Particle Swarm Optimization Approach for Protein Structure Prediction in the 3D HP Model

Article *in* Interdisciplinary Sciences Computational Life Sciences · September 2012

DOI: 10.1007/s12539-012-0131-z · Source: PubMed

3 authors:

Nashat Mansour
Lebanese American University
90 PUBLICATIONS **1,096** CITATIONS
SEE PROFILE

Fatima Kanj
Lebanese American University
4 PUBLICATIONS **107** CITATIONS
SEE PROFILE

Hassan M. Khachfe
Lebanese International University
35 PUBLICATIONS **148** CITATIONS
SEE PROFILE

Interdiscip Sci Comput Life Sci (2012) 4: 190–200
DOI: 10.1007/s12539-012-0131-z

Particle Swarm Optimization Approach for Protein Structure Prediction in the 3D HP Model

Nashat MANSOUR[1*], Fatima KANJ[1], Hassan KHACHFE[2]
[1](Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon)
[2](Department of Biology and Biomedical Sciences, Lebanese International University, Beirut, Lebanon)

Received 14 April 2011 / Revised 16 March 2012 / Accepted 17 June 2012

Abstract: The primary structure of proteins consists of a linear chain of amino acids that can vary in length. Proteins fold, under the influence of several chemical and physical factors, into their 3D structures, which determine their biological functions and properties. Misfolding occurs when the protein folds into a 3D structure that does not represent its native structure, which can lead to diseases. 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 hydrophobic polar model. Starting from a small set of candidate solutions, our algorithm efficiently explores the search space and returns 3D protein structures with minimal energy. To test our algorithm, we used two sets of benchmark sequences of different lengths and compared our results to published results. Our algorithm performs better than previous algorithms by finding lower energy structures or by performing fewer numbers of energy evaluations.
Key words: *ab initio* approach, HP model, particle swarm optimization, protein structure prediction.

1 Introduction

Proteins perform many biological functions and represent the building blocks of organisms. They are complex organic compounds of which the basic forming unit is the amino acid. Proteins are initially linear chains of amino acids which can vary in length from a few up to thousands of amino acids. Proteins fold, under the influence of several chemical and physical factors, into their unique 3D structures which determine their biological functions and properties. Misfolding occurs when the protein folds into a 3D structure that does not represent its correct native structure, which can lead to many diseases, such as Alzheimer, several types of cancer, *etc.* (Prusiner,1998). Due to the importance of this issue to human life, scientists have developed laboratory techniques such as X-ray crystallography and nuclear magnetic resonance (NMR) 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 protein structure is a linear sequence of amino acids connected together via peptide bonds. Proteins fold due to hydrophobic effect, van der Waals interactions, electrostatic forces, hydrogen bonding, *etc.* 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 some separate protein sequences forms the so called quaternary structures (Rylance, 2004).

The protein structure prediction (PSP) problem is intractable (Unger and Moult, 1993a). Hence, the main computational approaches are heuristics and can be classified as homology modeling, threading, and ab initio methods (Sikder and Zomaya, 2005). 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.

Homology modeling uses sequences of known structures in the protein data banks to align with the given protein sequence whose 3D structure is to be predicted. This approach is based on the assumption that new proteins have evolved from previous ones after passing through a set of mutations, which change some amino

*Corresponding author.
E-mail: nmansour@lau.edu.lb

acids without affecting the 3D structures. Examples of methods and empirical work on homology modeling are Kopp and Schwede (2004) and Pandit *et al.* (2006).

Threading is similar to homology modeling. But, instead of finding similar sequences to deduce the native conformation of the target protein, threading assumes that the target structure is similar to another existing structure, which should be searched for. Examples of threading methods are Lathrop *et al.* (1998) and Jones (1998).

Ab initio methods predict the 3D structure of proteins given their primary sequences without relying on protein databases. The underlying strategy is to 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 (Anfinsen, 1973). 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.

Detailed models consider the interactions between all atoms of the protein sequence. Therefore, the search space is huge, taking into consideration an overwhelming number of possible degrees of freedom and interactions between the different atoms. The energy function is usually based on molecular mechanics and force fields components such as bond lengths, bond angles, dihedral angles, van der Waals interactions, electrostatic forces, *etc.* Genetic Algorithms have been proposed for PSP in the detailed model (Schulze-Kremer, 2000). Cui *et al.* (1998) has proposed a genetic algorithm which constrains the search space of the native conformations by making use of the super secondary structures of the input protein which are predicted using an artificial neural network. A method based on molecular dynamics has been proposed (Klepeis and Floudas, 2003). A hybrid approach combining a particle swarm optimization (PSO) algorithm and artificial neural network has been developed by Datta *et al.* (2008). Recently, a scatter search algorithm has been proposed for tertiary structure prediction (Mansour *et al.*, 2009). Rosetta and TASSER are also well known techniques that employ previous protein fragments (Das and Baker, 2008; Roy *et al.*, 2010). Floudas (2007) surveyed more computational methods.

Hydrophobic-Polar (HP) models represent each amino with all of its atoms as one bead labeled as either hydrophobic (H) or polar (P). 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 remain 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 2-dimensional (2D) or 3-dimensional (3D), as illustrated in Fig. 1. In this work, we focus on cubic 3D models. The 3D or cubic lattice differs from the square lattice by representing amino acids using the z coordinate in addition to the x and y coordinates. The problem is about folding a string of Hs and Ps on a three dimensional coordinates system in a self-avoiding walk.

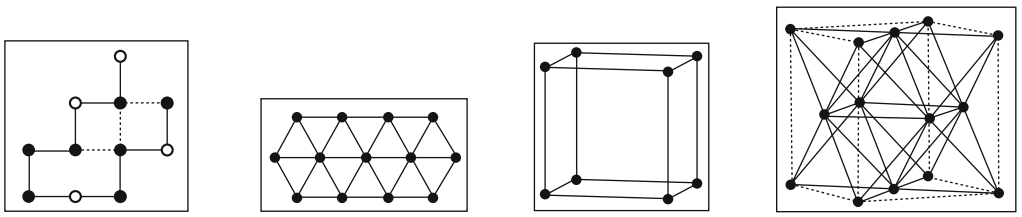

Fig. 1 From the left: square, triangular, cubic, and face-centered-cubic lattices (Hart and Newman, 2006)

The problem of predicting protein structures in the HP model is intractable. Hence, heuristic and metaheuristics algorithms have been reported for finding good sub-optimal solutions. In the early nineties of last century, Unger and Moult (1993b) developed genetic algorithms (GA) combined with the Monte Carlo method to fold proteins on 2D and 3D lattices. Later, a standard GA was developed and it outperformed that of Unger and Moult by reaching a higher number of hydrophobic contacts with a smaller number of energy evaluations (Patton *et al.*, 1995). Another genetic algorithm to fold proteins on a 3D lattice using a modified energy function was developed by Custódio *et al.* (2004). Recently, Johnson and Katikireddy (2006) proposed a genetic algorithm with a backtracking method to resolve the collision problem. Also, Bui and Sundarraj (2005) 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 constrained hydrophobic core construction algorithm (Yue and Dill, 1995) and the contact interactions method (Toma and Toma, 1996). A branch and bound algorithm was developed by Chen and Huang (2005). 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 (Bastolla *et al.*, 1998), an MC based growth algorithm (Hsu *et al.*, 2003), and the evolutionary MC algorithm (Liang and Wong, 2001). Further, a modified particle swarm optimization algorithm for the protein structure prediction problem in the 2D toy model was proposed by Zhang and Li (2007). An Ant Colony Optimization algorithm was proposed by Shmygelska and Hoos (2005) 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 a higher number of H-H contacts. Our algorithm produces better results than those of published methods that are based on genetic algorithms.

2 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 (1995). 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 the *Particles*, 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 are:

- 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 the vectors from the current position $X_d(t)$ to *pBest* ($P_{i,d}$) and *gBest*($P_{g,d}$), respectively.
- 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 $\overrightarrow{XP_i}$, which represents $(P_{i,d} - X_d(t))$, and is scaled with $c_1 * r_1$. The third vector is $\overrightarrow{XP_g}$, representing $(P_{g,d} - X_d(t))$, and it is scaled with $c_2 * r_2$.

3 PSO for protein structure prediction

In this section, we adapt the PSO algorithm for solving the PSP problem in the cubic HP model. Fig. 2 shows the different steps of the algorithm, which are described in the following subsections.

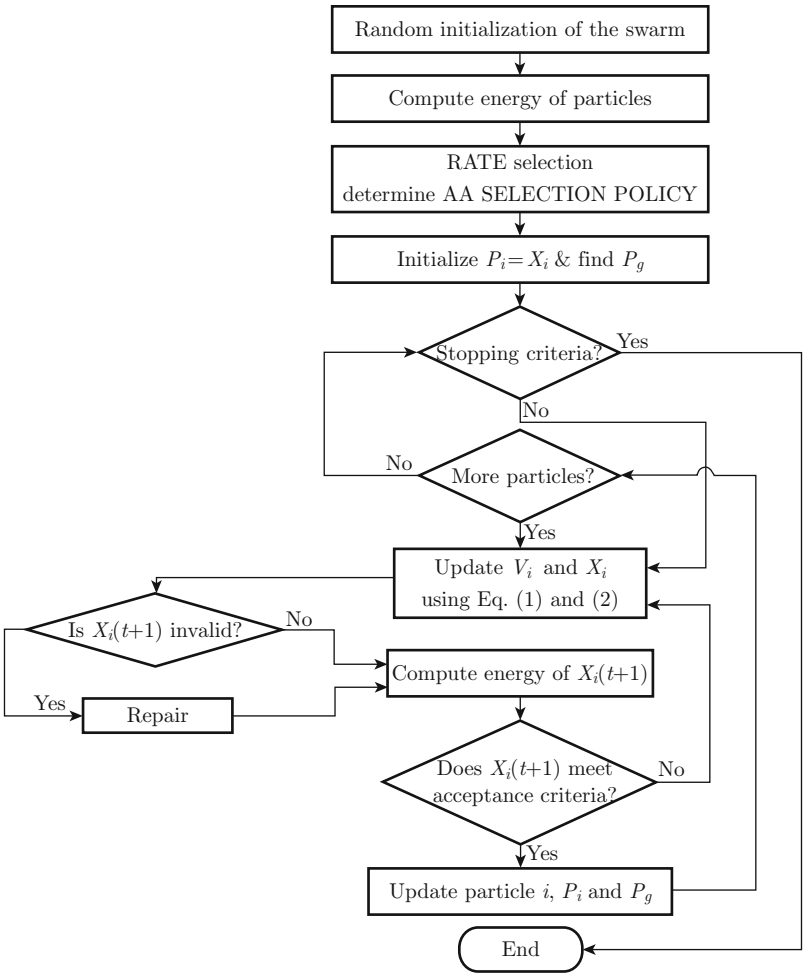

Fig. 2 PSO-based algorithm for PSP

3.1 Solution representation

A particle is a candidate solution represented by an array of length n (with index 0, 1, ..., 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}. We note that the position of the first amino acid (at array index 0) in the protein chain is assumed to be fixed to provide a reference for the position of the other amino acids. In Fig. 3, a 3D structure sample is illustrated. This structure is represented as bbburdfulurrur, which is an array of directions of length 14 representing a protein sequence containing 15 amino acids where the first amino acid is omitted since it is the reference point. The grayish balls represent the polar amino acids whereas the black balls represent the hydrophobic ones.

Initially, the swarm is populated with a set of N candidate solutions which are randomly generated. That

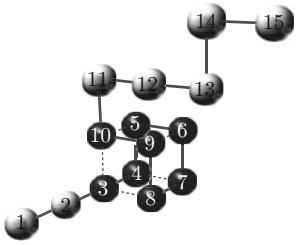

Fig. 3 A sample solution

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.

3.2 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. Fig. 4 provides an outline of this function. 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 whose locations can be modified. If more than three amino acids have been searched or if none can be modified, then it is assumed that the particle cannot be repaired and the initial input particle is returned.

```
REPAIR_PARTICLE (Particle)
1.  for every AA i in the particle, i = 1, 2, . . . ., n-1
2.    Find the candidate vacant spots with respect to the position of the preceding AA
3.    if the number of vacant spots != 0 then
4.      if current direction of AA i is marked as occupied then    //Collision
5.        Randomly choose another vacant spot
6.      endif
7.    else      //No vacant spots
8.      Search previous AA whose direction can be changed
9.      if (None is available OR number of searched AAs > 3) then
10.       Return Particle
11.     else
12.       i ← index of AA whose direction was changed    //i decremented
13.     end if
14.   end if
15. end for
16. Return Repaired_Particle
```

Fig. 4 Repair algorithm

3.3 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 (non-bonded) contacts per amino acid is four, since each amino acid has 2 bonds with the two adjacent AAs (neighbors) in the peptide chain and the maximum number of neighbors in a cubic lattice is six. The first and last amino acids might have up to five contacts, since they only have one bond with one adjacent AA. 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 Fig. 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.

3.4 Position update

The positions of the amino acids in each particle are updated using the procedure UPDATE_POSITION_V1, which employs Equations 1 and 2. In this procedure, the direction of each amino acid (AA) i in a particle, except the first one, is updated 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 dependence on the preceding AA in the peptide chain is consistent with description of the solution representation in section 3.1. 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 using Equation 1. Then, the position of the amino acid is updated, using Equation 2, along the same randomly chosen axis. 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. That is, if the position turns out to be (0.2, 0.7, 0.5) after applying Equation 2 and the randomly selected axis for this particular position update step is the z-axis, then the new position is translated to (0,0,1) with respect to the existing position, since the

z-value is positive.

3.5 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'.

3.6 Acceptance criterion of new particle and stopping criterion

In this adapted version of PSO, we used 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.

The algorithm will terminate when no improvement in the results is observed for two successive iterations.

4 Experimental results and discussion

In this section, we present the experimental results obtained upon running our PSO algorithm and compare them to those of published techniques on various date sets. One technique is reported by Patton *et al.* (1995), which proposed a standard genetic algorithm for this problem and reported better results than those achieved by Unger and Moult (1993b); the second technique is by Johnson and Katikireddy (2006), which reported better results than those achieved by Patton *et al.* for the smaller sequences. We also experimented with some variants of our PSO algorithm.

4.1 Experimental procedure

4.1.1 Data

We used two sets of benchmark sequences used first by Unger and Moult (1993b). These are amino acid sequences of Hs and Ps generated randomly: 10 sequences are of length 27 and 10 sequences of length 64 which are given in Tables 1 and 2.

4.1.2 PSO algorithm versions

In order to gain confidence about the design choices in our PSO algorithm, we have implemented a few versions of this algorithm based on different design decisions. These versions are illustrated in Table 3, where Version 1 refers to our proposed algorithm. The other versions are associated with different combinations of parameter values and design choices. Versions 2-5 are based on changing one parameter at a time, whereas Versions 6-8 involve changes in more than one parameter.

In UPDATE_POSITION_V2 used in Table 3 for some versions, we found the new velocity vector for amino acids in a particle, with a specific RATE, and its "temporary" new position using Equations 1 and 2. Then, in contrast with UPDATE_POSITION_V1, we calculated the Euclidean distance from this temporary position to the vacant candidate spots in the lattice around the preceding amino acid. The new position of the current amino acid will be the one with the minimum Euclidean distance. The 'Any' policy used in some versions for accepting a newly generated particle to replace an incumbent particle refers to always accepting the new particle regardless of its energy function value.

4.1.3 Metrics used

We evaluated 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 algorithms.
- 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.

4.1.4 Parameters

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

- Inertia (ω): It is typically set between 0.4 and 0.9 (Wilke, 2005). For all of our versions, we set it to 0.5.
- Self confidence (c_1) and swarm confidence (c_2) values were set to 2 (Eberhart and Kennedy, 1995).
- r_1 and r_2: are real random numbers which can range between 0 and 1 (Das *et al.*, 2008).
- 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 (Wilke, 2005). We used a swarm size of 5 for the smaller sequences and 10 for the longer sequences.
- Number of iterations: we allowed the algorithm to run for a maximum of 10000 iterations for the 27-long sequences and 40000 iterations for the 64-long sequences. However, we recorded the results at the point beyond which no improvement took place, if this occurred before reaching the maximum number of iterations.

Table 1 Benchmark sequences of length 27

Seq #	Sequence	Lowest Energy in Patton *et al.* (1995) & Johnson and Katikireddy (2006)
273d.1	phphphhhpphphpppppppppphhp	−9
273d.2	phhpppppppppphhpphhpphpphph	−10
273d.3	hhhhpppppphpppppphhhpppppppph	−8
273d.4	hhhpphhhhpppphphpphhpphpphh	−15
273d.5	hhhhpppphphhpppphhpppppppppp	−8
273d.6	hpppppphphhhpphhpppphpppphph	−11
273d.7	hpphphhpppphpppppphhhphphphh	−13
273d.8	hpppppppppppphphpppppppphphh	−4
273d.9	pppppppphhhpppphphpppphpphppp	−7
273d.10	pppppphhphphphphpphhphhphppp	−11

Table 2 Benchmark sequences of length 64

Seq #	Sequence	Lowest Energy in Patton *et al.* (1995)
643d.1	pphhhhhpppphhpppppphhpppphpppppphphpppphpphpphpppppphpppphhphhpphpphp	−27
643d.2	pphphpphpphhhphhhhpphhhpppphphpppphphpppphphpppppphphpphphpppphpphpp	−30
643d.3	hphhpphhphpppppphhhphhhhpphpphphhpppphphpphhhphhphpppppphhhhhhhhppp	−38
643d.4	hpphhpphpphphpphpppphpppppphphphhhpphphpppphphpphhpphpphpphphhhph	−34
643d.5	hppphhpphphpppphpppphphhpppphhphphhphpphpppphpphphhhpphpphpphhhphhhh	−36
643d.6	hpphhphhhhpppppphhpphpppphhpppphpphphhphpppphhpppphpppppphpppphphh	−31
643d.7	pppphpppphpppphhhphphpppppphpphphhphphpppppphpppppppppphhhhpppphhpph	−25
643d.8	ppphhhpphphpphpphhpppphpphpphphhphpppphpppppppphphhhphhhhhpphhpppphpph	−34
643d.9	hpphppphhhpppphphpppphphhphhhhhpppphphphpppphphpppphhphpppphpphhphp	−33
643d.10	pphpphpphhhpppphphpphpphpppppphpphhhpphpphpphphpppppphhhpppppphphp	−26

Table 3 Algorithm versions

Version #	Update_Position Function	RATE	Acceptance Criterion of New Particle	AA Selection Policy
Version 1	UPDATE_POSITION_V1	0.1	Greedy	Sequential
Version 2	UPDATE_POSITION_V2	0.1	Greedy	Sequential
Version 3	UPDATE_POSITION_V1	1	Greedy	Sequential
Version 4	UPDATE_POSITION_V1	0.1	Any	Sequential
Version 5	UPDATE_POSITION_V1	0.1	Greedy	Random
Version 6	UPDATE_POSITION_V2	1	Greedy	Sequential
Version 7	UPDATE_POSITION_V2	1	Any	Sequential
Version 8	UPDATE_POSITION_V2	0.9	Any	Random

4.2 Results

Tables 4 and 5 present the results of the PSO algorithm versions and also include the previously published results, for proteins with lengths 27 and 64 amino acids, respectively. Tables 6 and 7 provide a summary of the results of PSO algorithm Version 1 with the best previously published results.

Based on Tables 4 and 5, we make the following comments:

a. PSO Version 2 yields the same energy values as PSO Version 1 for the short sequences in 9 out of 10 cases, where the remaining case, 273d.6, has a higher energy value. For these sequences, the number of energy evaluations is larger for 80% of the cases. However, for the long sequences, PSO Version 2 fails to compete with Version 1 on all 10 cases. These results indicate that the policy UPADTE_POSITION_V1 is more appropriate than UPDATE_POSITION_V2.

b. For PSO Version 3, a similar assessment to that of PSO Version 2 holds, except that the number of energy evaluations for the short sequences

Table 4 Results for sequences of length 27

Seq #	Johnson and Katikireddy (2006)		Version 1		Version 2		Version 3		Version 4		Version 5		Version 6	Version 7	Version 8	
	Energy	Energy Eval.	Energy	Energy Eval.	Energy	Energy Eval.	Energy	Energy Eval.	Energy	Energy Eval.	Energy	Energy Eval.	Energy	Energy	Energy	Energy Eval.
273d.1	-9	15,854	-9	3,158	-9	73,253	-9	326,547	-7	94,710	-9	4,849	-7	-6	-9	298,456
273d.2	-10	19,965	-10	5,771	-10	11,546	-10	345,290	-7	322,090	-10	4,329	-8	-7	-10	245,756
273d.3	-8	7,991	-8	2,667	-8	2,584	-8	88,369	-7	745,190	-8	3,456	-6	-5	-8	84,134
273d.4	-15	23,525	-15	8,556	-15	59,497	-15	608,494	-12	315,430	-15	9,267	-11	-10	-15	190,348
273d.5	-8	3,561	-8	893	-8	10,569	-8	161,510	-7	128,470	-8	1094	-6	-5	-8	70,876
273d.6	-11	14,733	-12	12,790	-11	21,182	-11	450,983	-8	117,970	-11	13,664	-8	-7	-11	260,564
273d.7	-13	23,112	-13	17,024	-13	5,251	-12	80,307	-10	325,430	-13	18,098	-9	-8	-13	201,543
273d.8	-4	889	-4	149	-4	734	-4	8,027	-4	12,510	-4	558	-3	-3	-4	9,700
273d.9	-7	5,418	-7	1,915	-7	3,306	-7	196,229	-6	345,230	-7	4,508	-6	-5	-7	67,900
273d.10	-11	5,592	-11	2,638	-11	12,808	-11	609,229	-9	58930	-11	3,849	-9	-7	-11	23,981

Table 5 Results for sequences of length 64

Seq #	Patton *et al.* (1995)		Version 1		Version 2	Version 3	Version 4	Version 5	Version 6	Version 7	Version 8
	Energy	Energy Eval.	Energy	Energy Eval.	Energy	Energy	Energy	Energy	Energy	Energy	Energy
643d.1	-27	433,533	-28	1,131,552	-24	-18	-14	-27	-18	-12	-21
643d.2	-30	167,017	-31	456,877	-26	-18	-16	-30	-21	-13	-24
643d.3	-38	172,192	-39	113,315	-36	-24	-24	-39	-27	-14	-30
643d.4	-34	107,143	-36	1,730,129	-32	-25	-15	-35	-22	-10	-25
643d.5	-36	154,168	-38	1,602,646	-33	-24	-19	-36	-24	-15	-26
643d.6	-31	454,727	-31	410,586	-27	-21	-18	-31	-19	-12	-23
643d.7	-25	320,396	-27	1,296,319	-24	-17	-14	-26	-16	-11	-17
643d.8	-34	315,036	-35	1,113,330	-29	-19	-18	-34	-22	-14	-33
643d.9	-33	151,705	-35	404,199	-27	-24	-17	-35	-20	-12	-30
643d.10	-26	191,019	-27	175,053	-26	-17	-14	-26	-16	-8	-24

Table 6 Results of Version 1 compared to Johnson et al. (2006) (Seqs. of length 27)

Seq #	Johnson and Katikireddy (2006)		Version 1			Difference in Energy	Relative % Eval. Used
	Energy	Energy Eval.	Energy	Energy Eval.	Time (Sec)		
273d.1	-9	15,854	-9	3,158	7	0	19.91
273d.2	-10	19,965	-10	5,771	10	0	28.9
273d.3	-8	7,991	-8	2,667	7	0	33.37
273d.4	-15	23,525	-15	8,556	14	0	36.36
273d.5	-8	3,561	-8	893	2	0	25.07
273d.6	-11	14,733	-12	12,790	22	-1	86.81
273d.7	-13	23,112	-13	17,024	28	0	73.65
273d.8	-4	889	-4	149	0.5	0	16.76
273d.9	-7	5,418	-7	1,915	5	0	35.34
273d.10	-11	5,592	-11	2,638	8	0	47.17

Table 7 Results of Version 1 compared to Patton et al. (1995) (Seqs. of length 64)

Seq #	Patton *et al.* (1995)		Version 1		Version 1			Difference in Energy	Relative % Eval. Used
	Energy	Energy Eval.	Energy	Energy Eval.	Energy	Energy Eval.	Time (min)		
643d.1	-27	433,533	-27	422,373	-28	1,131,552	12	-1	261
643d.2	-30	167,017	-30	159,873	-31	456,877	5	-1	273.55
643d.3	-38	172,192	-38	109,541	-39	113,315	1	-1	65.8
643d.4	-34	107,143	-34	167,879	-36	1,730,129	18	-2	1614.78
643d.5	-36	154,168	-36	189,634	-38	1,602,646	17	-2	1039.54
643d.6	-31	454,727	-31	410,586	-31	410,586	5	0	90.29
643d.7	-25	320,396	-25	309,532	-27	1,296,319	3	-2	404.59
643d.8	-34	315,036	-34	410,813	-35	1,113,330	12	-1	353.39
643d.9	-33	151,705	-33	143,182	-35	404,199	4	-2	266.43
643d.10	-26	191,019	-26	165,762	-27	175,053	2	-1	91.65

is significantly larger. This indicates that the low/moderate RATE values for the number of AAs of a particle that are updated per iteration are more appropriate than high rates.

c. PSO Version 4 fails badly in comparison with PSO Version 1 on both the energy values and the number of energy evaluations. This demonstrates the merit of the greedy policy for accepting newly generated particles.

d. PSO Version 5 yields fairly comparable results to PSO Version 1 for the short sequences in terms of energy values and number of energy evaluations. For the larger sequences, it produces 7 out of 10 worse energy values. This indicates that the sequential policy of selecting AA for updating is more favorable than the random selection policy.

e. PSO Version 6 and 7 yield much worse energy values than those of PSO Version 1 for short and long sequences. PSO Version 8 also fails badly on the long sequences although it gives comparable energy values for the short sequences, for a much larger number of energy evaluations. This shows that the combinations of parameter values employed in these versions are not appropriate.

f. The results of Versions 3 and 6 show that UPDATE_POSITION_V1 and UPDATE_POSITION_V2 do not lead to the lowest energy solutions if applied on 100% of the amino acids (RATE = 1). That is, regardless of the function used to update the position, we note that when the rate of modifying the directions is high, the quality of the results becomes lower.

g. Based on the observations made in parts a-f, the PSO Version 1 produces better solutions than all other versions. Version 1 represents a combination of appropriate choices which improve the results. In this version, we are keeping 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.

Based on Tables 6 and 7, we infer the following:

h. For the 27-long sequences, the energies recorded by PSO Version 1 for all sequences, except for sequence 273d.6, were equal to those of Johnson *et al.* (2006) but with a fewer number of energy evaluations. For sequence 273d.6, PSO Version 1 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.

i. For the 64-long sequences, PSO Version 1 found lower energy values than Patton *et al.* (1995) for 9 out of the 10 sequences, although for a higher number of energy evaluations. However, for finding the same energy values, PSO Version 1 runs for a comparable number of energy evaluations.

Therefore, PSO Version 1 produces better results than those of the published algorithms, 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, our proposed algorithm 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 7, the PSO algorithm (Version 1) performs a larger number of energy evaluations.

5 Conclusion

We have presented a PSO based algorithm for solving the PSP problem in the 3D HP model. Given a sequence of Hs and Ps, where H represents a hydrophobic amino acid and P represents a polar one, we aim to find the 3D structure, which is a map of the given sequence into a 3D lattice characterized with a hydrophobic core, such that the number of H-H contacts is maximized.

Our proposed PSO algorithm (Version 1) efficiently explores the search space of possible solutions and returns the 3D structure with low energy. It starts with a set of randomly created potential solutions or particles gathered in a swarm. These particles are evaluated for their energy function values. At every iteration, this swarm is updated using a function that updates the velocity of the particle, which is the main operator of the algorithm. This operator's task is to explore new areas of the search space to find the optimal solutions. The performance of our algorithm is evaluated by comparing it to the results of previous algorithms using the same set of benchmark sequences. Our PSO algorithm has been shown to produce better results than these published results by reaching the same energy values with a fewer number of energy evaluations for the small sequences and by finding lower energy structures for the longer ones.

Further work can be done. The 3D model can be extended to more complex geometric shapes that represent more similarity to the real proteins like the face centered cubic lattice. This might allow us to use real proteins as our HP sequences and study the effects of hydrophibicity on the protein folding kinetics. Moreover, the energy function can be extended by taking into consideration the H-P and the H-Solvent contacts to study the effects of such integration on the overall performance of the algorithm (Custódio *et al.*, 2004). Furthermore, a parallel PSO might further improve the results for the longer sequences by providing a faster

and more efficient means to explore the search space.

Acknowledgements This work was partially supported by the Lebanese American University and the National Council for Scientific Research. The initial idea of this work was developed during a visit of the first author to the Centre for Distributed and High Performance Computing, School of IT, University of Sydney. We thank the anonymous referees whose comments improved the presentation of the paper.

References

[1] Anfinsen, C.B. 1973. Principles that govern the folding of proteins. Science 181, 223–230.

[2] Bastolla, U., Fravenkron, H., Gestner, E., Grassberger, P., Nadler, W. 1998. Testing a new Monte Carlo algorithm for the protein folding problem. Proteins 32, 52–66.

[3] Bui, T.N., Sundarraj, G. 2005. An efficient genetic algorithm for predicting protein tertiary structures in the 2D HP model. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, USA, 385–392.

[4] Chen, M., Huang, W. 2005. A branch and bound algorithm for the protein folding problem in the HP lattice model. Genomics Proteomics Bioinf 3, 225–230.

[5] Cui, Y., Chen, R.S., Wong, W.H. 1998. Protein folding simulation with genetic algorithm and supersecondary structure constraints. Proteins 31, 247–257.

[6] Custódio, F., Barbosa, H., Dardenne, L. 2004. Investigation of the three-dimensional lattice HP protein folding model using a genetic algorithm. Genet Mol Biol 27, 611–615.

[7] Das, S., Abraham, A., Konar, A. 2008. Swarm intelligence algorithms in bioinformatics. In: Studies in Computational Intelligence. Springer, Berlin, 113–147.

[8] Das, R., Baker, D. 2008. Macromolecular modeling with Rosetta. Annu Rev Biochem 77, 363–382.

[9] Datta, A., Talukdar, V., Konar, A., Jain, L.C. 2008. Neuro-swarm hybridization for protein tertiary structure prediction. Int J Hybrid Intell Syst 5, 153–159.

[10] Floudas, C. 2007. Computational methods in protein structure prediction. Biotechnol Bioeng 97, 207–213.

[11] Hart, W.E., Newman, A. 2006. Protein structure prediction with lattice models. In: Aluru, S. (Ed.) Handbook of Molecular Biology, CRC Press, New York, 1–24.

[12] Hsu, H.P., Mehra, V., Nadler, W., Grassberger, P. 2003. Growth algorithm for lattice heteropolymers at low temperatures. J Chem Phys 118, 444–451.

[13] Johnson, C., Katikireddy, A. 2006. A genetic algorithm with backtracking for protein structure prediction. In: Proceedings of the 8^{th} Annual Conference on Genetic and Evolutionary Computation, USA, 299–300.

[14] Jones, D.T. 1998. THREADER: Protein sequence threading by double dynamic programming. In: Salzberg, S., Searl, D., Kasif, S. (Eds.) Computational Methods in Molecular Biology, Elsevier Science, Amsterdam, 285-312.

[15] Kennedy, J., Eberhart, R.C. 1995. Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference of Neural Networks, Australia, 1942–1948.

[16] Klepeis, J.L., Floudas, C.A. 2003. Ab initio tertiary structure prediction of proteins. J Global Optim 25, 113–140.

[17] Kopp, J., Schwede, T. 2004. Automated protein structure homology modeling: A progress report. Pharm J 5, 405–416.

[18] Lathrop, R.H., Rogers, R.G. Jr., Bienkowska, J., Bryant, B.K.M., Buturovic, L.J., Gaitatzes, C., Nambudripad, R., White, J.V., Smith, T.F. 1998. Analysis and algorithms for protein sequence-structure alignment. In: Salzberg, S.L., Searls, D.B., Kasif, S. (Eds.) Computational Methods in Molecular Biology, Elsevier Science, Amsterdam, 227–283.

[19] Liang, F., Wong, W.H. 2001. Evolutionary Monte Carlo for protein folding simulations. J Chem Phys 115, 3374–3380.

[20] Mansour, N., Kehyayan, C., Khachfe, H. 2009. Scatter search algorithm for protein structure prediction. Int J Bioinf Res Appl 5, 501–515.

[21] Pandit, S.B., Zhang, Y., Skolnick, J. 2006. Tasser-lite: An automated tool for protein comparative modeling. Biophys J 91, 4180–4190.

[22] Patton, A.L., Punch, W.F., Goodman, E.D. 1995. A standard GA approach to native protein conformation prediction. In: Proceedings of the Sixth International Conference on Genetic Algorithms, USA, 574–581.

[23] Prusiner, S.B. 1998. Prions. Proc Natl Acad Sci USA 95, 13363–13383.

[24] Roy, A., Kucukural, A., Zhang, Y. 2010. I-TASSER: A unified platform for automated protein structure and function prediction. Nat Protoc 5, 725–738.

[25] Rylance, G. 2004. Applications of genetic algorithms in protein folding studies. First year report, School of Chemistry, University of Birmingham, England.

[26] Schulze-Kremer, S. 2000. Genetic algorithms and protein folding. Methods Mol Biol 143, 175–222.

[27] Shmygelska, A., Hoos, H.H. 2005. An Ant Colony optimization algorithm for the 2D and 3D hydrophobic polar protein folding problem. BMC Bioinf 6, 30.

[28] Sikder, A.R., Zomaya, A.Y. 2005. An Overview of protein-folding techniques: issues and perspectives. Int J Bioinf Res Appl 1, 121–143.

[29] Toma, L., Toma, S. 1996. Contact interactions method: A new algorithm for protein folding simulations. Protein Sci 5, 147–153.

[30] Unger, R., Moult, J. 1993a. Finding the lowest free energy conformation of a protein is an NP-Hard problem: Proof and implications. Bull Math Biol 55, 1183–1198.

[31] Unger, R., Moult, J. 1993b. Genetic algorithms for protein folding simulations. J Mol Biol 231, 75–81.

[32] Wilke, D.N. 2005. Analysis of the Particle Swarm Optimization Algorithm. Master dissertation, University of Pretoria.

[33] Yue, K., Dill, K.A. 1995. Forces of tertiary structural organization in globular proteins. Proc Natl Acad Sci USA 92, 146–150.

[34] Zhang, X., Li, T. 2007. Improved particle swarm optimization algorithm for 2D protein folding prediction. In: Proceedings of the 1st International Conference on Bioinformatics and Biomedical Engineering, China, 53–56.