Heliyon

Received:
26 July 2018
Revised:
27 December 2018
Accepted:
27 February 2019

Cite as: Akram Emdadi, Fatemeh Ahmadi Moughari, Fatemeh Yassaee Meybodi, Changiz Eslahchi. A novel algorithm for parameter estimation of Hidden Markov Model inspired by Ant Colony Optimization.
Heliyon 5 (2019) e01299.
doi: 10.1016/j.heliyon.2019.e01299

A novel algorithm for parameter estimation of Hidden Markov Model inspired by Ant Colony Optimization

Akram Emdadi [a], **Fatemeh Ahmadi Moughari** [a], **Fatemeh Yassaee Meybodi** [a], **Changiz Eslahchi** [a,b,*]

[a] *Department of Mathematics, Shahid-Beheshti University, Tehran, Iran*

[b] *School of Biological Sciences, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran*

* Corresponding author.
E-mail address: ch-eslahchi@sbu.ac.ir (C. Eslahchi).

Abstract

HMM is a powerful method to model data in various fields. Estimation of Hidden Markov Model parameters is an NP-Hard problem. We propose a heuristic algorithm called "AntMarkov" to improve the efficiency of estimating HMM parameters. We compared our method with four algorithms. The comparison was conducted on 5 different simulated datasets with different features. For further evaluation, we analyzed the performance of algorithms on the prediction of protein secondary structures problem. The results demonstrate that our algorithm obtains better results with respect to the results of the other algorithms in terms of time efficiency and the amount of similarity of estimated parameters to the original parameters and log-likelihood.

The source code of our algorithm is available in https://github.com/emdadi/HMMPE.

Keywords: Computer science

1. Introduction

The Hidden Markov Model (HMM) is a statistical extension of finite state automata to model the sequential data. It can be considered as a machine learning approach which is applied to elicit insight about model parameters. In an HMM, transitions between states and emission of symbols are stochastic. HMMs are applied in many fields such as bioinformatics [1, 2, 3, 4, 5], econometric [6, 7], population genetics [8, 9] and etc. The main problem related to HMM is estimating parameters of the model by learning from training data. This problem is very important in real applications that is an NP-hard problem. Several approaches have been proposed to work out this problem, such as Baum-Welch [2], Viterbi-Training [2], Tabu-Search [10], SNNMF algorithm [11].

The goal of the Baum-Welch algorithm is to find the maximum likelihood estimators of HMM parameters. It is essentially the Expectation-Maximization algorithm that re-estimates the hidden parameters with the calculation of forward, backward and posterior probabilities of sequences in each iteration [2]. This algorithm suffers from slow convergence.

The Viterbi-training applies the Viterbi algorithm on input sequences to get the most likely paths and uses them to re-estimate the hidden parameters [2]. This method is affected by the low accuracy of parameter estimation and high dependency to initial guesses.

The Tabu-Search algorithm is a generalized heuristic global search technique which can be applied in many problems. In applying Tabu-Search on HMMs, every solution of the problem is considered as an array which is a concatenation of parameters of the model. The neighbors of a solution can be obtained by swapping two of its elements. By starting with a random solution, Tabu-Search algorithm explores the solution space by visiting the neighbors of the current solution in case of not being in tabu list [10]. This algorithm is so time-consuming, such that if the number of states of the model is more than 9, it takes several hours to find a solution.

The structured nonnegative matrix factorization algorithm (SNNMF), is another approach for parameter estimation of HMM that ensures the non-negativity of the estimated parameters. It uses a non-negative matrix factorization by applying stochastic constraints [11]. The main issue of this method is that it cannot be applied to models that their number of states are more than the number of distinct emitted characters.

Due to the high importance of the problem of estimating HMM parameters and numerous drawbacks of previous methods, proposing novel method that can estimate HMM parameters efficiently and accurately is in need. Stochastic methods such as Ant colony optimization are good alternatives to achieve this goal.

Ant Colony Optimization (ACO) [12, 13] is a metaheuristic method that simulates foraging manner of ant colonies. Ant colonies govern their foraging behavior without any centralized management [14] and just by deposited pheromones on paths. The pheromone trails in ACO denoted as information communication media [12], which ants share their information via them. Pheromones also reflect the search experience. Ants use them as guidance to find the best path and solve the problem. ACO is used for solving hard combinatorial optimization problems [12, 15] such as scheduling [16], traveling salesman [17], tracking system [18], power system harmonics [19], intelligent traffic control system [20], water resource management [21], green cloud computing [22] and etc.

ACO has several advantages: it is easy to integrate with other algorithms [23, 24, 25], is able to run parallel computation [15, 23, 24, 26], has strong robustness [23], includes an intelligent search [23], and has good global optimization compared to other metaheuristic algorithms [16, 23]. Furthermore, ACO rapidly converges to good solutions [23, 24, 27] and has guaranteed convergence [24]. On average, it performs efficiently when applied to comparatively complex problems [23].

In this paper, we propose a novel method inspired by ACO. First, we present a short review of the Hidden Markov Model and Ant colony algorithm in the next section. Then by using Ant colony technique, we propose an algorithm to estimate parameters with phenomenal speed and high accuracy.

2. Background

Particularly, ACO is an appropriate method in problems that are similar to the shortest path in the graph. Since, we reformed the problem of estimating HMM parameters to the shortest path problem; thus, it is reasonable to use ACO.

2.1. Hidden Markov Model (HMM)

An HMM is a statistical model to represent a Markov process with hidden states. An output of HMM is a sequence with finite length called sequence of observation that is represented as $S = s_1, \dots, s_l$ and it is assumed to be emitted by sequences of states $T = t_1, t_2, \dots, t_l$. These states are hidden and cannot be observed.

An HMM $\mathcal{M}$ with n hidden states is represented by

- $\Sigma = \{c_1, \dots, c_m\}$ is the set of alphabets of all sequences.
- $Q = \{q_1, \dots, q_n\}$ is the set of states, each of which is able to emit symbols of alphabet Σ.
- π_i, $\forall i = 1, \dots, n$ is the probability to start with ith state ($\pi_i = \mathbb{P}(t_1 = q_i)$).

- $A = \left[a_{i,j}\right]_{i,j=1,\dots,n}$ which $a_{i,j}$ is the transition probability from q_i to q_j, i.e. if the step is k, $a_{i,j} = \mathbb{P}(t_{k+1} = q_j \mid t_k = q_i)$.
- $E = \left[e_{i,j}\right]_{i=1,\dots,n,j=1,\dots,m}$ where $e_{i,j}$ is the probability that state q_i emits c_j, i.e. if the step is k, $e_{i,j} = \mathbb{P}(s_k = c_j \mid t_k = q_i)$.

It is obvious that $\forall i$:

$$\sum_{i=1}^{n} \pi_i = 1, \quad \sum_{j=1}^{n} a_{i,j} = 1, \quad \sum_{j=1}^{m} e_{i,j} = 1 \tag{1}$$

There are two types of HMMs: homogeneous and non-homogeneous. In homogeneous HMMs the probability distributions of transitions and symbol emissions are always the same, while they change during the time in non-homogeneous HMMs. All HMMs in this paper are assumed to be homogeneous. States and emissions in this paper are discrete. An HMM $\mathcal{M}$ is specified by model parameters $\lambda = \{\pi, A, E\}$. Given model parameters λ, the probability of emitting a sequence of observation $S = s_1, s_2 \cdots, s_l$ by a sequence of states $T = t_1, t_2 \cdots, t_l$ is computed as follows.

$$\mathbb{P}(S, T \mid \lambda) = \pi_{t_1} \cdot \prod_{i=1}^{len} e_{t_i, s_i} \cdot a_{t_i, t_{i+1}} \tag{2}$$

where $a_{t_{len}, t_{len+1}} = 1$.

Example 2.1. Suppose the well-known Fair-Bet Casino HMM with $\sum = \{1$ (Head), 0 (Tail)$\}$, $Q = \{\text{Fair}, \text{Biased}\}$, $\pi = [\frac{1}{2}, \frac{1}{2}]$, $A = \begin{bmatrix} \frac{9}{10} & \frac{1}{10} \\ \frac{1}{10} & \frac{9}{10} \end{bmatrix}$, $E = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & \frac{1}{4} \end{bmatrix}$.

The probability of emitting sequence $S = 01011101001$ with path $T = FFFBBBBBFFF$ is

$$(\tfrac{1}{2} \times \tfrac{1}{2})(\tfrac{9}{10} \times \tfrac{1}{2})(\tfrac{9}{10} \times \tfrac{1}{2})(\tfrac{1}{10} \times \tfrac{3}{4})(\tfrac{9}{10} \times \tfrac{3}{4})(\tfrac{9}{10} \times \tfrac{3}{4})(\tfrac{9}{10} \times \tfrac{1}{4})(\tfrac{9}{10} \times \tfrac{3}{4})(\tfrac{1}{10} \times \tfrac{1}{2})(\tfrac{9}{10} \times \tfrac{1}{2})(\tfrac{9}{10} \times \tfrac{1}{2}) = 2.66 \times 10^{-6}$$

Generally, the probability of emitting a sequence S by $\mathcal{M}$ with model parameters λ, which is denoted by $\mathbb{P}(S \mid \lambda)$, is calculated as follows.

$$\mathbb{P}(S \mid \lambda) = \sum_{T} \mathbb{P}(S, T \mid \lambda), \tag{3}$$

where T is a possible sequence of states in HMM $\mathcal{M}$.

Let $S_1, \dots, S_N$ be N different sequences that are generated independently from HMM $\mathcal{M}$ with model parameters λ. The average of log-likelihood of emitting the sequences $S_1, \dots, S_N$ is defined by:

$$prob = \frac{1}{N} \sum_{i}^{N} \log \mathbb{P}(S_i \mid \lambda). \tag{4}$$

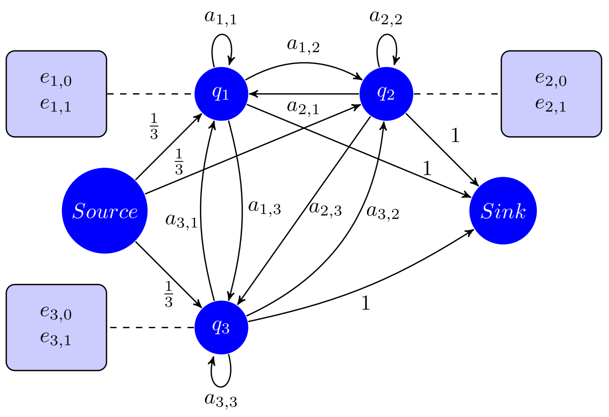

Figure 1. Associated Graph of HMM.

In this paper, we represent an HMM $\mathcal{M} = (\Sigma, Q, A, E, \pi)$ as a directed graph $G = (V, E, w, Z)$ called associated graph that $V = Q \cup \{q_{source}, q_{sink}\}$ and $E = \{(q_i, q_j) \mid \forall q_i, q_j \in Q\} \cup \{(q_{source}, q_i) \mid \forall q_i \in Q\} \cup \{(q_i, q_{sink}) \mid \forall q_i \in Q\}$. The elements of Q denote as state nodes and q_{source}, q_{sink} as non-state nodes. In this graph, non state nodes are silent, i.e. they do not emit any symbol, but state nodes are able to emit each symbol of alphabet. w is a function which assigns a weight to each edge as follows.

$$\forall q_i, q_j \in Q \quad w(q_i, q_j) = a_{ij}, \quad w(q_{source}, q_i) = \frac{1}{n}, \quad w(q_i, q_{sink}) = 1.$$

For sake of simplicity, we show $w(q_i, q_j)$ by w_{ij}. In addition, a vector $Z_i = (z_i(1), z_i(2), \dots, z_i(m))$ is assigned to each state node q_i, where

$$\forall c_j \in \Sigma, \quad z_i(j) = e_{ij}. \tag{5}$$

Example 2.2. Suppose HMM $\mathcal{M} = (\Sigma, Q, A, E, \pi)$, $\Sigma = \{0, 1\}$, $Q = \{q_1, q_2, q_3\}$, $A = [a_{ij}]_{i,j}$, $E = [e_{ij}]_{i,j}$. The associated graph of this HMM is shown in Figure 1.

In this paper, we use "path" instead of sequence of states and "sequence" instead of "sequence of observation".

There are three main problems related to HMM:

1. **Path Finding** (PF): HMM $\mathcal{M}$ with model parameters λ is given. The problem is to determine a path for a sequence S, such that the probability $\mathbb{P}(S, T \mid \lambda)$ is maximized.
2. **Known Parameter Estimation** (KPE): Suppose a set of sequences and their paths are given. The problem is to specify model parameters such that the average of log-likelihood of emitting all sequences by their paths is maximized.
3. **Unknown Parameter Estimation** (UPE): Assume a set of sequences are given and their paths are unknown. The problem is to identify model parameters with the aim of maximizing the average of log-likelihood of emitting all sequences.

The first and second problems have polynomial time algorithms. Viterbi algorithm [2] solves the path finding problem. Empirical distributions are the maximum likelihood estimators for model parameters in KPE problem [2]. However, UPE problem is NP-hard and if $P \neq NP$, its exact solution cannot be obtained in polynomial time [28].

The problem studied in this paper is the UPE problem. There are many papers that have tried to solve this problem by heuristic methods such as [10, 29, 30, 31, 32]. The UPE problem is to estimate matrices A , E. Thus, the problem of estimating HMM parameters is equivalent to finding weights of edges and vectors of each node in the associated graph. Here we utilize Ant Colony Optimization to solve UPE problem.

2.2. Ant Colony Optimization (ACO)

Social insects like ants act as a community. They are able to solve complex problems by means of cooperation. Ants can find the shortest path from the nest to the food source without visual information. They can also adapt to a changing environment. Ants communicate with each other by pheromones. While ants move, they drop some amount of pheromones. Ant Colony Optimization (ACO) is a probabilistic technique for solving optimization problems based on the behavior of ants in nature. Application of ACO on a graph with source and sink is as follows.

Every path from source to sink can be considered as a route for an ant to traverse. The graph contains all possible paths. Ants walk through the graph according to the distance and the amount of pheromones on each path. Thus, the probability of traversing the edge from node x to y can be formulated as follow.

$$\mathbb{P}_{x,y} = \begin{cases} \frac{(\tau_{x,y})^{\alpha} \cdot (\eta_{x,y})^{\beta}}{\sum_{z \in N_x} (\tau_{x,z})^{\alpha} \cdot (\eta_{x,z})^{\beta}} & y \in N_x, \\ 0 & otherwise, \end{cases} \tag{6}$$

where N_x is a set of the neighbor nodes of x, $\tau_{x,y}$ is the pheromone amount on the edge from x to y, $\eta_{x,y}$ is the desirability of transition from node x to y (usually the desirability is the inverse of distance) and α, β are two parameters to control the influence of variables.

When all ants move through the graph, pheromone quantities will be updated as follows.

$$\tau_{x,y} = (1 - \rho)(\tau_{x,y} + \sum_{\upsilon} \Delta^{\upsilon}_{x,y}) \tag{7}$$

where ρ is the pheromone evaporation coefficient and $\Delta^{\upsilon}_{x,y}$ is the amount of pheromone deposited by υth ant on the edge from x to y. A large value of ρ indicates fast evaporation. ACO algorithm includes several parameters, of which, the number of ants and the evaporation factor can highly influence the calculation time. The above parameters are determined in the subsequent sections in detail.

We use ACO technique to propose an algorithm to estimate HMM parameters.

3. Method

In the UPE problem, our inputs are a set of sequences and the number of states of the HMM. The goal is to determine model parameters of the HMM that maximizes the log-likelihood of sequences given model ($log\mathbb{P}(s|\lambda)$).

Unknown Parameter Estimation Problem:
Find the optimal model parameters.
Inputs: A set of sequences $S_1, S_2, \ldots, S_N$ generated by an HMM $\mathcal{M}(\Sigma, Q, A, E, \pi)$ where elements of A, E and π are not known.
Output: Values of $\lambda = (A,\ E,\ \pi)$ that maximizes $\frac{\sum_i Log\mathbb{P}(S_i|\lambda)}{N}$

As mentioned above, this problem is NP-hard. Considering the associated graph G, the main steps of our algorithm are as follows.

1. Find initial path for each sequence by random movements of ants.
2. Set initial pheromones on nodes and edges.
3. Estimate weights of edges and vectors of each node to re-estimate matrices $A,\ E$.
4. Find new paths for each sequence according to matrices $A,\ E$ and pheromones.
5. Update and evaporate the pheromone values.
6. Go to the third step if the termination criteria do not satisfy, otherwise stop.

These steps are explained elaborately in the subsequent subsections.

3.1. Find initial paths

For the first iteration, we assume that the weights of edges and vector of nodes are zero. We considered an ant for each input sequence. At the first step, for each sequence of observation, we consider a random movement of ant in graph G from source to sink as its sequence of states. The number of nodes visited in path except source and sink, is as long as the sequence of observation. For example, for sequence of observation $S = s_1 s_2 \ldots s_l$ ($\forall i, s_i \in \Sigma$) and its corresponding ant, the random path $T = q_{source} t_1 t_2 \ldots t_l q_{sink}$ is selected ($\forall i, t_i \in Q$). This path can be shown as follows:

$$< q_{source} >, < t_1, s_1 >, < t_2, s_2 >, \ldots, < t_l, s_l >, < q_{sink} > .$$

3.2. Initial pheromone

We defined two types of pheromone instead of $\tau_{x,y}$ described in the Section 2. Since the goal is to estimate two parameter types A, E, every ant deposits two kinds of pheromones: The first type of pheromones is related to edges and the second is related to nodes. For each edge $e = (q_i, q_j)$ we defined pheromone value τ_{ij}^A and for each node q_i, we defined pheromone vector τ_i^E of size m. The pheromone quantities on edges are proportionate to the number of ants visited those edges and the pheromone quantities on pairs $< t_i, c_i >$ are proportionate to the number of ants emitted those symbols from those states. The amount of pheromone put by one ant on an edge is Phr^A, and the amount of pheromone it puts on each node for any character is Phr^E.

If edge $e = (q_i, q_j)$ in G is passed by x ants in the first iteration, we defined

$$\tau_{ij}^A = (x \times Phr^A)(1-\rho), \tag{8}$$

where ρ is the pheromone evaporation coefficient.

If $< q_i, c_j >$ is visited y times in ants paths,

$$\tau_i^E(j) = (y \times Phr^E)(1-\rho). \tag{9}$$

3.3. Estimate weights of edges and vectors of each node

In each iteration, weights of edges and vectors of each node are updated using residual amount of pheromone τ_{ij}^A, τ_i^E as follows.

$$w'_{ij} = w_{ij} + \tau_{ij}^A \tag{10}$$

$$z'_i(j) = z_i(j) + \tau_i^E(j) \tag{11}$$

Then weights of edges and vectors of each node are normalized. Normalizing is done by

$$w_{ij} = \frac{w'_{ij}}{\sum_{k=1}^{n} w'_{ik}} \tag{12}$$

$$z_i(j) = \frac{z'_i(j)}{\sum_{k=1}^{m} z'_i(k)}. \tag{13}$$

And the values of matrices A, E are updated by

$$a_{ij} = w_{ij}, \quad \forall i, j \in [1, n] \tag{14}$$

$$e_{ij} = z_i(j) \quad \forall i \in [1, n],\ j \in [1, m]. \tag{15}$$

3.4. Finding new path

From this stage on, the movements of ants are based on weights of edges and vectors of each node. The decision rule for selecting the sequence of states for a sequence of observation $S = s_1 s_2 \dots s_l$ is as follows.

If the ant corresponding to sequence S is at node q_i in the one step of its movement and the next symbol is c_k, the state q_j is selected as its next node of path if:

$$j = \arg\max_d \frac{(log(w_{id}) + \tau_{id}^A) \times (log(z_d(k)) + \tau_d^E(k))}{\sum_{r=1}^n (log(w_{ir}) + \tau_{ir}^A) \times (log(z_r(k)) + \tau_r^E(k))} \tag{16}$$

and for the first step of movement from source, q_j is selected by ant if the first symbol is c_k and

$$j = \arg\max_i \frac{log(w_{source,i}) \times (log(z_i(k)) + \tau_i^E(k))}{\sum_{r=1}^n (log(w_{source,r})) \times (log(z_r(k)) + \tau_i^E(k))}. \tag{17}$$

3.5. Update and evaporate pheromone

The weights of edges and vectors of nods are updated in each iteration as follows.

If edge $e = (q_i, q_j)$ in G is passed by d ants in this iteration, the pheromone of this edge is updated by:

$$\tau_{ij}^A = \tau_{ij}^A + (d \times Phr^A)(1 - \rho). \tag{18}$$

If $< q_i, c_j >$ is visited f times in ants paths, the jth elements of τ_i^E will be updated by:

$$\tau_i^E(j) = (f \times Phr^E)(1 - \rho). \tag{19}$$

Then we go to step 3 to re-estimate matrices A, E.

3.6. Terminate or continue

The algorithm iterates until one of the following conditions is met.

(i) Number of iterations reaches to a predefined max iteration value.
(ii) Algorithm converges.

The algorithm converges when all the following constraints satisfy.

1. Norm-2 of subtraction of estimated matrices A in two consecutive iterations divided by the number of states, is lower than a predefined threshold. Particularly, $\frac{||A^t - A^{t-1}||_2}{n} < threshold$, where A^t is estimation of A in the tth iteration.

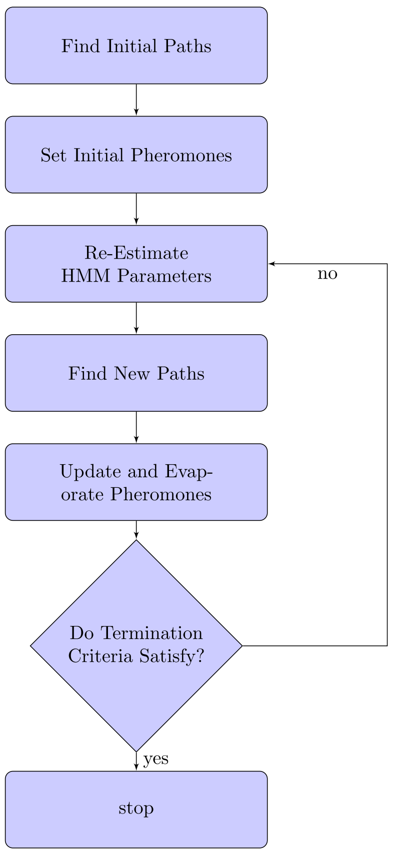
Find Initial Paths
Set Initial Pheromones
Re-Estimate HMM Parameters
no
Find New Paths
Update and Evaporate Pheromones
Do Termination Criteria Satisfy?
yes
stop

Figure 2. Scheme of AntMarkov algorithm.

2. Norm-2 of subtraction of estimated matrices for E in two consecutive iterations divided by the number of states, is lower than a predefined threshold. Particularly, $\frac{||E^t - E^{t-1}||_2}{n} < threshold$, where E^t is estimation of E in the tth iteration.
3. The following relation satisfies:

$$\frac{|prob^t - prob^{t-1}|}{prob^{t-1} + 1} < threshold \tag{20}$$

where $prob^t$ is the log-likelihood of sequences given model parameters estimated in tth iteration.

The main scheme of our algorithm is represented in Figure 2.

4. Results and discussion

4.1. Simulation

We generate 5 datasets in order to evaluate algorithms. The datasets have different characteristics. In this paper *StateNum* denotes the number of states, *CharNum* indicates the number of symbols in alphabet, *LineCount* is the number of sequences in each dataset and *SeqLen* is the length of sequences.

We considered distinct HMMs with dissimilar StateNum, CharNum and generated different cases from each HMMs with various SeqLen and LineCount. Each case includes a quartet (trainSet, testSet, A, E): two sets of sequence for training and testing and also the original matrices A, E. The number of sequences in trainSet and testSet is equal in each case. In each case, the lengths of sequences were considered to be different in order to evaluate algorithms on sequences with dissimilar lengths. In order to completely evaluate algorithms, the condition number of transition and emission matrices were different and in range $(1, 250)$. The distribution of transition and emission matrices were considered uniform unless where we declare another distribution. The sequences were generated by *hmmgenerate* function in Matlab giving the parameters of the model. In some generated cases, the log-likelihood of emitting sequences given the original model parameters, was $-\inf$. We eliminated such cases from further analyses.

We compare the mentioned algorithms on five datasets:

1. **DS1:** A dataset of 432 cases that are generated from the uniform probability distribution.
   We considered 24 distinct HMMs and generated 18 different cases from each HMMs. Values of parameters were set to the following numbers:
   - StateNum: 2, 3, 5, 8
   - CharNum: 2, 4, 8, 12, 16, 20
   - LineCount: 50, 100, 500, 1000, 2000, 5000
   - SeqLen: $(10, 20)$, $(50, 100)$, $(100, 200)$

   The values of parameters were selected based on natural applications of HMM, so the testbed is natural and reasonable. For example, in prediction of secondary structure, the StateNum=3 (Helix, Sheet, Coil) [33, 34, 35, 36, 37, 38] or StateNum=8 (H=alpha helix, B=beta-bridge, E= extended strand, G=3/10 helix, I=pi-helix, T=turn, S=bend, X=coil) [33, 35, 39, 40] and CharNum=20 (number of amino acids). In Fair-BetCasino problem, StateNum=2 (Fair, Biased) and CharNum=2 (Head, Tail) [2]. When modeling membrane protein topology prediction to HMM problem, we have CharNum=20 (number of amino acids) and StateNum=5 (outside loop, outside tail, membrane helix, inside tail, inside loop) [41]. In the prediction of Heterochromatin/Euchromatin regions from DNA sequence, the StateNum=2 (Heterochromatin, Euchromatin) and CharNum=4 (number of nucleic acids).
   SeqLen is considered small since the likelihood of long sequences is close to zero and their numerical calculations lead to underflow. The transition and emission

probability distributions are uniform. For further information about these cases refer to Additional files 1, 2.

2. **DS2:** A dataset of 432 cases that are generated from normal probability distribution.
   These cases are generated with similar settings (such as StateNum, CharNum, LineCount, SeqLen, the number of distinct HMMs, etc) to the first dataset, except that the transition and emission matrices were generated by the truncated normal distribution with $\sigma^2 = 0.25$ and $\mu = 0.5$. Thus, the generated numbers out of range $[0, 1]$ are set to 0.5. The selection of σ^2, μ is reasonable since the sampled elements in range $[0, 1]$ has mean$= 0.5$ and since in normal distribution almost all numbers are $\pm 2\sigma^2$ far from μ. This is common to consider normal and uniform distribution for generating datasets [42]. For further information about these cases refer to Additional files 3, 4.
3. **DS3:** A dataset of 30 cases with singular transition matrices
   In each case, all rows of transition matrix were generated randomly from the uniform distribution and the last row was computed based on a linear combination of other rows, this procedure ensures the singularity of the transition matrix. Since emission matrices are often not square, their singularity is not meaningful. Supplementary information about these cases is provided in Additional file 5 and 6.
4. **DS4:** A dataset of 90 cases with sparse transition matrices
   The cases in this dataset have sparse transition matrices with uniform probability distributions. In order to fully examining this kind of data, we considered the size of transition matrices from 5×5 to 50×50. Further information of these cases is presented in Additional files 7 and 8.
5. **DS5:** A dataset of 16 cases with absorbing states
   A common type of HMM with transient states is an absorbing one. In an absorbing HMM, every state can reach an absorbing state. An absorbing state is a state that, once entered, cannot be left. We simulated a dataset containing 16 cases with the random number of absorbing states. More information about this dataset is provided in Additional files 9 and 10.

For each case, we estimated model parameters via Baum-Welch, Viterbi-Training, Tabu-search, SNNMF and AntMarkov algorithms by trainSet.

We use off-the-shelf implementations of Viterbi-Training and Baum-Welch algorithm in Matlab. Also, SNNMF is implemented by [11]. Thus, we implement our algorithm and Tabu-Search in Matlab in order to provide a comparative analysis. The maximum number of iterations was set to 100 in all iterative algorithms. In Baum-Welch, Viterbi-Training and AntMarkov algorithms, the threshold used in convergence constraints, was set to 10^{-6}. Baum-Welch and Viterbi-Training algorithms take an initial transition and emission matrices as input. These initial matrices were generated from a uniform distribution for cases with uniform HMMs and from a normal distribution for cases with normal HMMs. In Tabu-Search algorithm, a number of neighbors considered in each step and the size of Tabu-list were set to 20, according to the suggestion of its paper [10].

Implementation of SNNMF algorithm is not able to deal with cases that their StateNum is greater than CharNum. Thus, we did not execute it in such cases.

We observed that the performance of AntMarkov algorithm is influenced severely by Phr^A and Phr^E values. Thus, we considered 4 different intervals for choosing Phr^A and 4 different intervals for choosing Phr^E. For each training case, Phr^A and Phr^E values were selected randomly from their intervals.

We tested 16 various values for pheromones in AntMarkov. Intervals for Phr^A and Phr^E were $(0, 1)$, $(1, 10)$, $(10, 100)$, $(100, 1000)$. We observed that on 80% cases, $(100, 1000)$ is the best interval for Phr^A and results were indifferent to the selection of an interval of Phr^E. To compare AntMarkov with other algorithms, we set Phr^A to a random number in the interval $(100, 1000)$ and Phr^E to a random number in the interval $(0, 1)$.

We evaluated the performance of algorithms according to the following criteria:

- Log-likelihood: is defined in formula (4). This is the most fundamental criterion for evaluating the estimation of HMM parameters.
- Time: the time consumed by the algorithm to produce the output. We recorded the time consumed by each algorithm for estimating parameters from trainingSet by Matlab commands.
- NormE: the norm of subtraction of estimated emission matrix with the original emission matrix.
- NormTR: the norm of subtraction of estimated transition matrix with original transition matrix.
  We compute Frobenius norm, infinity norm and 2-norm for both the normE and normTR. Results show that these three norms are equivalent, i.e. whenever algorithm A has better 2-norm than algorithm B, the Frobenius norm of algorithm A is better than that of algorithm B. The values of 2-norm for normTR and normE is in $[0, 2]$. It is obvious that the lower values of normTR (normE) means that the estimated transition (emission) matrix is more similar to the original transition (emission) matrix. Hence, normTR and normE can be interpreted as the accuracy of estimation. Since computing the difference of estimated and the true parameters in this way, is dependent on the ordering of states, we calculated normE and normTR for all permutations of states. Then for evaluating the algorithms, we considered the value of normE and normTR for

Table 1. Win percentage on uniform data (DS1).

criteria	AntMarkov	Baum-Welch	Viterbi-Training	SNNMF	Tabu-Search
Log likelihood	**0.43**	0.13	0.02	0.33	0.09
Norm-2-TR	**0.32**	0.25	0.00	0.15	0.28
Norm-2-E	**0.48**	0.15	0.01	0.24	0.13
Norm-Inf-TR	**0.39**	0.16	0.00	0.27	0.17
Norm-Inf-E	**0.45**	0.04	0.01	0.37	0.12
Norm-Fr-TR	**0.41**	0.11	0.00	0.34	0.14
Norm-Fr-E	**0.47**	0.00	0.01	0.41	0.11
time	**0.45**	0.00	0.32	0.23	0.00
iteration	**0.76**	0.00	0.24		

the permutation that the algorithm has the lowest sum of normTR and normE. In Tables 1, 2, 3, 4, 5, norm_2_TR, norm_inf_TR and norm_Fro_TR stand for normTR based on 2-norm, infinity and Frobenius norm. Similarly, norm_2_E, norm_inf_E, and norm_Fr_E denote normE based on 2-norm, infinity and Frobenius norm.

We considered the win percentage of algorithms according to the above criteria. The win percentage value of algorithm A based on criterion C means the percentage of cases that algorithm A performs better than others based on criterion C.

4.2. Results on simulated datasets

In order to fully investigate the performance of AntMarkov, we compared its results with the results of four other methods, namely Baum-Welch, Viterbi-Training, SNNMF and Tabu-Search method on simulated datasets.

4.2.1. Validation on cases with complete associated graphs

For each first two datasets DS1 and DS2, we generated 432 different cases. All mentioned algorithms were applied to these datasets. It should be noted that SNNMF cannot handle 90 cases in both datasets, because their CharNum values were less than StateNum. We compare all algorithms on 342 remaining cases. Tables 1 and 2 represent summaries of performance quality of algorithms on the uniform and normal datasets.

It is crystal clear from these tables that AntMarkov has the highest win percentage based on all criteria in both datasets. Results show that in most cases estimations obtained by AntMarkov has the highest log-likelihood and better accuracy. Moreover, this algorithm saves time in most cases in comparison of other algorithms. Nevertheless, this is not the end of the analysis, since these 864 cases cannot cover the space of all possible cases. We probed other kinds of cases such as those with sparse or singular transition matrix or those with absorbing states for further

Table 2. Win percentage on normal data (DS2).

criteria	AntMarkov	Baum-Welch	Viterbi-Training	SNNMF	Tabu-Search
Log likelihood	**0.49**	0.10	0.01	0.32	0.08
Norm-2-TR	**0.39**	0.18	0.00	0.20	0.23
Norm-2-E	**0.62**	0.09	0.00	0.26	0.03
Norm-Inf-TR	**0.46**	0.12	0.00	0.30	0.11
Norm-Inf-E	**0.47**	0.02	0.01	0.45	0.05
Norm-Fr-TR	**0.41**	0.13	0.00	0.37	0.09
Norm-Fr-E	**0.47**	0.01	0.02	0.46	0.04
time	**0.46**	0.00	0.30	0.24	0.00
iteration	**0.77**	0.00	0.23		

Table 3. Win percentage on data with singular transition matrix (DS3).

criteria	AntMarkov	Baum-Welch	Viterbi-Training	SNNMF	Tabu-Search
Log likelihood	**0.41**	0.12	0.00	**0.41**	0.06
Norm-2-TR	**0.76**	0.00	0.00	0.24	0.00
Norm-2-E	**0.71**	0.06	0.00	0.24	0.00
Norm-Inf-TR	**0.76**	0.00	0.00	0.18	0.06
Norm-Inf-E	**0.88**	0.06	0.00	0.06	0.00
Norm-Fr-TR	**0.76**	0.00	0.00	0.24	0.00
Norm-Fr-E	**0.94**	0.00	0.00	0.06	0.00
time	0.35	0.00	**0.47**	0.18	0.00
iteration	**0.72**	0.00	0.28		

assessments. For further information about these calculated criteria on each case refer to Additional files 11 and 12. For an instance, one estimated model parameters by each algorithm are presented in Additional files 13 and 14. The comparison on all cases (without eliminating 90 cases) is provided in Additional file 15.

4.2.2. Validation on cases with singular transition matrices

DS3 with 30 cases with singular transition matrices were provided to assess this issue. A report of their performance quality is shown in Table 3. The win percentage of cases that AntMarkov and SNNMF obtain the model with the highest likelihood, are the same. Moreover, AntMarkov has succeeded to obtain more accurate estimations of parameters. The accuracy of estimation obtained by AntMarkov is outstanding. It can be seen that AntMarkov can win in more than 70% cases regarding norm-TR and norm-E. Consequently, the overall performance of AntMarkov is better than others in these cases. More details about these results are provided in Additional File 16. For an instance one estimated model parameters by each algorithm is presented in Additional files 17.

4.2.3. Validation on cases with sparse transition matrices

Since almost all of these algorithms use matrix computing in their methods, analyzing their performance for estimating sparse matrices is of high importance.

Table 4. Win percentage on data with sparse transition matrix (DS4).

criteria	AntMarkov	Baum-Welch	Viterbi-Training	SNNMF	Tabu-Search
Log likelihood	0.12	0.26	0.02	**0.54**	0.06
Norm-2-TR	**0.47**	0.10	0.01	0.15	0.27
Norm-2-E	**0.45**	0.00	0.00	0.22	0.33
Norm-Inf-TR	**0.57**	0.09	0.00	0.18	0.16
Norm-Inf-E	**0.60**	0.00	0.00	0.21	0.19
Norm-Fr-TR	**0.48**	0.16	0.01	0.12	0.22
Norm-Fr-E	**0.58**	0.00	0.00	0.25	0.17
time	**0.43**	0.00	0.35	0.22	0.00
iteration	**0.60**	0.00	0.40		

Table 5. Win percentage on data with random number of absorbing states (DS5).

criteria	AntMarkov	Baum-Welch	Viterbi-Training	SNNMF	Tabu-Search
Log likelihood	0.19	**0.56**	0.00	0.25	0.00
Norm-2-TR	0.19	**0.31**	0.06	0.25	0.06
Norm-2-E	**0.38**	0.31	0.00	0.19	0.13
Norm-Inf-TR	0.13	**0.44**	0.13	0.00	0.13
Norm-Inf-E	**0.38**	0.31	0.00	0.19	0.13
Norm-Fr-TR	0.25	**0.38**	0.06	0.06	0.13
Norm-Fr-E	**0.44**	0.31	0.00	0.25	0.00
time	0.00	0.00	**0.56**	0.00	0.00
iteration	**0.81**	0.00	0.19		

We applied the mentioned methods on DS4 to accomplish this evaluation. Table 4 summarizes the performance of other algorithms. These results express that SNNMF can estimate the most probable estimations in more cases than other algorithms on these datasets. However, the accuracy of parameters estimated by AntMarkov is prominent. In sum, AntMarkov succeeded to obtain good results in these cases. Extra information about these results are provided in Additional file 18 and one estimated parameter instance for each algorithm is provided in Additional file 19.

4.2.4. Validation on cases with absorbing states

We examined the performance of algorithms for estimating parameters of absorbing HMMs. A report of performance quality of algorithms is presented in Table 5. It can be inferred that AntMarkov can estimate the emission matrices accurately, but it cannot obtain an accurate estimation of transition matrices and BaumWelch has a better win percentage for these matrices. Thus, the overall performance of BaumWelch algorithm according to log-likelihood and accuracy of estimated transition matrix is preferable in cases with absorbing HMM. More details about algorithm results are provided in Additional file 20 and the first estimated matrices of each algorithm are provided in Additional file 21.

4.2.5. *Summary of results on simulated data*

Taking all results of previous sections into account, AntMarkov has a superior performance in most conditions, such as the cases with complete transition graphs with the probabilities generated from uniform or normal distributions. Furthermore, AntMarkov has the best performance in cases with sparse transition matrix and singular transition matrix. It should be mentioned that in cases with sparse transition matrix, SNNMF obtained best log likelihood, while AntMarkov yield more accurate estimations.

SNNMF can obtain fairly good results and in most cases the performance of SNNMF is the second bast. In cases sparse transition matrix and singular transition matrix, SNNMF obtained log likelihood and fairly good accuracy.

BaumWelch performance was the best with absorbing states, while it cannot get good results in other cases.

Viterbi training performed poorly on almost all cases; this is also true about Tabu-search method.

4.3. Application on protein secondary structure prediction problem

One can conclude from the previous section that AntMarkov can estimate HMM parameters in simulated data accurately and efficiently. Nevertheless, it is precious to assess its performance in a natural application. Prediction of protein secondary structure is an important step towards predicting its three-dimensional structure, analyzing protein function and applications like drug design. This is a challenging problem in bioinformatics. It is widely accepted that amino acid sequence determines protein shape [43]. It means the primary structure of proteins uniquely determining their secondary structure. Three major secondary structure elements are helix (H), Sheet (S) and coil (C). The coil class is default descriptions for those amino acids do not belong to helix or sheet classes. In the secondary structure prediction, we usually assign a structural state from a three-letter alphabets H, S, C to each amino acid. An HMM with StateNum=3 and CharNum=20 is considered for this application. Each protein sequence is considered as a sequence of observation and its path denote the secondary structure. The dataset used for training HMM was CullPDB [44] that contains 1763 proteins generated by the PISCES server with the identity of less than 30%. The secondary structures of these proteins were derived from PDB [45]. We used 5 fold cross validation for assessing the quality of estimation. The dataset was divided randomly into 5 parts. In each fold, HMM parameters were learned by 4 parts and then the obtained HMM was tested on the remaining part. One of

Table 6. Average on 5-fold cross-validation on CullPDB (1763 Protein).

criteria	AntMarkov	Baum-Welch	Viterbi-Training	SNNMF	Tabu-Search
Time	**0.023**	679.149	15.280	5.259	1.259
Q3-Helix	0.332	0.298	0.327	0.146	**0.333**
Q3-Sheet	0.338	0.349	**0.352**	0.340	0.337
Q3-Coil	0.335	0.360	0.324	**0.552**	0.329
Q3-Overall	0.335	0.336	0.334	**0.346**	0.334
Prc-Helix	**0.346**	0.344	0.321	0.344	0.345
Prc-Sheet	0.230	0.230	0.225	**0.232**	**0.232**
Prc-Coil	0.427	0.433	**0.475**	0.422	0.423
Prc-Overall	0.335	0.334	**0.340**	0.321	0.333
F-m:Helix	**0.326**	0.282	0.285	0.202	0.325
F-m:Sheet	**0.264**	246	0.237	0.256	0.259
F-m:Coil	0.357	0.351	0.312	**0.469**	0.348
F-m:Overall	0.335	0.335	**0.337**	0.309	0.333

the estimated parameters by each algorithm is presented in Additional file 22. For each sequence in the last part, the path finding (PF) problem was solved and the answer was considered as estimated secondary structures by HMM. One can see one instance of the predicted structure suggested by each algorithm in Additional file 23. We compared the estimated structures with the original ones derived from PDB. The assessment has been done according to the following criteria:

- Q3 (Recall): $\frac{TP}{TP+FN}$
- Precision: $\frac{TP}{TP+TN}$
- F-measure: $\frac{2\times Precision\times Recall}{Precision+Recall}$

In Table 6, Prc stand for Precision and F-m denotes F-measure.

These criteria can be calculated for each structure (H, E, C), separately. For example, the Q3 criterion based on helix is the number of truly detected helices over the sum of correctly detected helices and false detected non-helices.

$$Q3-Helix=\frac{TP_{Helix}}{TP_{Helix}+FN_{Helix}} \tag{21}$$

Other criteria such as Q3-Sheet, Q3-Coil, Prc-helix, Prc-Sheet, $\cdots$ are defined similarly. The overall criteria (Q3-Overall, Prc-Overall, and F-m-Overall) are computed by averaging these criteria in each structure. For example,

$$Q3-Overall=\frac{Q3-Helix+Q3-Sheet+Q3-Coil}{3}. \tag{22}$$

We compute all mentioned criteria for each algorithm based on all permutations of states. Finally, the reported criteria of each algorithm are selected based on the permutation that has the maximum F-measure-overall. Unfortunately, some papers have considered only Q3 for their analysis, while this is not sufficient. Considering Precision and especially F-measure besides Q3 can help us to conduct a fairer and better assessment. Precision and Q3 have trade-off and F-measure is a harmonic average of Precision and Q3. So F-measure can suitably capture the quality of performance. It should be noted that in CullPDB dataset, the ratio of Helix, Sheet, and Coil is 35%, 23%, and 42%, respectively. Since the partition of whole datasets is conducted randomly, we repeated the five-fold cross-validation on CULLPDB dataset 100 times and computed Q3, Precision, and F-measure for each execution. Then, we calculated the total criteria by averaging on executions. For example, Q3-Helix for AntMarkov on a fold is calculated by averaging Q3-Helix on each of 100 executions of AntMarkov on that fold. In this way, in each fold, all criteria can be computed for each algorithm. These results are provided in Additional File 24. F-m of algorithms in each fold is listed in Table 6. These tables can get us a summary on performance quality of each algorithm. It can be seen from the table that each of methods has good performance according to some criteria. But all criteria do not have equal worthiness. The last row of this table shows F-m-Overall which is the most expressive criterion; thus, it has the highest worthiness in our evaluation. Because Q3 and Precision have trade-off, i.e. maximizing one of them leads to the minimization of the other. Therefore, their geometric average which is F-measure, is more expressive than precision and recall. On the other hand, achieving good results on just one type of secondary structure such as helix, coil or strand is not good enough. The overall performance on all types of secondary structures shall be good. Put all these considerations into account, one can conclude that F-m-Overall is the most important criterion in this area.

It should be noted that SNNMF and Viterbi-training has high performance in these cases. According to the results, Viterbi-training is a reasonable method in this application. AntMarkov succeeded to obtain best results according to 4 criteria. Although, it cannot obtain good F-m-overall. Its F-m-Overall is very close to the best F-m-Overall. Although AntMarkov cannot achieve the highest performance according to some criteria, it is a good alternative to predict protein secondary structure, since it has a highly good F-m-Overall.

5. Conclusions

We proposed a method for estimating HMM parameters in this paper and compared it with previous methods such as Baum-Welch, Tabu-Search, Viterbi-Training and SNNMF method on real data and simulated data. The evaluations are conducted in two steps: Simulation data and real data. This study showed that in most cases our algorithm outperformed other algorithms in matter of all criteria on simulation data and obtain good results on real data.

According to the results on simulation data, if the sequences are emitted from transition and emission matrices with a normal or uniform probability distribution and with no specification (like DS1 and DS2) or when the original transition matrix is singular (like DS3), AntMarkov is the best algorithm with regard to log-likelihood. Whereas if the original transition matrices are sparse (like DS4), SNNMF can obtain better results. On the other hand, if the HMM has absorbing states (like DS5), BaumWelch is preferred. With respect to time or accuracy of estimated parameters, according to the results on simulation datasets AntMarkov had the best performance on all datasets DS1, DS2, DS3, DS4 and DS5 and it succeeded to achieve estimations that are very close to their original values.

A summary of the performance of algorithms on all simulated datasets is as follows. It should be noted that the performance of Tabu-search on cases with singular transition matrix or on HMMs with absorbing states, was hopeless. Moreover, Viterbi Training usually obtained its final estimations quickly but the quality of its estimations was the worst with respect to likelihood and the difference of estimations with original parameters. The likelihood of BaumWelch results was relatively good; however, it took a lot of time to estimate parameters. Although SNNMF could obtain the results with a relatively high likelihood, its estimations were not very close to the original parameters. Conversely, AntMarkov could obtain results quickly with almost the highest likelihood among other algorithms and its estimations were very close to the original parameters. In sum, according to the results on simulated datasets, AntMarkov is the best alternative for estimating HMM parameters.

In the next stage of evaluation, we considered an application of HMM parameter estimation in a real biological problem. To do this, we applied the mentioned algorithm on the problem of estimating the secondary structure of proteins. The evaluation criteria for this application was Q3 (recall), Precision and F-measure. We applied algorithms on CullPDB dataset with 5 fold cross validation 100 times and computed the average of Q3, Precision, and F-measure. The results showed that Viterbi-training obtained best F-measure and AntMarkov succeeded to obtain a good average of F-measure, which is the most important and expressive criterion. Taking the good performance of AntMarkov in simulated datasets and high F-measure in real data, AntMarkov can be known as an efficient algorithm for estimating HMM parameters.

The source code of our algorithm is available in https://github.com/emdadi/HMMPE. We gather the implementation of other algorithms in this address. Also, we develop a program that gets the input sequences and executes the mentioned algorithms altogether and outputs all results.

Declarations

Author contribution statement

Akram Emdadi, Fatemeh A. Moughari, Fatemeh Y. Meybodi, Changiz Eslahchi: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.
First three authors contributed equally.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Competing interest statement

The authors declare no conflict of interest.

Additional information

Data associated with this study has been deposited at https://github.com/emdadi/HMMPE. Supplementary content related to this article has been published online at https://doi.org/10.1016/j.heliyon.2019.e01299.

References

[1] David Haussler, Anders Krogh, I. Saira Mian, K. Sjolander, Protein modeling using hidden Markov models: analysis of globins, in: Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences, vol. 1, 1993, IEEE, 1993, pp. 792–802.

[2] Richard Durbin, Sean R. Eddy, Anders Krogh, Graeme Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.

[3] Gunasekaran Manogaran, V. Vijayakumar, R. Varatharajan, Priyan Malarvizhi Kumar, Revathi Sundarasekar, Ching-Hsien Hsu, Machine learning based big data processing framework for cancer diagnosis using hidden Markov model and GM clustering, Wirel. Pers. Commun. 102 (3) (2018) 2099–2116.

[4] Alison C. Testa, James K. Hane, Simon R. Ellwood, Richard P. Oliver, CodingQuarry: highly accurate hidden Markov model gene prediction in fungal genomes using RNA-seq transcripts, BMC Genomics 16 (1) (2015) 170.

[5] Eugenio Marco, Wouter Meuleman, Jialiang Huang, Kimberly Glass, Luca Pinello, Jianrong Wang, Manolis Kellis, Guo-Cheng Yuan, Multi-scale chromatin state annotation using a hierarchical hidden Markov model, Nat. Commun. 8 (2017) 15011.

[6] Sangjoon Kim, Neil Shephard, Siddhartha Chib, Stochastic volatility: likelihood inference and comparison with ARCH models, Rev. Econ. Stud. 65 (3) (1998) 361–393.

[7] Peter Nystrup, Henrik Madsen, Erik Lindstrom, Long memory of financial time series and hidden Markov models with time varying parameters, J. Forecast. 36 (8) (2017) 989–1002.

[8] Joseph Felsenstein, Gary A. Churchill, A hidden Markov model approach to variation among sites in rate of evolution, Mol. Biol. Evol. 13 (1) (1996) 93–104.

[9] Russell Corbett-Detig, Rasmus Nielsen, A hidden Markov model approach for simultaneously estimating local ancestry and admixture time using next generation sequence data in samples of arbitrary ploidy, PLoS Genet. 13 (1) (2017) e1006529.

[10] Tsong-Yi Chen, Xiao-Dan Mei, Jeng-Shyang Pan, Sheng-He Sun, Optimization of HMM by the tabu search algorithm, J. Inf. Sci. Eng. 20 (5) (2004) 949–957.

[11] Carl Mattfeld, Implementing spectral methods for hidden Markov models with real-valued emissions, arXiv preprint, arXiv:1404.7472, 2014.

[12] Marco Dorigo, Thomas Stutzle, Ant colony optimization: overview and recent advances, in: Handbook of Metaheuristics, Springer, Cham, 2019, pp. 311–351.

[13] Christian Blum, Ant colony optimization: introduction and recent trends, Phys. Life Rev. 2 (4) (2005) 353–373.

[14] Bert Holldobler, Edward O. Wilson, The Ants, Harvard University Press, 1990.

[15] Yi Zhou, Fazhi He, Neng Hou, Yimin Qiu, Parallel ant colony optimization on multi-core SIMD CPUs, Future Gener. Comput. Syst. 79 (2018) 473–487.

[16] Orhan Engin, Abdullah Guclu, A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems, Appl. Soft Comput. 72 (2018) 166–176.

[17] aban Gulcu, Mostafa Mahi, Omer Kaan Baykan, Halife Kodaz, A parallel cooperative hybrid method based on ant colony optimization and 3-Opt

algorithm for solving traveling salesman problem, Soft Comput. 22 (5) (2018) 1669–1685.

[18] Chieh-Yuan Tsai, Hui-Ting Chang, Ren Jieh Kuo, An ant colony based optimization for RFID reader deployment in theme parks under service level consideration, Tour. Manag. 58 (2017) 1–14.

[19] Nor H.B. Abdul Kahar, Ahmed F. Zobaa, Application of mixed integer distributed ant colony optimization to the design of undamped single-tuned passive filters based harmonics mitigation, Swarm Evol. Comput. (2018).

[20] Priyan Malarvizhi Kumar, Usha Devi, Gunasekaran Manogaran, Revathi Sundarasekar, Naveen Chilamkurti, Ramachandran Varatharajan, Ant colony optimization algorithm with Internet of vehicles for intelligent traffic control system, Comput. Netw. 144 (2018) 154–162.

[21] Abbas Afshar, Fariborz Massoumi, Amin Afshar, Miquel A. Marino, State of the art review of ant colony optimization applications in water resource management, Water Resour. Manag. 29 (11) (2015) 3891–3904.

[22] Fahimeh Farahnakian, Adnan Ashraf, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Ivan Porres, Hannu Tenhunen, Using ant colony system to consolidate VMs for green cloud computing, IEEE Trans. Serv. Comput. 8 (2) (2015) 187–198.

[23] Jiaxu Ning, Qin Zhang, Changsheng Zhang, Bin Zhang, A best-path-updating information-guided ant colony optimization algorithm, Inf. Sci. 433 (2018) 142–162.

[24] Zeineb Abdmouleh, Adel Gastli, Lazhar Ben-Brahim, Mohamed Haouari, Nasser Ahmed Al-Emadi, Review of optimization techniques applied for the integration of distributed generation from renewable energy sources, Renew. Energy 113 (2017) 266–280.

[25] Qiang Yang, Wei-Neng Chen, Zhengtao Yu, Tianlong Gu, Yun Li, Huaxiang Zhang, Jun Zhang, Adaptive multimodal continuous ant colony optimization, IEEE Trans. Evol. Comput. 21 (2) (2017) 191–205.

[26] Jose M. Cecilia, Antonio Llanes, Jose L. Abellan, Juan Gomez-Luna, Li-Wen Chang, Wen-Mei W. Hwu, High-throughput ant colony optimization on graphics processing units, J. Parallel Distrib. Comput. 113 (2018) 261–274.

[27] Arvind Lal, C. Rama Krishna, Critical path-based ant colony optimization for scientific workflow scheduling in cloud computing under deadline constraint, in: Ambient Communications and Computer Systems, Springer, Singapore, 2018, pp. 447–461.

[28] Ion Mandoiu, Alexander Zelikovsky, Bioinformatics Algorithms: Techniques and Applications, vol. 3, John Wiley & Sons, 2008.

[29] Jiyi Xiao, Lamei Zou, Chuanqi Li, Optimization of hidden Markov model by a genetic algorithm for web information extraction, in: ISKE-2007 Proceedings, 2007.

[30] Kwang-Eun Ko, Seung-Min Park, Jun-Heong Park, Kwee-Bo Sim, Training HMM structure and parameters with genetic algorithm and harmony search algorithm, J. Electr. Eng. Technol. 7 (1) (2012) 109–114.

[31] Jong-Seok Lee, Cheol Hoon Park, Training hidden Markov models by hybrid simulated annealing for visual speech recognition, in: IEEE International Conference on Systems, Man and Cybernetics, vol. 1, 2006, SMC'06, IEEE, 2006, pp. 198–202.

[32] D. Paul, Training of HMM recognizers by simulated annealing, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 10, ICASSP'85, IEEE, 1985, pp. 13–16.

[33] Sheng Wang, Jian Peng, Jianzhu Ma, Jinbo Xu, Protein secondary structure prediction using deep convolutional neural fields, Sci. Rep. 6 (2016) 18962.

[34] Quan Le, Fabian Sievers, Desmond G. Higgins, Protein multiple sequence alignment benchmarking through secondary structure prediction, Bioinformatics 33 (9) (2017) 1331–1337.

[35] Xuhua Xia, Hidden Markov models and protein secondary structure prediction, in: Bioinformatics and the Cell, Springer, Cham, 2018, pp. 145–172.

[36] James A. Cuff, Geoffrey J. Barton, Application of multiple sequence alignment profiles to improve protein secondary structure prediction, Proteins, Struct. Funct. Bioinform. 40 (3) (2000) 502–511.

[37] Yehong Chen, Yihui Liu, Jinyong Cheng, Yanchun Wang, Prediction of protein secondary structure using SVM-PSSM classifier combined by sequence features, in: Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC, 2016, IEEE, 2016, pp. 103–106.

[38] Seyed Amir Malekpour, Sima Naghizadeh, Hamid Pezeshk, Mehdi Sadeghi, Changiz Eslahchi, Protein secondary structure prediction using three neural networks and a segmental semi Markov model, Math. Biosci. 217 (2) (2009) 145–150.

[39] Wang Ding, Dongbo Dai, Jiang Xie, Huiran Zhang, Wu Zhang, Hao Xie, PRT-HMM: a novel hidden Markov model for protein secondary structure

prediction, in: 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, ICIS, 2012, IEEE, 2012, pp. 207–212.

[40] Zhiyong Wang, Feng Zhao, Jian Peng, Jinbo Xu, Protein 8-class secondary structure prediction using conditional neural fields, in: 2010 IEEE International Conference on Bioinformatics and Biomedicine, BIBM, IEEE, 2010, pp. 109–114.

[41] Gabor E. Tusnady, Istvan Simon, Principles governing amino acid composition of integral membrane proteins: application to topology prediction, J. Mol. Biol. 283 (2) (1998) 489–506.

[42] Bernard Robles, Manuel Avila, Florent Duculty, Pascal Vrignat, Frederic Kratz, Statistical evaluation of hidden Markov models topologies, based on industrial synthetic model, in: INCOM 2012, 14th IFAC Symposium on Information Control Problems in Manufacturing Information Control Problems in Manufacturing, vol. 14, no. 1, Elsevier Ltd on IFAC-PapersOnLine.net, 2012, pp. 1029–1034.

[43] Jeremy M. Berg, John L. Tymoczko, Lubert Stryer, J.M. Berg, J.L. Tymoczko, L. Stryer, Biochemistry: International Version (Hardcover), 2002.

[44] Guoli Wang, Roland L. Dunbrack Jr., PISCES: a protein sequence culling server, Bioinformatics 19 (12) (2003) 1589–1591.

[45] https://www.rcsb.org.