Applied Soft Computing 26 (2015) 213–223

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A novel state space representation for the solution of 2D-HP protein folding problem using reinforcement learning methods

Berat Doğan*, Tamer Ölmez

Istanbul Technical University, Department of Electronics and Communication Engineering, Turkey

ARTICLE INFO

Article history:
Received 22 August 2013
Received in revised form 13 June 2014
Accepted 16 September 2014
Available online 17 October 2014

Keywords:
Reinforcement learning
Q-learning
Ant colony optimization
Protein folding
2D-HP model

ABSTRACT

In this study, a new state space representation of the protein folding problem for the use of reinforcement learning methods is proposed. In the existing studies, the way of defining the state-action space prevents the agent to learn the state space for any amino-acid sequence, but rather, the defined state-action space is valid for only a particular amino-acid sequence. Moreover, in the existing methods, the size of the state space is strictly depends on the amino-acid sequence length. The newly proposed state-action space reduces this dependency and allows the agent to find the optimal fold of any sequence of a certain length. Additionally, by utilizing an ant based reinforcement learning algorithm, the Ant-Q algorithm, optimum fold of a protein is found rapidly when compared to the standard Q-learning algorithm. Experiments showed that, the new state-action space with the ant based reinforcement learning method is much more suited for the protein folding problem in two dimensional lattice model.

1. Introduction

The protein folding problem is a widely studied optimization problem which is known to be NP-complete. Once the proteins are synthesized, they fold a unique three-dimensional structure that makes them functional or biologically active. The mechanism behind the folding process is still unknown, but there are some mathematical models proposed to simulate the folding process and to find the correct fold of a protein from its amino-acid sequence. Perhaps the most widely studied model is the hydrophobic-polar (HP) lattice model, which is firstly proposed by Dill [1]. In this model, each amino-acid is treated either hydrophobic (H) or polar (P) and represented as a point on a two or three dimensional lattice structure.

Lattices are grid like structures that guide the algorithms to form self-avoiding protein configurations, in which each amino-acid in the sequence is mapped to only a particular point on the grid. This mapping process is usually handled in two different ways. In the first one, the amino-acid sequence is considered as a constant chain and folding is performed by iteratively modifying the positions of each amino-acid on the grid without breaking this chain. While in the second one, each amino-acid in the sequence is consecutively added to form continuous and self-avoiding amino-acid chains on the grid which can be considered as a navigation problem or a robot path planning problem.

It is shown that, reinforcement learning methods perform well on the solution of the robot path planning problems [2,3]. Thus, in this study the reinforcement learning methods are used for the solution of the protein-folding problem in two dimensional lattice model. There exist many studies [4–9] in literature that proposed different methods for the solution of this problem, but the use of reinforcement learning methods are quite new. In [10–13], authors used the Q-learning algorithm to solve the protein folding problem in two dimensional hydrophobic-polar (2D-HP) model.

In order to use the reinforcement learning methods for the solution of the protein folding problem in 2D-HP model, first a state-action space should be defined properly. Thus, each move of the agent on the grid could be easily mapped into the defined state-action space.

In the existing studies [10–13], a state-action space is defined for this purpose. However, in these studies the size of the defined state-action space is highly affected by the amino-acid sequence length. As the amino-acid sequence length increases, the size of the proposed state-action space also increases, dramatically. So, even for the small sized amino-acid sequences it is not computationally possible to create the state-action space at the beginning of the algorithm. The only way is to create the state-action space dynamically during the learning process, which is not desirable. Moreover, in these studies the state-action space is created for all amino-acid sequences individually. So, all amino-acid sequences have a unique state-action space and the algorithm must learn all

* Corresponding author. Tel.: +90 0212 2853645.
E-mail addresses: bdogan@itu.edu.tr (B. Doğan), olmezt@itu.edu.tr (T. Ölmez).

http://dx.doi.org/10.1016/j.asoc.2014.09.047
1568-4946/

of these state-action spaces separately. By this way, after a learning process, the proposed method could not be able to find the optimal fold of another amino-acid sequence, which conflicts with the philosophy of the term "learning".

In this study, to overcome above mentioned drawbacks a new state-action space is proposed. The proposed state-action space allows the agent to find the optimal fold of any amino-acid sequence (protein) with a particular length. This is achieved by incorporating the "learning" concept to the 2D HP protein folding problem by using the newly proposed state-action space. Moreover, by utilizing a swarm based reinforcement method (Ant-Q algorithm) the optimal fold is found rapidly when compared to the traditional Q-learning algorithm.

The remaining part of this paper is organized as follows; in the following section the protein folding problem in 2D-HP model is introduced. In Section 3, existing state space representation of the protein folding problem in 2D-HP model is given. Then, the newly proposed state-action space is introduced. In Section 4, reinforcement learning algorithms, the Q-learning algorithm and the Ant-Q algorithm is given. Section 5 covers the experiments, results and discussions for the proposed method. Finally, Section 6 concludes the work.

2. The protein folding problem in two dimensional hydrophobic-polar model

An amino acid sequence (or chain) is known to be the primary structure of a protein which is synthesized by using the information encoded in genes. This primary structure is then folded into a unique three dimensional structure which makes the protein functional. In literature, the challenge of inferring this three dimensional structure (tertiary structure) from the amino acid sequence is known as the "protein folding problem". Since the discovery of the proteins three dimensional structures can provide important clues about the functionalities of the proteins, the protein folding problem is of crucial importance to the biological community.

There are two main methods to experimentally determine the protein three dimensional structure; X-ray crystallography and NMR spectroscopy, both of which can provide information at atomic resolution. Unfortunately, there exist classes of proteins for which three dimensional structure reconstruction is not possible by using these experimental methods. Moreover, these experimental methods are very expensive and it is usually very time consuming to obtain the 3D structure of a protein by using these methods. For these reasons, computational methods are proposed to find out the three dimensional structure of the proteins from their amino acid sequences. However, it is also a problematic task to find the optimum fold a sequence computationally. Because, the number of possible protein conformations dramatically increases with the increased amino acid sequence length. Therefore, the proposed computational methods should explore the search space efficiently in a reasonable time. In order to achieve this, in computational methods the protein folding process is modeled with some mathematical energy functions. It is thought that, the three dimensional structure of a protein is its native state, which has the lowest possible energy conformation. The problem is thus evolved to find out the lowest energy conformation by minimizing these energy functions.

Perhaps the most widely studied model is the hydrophobic–hydrophilic (HP) lattice model in both two and three dimensions, which is firstly proposed by Dill [1]. In this model, each amino acid side chain is classified either as 'H' hydrophobic (repelled by water) or 'P' hydrophilic or polar (liking water). Dill's survey of proteins identified the interactions between hydrophobic residues (amino acids) to be the dominant force in protein folding [14].

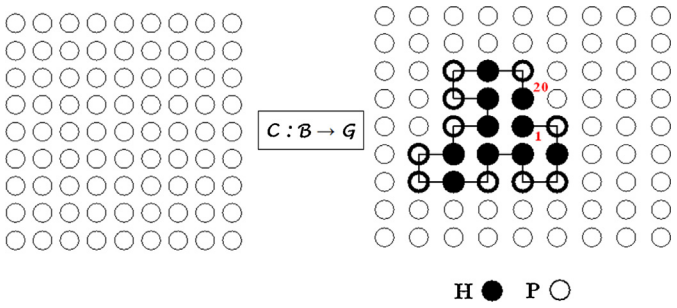

Fig. 1. A sample configuration with energy −9 for the protein. $\mathcal{P}$ = HPHPPHHPHPPHPHHPPHPH after mapping process.

Let us, define the primary structure of a protein consists of n amino acid as $\mathcal{P}$. In 2D-HP lattice model this protein could be mathematically defined as below;

$$\mathcal{P} = p_1 p_2 p_3 \ldots . p_n, \quad p_i \in \{H, P\}, \quad \forall 1 \le i \le n \tag{1}$$

Here, $p_i \in \{H, P\}$ represents each amino acid in the chain which are either hydrophobic or hydrophilic (polar). A valid protein structure is defined with a function $\mathcal{C}$, such that each residue of the amino acid chain is mapped to the lattice points in Cartesian coordinates by this function. This could be mathematically defined as in Eq. (2).

$$\begin{aligned} &\mathcal{B} = \{P = p_1 p_2 p_3 \ldots . p_n | p_i \in \{H, P\}, \quad \forall 1 \le i \le n, \quad n \in N\} \\ &\mathcal{G} = \{G = (x_i, y_i) | x_i, y_i \in \Re, \quad 1 \le i \le n\} \\ &\mathcal{C} : \mathcal{B} \to \mathcal{G} \end{aligned} \tag{2}$$

Here, $\mathcal{C} : \mathcal{B} \to \mathcal{G}$ represents the mapping process of a residue $p_i \in \{H, P\}$ to a lattice point (x_i, y_i) in Cartesian coordinates. After this mapping process, for $\forall 1 \le i, j \le n$ with $|i-j| \ge 2$ the energy of the resulting protein structure in 2D-HP lattice model is defined as in Eq. (3).

$$E(C) = \sum_{i,j} I(i,j)$$

$$I(i,j) = \begin{cases} -1, & \text{if } p_i = p_j = H \text{ and } |x_i - x_j| + |y_i - y_j| = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where (x_i, y_i) represents the position of the amino acid $p_i \in \{H, P\}$ and (x_j, y_j) represents the position of the amino acid $p_j \in \{H, P\}$ in Cartesian coordinates. More clearly, the energy function is decreased by 1 for each two amino acids that are mapped by $\mathcal{C}$ on neighboring positions in the lattice, but that are not consecutive in the primary structure $\mathcal{P}$. Such two amino acids are called as topological neighbors [12]. In Fig. 1, a sample configuration with energy −9 for the protein $\mathcal{P}$ = HPHPPHHPHPPHPHHPPHPH is given.

3. State space representations for the solution of the protein folding problem in two-dimensional hydrophobic-polar lattice model

A valid protein configuration forms a self avoiding path which means, the mapped positions of two different amino acids must not be same in the 2D grid. By considering the resulting self avoiding path, a solution could be represented by an $n-1$ length sequence $\pi = \pi_1 \pi_2 \pi_3 \cdots \pi_{n-1}$, $\pi_i \in \{L, R, U, D\}$, $\forall 1 \le i \le n-1$ of directions, which encodes the relative positions of the current amino acid to the previous one. Let us consider the configuration given in Fig. 1. The resulting sequence for this protein then could be given as π = *RDDLULDLLURURULURRD*.

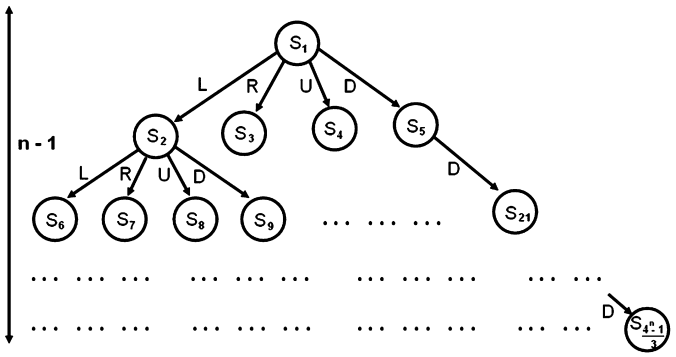

Fig. 2. State space for the reinforcement learning method given in [11].

In order to study the protein folding problem in 2D-HP lattice model with reinforcement learning methods, first a state-action space that encodes the above mentioned sequence of directions should be proposed. In the following sections, the state-action space defined in the existing studies [10–13] and the state-action space defined in this study is given, respectively. (Note that, throughout the paper same notation is used with the Ref. [11].)

3.1. The existing state space representation

The state space $\mathcal{S}$ proposed by Czibula et al. [10–13] consists of $(4^n-1)/3$ states i.e. $\mathcal{S}=\{s_1, s_2, \ldots, s_{(4^n-1)/3}\}$ which is given in Fig. 2. At the beginning, the agent is at state s_1. A state $s_{i_k} \in \mathcal{S}(i_k \in [1, (4^n-1)/3])$ is reached by the agent at a given moment after it has visited states $s_1, s_{i_1}, s_{i_2}, \ldots, s_{i_{k-1}}$ is a terminal state if the number of states visited by the agent in the current sequence is n-1 i.e. $k=n-2$. A path from the initial to the final state forms a configuration for the protein $\mathcal{P}$. In Table 1 resulting states at each step are given as an example for the protein sequence $\mathcal{P}=$ HPHPPH.

The action space $\mathcal{A}$ consists of 4 actions *L (Left)*, *R (Right)*, *U (Up)*, *D (Down)* which are the relative directions of the current position of the agent to the previous one. So, the action space could be given as $A=\{a_1, a_2, a_3, a_4\}$, where $a_1=L$, $a_2=R$, $a_3=U$ and $a_4=D$.

Table 1
An example of the existing state space representation for the sequence $\mathcal{P}=$ HPHPPH.

Agent in state	Can choose action	Resulting states
S1	L, R, U, D	S2 = L S3 = R S4 = U S5 = D
S2	L, R, U, D	S6 = LL S7 = LR S8 = LU S9 = LD
S3	L, R, U, D	S10 = RL S11 = RR S12 = RU S13 = RD
S4	L, R, U, D	S14 = UL S15 = UR S16 = UU S17 = UD
S5	L, R, U, D	S18 = DL S19 = DR S20 = DU S21 = DD
⋮	L, R, U, D	⋮ ⋮
S1361	L, R, U, D	S1362 = DDDDDL S1363 = DDDDDR S1364 = DDDDDU S1365 = DDDDDD

At a given moment, from a state $s \in \mathcal{S}$ agent can move in 4 successor states, by executing one of the 4 possible actions. Thus, the transition probability between a state s and each neighbor state s' of s is equal to 0.25.

3.1.1. Comments on the existing state space representation

In the existing method, the state space $\mathcal{S}$ (the agent's environment) consists of $(4^n-1)/3$ states i.e. $\mathcal{S}=\{s_1, s_2, \ldots, s_{(4^n-1)/3}\}$. Let us consider the sequence given in Fig. 1 which has a total number of 20 amino acids. So, for this sequence the state space $\mathcal{S}$ consists of $(4^{20}-1)/3 = 3.665 \times 10^{11}$ states. Thus, even to create the state-action space for a sequence length of 20, the required computational time is dramatically huge and it is not possible to create such a big space by an average PC. The only way is to create the state space dynamically, but there is not any information provided about this point in the above mentioned studies. To sum up, in the existing method the size of the state space $\mathcal{S}$ is strictly depends on the length of the amino acid sequence. As the length of the amino acid sequence n increases, the size of the state space also increases dramatically.

Another point that should be comment on is the learning stage of the existing method. Let us again consider the amino acid sequence given in Fig. 1. Since the state space $\mathcal{S}$ encodes the relative positions of the current amino acid to the previous one, after a learning process all we have is a sequence of directions where the optimal one is given as RDDLULDLLURURULURRD for the sequence given in Fig. 1. Here, it should be noted that at the end of the learning process there is not any information provided about the characteristics of the amino acids whether they are hydrophobic or hydrophilic. This information is totally lost at the end of the learning process. Thus, for an another protein sequence the state-action space must be re-initialized and the agent must learn the environment for this new sequence. This situation conflicts with the philosophy of the term "learning". Because in a learning process the previous information must be conserved.

3.2. The proposed state space representation

In the previous section the drawbacks of the existing reinforcement learning method are discussed. In this study, to overcome these drawbacks a new state space representation is proposed. Unlike the existing one, the proposed state-action space comprises the characteristics of the amino acids. By this way, the information stored in the state-action space is conserved. So, there is no need to re-initialize the state-action space for another amino acid sequence.

This section mainly covers the definition of the proposed state action space. To allow a comparison to the existing method the proposed state-action space is studied in two different scenarios.

Scenario 1. In the first scenario, the agent tries to find the optimal policy for only a particular amino acid sequence which is also the case in the above mentioned existing method. In Fig. 3 the proposed state-action space is given for this case. As it can be shown in Fig. 3, the new state-action space has a matrix like structure in which each column represents an element of the amino acid sequence that is hydrophobic or polar. Again there are four possible directions $\{L, R, U, D\}$ that agent can move when it is at a state s.

In this case, the total number of states $\mathcal{S}$ consists of only $[4\cdot(n-1)+1]$ states for a n length amino acid sequence. Let us again consider the amino acid sequence $\mathcal{P}_1=$ HPHPPHHPHPPHPHHPPHPH given in Fig. 1. The total number of states is $[4\cdot(20-1)+1]=77$ states, which is very small when compared to the existing one (3.665×10^{11}). For better

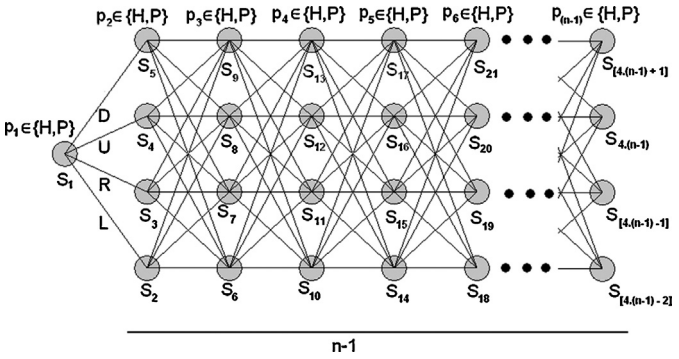

Fig. 3. The proposed state space for Scenario 1.

Table 2
State-action space for the sequence $\mathcal{P}_2$ = HPHPPH (Scenario 1).

Agent in state	Can choose action	Resulting states
S1 = H	L, R, U, D	S2 = HPL S3 = HPR S4 = HPU S5 = HPD
S2 = HPL S3 = HPR S4 = HPU S5 = HPD	L, R, U, D	S6 = HPHL S7 = HPHR S8 = HPHU S9 = HPHD
S6 = HPHL S7 = HPHR S8 = HPHU S9 = HPHD	L, R, U, D	S10 = HPHPL S11 = HPHPR S12 = HPHPU S13 = HPHPD
S10 = HPHPL S11 = HPHPR S12 = HPHPU S13 = HPHPD	L, R, U, D	S14 = HPHPPL S15 = HPHPPR S16 = HPHPPU S17 = HPHPPD
S14 = HPHPPL S15 = HPHPPR S16 = HPHPPU S17 = HPHPPD	L, R, U, D	S18 = HPHPPHL S19 = HPHPPHR S20 = HPHPPHU S21 = HPHPPHD

understanding in Table 2 all of the state action pairs are given for a short sequence $\mathcal{P}_2$ = HPHPPH.

Note that, as in the existing method here also it is not possible to talk about an actual "learning". Because, the agent learns the space for only the corresponding amino acid sequence and thus, the Q-table only consists of the state-action pairs for this individual amino acid sequence. However, when compared to the existing method, the new state-action space still has advantages. First, the size of the state-action space is reduced dramatically which allows creating the Q-table at the beginning of the algorithm. For the second advantage let us consider the sequence $\mathcal{P}_2$ = HPHPPH. Since $\mathcal{P}_2 \subset \mathcal{P}_1$, there is no need to learn the space for the $\mathcal{P}_2$ again. The Q-table created for the sequence $\mathcal{P}_1$ already comprises the solution for the $\mathcal{P}_2$. But since the existing method only encodes the directions, it is not possible to deduce whether $\mathcal{P}_1$ comprises $\mathcal{P}_2$ or not.

Scenario 2. In the previous scenario, only a particular amino acid sequence is considered that is why the Q-table only consists of the state-action pairs for the corresponding sequence and hence the subsets of this sequence. However, to talk about an actual learning the state-action space should comprise all of the possible combinations of an n length amino acid sequence. In Fig. 4 the proposed state-action space for this case is given. Since an amino acid could be either H or P, for a n length amino acid sequence there are 2^n different sequences. Thus, the state-action space should be designed to allow an agent to learn all of these sequences. So, after a learning process an agent could find the optimal fold of a sequence with the help of resulting Q-table which covers all of the state-action

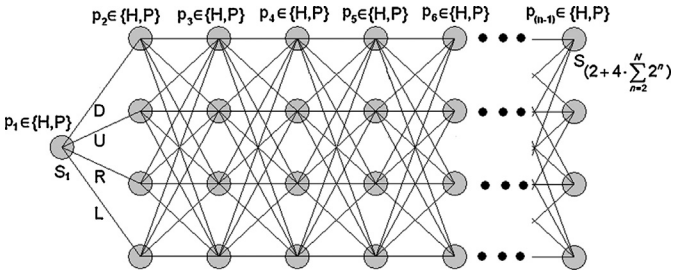

Fig. 4. The proposed state space for Scenario 2.

Table 3
State-action space for n = 3 (Scenario 2).

Agent in state	Can choose action	Resulting states
S1 = H or P (**let us consider S1 = H**. Note that, all of this process must be repeated for **S1 = P**)	L, R, U, D	**if p2 = H** S2 = HHL S3 = HHR S4 = HHU S5 = HHD **if p2 = P** S6 = HPL S7 = HPR S8 = HPU S9 = HPD
S2 = HHL S3 = HHR S4 = HHU S5 = HHD	L, R, U, D	**if p3 = H** S10 = HHHL S11 = HHHR S12 = HHHU S13 = HHHD **if p3 = P** S14 = HHPL S15 = HHPR S16 = HHPU S17 = HHPD
S6 = HPL S7 = HPR S8 = HPU S9 = HPD	L, R, U, D	**if p3 = H** S18 = HPHL S19 = HPHR S20 = HPHU S21 = HPHD **if p3 = P** S22 = HPPL S23 = HPPR S24 = HPPU S25 = HPPD

pairs. In Table 3 the state-action space for n = 3 is given for better understanding.

Note that, in Table 3 the total number of states is 25. But this number must be doubled because the initial state S1 is considered as H but it could also be P. So, in total for n = 3 there are 50 states. In this case, the total number of states is given as $\mathcal{S} = \left(2 + 4 \cdot \sum_{n=2}^{N} 2^n\right)$, where N represents the length of the amino acid sequence. When compared to the existing method (in which total number of states for n = 3 is $(4^3 - 1)/3$ =21) for n = 3 the size of the proposed state-action space is bigger. However, as n increases the total number of states becomes much smaller than the existing one. For example for n = 20 the size of the proposed state-action space is $\mathcal{S} = \left(2 + 4 \cdot \sum_{n=2}^{20} 2^n\right) \approx 8.4 \times 10^6$ which is much smaller than the existing one (3.665×10^{11}).

As it can be shown, the newly proposed state-action space reduced the size of the state-action space dramatically for both scenarios. However, for Scenario 2 the size of the state-action space is still highly depends on the amino acid sequence length n. But it should be noted that, the proposed state action space allows the agent to predict the optimal fold for $n + a$ length sequences, where $a \geq 1$. If the agent learns the space for all of the n length sequences,

Table 4
A short description of the Q-learning algorithm.

```
Initialize Q(s,a)=0 for each pair of (s,a)
Repeat (for each episode)
      Select the initial state s
      Choose a from s using policy derived from Q (e.g., ε greedy)
      Repeat (for each step of the episode)
            Take action a, observe r,s'.
            Update the table entry Q(s,a) as follows
            Q(s,a)←r(s,a)+γmax_a' Q(s',a')
            s→s'
      until s is terminal
until the maximum number of episodes is reached.
```

where $Q(s,a)$ is the table entry for the corresponding state-action pair, $r(s,a)$ is the reward of the state-action pair (s,a), y is the discount factor and finally $Q(s', a')$ is the table entry for the state-action pairs those can be reached from the state s. Once the training process is completed the agent learns the environment and constructs the optimal solution by starting from the initial state. For this purpose the Greedy selection mechanism is used. In Greedy selection mechanism, the agent selects the next state according to the Q-values stored in the Q-table for each $Q(s, a)$ pairs. Thus, the agent moves from state i to a neighbor state j that have the maximum Q-value stored in the Q-table.

it will be possible to extend the space and predict the optimal fold for longer sequences. This is another important advantage of the proposed state-action space and could be studied separately.

4. Reinforcement learning algorithms

A basic reinforcement learning model consists of a set of environment states, a set of actions, rules of transitioning between states, and rules that determine the scalar reward of a transition. Having defined the state-action spaces, now the transition rules and rules that determine the reward of a transition is given. These parts could be associated to the learning stage of a reinforcement learning method.

In [10–13], transition rules and the reward of a transition is defined as below.

If the transition generates a configuration that is not valid (i.e. self-avoiding) the received reward is 0.01; the reward received after a transition to a non terminal state is small positive constant greater than 0.01 (e.g. 0.1); the reward received after a transition to a final state π_{n-1} is the minus energy of the protein $\mathcal{P}$ corresponding to the path π. These definitions are mathematically given in Eq. (4).

$$r(\pi_k|s_1, \pi_1, \pi_2, \ldots, \pi_{k-1}) = \begin{cases} 0.01 & \text{if } a_\pi \text{ is not valid} \\ -E_\pi & \text{if } k = n-1 \\ 0.1 & \text{otherwise} \end{cases} \quad (4)$$

where $r(\pi_k|s_1, \pi_1, \pi_2, \ldots, \pi_{k-1})$ represents the reward received by the agent in state π_k, after it has visited states $\pi_1, \pi_2, \ldots, \pi_{k-1}$ and E_π represents the energy of the configuration formed by the path π.

With the above defined transition rules and the reward of a transition, Czibula et al. [10–13] used the Q-learning algorithm as the reinforcement learning method to find the optimal policy.

The Q-learning algorithm is known to be performed well on the small-sized state action spaces. However, as the size of the state-action space increases random choices of the actions prevents the agent to converge the optimal solution. To avoid this problem, recently swarm based reinforcement learning methods are proposed [15–20]. In these methods, a number of agents learn concurrently by exchanging the information that they gain during the individual learning.

The performances of the swarm based reinforcement learning methods are highly affected by the way of exchanging the information among the agents. In [15], authors proposed an information exchange method based on ant colony optimization [21], which is inspired from behavior of real ants. Real ants deposit pheromone trails over the paths from their nest to the food source. Once the amount of the pheromone trail of a particular path is increased over time, this path becomes more attractive for the members of the ant colony. In [15], the same concept is used for the Q-learning algorithm, which is a widely used reinforcement learning algorithm. In this study, pheromone trails are deposited over the state-action space. Thus, the agents could avoid from random movements which helps to find the optimal policy rapidly, even for the complicated problems. In the above mentioned study, the authors also compared the performance of the proposed ant based reinforcement learning method, namely the PHE-Q method, with the Q-learning algorithm and some other swarm based reinforcement learning methods (BEST-Q, AVE-Q and PSO-Q) that they proposed earlier. The performance of the PHE-Q method is examined by applying it to two shortest path problems. After several experiments, it is observed that the PHE-Q algorithm could find better policies in a small number of episodes when compared to the other methods. In another study, an ant based reinforcement method Ant-Q is proposed for the solution of traveling salesman problem [20]. After several experiments, it is shown that the Ant-Q algorithm with delayed reinforcement is much more efficient than the other well known heuristic methods such as, Elastic Net (EN), Simulated Annealing (SA), Self Organizing Map (SOM), and Farthest Insertion (FI). Although, there are small differences between the Ant-Q and PHE-Q algorithms, in principle both algorithms use the same concept, the pheromone trails, to find out the optimal solution.

In this study, the Ant-Q algorithm [20] is used to overcome above mentioned drawbacks of the standard Q-learning algorithm. Different from the existing methods given above, the Ant-Q algorithm uses different state transition rules and rewarding mechanism. Details of the transition rules and the rewarding mechanism can be found in Section 4.2.

4.1. The Q-learning algorithm

The Q-learning algorithm is an off-policy algorithm which is proposed to optimize the solutions in Markov decision process problems. It can be proven that, under sufficient training the

Table 5
A short description of the Ant-Q algorithm.

```
Initialize AQ(s,a) = (0,1] for each pair of (s,a)
Repeat for each iteration
   Repeat for each agent
     Select the initial state s
     Repeat (for each step of the episode)
          Choose a from s by using the Eq.7
          Take action a, observe s'.
          Update the table entry AQ(s,a) as follows
          AQ(s,a) ← AQ(s,a) + γmax_a' AQ(s',a'),  (ΔAQ(s,a) = 0)
          s → s'
     until s is terminal
    end

    Repeat for each agent
     Compute the delayed reinforcement ΔAQ(s,a) by using the Eq.8
     Update the table entry AQ(s,a) as follows
     AQ(s,a) ← (1 − α)AQ(s,a) + α · ΔAQ(s,a),  (γmax_a' AQ(s',a') = 0)
    end
until the maximum number of iterations is reached.
```

algorithm converges to a close approximation of the action-value function for an arbitrary target policy. In Table 4 a short description of the Q-learning algorithm is given.

4.2. The Ant-Q algorithm

Ant-Q algorithm is proposed by Gamberdalla and Dorigo [20] for the solution of symmetric and asymmetric travelling salesman problem. The Ant-Q algorithm was inspired by work on the ant system (AS), a distributed algorithm for combinatorial optimization based on the metaphor of ant colonies proposed in [22,23].

In this study the Ant-Q algorithm is adopted to the 2D-HP protein folding problem as follows:

Let $AQ(s, a)$ be a positive real value associated to the state-action pair (s, a). The AQ-values are the Ant-Q counterpart of Q-learning Q-values and is intended to indicate how useful it is to choose action a when the agent is in state S.

Let $HE(s,a)$ be a heuristic value associated to the state-action pair (s,a) which allows an heuristic evaluation of which transitions are better. In this study, the heuristic value is determined by the exponential of the number of new H–H contacts.

Let k be an agent whose task is to find the optimum folding starting from the initial state. Associated to k there is the list $J_k(s)$ of allowed states of to be visited, where s is the current state. This list implements a kind of memory, and is used to constrain agents to find feasible configurations, that is self-avoiding. When the agent is in the state s normally there are four possible actions. But since the resulting configuration must be self-avoiding all of these four actions are not always available. Here, $J_k(s)$ stores the states which could be reached by the agent k by performing an available action when the agent in the state s.

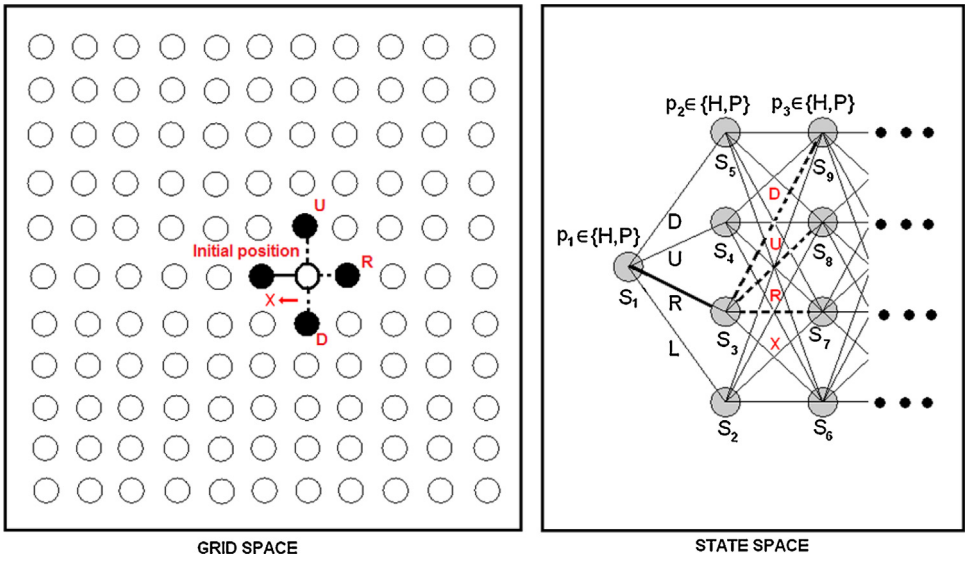

Fig. 5. Agent's move over the grid space and the corresponding state transitions.

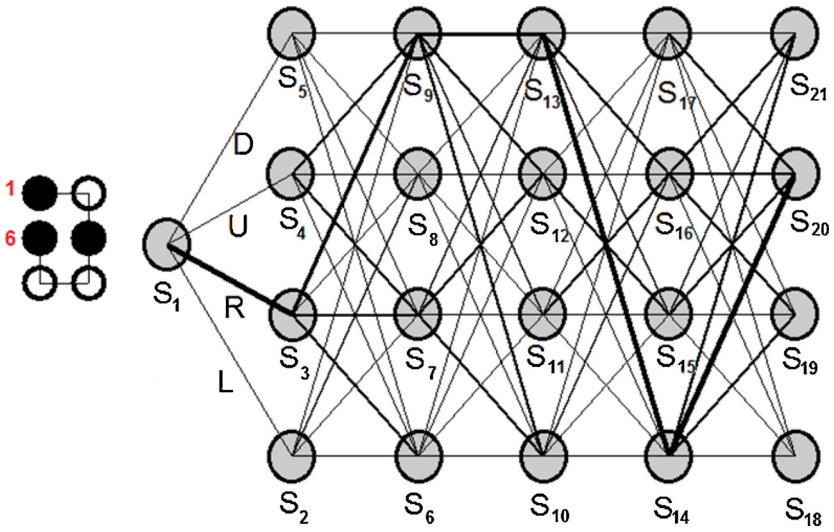

Fig. 6. Optimum fold and the resulting state-action space for $\mathcal{P}_2$ = HPHPPH.

Based on the above considerations, when the agent is in the state s an action a is chosen with Eq. (5) which is known as the action choose rule (or the state transition) rule.

$$a = \begin{cases} arg \underset{HE(s,u)}{\max} \left\{ [AQ(s,u)]^{\delta} \cdot [HE(s,u)]^{\beta} \right\} & \text{if } q \le q_0 \\ r & \text{otherwise} \end{cases} \tag{5}$$

where δ and β are parameters which weigh the relative importance of the learned AQ-values and heuristic values, q is a value chosen randomly with uniform probability in [0, 1], q_0 ($0 \le q_0 \le 1$) is a parameter such that the higher q_0 the smaller the probability to make a random choice, and r is a random variable selected according to a probability distribution given by a function of the $AQ(s,u)$'s and $HE(s,u)$'s, with $u \in J_k(s)$.

In Ant-Q algorithm m agents cooperates to find out the best solutions. Ant-Q values are updated by the following equation.

$$AQ(s,a) \leftarrow (1-\alpha) \cdot AQ(s,a) + \alpha \cdot (\Delta AQ(s,a) + \gamma \cdot \underset{z \in J_k(s)}{Max}\, AQ(s,z)) \tag{6}$$

The update term given in Eq. (6) is composed of a reinforcement and of the discounted evolution of the next state. α and γ parameters are known as the learning step and the discount factor. In Ant-System (AS) and Ant-Q algorithm, the reinforcement term ΔAQ is always zero until the agent forms a complete fold. Different from the Q-learning algorithm in Ant-Q the reinforcement term $\Delta AQ(s,a)$ is computed at the end of the agent's tour. The computation method for this delayed reinforcement is given in the following parts of this section.

In Table 5 a short description of the Ant-Q algorithm is given. In [20] a generic description of the Ant-Q algorithm also can be found. The algorithm given in [20] is called generic because in their studies Gamberdella and Dorigo tried some other action choice rules (*pseudo-random*, *pseudo-random-proportional*, and *random-proportional*) and delayed reinforcement methods (*global-best* and *iteration-best*) to further improve the performance. In [20] it is shown that the pseudo-random-proportional action choice rule with the iteration-best delayed reinforcement method performs better when compared to the other options. In Eq. (7), the pseudo-random-proportional action choice rule for the agent k is given.

$$p_k(s,a) = \begin{cases} \dfrac{[AQ(s,a)]^{\delta} \cdot [HE(s,a)]^{\beta}}{\sum_{u \in J_k(s)} [AQ(s,a)]^{\delta} \cdot [HE(s,a)]^{\beta}} & \text{if } a \in J_k(s) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $p_k(s,a)$ defines the probability of choosing an action a when the agent k is located in the state s.

As mentioned before, different from the Q-algorithm the Ant-Q algorithm uses a delayed reinforcement method. In fact, due to the nature of the protein folding problem in 2D-HP model it is much more convenient to use the delay reinforcement. As in the travelling salesman problem here also the complete tour (or configuration) defines whether the solution is good or not. Thus, instead of the immediate reward, the final configuration should be considered and rewarded. In Eq. (8) the way of computing the iteration-best delayed reinforcement is given.

$$\Delta AQ(s,a) = \begin{cases} \dfrac{E_{k_{ib}}}{L} & \text{if } (s,a) \in \text{configuration found by the agent } k_{ib} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where $E_{k_{ib}}$ represents the energy of the configuration found by the agent k and L represents the length of the amino acid sequence.

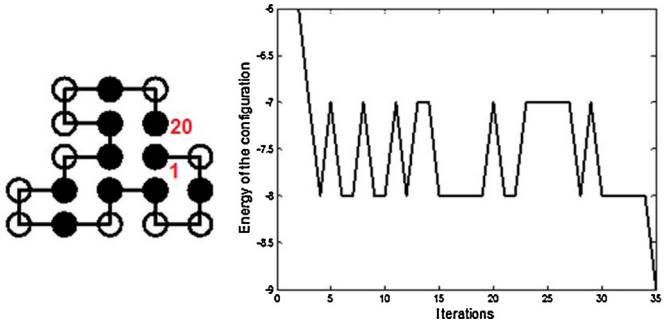

Fig. 7. Optimum configuration and corresponding fitness evaluation for the sequence $\mathcal{P}_1$ = HPHPPHHPHPPHPHHPPHPH found by Ant-Q algorithm.

Table 6
Solutions found by Ant-Q for some benchmark sequences.

Seq ID	Length	Sequence	Optimum	Ant-Q solution
Seq1	24	HHPPHPPHPPHPPHPPHPPHPPHH	−9	−9
Seq2	36	PPPHHPPHHPPPPPHHHHHHHPPHHPPPPHHPPHPP	−14	−13
Seq3	48	PPHPPHHPPHHPPPPPHHHHHHHHHHPPPPPHHPPHHPPHPPHHHHH	−23	−19
Seq4	64	HHHHHHHHHHHHPHPHPPHHPPHHPPHPPHHPPHHPPHPPHHPPHPPHHPPHPPHPHHHHHHHHHHHH	−42	−36

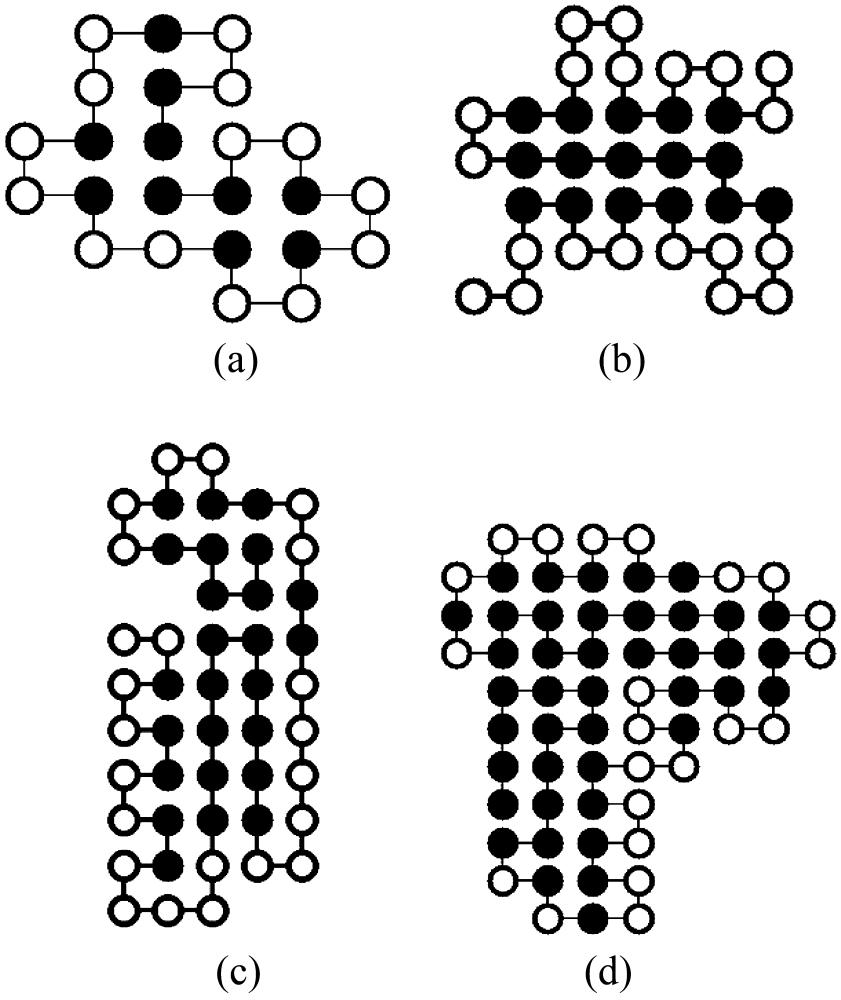

Fig. 8. Example solutions found by Ant-Q algorithm for the sequences given in Table 5: (a) Seq1; (b) Seq2; (c) Seq3; (d) Seq4.

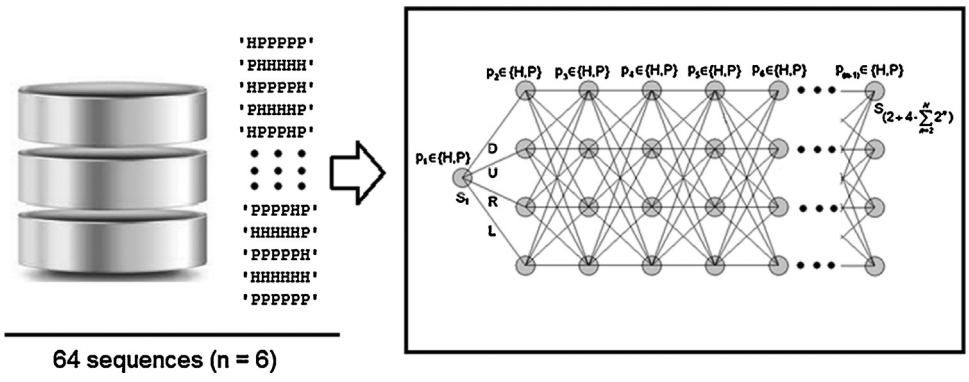

Fig. 9. Agent learns the state-action space for all of the n length sequences in Scenario 2.

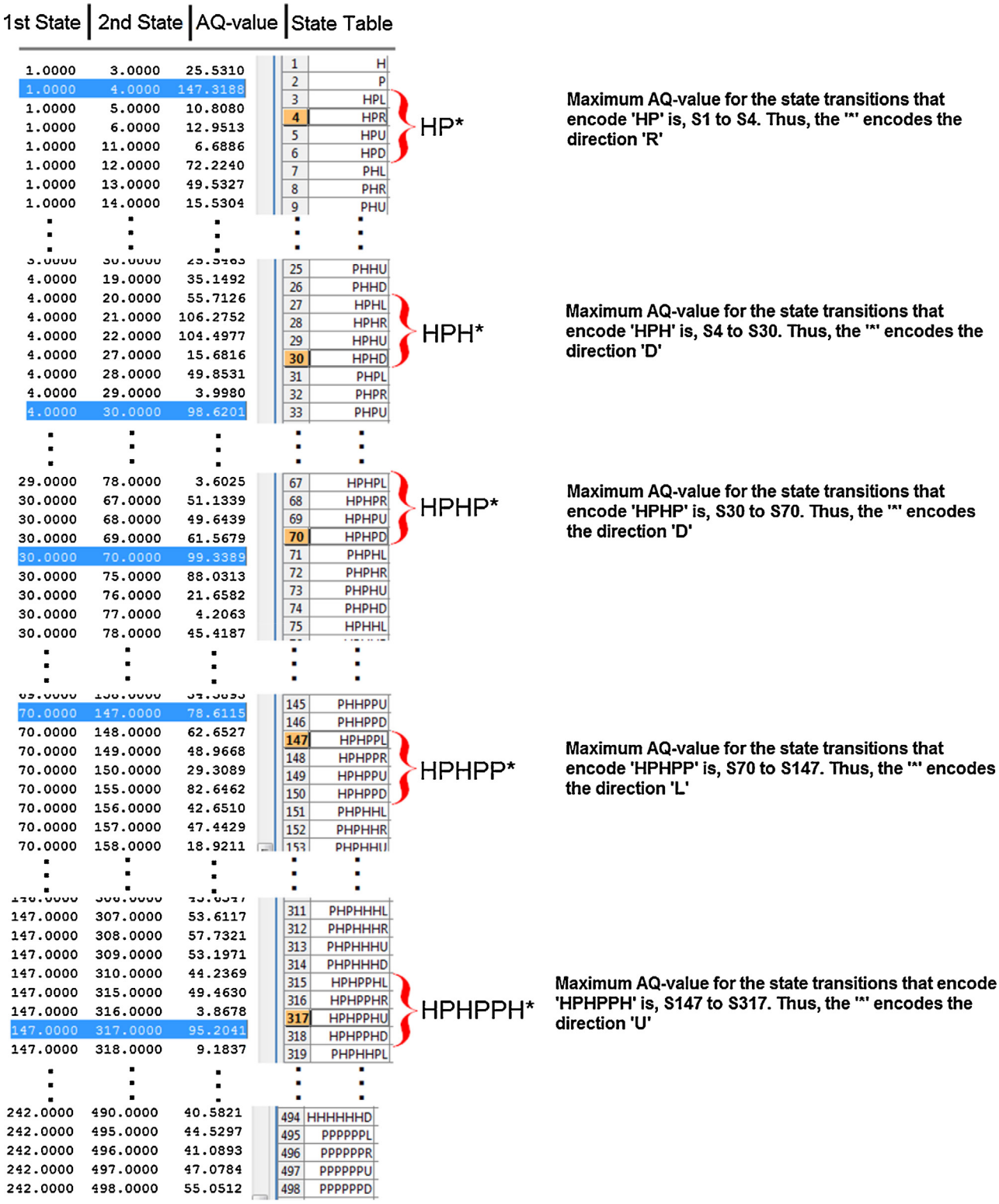

Fig. 10. After the learning process the universal AQ-table guides the agent to form the optimum configuration for the sequence $\mathcal{P}_2$ = HPHPPH. The resulting state transition chain is 1-4-30-70-147-317 which encodes the sequence of directions RDDLU as in Fig. 6.

5. Results and discussion

In this section experimental results for both scenarios are given and results are compared to the standard Q-learning algorithm.

To allow the agent to form self-avoiding configurations (to guide the agent) an MxM grid is first created and the agent is located in the center of this grid. Without loss of generality the agent's first move could be chosen to any direction and in this study the agent is first moved to 'R'. At each move of the agent the corresponding state-action pairs are found from the AQ-table and the resulting state transitions are stored. Thus, at the end of the tour the AQ-values for these transitions could be updated by using the total reward taken by the agent. In Fig. 5, a schematic representation of the agent's moves is given.

As it can be shown in Fig. 5, the state-action pairs could be further reduced. Because the agent has 3 or less (due to the self-avoiding constraint) possible moves when it is at a certain position.

Given the above grid and state spaces in Fig. 5, k agents cooperate to find optimum protein folds by using the Ant-Q algorithm with the following parameters: $\delta = 1, \beta = 2, q_0 = 1, \alpha = 0.1$, and $\gamma = 0.3$. The number of agents k is chosen to be equal to the length of corresponding amino acid sequence.

Having established the parameters for the Ant-Q algorithm, the results for the scenarios given in Section 4 is examined, respectively.

Scenario 1. As it is stated in Section 4, in the first scenario the agent tries to find the optimum fold for a particular amino acid sequence. Let us again consider the sequences $\mathcal{P}_1 =$ HPHPPHHPHPPHPHHPPHPH and $\mathcal{P}_2 =$ HPHPPH. In Fig. 6, one of the optimum configurations found by Ant-Q algorithm and the resulting state-action space for the $\mathcal{P}_2 =$ HPHPPH is given.

The edges between the states shown in Fig. 6 are weighted according to the resulting AQ-values. Thus, the state transitions for the optimum fold of amino acid sequence $\mathcal{P}_2 =$ HPHPPH could be given as S1-S3-S9-S13-S14-S20. From Table 2 it could be shown that the resulting sequence of directions is RDDLU for the amino acid sequence $\mathcal{P}_2 =$ HPHPPH as verified in Fig. 6. Note that, the resulting sequence of directions is not directly encoded as in the existing method but it is combined with the characteristics of the amino acids. Thus, for the longer sequence of $\mathcal{P}_1 =$ HPHPPHHPHPPHPHHPPHPH (where $\mathcal{P}_2 \subset \mathcal{P}_1$) a part of the state action space is already learned by the agent.

The Ant-Q algorithm is able to find the optimum configuration in nine out of ten experiments for the sequence $\mathcal{P}_1 =$ HPHPPHHPHPPHPHHPPHPH. All of the experiments are performed in MATLAB and the average computation time of ten experiments is 46.64 s in a Pentium IV 3 GHz 1.5GB RAM PC. In Fig. 7 one of the optimum configurations and the corresponding fitness evaluation for each iteration is given. As it can be shown in Fig. 7, the Ant-Q algorithm found the optimum configuration in 35 iterations which takes 34 s of computational time. In Table 6 and Fig. 8 solutions found by Ant-Q algorithm for a number of benchmark amino acid sequences also provided. In literature, there exist many studies proposed to find out the optimum fold for these sequences. In these studies, the problem generally handled in two phases. In the first phase an algorithm is used to find a local minimum protein configuration (like Ant-Q) and then another algorithm (local search methods) tries to further improve this local configuration to find out the global optimum fold. However, in this study it is mainly aimed to show the advantages of the proposed state action space. Thus, the second phase is not used and for this reason, except the Seq1 the obtained configurations are local minimums.

Czibula et al. [10–13] also used the sequence $\mathcal{P}_1 =$ HPHPPHHPHPPHPHHPPHPH for their studies. However, as mentioned before with the existing method it is not possible to create state-action space at the beginning of the algorithm. So, in this study to compare the performance of the proposed method with the standard Q-learning algorithm, again the newly proposed state-action space is used with delayed reinforcement.

As mentioned before, in the standard Q-learning algorithm the agent chooses an action randomly with a probability of 0.25. When the amino acid sequence is not long enough these random movements converges the agent to optimum solution in a number of iterations. However, as the length of the amino acid sequence increases, the number of the configurations that must be visited by the agent also increases dramatically. Thus, in such a situation agent could not be able to find the optimum configuration. For example, the agent finds the optimum configuration for the $\mathcal{P}_2 =$ HPHPPH. However, when it comes to sequence $\mathcal{P}_1 =$ HPHPPHHPHPPHPHHPPHPH the agent fails to find the optimum configuration even in 100,000 iterations. Because, the state space is not thoroughly explored by the agent even in 100,000 iterations with the standard Q-learning algorithm.

Scenario 2. In this case, the agent learns the state-action space for all of the combinations of an n length amino acid sequence. As it can be shown from Fig. 9, for $n = 6$ there are $2^6 = 64$ sequences that agent must be learned.

Let us consider the agent wants to learn state-action space for $n = 6$. For this purpose, first the state-action space which consists of $\mathcal{S} = \left(2 + 4 \cdot \sum_{n=2}^{6} 2^n\right) = 498$ states is created and then the AQ-table is initialized according to the possible state transitions. Note that, in this case the number of transitions are much more than the one in Scenario 1. Because, all of the sequences are considered. Then, the agent picks up one of the sequences and tries to learn state-action space for this particular sequence in a number of iterations. In this study, for $n = 6$ the maximum number of iterations is selected as 100 for each sequence. Once the agent learns the state-action space for a particular sequence the resulting AQ-table is stored for the next sequence. Thus, additively updating the AQ-table, at the end of the learning process a universal AQ-table is obtained. This table is called universal because, after the learning process for any sequence of length 6, this table can guide the agent to form optimum configurations. In Fig. 10, an example for the sequence $\mathcal{P}_2 =$ HPHPPH is given. Note that, in the proposed method in order to find the optimum configuration all the agent needs is the amino acid sequence itself but not the directions as in the existing methods. In the existing methods the state space only encodes the folding directions of a particular amino acid sequence. So, it is not possible to obtain optimum configuration for any sequence after the learning process.

6. Conclusion

The newly proposed state-action space allows the protein folding problem to be studied by using reinforcement learning methods. Compared to the existing one the newly proposed state-action space has several advantages. First, the size of the state-action space is less dependent to the length of the amino-acid sequence. In the existing methods, the size of the state-action space is highly affected by the amino-acid sequence length. Moreover, if the problem is handled in 3D lattice structures the size of the state-action space will be further increased. Because in 3D lattice structures there are two additional positions that agent could be moved. Thus, with the existing definition the size of the state-action space will change by 6^n.

The second advantage of the proposed state space representation is its learning ability. By using the newly proposed state-action space, the optimal fold of any amino acid sequence of a particular length can be found. Because, in the proposed state-action space the characteristics of each amino acids is conserved in addition to direction information. Thus, it can be concluded that the proposed state-action space is more generic than the existing one. These important advantages of the proposed state-action space could be combined with the existing methods for the protein folding problem in 2D or 3D lattice models.

Additionally, the Ant-Q algorithm has proven to be a good candidate for the solution of the protein folding problem. It has been demonstrated that the Ant-Q algorithm overwhelmingly outperforms the standard Q-learning algorithm which is used in the existing methods.

Besides its advantages, the proposed state-action space has also some limitations. Especially, for long sequences the resulting state-action space (for Scenario 2) is still huge. As a further study, a recursive algorithm could be proposed to predict the optimum fold of $n + a$ length sequences, where $a \geq 1$, from the learned AQ-values of n length sequences.

Future directions include the design of the above mentioned recursive method to handle the problems for long sequences. Additionally, a comparison of the swarm based algorithms over the newly proposed state-action space in 2D and 3D will also be studied.

References

[1] K.A. Dill, Theory for the folding and stability of globular proteins, Biochemistry 24 (6) (1985) 1501–1509.

[2] X. Ma, Y. Xu, G. Sun, L. Deng, Y. Li, State-chain sequential feedback reinforcement learning for path planning of autonomous mobile robots, J. Zhejiang Univ. Sci. C 3 (March) (2013) 167–178.

[3] S. Sendari, S. Mabu, K. Hirasawa, Fuzzy genetic Network Programming with Reinforcement Learning for mobile robot navigation, in: IEEE International Conference on Systems, Man, and Cybernetics (SMC), 9–12 October, 2011, pp. 2243–2248, http://dx.doi.org/10.1109/ICSMC.2011.6084011.

[4] C. Rego, H. Li, F. Glover, A filter-and-fan approach to the 2D HP model of the protein folding problem, Ann. Oper. Res. 188 (1) (2009) 389–414.

[5] H. Guo, Q. Lu, J. Wu, X. Huang, P. Qian, Solving 2D HP protein folding problem by parallel ant colonies, in: BMEI'09. 2nd International Conference on Biomedical Engineering and Informatics, 2009, 17–19 October, 2009, pp. 1–5.

[6] J. Liu, G. Li, J. Yu, Protein-folding simulations of the hydrophobic–hydrophilic model by combining pull moves with energy landscape paving, Phys. Rev. E 84 (3) (2011) 031934.

[7] C. Huang, X. Yang, Z. He, Protein folding simulations of 2D HP model by the genetic algorithm based on optimal secondary structures, Comput. Biol. Chem. 34 (June (3)) (2010) 137–142, http://dx.doi.org/10.1016/j.compbiolchem.2010.04.002, ISSN 1476-9271.

[8] B. Andrei, H. Luchian, Protein structure prediction in lattice models with particle swarm optimization Lecture Notes in Computer Science, vol. 6234, 2010, pp. 512–519.

[9] C.-J. Lin, M.-H. Hsieh, An efficient hybrid Taguchi-genetic algorithm for protein folding simulation, Expert Syst. Appl. 36 (December (10)) (2009) 12446–12453, http://dx.doi.org/10.1016/j.eswa.2009.04.074, ISSN 0957-4174.

[10] G. Czibula, M.I. Bocicor, I.G. Czibula, A distributed reinforcement learning approach for solving optimization problems, Recent Res. Commun. IT (2011).

[11] G. Czibula, M.I. Bocicor, I.G. Czibula, A reinforcement learning model for solving the folding problem, Int. J. Comp. Technol. Appl. (2011) 171–182.

[12] G. Czibula, M.I. Bocicor, I.G. Czibula, An experiment on protein structure prediction using reinforcement learning, Informatica LVI (1) (2011).

[13] G. Czibula, M.I. Bocicor, I.G. Czibula, Solving the protein folding problem using a distributed Q-learning approach, Int. J. Comput. 5 (3) (2011).

[14] A. Stewart, Face-centered Cubic (FCC) Lattice Models for Protein Folding: Energy Function Inference and Biplane Packing (Ph.D. thesis), Brown University, USA, 2010.

[15] H. Iima, Y. Kuroe, S. Matsuda, Swarm reinforcement learning method based on ant colony optimization, in: IEEE International Conference on Systems Man and Cybernetics (SMC), 10–13 October, 2010, pp. 1726–1733.

[16] H. Iima, Y. Kuroe, Swarm reinforcement learning algorithms based on particle swarm optimization, in: IEEE International Conference on Systems, Man and Cybernetics, 2008. SMC 2008, 12–15 October, 2008, pp. 1110–1115.

[17] H. Iima, Y. Kuroe, Swarm reinforcement learning algorithms based on Sarsa method, in: SICE Annual Conference, 20–22 August, 2008, pp. 2045–2049.

[18] H. Iima, Y. Kuroe, Swarm reinforcement learning method based on an Actor-Critic method Simulated Evolution and Learning, Lecture Notes in Computer Science, vol. 6457, 2010, pp. 279–288.

[19] H. Iima, Y. Kuroe, K. Emoto, Swarm reinforcement learning methods for problems with continuous state-action space, in: IEEE International Conference on Systems, Man, and Cybernetics (SMC), 9–12 October, 2011, pp. 2173–2180.

[20] L.M. Gambardella, M. Dorigo, Ant-Q: a reinforcement learning approach to the traveling salesman problem, in: Proceedings of Twelfth International Conference on Machine Learning, Morgan Kaufmann, 1995, pp. 252–260.

[21] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.

[22] M. Dorigo, Optimization, Learning and Natural Algorithms (Ph.D. thesis), Politecnico di Milano, Italy, EU, 1992 (in Italian).

[23] M. Dorigo, V. Maniezzo, A. Colorni, The Ant System: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. 26 (1996) 2.