on Bioinformatics and Biomedical Engineering*, 2007, pp. 53-56.

[21] A. Shmygelska, and H.H. Hoos, “An ant colony optimization algorithm for the 2D and 3D hydrophobic polar protein folding problem,” *BMC Bioinformatics*, Vol. 6, No. 30, 2005.

[22] R.C. Eberhart, and J. Kennedy, “Particle swarm optimization,” *Proceedings of the IEEE International Conference of Neural Networks*, 1995, pp. 1942-1948.

[23] D.N. Wilke, *Analysis of the Particle Swarm Optimization Algorithm*, Masters Dissertation, University of Pretoria. 2005.

[24] S. Das, A. Abraham, and A. Konar, “Swarm intelligence algorithms in bioinformatics,” in *Computational Intelligence in Bioinformatics*, A. Kelemen, A. Abraham, and Y. Chen, Eds. Berlin: Springer, 2008, pp. 113-147.

TABLE 1. RESULTS FOR SEQUENCES OF LENGTH 27

Seq #	**Johnson et al. (2006)**		**PSO**		
	Energy	***#Energy Evaluations***	***Energy***	***#Energy Evaluations***	***Time (sec)***
273d.1	-9	15,854	-9	3,158	7
273d.2	-10	19,965	-10	5,771	10
273d.3	-8	7,991	-8	2,667	7
273d.4	-15	23,525	-15	8,556	14
273d.5	-8	3,561	-8	893	2
273d.6	-11	14,733	-12	12,790	22
273d.7	-13	23,112	-13	17,024	28
273d.8	-4	889	-4	149	0.5
273d.9	-7	5,418	-7	1,915	5
273d.10	-11	5,592	-11	2,638	8

TABLE 2. RESULTS FOR SEQUENCES OF LENGTH 64

Seq #	**Patton et al. (1995)**		**PSO**		**PSO**		
	Energy	***#Energy Evaluations***	***Energy***	***#Energy Evaluations***	***Energy***	***#Energy Evaluations***	***Time (min)***
643d.1	-27	433,533	-27	422,373	-28	1,131,552	12
643d.2	-30	167,017	-30	159,873	-31	456,877	5
643d.3	-38	172,192	-38	109,541	-39	113,315	1
643d.4	-34	107,143	-34	167,879	-36	1,730,129	18
643d.5	-36	154,168	-36	189,634	-38	1,602,646	17
643d.6	-31	454,727	-31	410,586	-31	410,586	5
643d.7	-25	320,396	-25	309,532	-27	1,296,319	3
643d.8	-34	315,036	-34	410,813	-35	1,113,330	12
643d.9	-33	151,705	-33	143,182	-35	404,199	4
643d.10	-26	191,019	-26	165,762	-27	175,053	2