International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 6 (2014) pp. 484-493


# Distributed Learning Automata based Algorithm for Solving Maximum Clique Problem in Stochastic Graphs

**Mohammad Soleimani-Pouri [1], Alireza Rezvanian[2] and Mohammad Reza Meybodi[3]**

[1] Department of Electrical, Computer & Biomedical Engineering,
Qazvin branch, Islamic Azad University, Qazvin, Iran
m.soleimani@qiau.ac.ir

[2] Soft Computing Laboratory, Computer Engineering & Information Technology Department,
Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
a.rezvanian@aut.ac.ir

[3] Soft Computing Laboratory, Computer Engineering & Information Technology Department,
Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
mmeybodi@aut.ac.ir

***Abstract*: Many real world systems modeled as graph or networks, which the characteristics of interaction between vertices are stochastic and the probability distribution function of the vertex weight is unknown. Finding the maximum clique in a given graph is known as a NP-Hard problem, motivated by the social networks analysis. The maximum clique of an arbitrary graph G is the sub-graph C of G, Such that all vertices in C are adjacent in G and have maximum cardinality. In this paper an algorithm based on distributed learning automata is presented to solve maximum clique problem in the stochastic graph. Several experiments are designed to evaluate the proposed algorithm. Experimental results indicate that the proposed algorithm have a good performance in stochastic graph.**

***Keywords*:** maximum clique problem, NP-Hard, stochastic graph, learning automata, distributed learning automata, social networks.

## I. Introduction

Let $G=(V,E)$ be an undirected graph with vertex set $V=\{1, 2, \ldots, n\}$ and edge set $E \subseteq V \times V$, A clique [1]–[4]of $G$ is the set of vertices $C \subseteq V$, such that $i,j \in E$ for all $i,j \in C$. A maximum clique is a clique with maximum cardinality among all cliques of $G$. Due to its numerous applications, the maximum clique problem is one of the most important NP-hard problems [5] and it has been extensively studied in the literature such as clustering [6], [7] ,wireless sensor networks [8], social network analysis [9], etc.

One of the most interests in the networks applications is finding dense subsets of vertices such as clique, which represents a group of entities or people, any two of which have a certain type of relationship with each other in social networks [10].

In all existing methods for solving maximum clique, it is assumed that the graph is deterministic and thus the weight of its vertices fixed. But in the real world application, this assumption does not hold true, for example availability of people as nodes in the social networks or activity of routers in communication networks is varying over the time. So a stochastic is proposed based on this idea [11], in this paper the maximum clique problem in stochastic graphs is introduced, and then a Distributed learning automata-based algorithm is proposed for solving this problem, when the probability distribution function of the weight of the vertices is unknown. So far various methods have been proposed for this problem Including methods based on ant colony [12], immune genetic algorithm [13], reactive evolutionary algorithm[14], branch-and-bound algorithm [15], [16], etc. All methods presented cannot be used in new construction because they can be used only for certain structures.

To evaluate the performance of the proposed algorithm, the number of samples needs to be taken by it from the vertices of the stochastic graph is compared to that of the standard sampling method. According to the simulation results the proposed algorithm in terms of the number of samplings is acceptable. The rest of this paper is organized as follows. In the section II, learning automata is introduced. Distributed learning automata is described in section III. In section IV, Structure of the stochastic graphs are introduced. the proposed algorithm based on learning automata for solving maximum clique in stochastic graph is presented in section V. The performance of the proposed algorithm is evaluated through the simulation in section VI. Finally section VII concludes the paper.

## II. Learning Automata

A learning automaton [11], [17]–[20] is an adaptive decision making unit that improves its performance by learning how choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action is chosen at random based on a probability distribution kept over the action-set and at each instant the given action is served as the input to the random environment. The environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action-set so that the average penalty received from the environment is minimized.

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents the finite set of the inputs, $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ denotes the set of the values can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, ..., c_r\}$ denotes the set of the penalty probabilities, where the element $c_i$ is associated with the given action $\alpha_i$. If the penalty probabilities are constant, the random environment is said to be a stationary random environment, and if they vary with time, the environment is called a non-stationary environment. The environments depending on the nature of the reinforcement signal β can be classified into *P*-model, *Q*-model, and *S*-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as *P*-model environments. Another class of the environment allows a finite number of the values in the interval [0, 1] can be taken by the reinforcement signal. Such an environment is referred to as *Q*-model environment. In *S*-model environments, the reinforcement signal lies in the interval [a, b].

Learning automata can be classified into two main families: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple <β, α, *T*>, where β is the set of inputs, *Q* is the set of actions, and *T* is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $a(k)$ and $p(k)$ denote the action chosen at instant k and the action probability vector on which the chosen action is based, respectively. The recurrence equation shown by (1) and (2) is a linear learning algorithm by which the action probability vector $\underline{p}$ is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant *k*.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)\,p_j(n) \qquad \forall j, \quad j \neq i \end{aligned} \tag{1}$$

When the taken action is rewarded by the environment (i.e. $\beta(n)=0$) and

$$\begin{aligned} p_i(n+1) &= (1-b)\,p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)\,p_j(n) \qquad \forall j, \quad j \neq i \end{aligned} \tag{2}$$

When the taken action is penalized by the environment (i.e. $\beta(n)=1$). r is the number of actions which can be chosen by the automaton, $a(k)$ and $b(k)$ denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If $a(k) = b(k)$, the recurrence equations (1) and (2) are called linear reward-penalty ($L_{R-P}$) algorithm, if $a(k) >> b(k)$ the given equations are called linear reward-ε-penalty ($L_{R-\varepsilon P}$), and finally if $b(k) = 0$ they are called linear reward-inaction ($L_{R-I}$). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment. In the multicast routing algorithm presented in this paper, each learning automaton uses a linear reward-inaction learning algorithm to update its action probability vector.

## III. Distributed Learning Automata

A Distributed learning automata (DLA) [21] is a network of interconnected learning automata which collectively cooperate to solve a particular problem with many applications including: wireless Adhac networks[22] , data mining[23] , etc.

The number of actions for a particular LA in DLA is equal to the number of LA's that are connected to this LA. Selection of an action by a LA in the network activates one LA corresponding to the action. Formally, a DLA can be defined by a quadruple $\langle A, E, T, A_0 \rangle$, where $A=\{A_1, A_2, \ldots, A_n\}$ is the set of learning automata, $E \subset A \times A$ is the set of the edges in which edge $e_{i,j}$ corresponds to the action $\alpha_{ij}$ of the automaton $A_i$, T is the set of learning schemes with which the learning automata update their action probability vectors, and $A_0$ is the root automaton of DLA from which the automaton activation is started.

The operation of a DLA can be described as follows: At first, the root automaton randomly chooses one of its outgoing edges (actions) according to its action probabilities and activates the learning automaton at the other end of the selected edge. The activated automaton also randomly selects an action which results in activation of another automaton. The process of choosing the actions and activating the automata is continued until a leaf automaton (an automaton which interacts with the environment) is being reached. The chosen actions, along the path induced by the activated automata between the root and leaf, are applied to the random environment. The environment evaluates the applied actions and emits a reinforcement signal to the DLA. The activated learning automata along the chosen path update their action probability vectors on the basis of the reinforcement signal by using the learning schemes. The paths from the unique root automaton to one of the leaf automata are selected until the probability with which one of the chosen paths is close enough to unity. Each DLA has exactly one root automaton which is always activated, and at least one leaf automaton which is activated probabilistically. For example in the figure 3, every automaton has two actions. If automaton $A_1$ selects $\alpha_3$ from its action set, then it will be activate automaton of $A_3$. Afterward, automaton of $A_3$ will choose one of its possible actions and so on.

## IV. Stochastic Graph

As mentioned, in the most scenarios of network applications, the weight of the vertices of graph is assumed

to be fixed, but in real world applications this is not always true and it varies with time. So we introduce stochastic graph [24]–[26] for modeling the real networks applications.

A stochastic graph $G$ can be defined by a triple $G = \langle \boldsymbol{V}, \boldsymbol{E}, F \rangle$, where $V = \{v_1, v_2, ..., v_n\}$ denotes the set of vertices, $\boldsymbol{E} \subset \boldsymbol{V}\times\boldsymbol{V} = \{e_1, e_2, ..., e_m\}$ is the edge set, and $F=\{f_1, f_2, ..., f_n\}$ is the probability distribution describing the statistics of vertex weights. In particular, weight $w_i$ of vertex $v_i$ is assumed to be positive random variable with $f_i$ as its probability density function, which is supposed to be unknown for the proposed algorithm. In stochastic graph $G$, an maximum clique $\phi_i \subset V$ with weight of $W(v_i)$ vertices and expected weight of $\bar{W}(\phi_i)$ defined as $\Phi=\{\phi_1, \phi_2, ..., \phi_r\}$ the set of all its maximum clique such that for all arbitrary vertices of $v_i, v_j \in \gamma_i$, $v_i$ and $v_j$ are adjacent. The maximum clique is defined as an clique with maximum expected weight. In other word, stochastic maximum clique $\phi^*$ specifies as follow

$$\phi^* = \arg\max_{\forall \phi_i \in \Phi} \left\{ \bar{W}(\phi_i) \right\} \qquad (3)$$

where $\bar{W}(\phi_i)$ is the expected weight of the clique $\phi_i$ and the defined as below

$$\bar{W}(\phi_i) = \frac{\sum_{v_i \in \phi_i} \bar{W}(v_i)}{|\phi^t|} \qquad (4)$$

Where $\bar{W}(v_i)$ denotes the expected weight of vertex $v_i$. Therefore, the stochastic maximum clique of a given stochastic graph $G$ is defined as the stochastic clique with the maximum expected weight.

## V. Proposed algorithm

In this section, the proposed algorithm based on Distributed learning automata is described for solving the maximum clique problem in stochastic graphs. In this paper, weight of the vertices of the graph is assumed to be positive random variables and the parameters of the underlying probability distribution of the vertex weight are unknown. Therefore it is required to estimate the parameters by a statistical method. In the proposed algorithm, each vertices of graph, equipped with a learning automaton, indeed, a network of learning automata isomorphic to the stochastic graph. In this case, the network of automata can be formulated by a triple $<\underline{A}, \alpha, C>$, where $\underline{A}$ denotes the set of the learning automata, $\alpha$ is the set of actions in which $\alpha_i$ specifies the set of actions can be taken by learning automata $A_i$, for each $\alpha_i \in \alpha$, and $W=\{w_1,...,w_n\}$ is the set of weights such that $w_i$ ($\forall_{i \in \{1, 2, ..., n\}}$) is the random weight associated with automata $A_i$. The action set of each learning automata $v_i$ equals to its adjacent vertices of $v_i$. So the learning automaton assigned to each vertex $v_i$ of the stochastic graph, referred to as $\alpha_i$, has $n_i=(d_i-1)$ actions such that $\alpha_i=\{\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_{ni}\}$. To evaluate properly, At first , according to the central limit theorem each vertex is sampled 30 times, At each step of the algorithm, after sampling from some vertices and compute the expected weight of vertices, the candidate maximum clique is constructed by cooperation of learning automata. The learning automata iteratively construct the candidate maximum clique and update the action probability vectors until they find a near optimal solution to the maximum clique problem. The detail of the proposed algorithm is depicted as follow.

In the initialization, a learning automaton $A_i$ is assigned to each vertices $v_i$ of graph $G$ and action probability vector of them are initialized equal by $1/d_i$. Moreover, the candidate maximum clique consider as empty set ($\phi^0=\{\}$).

In the proposed algorithm, the following steps repeated until the stopping criteria are met. In this algorithm, the stopping criteria are considered as predefined total number of iterations and exceed the predefined threshold value of probability vector of the maximum clique as follows.

$$P(\phi^t) = \prod_{v_i \in \phi^t} p(v_i) \qquad (5)$$

Where $\phi^t$ denotes the set of vertices in the candidate maximum clique in the step $t$, $p(v_i)$ is the probability vector of the $v_i$.

**1.** All automata are activated and an automaton was selected randomly as $A_i$, and added into candidate maximum clique set afterward, all automata nonadjacent of $A_i$ is deactivated. Now, automaton $A_i$ chooses one of its actions according to its action probability vector, and deactivates nonadjacent automaton of $A_j$. Then, the new selected vertex ($j$) inserted in the candidate maximum clique set as $\phi^t$. This process repeated iteratively until there is no any active automaton.

**2.** Weight of the candidate maximum clique ($\phi^t$) which constructed in the step of $t$ is computed according to equation (6).

$$\bar{w}(\phi^t) = \frac{\sum_{v_i \in \phi^t} \bar{w}(v_i)}{|\phi^t|} \qquad (6)$$

Where $\bar{w}(v_i)$ and $\bar{w}(\phi^t)$ are the expected weight of vertex $v_i$ and the expected weight of clique $\phi^t$ respectively. $\phi^t$ specifies the vertex set of candidate maximum clique in the step of $t$ and $|\phi^t|$ denotes the cardinality of candidate maximum clique $\phi^t$.

**3.** The candidate maximum clique, which obtained in the step of $t$ in comparison with the best candidate maximum clique up to now is evaluated. If the cardinality of current candidate maximum clique is greater than the cardinality of the all candidate maximum clique that found until now, then all chosen learning automata are rewarded and candidate maximum clique was replaced by current maximum clique, otherwise the chosen learning automata are penalized.

According to the description, the pseudo code of proposed algorithm is presented in the figure 1.

**Proposed algorithm**

**Input** Stochastic Graph G⟨V, E, F⟩, Threshold probability P, Threshold iteration T
**Output** The stochastic maximum clique
**Assumptions**
Assign an automaton $A_i$ to each vertex $v_i$
Let $\phi^t$ denotes the vertex set of candidate maximum clique at step *t*
**Begin**
Let *t* denotes the step number and is initially set to 1
Let $\tau^t$ denotes the dynamic probability thresholdat step t
$\phi^t \leftarrow \{\}$
$\phi^* \leftarrow \{\}$
**While** (t<T **and** $\tau^t$< P) **Do**
Enable all learning automata
Select a learning automaton randomly as $A_i$
$\phi^t \leftarrow v_i$
**While** exist enabled LA **Do**
Disable all its non-adjacent $A_i$
Choose one of adjacent vertex of $v_i$by learning automaton $A_i$
$\phi^t \leftarrow \phi^t + v_j$
Enable $A_j$
$A_i \leftarrow A_j$
**End While**
**If** ($\hat{w}(\phi^t) \geq \hat{w}(\phi^*)$) Then
The chosen actions corresponding to edges of current maximum clique by all the learning automata are rewarded
$\phi^* \leftarrow \phi^t$
**Else**
The chosen actions corresponding to edges of current maximum clique by all the learning automata are penalized
**End If**
Calculate $\tau^t$ by equation (5)
$t \leftarrow t+1$
**End While**
**End Algorithm**

**Figure 1:** psoude code of proposed algorithm

The Best result for maximum clique in the stochastic graph obtained in the end of the algorithm.

## VI. Simulation Results

### *A. Experimental Study*

To evaluate the performance of the proposed algorithm, experiments are accomplished on the following stochastic graphs, which details of them are listed in [27], and are demonstrated in figure 2. These graph models are social networks, which the weight of activity/availability of vertices to be random variables. Stochastic graph descriptions have been presented in table 1.

*Table 1:* Stochastic graph description for experiments

| Graph name | Vertices | Edges | Type |
|---|---|---|---|
| Stnd-0.7-4-15-20 | 20 | 70 | Social networks |
| Stnd-0.8-4-15-30 | 30 | 102 | Social networks |
| Stnd-0.8-4-15-40 | 40 | 138 | Social networks |
| Stnd-0.8-4-15-50 | 50 | 247 | Social networks |
| Stnd-0.9-4-15-100 | 100 | 518 | Social networks |

### *B. Experimental Results*

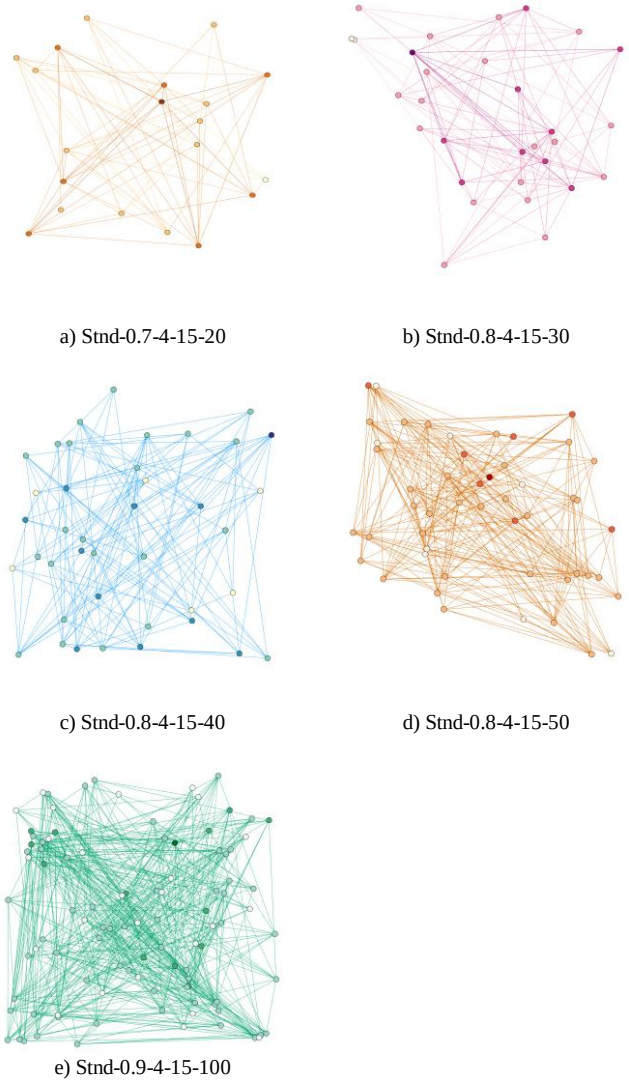
In all simulation presented in this paper, the number of samples taken by the proposed algorithm from the stochastic graph to construct the maximum clique is demonstrated. The updating scheme for action probability vectors of learning automata is linear reward-inaction ($L_{R-I}$). The stopping criteria are set to pre-defined number of steps (40000) and threshold value of function on probability vector. The number of total samples taken, iterations of each confidence level and average clique weights of sampling are listed in table 2 to 16 for stochastic graphs.

The results of proposed algorithm for averaged over 10 runs are presented.

The simulation results in the tables of 2 and 6 have demonstrated the average total samples for the maximum clique by the proposed algorithm, the effect of learning parameter of learning automata and confidence level is evaluated.

In another experiment, Average Iterations for stochastic graphs are presented in table 7 to 11, in these tables the effect of learning parameter of learning automata and confidence level are also listed.

The final test, average clique weights of sampling is presented; they are demonstrated in table 12 to 16.
According to evaluation results, the proposed algorithm is more inclined to maximum clique with a maximum average weight. In The first row of the tables shows the learning rate parameters and the first column describes the confidence levels (threshold).In these tables the effect of different values of learning parameter specifies the accuracy of algorithm with increasing the learning parameter in terms of average clique weight, average number of samples, and average iterations of algorithm.



a) Stnd-0.7-4-15-20 b) Stnd-0.8-4-15-30

c) Stnd-0.8-4-15-40 d) Stnd-0.8-4-15-50

e) Stnd-0.9-4-15-100

**Figure 2:** Stochastic graphs for experiments

*Table 2:* Average Total Samples of the proposed algorithm for *Stnd-0.7-4-15-20* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 758.8 | 679.9 | 651.8 | 639.5 | 630.6 | 630.2 | 625.1 | 623.9 | 621.2 | 616.6 |
| 0.45 | 791.9 | 696.6 | 664.5 | 646.5 | 639.2 | 632.4 | 627.7 | 624.2 | 625.0 | 620.7 |
| 0.50 | 831.6 | 714.8 | 675.2 | 659.3 | 648.4 | 640.5 | 632.7 | 629.0 | 628.4 | 630.4 |
| 0.55 | 871.7 | 734.7 | 692.2 | 666.6 | 655.2 | 648.1 | 638.6 | 635.7 | 632.1 | 627.7 |
| 0.60 | 920.4 | 759.3 | 708.0 | 679.6 | 664.2 | 651.2 | 648.8 | 642.2 | 635.5 | 632.9 |
| 0.65 | 971.7 | 788.1 | 725.4 | 692.7 | 675.2 | 664.0 | 651.7 | 647.6 | 639.1 | 633.8 |
| 0.70 | 1036.6 | 814.1 | 742.8 | 707.8 | 688.3 | 672.8 | 660.3 | 655.9 | 647.3 | 644.8 |
| 0.75 | 1107.2 | 852.0 | 767.1 | 727.6 | 700.1 | 684.1 | 671.8 | 663.3 | 655.6 | 650.6 |
| 0.80 | 1195.4 | 894.7 | 794.9 | 749.7 | 719.9 | 699.9 | 683.8 | 671.9 | 664.7 | 659.1 |
| 0.85 | 1311.2 | 956.3 | 834.6 | 775.5 | 740.5 | 714.7 | 699.7 | 689.4 | 677.4 | 666.3 |
| 0.90 | 1472.4 | 1035.1 | 888.0 | 815.8 | 771.9 | 744.2 | 724.8 | 707.5 | 696.1 | 684.7 |

*Table 3:* Average Total Samples of the proposed algorithm for *Stnd-0.8-4-15-30* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 1186.5 | 1044.7 | 996.8 | 971.5 | 956.7 | 950.1 | 940.7 | 933.8 | 931.8 | 927.6 |
| 0.45 | 1230.5 | 1067.6 | 1011.6 | 981.7 | 966.5 | 952.9 | 945.9 | 941.7 | 934.8 | 931.2 |
| 0.50 | 1278.0 | 1086.5 | 1025.1 | 994.6 | 974.7 | 963.0 | 954.4 | 945.8 | 940.1 | 936.9 |
| 0.55 | 1328.1 | 1114.0 | 1042.7 | 1006.9 | 984.5 | 968.7 | 960.6 | 951.5 | 944.5 | 943.0 |
| 0.60 | 1382.4 | 1141.3 | 1059.7 | 1019.6 | 995.5 | 977.8 | 965.6 | 959.1 | 952.2 | 946.6 |
| 0.65 | 1440.6 | 1171.7 | 1079.4 | 1035.8 | 1006.1 | 986.5 | 976.0 | 966.0 | 957.9 | 950.3 |
| 0.70 | 1509.5 | 1201.9 | 1102.3 | 1052.3 | 1021.7 | 1000.2 | 986.3 | 973.3 | 966.7 | 958.7 |
| 0.75 | 1591.9 | 1246.0 | 1128.1 | 1071.3 | 1036.3 | 1014.0 | 996.3 | 984.4 | 976.1 | 967.1 |
| 0.80 | 1686.2 | 1292.3 | 1160.9 | 1093.6 | 1053.7 | 1029.0 | 1011.8 | 993.8 | 985.0 | 974.5 |
| 0.85 | 1806.5 | 1352.9 | 1201.8 | 1124.5 | 1078.5 | 1048.5 | 1026.2 | 1010.0 | 996.6 | 985.6 |
| 0.90 | 1968.9 | 1435.2 | 1255.0 | 1166.3 | 1110.5 | 1073.8 | 1050.0 | 1030.1 | 1013.6 | 1002.5 |

*Table 4:* Average Total Samples of the proposed algorithm for *Stnd-0.8-4-15-40* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 1491.1 | 1345.7 | 1298.9 | 1270.8 | 1256.7 | 1248.3 | 1240.5 | 1234.0 | 1231.5 | 1228.3 |
| 0.45 | 1538.3 | 1368.1 | 1312.3 | 1283.2 | 1266.0 | 1256.2 | 1245.0 | 1239.5 | 1236.3 | 1233.4 |
| 0.50 | 1586.0 | 1396.2 | 1329.3 | 1297.6 | 1273.7 | 1262.5 | 1253.0 | 1247.0 | 1241.0 | 1237.7 |
| 0.55 | 1631.7 | 1419.4 | 1343.2 | 1308.3 | 1288.4 | 1269.1 | 1259.5 | 1253.0 | 1247.3 | 1241.4 |
| 0.60 | 1690.5 | 1444.3 | 1362.5 | 1322.2 | 1296.7 | 1278.9 | 1269.4 | 1257.9 | 1252.0 | 1246.8 |
| 0.65 | 1748.8 | 1473.0 | 1382.9 | 1336.0 | 1306.7 | 1291.6 | 1275.3 | 1267.2 | 1258.0 | 1253.4 |
| 0.70 | 1818.0 | 1510.7 | 1403.3 | 1354.5 | 1321.1 | 1301.0 | 1286.0 | 1275.4 | 1266.1 | 1257.7 |
| 0.75 | 1900.6 | 1544.1 | 1431.7 | 1373.2 | 1336.9 | 1313.8 | 1296.0 | 1283.6 | 1274.4 | 1264.8 |
| 0.80 | 1991.2 | 1594.1 | 1461.0 | 1393.1 | 1353.7 | 1329.4 | 1311.6 | 1295.1 | 1284.6 | 1274.5 |
| 0.85 | 2110.3 | 1652.6 | 1502.2 | 1424.4 | 1378.2 | 1349.9 | 1327.1 | 1309.8 | 1298.7 | 1287.0 |
| 0.90 | 2279.4 | 1733.9 | 1559.5 | 1464.9 | 1410.8 | 1375.1 | 1349.7 | 1329.7 | 1314.0 | 1301.8 |

*Table 5:* Average Total Samples of the proposed algorithm for *Stnd-0.8-4-15-50* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 1823.6 | 1661.0 | 1608.6 | 1580.6 | 1562.2 | 1552.9 | 1545.7 | 1539.6 | 1535.5 | 1530.2 |
| 0.45 | 1864.6 | 1684.8 | 1623.3 | 1589.0 | 1571.9 | 1560.1 | 1550.7 | 1544.1 | 1537.3 | 1535.0 |
| 0.50 | 1915.1 | 1705.8 | 1639.7 | 1601.5 | 1582.4 | 1567.9 | 1558.3 | 1550.4 | 1545.5 | 1538.9 |
| 0.55 | 1963.5 | 1733.3 | 1652.4 | 1614.8 | 1590.3 | 1576.4 | 1564.0 | 1557.5 | 1549.5 | 1543.4 |
| 0.60 | 2018.3 | 1756.7 | 1672.5 | 1630.3 | 1602.3 | 1584.2 | 1572.9 | 1563.7 | 1555.5 | 1549.4 |
| 0.65 | 2079.8 | 1789.3 | 1690.4 | 1643.1 | 1614.4 | 1594.0 | 1580.2 | 1570.5 | 1561.7 | 1554.2 |
| 0.70 | 2150.4 | 1819.8 | 1712.4 | 1657.9 | 1627.6 | 1606.2 | 1590.1 | 1579.5 | 1569.1 | 1562.2 |
| 0.75 | 2227.4 | 1861.2 | 1740.8 | 1682.1 | 1643.6 | 1618.2 | 1600.2 | 1587.2 | 1577.7 | 1569.5 |
| 0.80 | 2322.5 | 1907.9 | 1772.9 | 1702.5 | 1661.0 | 1635.8 | 1612.9 | 1599.4 | 1585.4 | 1577.6 |
| 0.85 | 2441.2 | 1969.4 | 1813.4 | 1731.6 | 1685.4 | 1654.9 | 1631.1 | 1613.9 | 1600.4 | 1590.1 |
| 0.90 | 2606.9 | 2050.3 | 1865.5 | 1773.5 | 1717.8 | 1680.0 | 1654.1 | 1634.0 | 1619.1 | 1607.0 |

*Table 6:* Average Total Samples of the proposed algorithm for *Stnd-0.9-4-15-100* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 3316.3 | 3160.7 | 3103.6 | 3075.8 | 3061.2 | 3050.3 | 3043.3 | 3034.6 | 3033.3 | 3027.1 |
| 0.45 | 3365.1 | 3178.3 | 3119.6 | 3088.1 | 3067.8 | 3056.4 | 3046.9 | 3040.9 | 3034.0 | 3029.7 |
| 0.50 | 3411.0 | 3202.3 | 3132.4 | 3099.0 | 3079.8 | 3063.1 | 3054.2 | 3047.8 | 3040.3 | 3035.8 |
| 0.55 | 3460.3 | 3228.5 | 3149.6 | 3112.4 | 3087.7 | 3071.7 | 3061.7 | 3052.1 | 3047.5 | 3043.1 |
| 0.60 | 3514.1 | 3253.6 | 3166.5 | 3126.3 | 3099.8 | 3079.5 | 3070.5 | 3060.0 | 3051.4 | 3047.7 |
| 0.65 | 3577.4 | 3286.0 | 3188.1 | 3138.8 | 3111.5 | 3091.7 | 3077.1 | 3066.6 | 3058.9 | 3052.0 |
| 0.70 | 3642.7 | 3303.1 | 3209.1 | 3154.1 | 3123.8 | 3102.1 | 3086.0 | 3074.1 | 3067.9 | 3059.6 |
| 0.75 | 3721.6 | 3357.4 | 3236.6 | 3176.2 | 3139.9 | 3114.1 | 3097.9 | 3082.4 | 3075.5 | 3066.4 |
| 0.80 | 3742.3 | 3404.3 | 3269.0 | 3198.7 | 3156.9 | 3131.1 | 3110.2 | 3094.8 | 3083.1 | 3075.6 |
| 0.85 | 3792.2 | 3464.1 | 3307.6 | 3228.8 | 3180.1 | 3150.7 | 3127.0 | 3109.5 | 3096.7 | 3088.7 |
| 0.90 | 3751.2 | 3549.8 | 3361.2 | 3271.8 | 3215.3 | 3177.2 | 3150.9 | 3129.8 | 3115.1 | 3103.7 |

*Table 7:* Average Iteration of the proposed algorithm for *Stnd-0.7-4-15-20* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 2402.3 | 1196.1 | 475.3 | 292.7 | 306.4 | 125.1 | 171.3 | 76.6 | 164.1 | 43.2 |
| 0.45 | 2526.7 | 1181.2 | 891.6 | 441.3 | 298.4 | 236.5 | 250.0 | 149.0 | 340.4 | 99.1 |
| 0.50 | 3413.8 | 1705.6 | 1218.1 | 683.1 | 607.9 | 553.2 | 299.5 | 138.6 | 146.3 | 137.4 |
| 0.55 | 4425.8 | 1902.5 | 1481.9 | 903.2 | 683.0 | 524.4 | 341.5 | 265.8 | 221.0 | 210.6 |
| 0.60 | 5197.1 | 2627.3 | 1720.8 | 1283.2 | 876.8 | 648.1 | 652.6 | 362.2 | 316.1 | 230.2 |
| 0.65 | 6286.2 | 3381.6 | 2279.3 | 1341.3 | 1125.1 | 1061.9 | 670.6 | 617.7 | 564.6 | 284.6 |
| 0.70 | 8105.7 | 3633.7 | 2796.7 | 1846.0 | 1232.5 | 897.2 | 774.5 | 596.2 | 494.7 | 411.8 |
| 0.75 | 8437.8 | 4523.2 | 2587.1 | 1882.6 | 1493.5 | 998.5 | 1058.5 | 1437.7 | 984.4 | 887.9 |
| 0.80 | 9922.3 | 5280.1 | 3417.2 | 2219.0 | 1991.8 | 1583.0 | 980.1 | 1415.4 | 753.9 | 677.7 |
| 0.85 | 12823.3 | 6470.3 | 4882.4 | 2802.7 | 2223.7 | 1866.3 | 1359.5 | 1641.9 | 1367.2 | 894.3 |
| 0.90 | 16927.5 | 7923.0 | 4761.2 | 3806.5 | 3122.1 | 2326.6 | 1971.7 | 1952.7 | 1470.4 | 1509.0 |

*Table 8:* Average Iteration of the proposed algorithm for *Stnd-0.8-4-15-30* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 3265.6 | 1705.2 | 904.2 | 692.0 | 501.6 | 316.2 | 211.7 | 221.5 | 187.3 | 129.4 |
| 0.45 | 3851.2 | 2024.2 | 1173.2 | 897.8 | 545.6 | 389.5 | 310.1 | 256.4 | 191.7 | 180.7 |
| 0.50 | 4433.8 | 2234.4 | 1324.6 | 836.0 | 581.2 | 462.5 | 375.6 | 372.4 | 277.5 | 287.7 |
| 0.55 | 5067.2 | 2248.9 | 1375.8 | 981.6 | 803.8 | 515.0 | 501.3 | 373.7 | 291.0 | 171.7 |
| 0.60 | 5863.2 | 2710.9 | 1838.8 | 1087 | 769.2 | 575.1 | 640.0 | 457.1 | 292.3 | 266.3 |
| 0.65 | 5878.9 | 2800.0 | 1591.5 | 1414.7 | 1067.0 | 761.1 | 540.1 | 577.4 | 510.5 | 261.2 |
| 0.70 | 6710.5 | 2911.0 | 2062.3 | 1242.7 | 1148.1 | 800.2 | 633.6 | 800.1 | 488.1 | 418.0 |
| 0.75 | 7090.3 | 3584.6 | 2224.4 | 1506.8 | 1241.8 | 1129.8 | 850.3 | 563.1 | 513.9 | 489.5 |
| 0.80 | 8116.5 | 3666.6 | 2639.9 | 1691.3 | 1444 | 1189.5 | 920.6 | 705.8 | 747.7 | 558.2 |
| 0.85 | 8956.5 | 4401.6 | 2830.2 | 2057.5 | 1409.3 | 1202.8 | 968.6 | 896.1 | 755.3 | 696.9 |
| 0.90 | 10383.8 | 5086.5 | 3411.1 | 2335.9 | 1827.3 | 1689.1 | 1315.5 | 1180.5 | 811.2 | 745.4 |

*Table 9:* Average Iteration of the proposed algorithm for *Stnd-0.8-4-15-40* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 3747.2 | 1848.1 | 915.2 | 708.0 | 460.2 | 375.5 | 330.2 | 238.2 | 281.0 | 126.6 |
| 0.45 | 3932.3 | 1981.2 | 1080.8 | 1037.5 | 533.5 | 410.3 | 399.5 | 277.0 | 211.1 | 318.2 |
| 0.50 | 4498.3 | 2087.1 | 1414.5 | 922.0 | 732.0 | 525.2 | 417.3 | 339.2 | 294.2 | 238.9 |
| 0.55 | 5594.3 | 2473.5 | 1465.5 | 894.1 | 1071.6 | 587.6 | 452.0 | 368.8 | 391.4 | 350.2 |
| 0.60 | 5605.2 | 2982.6 | 1609.4 | 1245.5 | 1062.7 | 612.1 | 655.8 | 443.4 | 390.5 | 335.7 |
| 0.65 | 6057.0 | 3078.5 | 2026.6 | 1577.5 | 1114.7 | 979.3 | 682.4 | 551.1 | 590.3 | 475.6 |
| 0.70 | 7008.8 | 3477.0 | 2095.5 | 1523.0 | 1273.1 | 1049.4 | 985.2 | 675.8 | 512.5 | 480.3 |
| 0.75 | 8087.6 | 3762.5 | 2401.3 | 1759.9 | 1164.9 | 1174.3 | 927.5 | 741.9 | 642.6 | 534.2 |
| 0.80 | 9074.1 | 4343.1 | 2943.8 | 1978.9 | 1581.5 | 1253.6 | 1076.5 | 1031.8 | 810.5 | 650.1 |
| 0.85 | 10295.9 | 5009.7 | 3250.7 | 2461.6 | 1595.7 | 1469.1 | 1159.7 | 1063.9 | 917.6 | 717.4 |
| 0.90 | 11888.4 | 5756.9 | 3832.8 | 3039.3 | 2195.0 | 1761.5 | 1451.4 | 1214.2 | 1042.0 | 969.9 |

*Table 10:* Average Iteration of the proposed algorithm for *Stnd-0.8-4-15-50* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 9051.5 | 3825.7 | 2551.4 | 1694.2 | 1661.1 | 682.3 | 722.7 | 636.9 | 491.3 | 262.6 |
| 0.45 | 11558.7 | 5205.2 | 2731.6 | 1917.4 | 1486.3 | 1103.0 | 873.3 | 517.4 | 439.4 | 329.2 |
| 0.50 | 11712.6 | 5203.5 | 3526.1 | 2080.5 | 1624.9 | 1270.1 | 1058.9 | 849.5 | 753.0 | 617.2 |
| 0.55 | 12237.4 | 6114.3 | 4315.6 | 2556.2 | 2107.0 | 1677.9 | 1404.3 | 857.6 | 734.2 | 423.6 |
| 0.60 | 14841.3 | 6279.9 | 4486.5 | 2843.6 | 2126.0 | 1629.2 | 1394.0 | 1119.1 | 1013.1 | 771.7 |
| 0.65 | 15436.6 | 7320.1 | 4754.5 | 3240.4 | 2790.8 | 2295.2 | 1601.6 | 1271.7 | 1234.2 | 1031.9 |
| 0.70 | 19821.9 | 8946.3 | 4657.1 | 3617.0 | 2734.3 | 2556.5 | 1918.9 | 2171.5 | 1196.4 | 1026.0 |
| 0.75 | 21097.8 | 10053.0 | 6478.9 | 4167.8 | 3150.5 | 2961.5 | 1824.2 | 1726.7 | 1500.1 | 1483.0 |
| 0.80 | 23132.2 | 10430.3 | 7339.6 | 4912.9 | 3582.7 | 3380.3 | 2429.7 | 2216.1 | 1730.7 | 1708.6 |
| 0.85 | 23953.1 | 12626.5 | 7663.5 | 7271.5 | 4370.5 | 4266.4 | 2772.1 | 2108.1 | 2483.3 | 2039.6 |
| 0.90 | 30673.1 | 15344.8 | 9271.5 | 6682.8 | 5262.7 | 4111.7 | 3865.6 | 3182.4 | 2717.1 | 2171.2 |

*Table 11:* Average Iteration of the proposed algorithm for *Stnd-0.9-4-15-100* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 18040.4 | 7190.4 | 4591.2 | 3359.3 | 2345.7 | 1391.0 | 958.2 | 1064.4 | 704.7 | 474.6 |
| 0.45 | 19421.9 | 9236.2 | 6048.3 | 3319.3 | 2692.5 | 2248.9 | 1759.7 | 1052.3 | 696.5 | 534.4 |
| 0.50 | 21841.7 | 10957.0 | 5984.4 | 4099.8 | 3041.7 | 2382.4 | 2753.8 | 1198.4 | 1276.5 | 734.1 |
| 0.55 | 25687.5 | 10823.7 | 6970.4 | 4987.2 | 3091.6 | 2581.0 | 2536.1 | 1400.5 | 1590.2 | 1019.0 |
| 0.60 | 26440.9 | 11403.2 | 7484.6 | 5603.9 | 4091.8 | 2932.7 | 2584.6 | 1735.2 | 1424.2 | 1208.7 |
| 0.65 | 28289.1 | 15433.3 | 9056.4 | 5564.4 | 4339.3 | 3047.5 | 2706.9 | 2431.5 | 2140.2 | 1739.6 |
| 0.70 | 32470.4 | 18864.6 | 10242.3 | 6469.7 | 5130.5 | 3997.8 | 3561.7 | 3967.0 | 2692.5 | 2903.2 |
| 0.75 | 34451.1 | 16816.4 | 10300.4 | 8076.3 | 5380.6 | 4565.0 | 4801.1 | 3123.1 | 3508.5 | 2014.7 |
| 0.80 | 39731.7 | 20028.9 | 12822.7 | 9926.8 | 7574.6 | 5811.8 | 4541.0 | 3240.2 | 3113.7 | 2950.2 |
| 0.85 | 40000+ | 21629.7 | 13357.0 | 11011.9 | 7070.7 | 6693.3 | 5771.5 | 4350.5 | 3610.4 | 3201.4 |
| 0.90 | 40000+ | 25779.0 | 16185.1 | 13069.7 | 9367.4 | 7423.2 | 6713.1 | 5076.0 | 4737.4 | 3527.1 |

*Table 12:* Average clique weight of the proposed algorithm for *Stnd-0.7-4-15-20* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 10.291 | 10.332 | 10.286 | 10.248 | 10.346 | 10.049 | 10.202 | 10.038 | 9.906 | 9.804 |
| 0.45 | 10.300 | 10.295 | 10.408 | 10.318 | 10.355 | 10.322 | 10.190 | 10.177 | 10.233 | 10.091 |
| 0.50 | 10.274 | 10.339 | 10.318 | 10.295 | 10.322 | 10.323 | 10.184 | 10.249 | 10.173 | 10.015 |
| 0.55 | 10.270 | 10.302 | 10.317 | 10.304 | 10.290 | 10.311 | 10.260 | 10.238 | 10.249 | 10.291 |
| 0.60 | 10.326 | 10.322 | 10.331 | 10.329 | 10.353 | 10.297 | 10.305 | 10.298 | 10.268 | 10.172 |
| 0.65 | 10.269 | 10.342 | 10.296 | 10.336 | 10.291 | 10.290 | 10.306 | 10.372 | 10.251 | 10.282 |
| 0.70 | 10.288 | 10.340 | 10.290 | 10.318 | 10.395 | 10.344 | 10.359 | 10.331 | 10.328 | 10.384 |
| 0.75 | 10.361 | 10.314 | 10.348 | 10.332 | 10.266 | 10.326 | 10.323 | 10.250 | 10.305 | 10.299 |
| 0.80 | 10.326 | 10.322 | 10.312 | 10.378 | 10.296 | 10.336 | 10.340 | 10.307 | 10.284 | 10.304 |
| 0.85 | 10.331 | 10.348 | 10.303 | 10.337 | 10.344 | 10.322 | 10.292 | 10.300 | 10.295 | 10.240 |
| 0.90 | 10.324 | 10.329 | 10.295 | 10.296 | 10.285 | 10.363 | 10.325 | 10.243 | 10.288 | 10.254 |

*Table 13:* Average clique weight of the proposed algorithm for *Stnd-0.8-4-15-30* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 10.695 | 10.730 | 10.782 | 10.690 | 10.828 | 10.781 | 10.837 | 10.990 | 10.529 | 10.522 |
| 0.45 | 10.729 | 10.787 | 10.750 | 10.768 | 10.738 | 10.784 | 10.613 | 10.482 | 10.560 | 10.538 |
| 0.50 | 10.778 | 10.817 | 10.797 | 10.906 | 10.818 | 10.705 | 10.609 | 10.781 | 10.776 | 10.793 |
| 0.55 | 10.751 | 10.757 | 10.806 | 10.803 | 10.792 | 10.875 | 10.731 | 10.819 | 10.882 | 10.613 |
| 0.60 | 10.826 | 10.739 | 10.625 | 10.851 | 10.661 | 10.700 | 10.898 | 10.617 | 10.702 | 10.750 |
| 0.65 | 10.801 | 10.828 | 10.813 | 10.780 | 10.863 | 10.783 | 10.711 | 10.739 | 10.805 | 10.811 |
| 0.70 | 10.772 | 10.745 | 10.780 | 10.678 | 10.735 | 10.755 | 10.773 | 10.849 | 10.789 | 10.813 |
| 0.75 | 10.741 | 10.771 | 10.702 | 10.708 | 10.734 | 10.737 | 10.705 | 10.664 | 10.884 | 10.901 |
| 0.80 | 10.740 | 10.709 | 10.778 | 10.726 | 10.817 | 10.817 | 10.825 | 10.780 | 10.696 | 10.726 |
| 0.85 | 10.767 | 10.757 | 10.797 | 10.781 | 10.728 | 10.831 | 10.742 | 10.792 | 10.874 | 10.612 |
| 0.90 | 10.774 | 10.820 | 10.731 | 10.736 | 10.812 | 10.687 | 10.709 | 10.777 | 10.701 | 10.775 |

*Table 14:* Average clique weight of the proposed algorithm for *Stnd-0.8-4-15-40* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 10.014 | 10.005 | 10.023 | 10.001 | 10.013 | 9.984 | 10.030 | 10.007 | 9.992 | 10.215 |
| 0.45 | 10.013 | 10.005 | 10.017 | 10.036 | 10.000 | 10.016 | 10.028 | 10.018 | 9.806 | 10.028 |
| 0.50 | 10.003 | 10.004 | 10.004 | 10.019 | 10.022 | 10.014 | 10.009 | 9.847 | 10.009 | 10.000 |
| 0.55 | 10.015 | 10.021 | 10.010 | 10.020 | 10.010 | 9.999 | 10.013 | 10.007 | 10.008 | 10.018 |
| 0.60 | 10.021 | 10.001 | 10.030 | 10.024 | 10.019 | 10.009 | 10.017 | 10.021 | 10.011 | 10.032 |
| 0.65 | 10.012 | 10.016 | 10.007 | 10.021 | 9.996 | 10.032 | 9.989 | 10.022 | 10.023 | 10.000 |
| 0.70 | 10.018 | 10.023 | 10.012 | 10.024 | 10.019 | 10.033 | 10.028 | 10.032 | 10.000 | 10.036 |
| 0.75 | 10.013 | 10.013 | 10.023 | 9.983 | 10.040 | 10.001 | 10.004 | 10.004 | 10.004 | 10.030 |
| 0.80 | 10.015 | 10.006 | 10.020 | 10.001 | 9.996 | 10.040 | 10.007 | 10.005 | 10.023 | 9.971 |
| 0.85 | 10.021 | 10.021 | 10.000 | 10.019 | 10.006 | 10.005 | 9.988 | 9.989 | 10.016 | 10.032 |
| 0.90 | 10.006 | 10.017 | 10.024 | 10.026 | 10.004 | 9.972 | 10.009 | 9.989 | 10.001 | 10.004 |

*Table 15:* Average clique weight of the proposed algorithm for *Stnd-0.8-4-15-50* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 10.491 | 10.533 | 10.528 | 10.425 | 10.583 | 10.492 | 10.415 | 10.494 | 10.277 | 10.188 |
| 0.45 | 10.541 | 10.508 | 10.420 | 10.524 | 10.493 | 10.501 | 10.464 | 10.442 | 10.489 | 10.488 |
| 0.50 | 10.469 | 10.526 | 10.387 | 10.451 | 10.459 | 10.497 | 10.421 | 10.373 | 10.559 | 10.584 |
| 0.55 | 10.467 | 10.483 | 10.421 | 10.436 | 10.496 | 10.525 | 10.453 | 10.460 | 10.443 | 10.503 |
| 0.60 | 10.549 | 10.436 | 10.522 | 10.533 | 10.496 | 10.455 | 10.507 | 10.545 | 10.496 | 10.502 |
| 0.65 | 10.446 | 10.425 | 10.423 | 10.448 | 10.464 | 10.459 | 10.453 | 10.531 | 10.537 | 10.462 |
| 0.70 | 10.462 | 10.441 | 10.419 | 10.424 | 10.478 | 10.464 | 10.545 | 10.472 | 10.431 | 10.635 |
| 0.75 | 10.472 | 10.498 | 10.420 | 10.405 | 10.406 | 10.513 | 10.461 | 10.443 | 10.452 | 10.424 |
| 0.80 | 10.450 | 10.507 | 10.470 | 10.510 | 10.519 | 10.497 | 10.596 | 10.547 | 10.504 | 10.560 |
| 0.85 | 10.434 | 10.525 | 10.471 | 10.510 | 10.517 | 10.515 | 10.520 | 10.523 | 10.496 | 10.549 |
| 0.90 | 10.502 | 10.423 | 10.463 | 10.507 | 10.477 | 10.539 | 10.512 | 10.495 | 10.502 | 10.593 |

*Table 16:* Average clique weight of the proposed algorithm for *Stnd-0.9-4-15-100* with different confidence level and different learning parameter

| Learning rate / Confidence | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.40 | 9.216 | 9.230 | 9.150 | 9.202 | 9.211 | 9.218 | 9.235 | 9.199 | 8.977 | 9.414 |
| 0.45 | 9.192 | 9.172 | 9.208 | 9.157 | 9.149 | 9.176 | 9.158 | 9.210 | 8.813 | 9.329 |
| 0.50 | 9.249 | 9.182 | 9.204 | 9.218 | 9.217 | 9.274 | 9.201 | 9.205 | 9.179 | 9.158 |
| 0.55 | 9.241 | 9.203 | 9.275 | 9.171 | 9.251 | 9.194 | 9.229 | 9.272 | 9.046 | 9.172 |
| 0.60 | 9.203 | 9.146 | 9.196 | 9.181 | 9.156 | 9.151 | 9.196 | 9.218 | 9.115 | 9.142 |
| 0.65 | 9.177 | 9.200 | 9.219 | 9.264 | 9.172 | 9.270 | 9.111 | 9.169 | 9.094 | 9.206 |
| 0.70 | 9.193 | 9.188 | 9.139 | 9.219 | 9.191 | 9.215 | 9.241 | 9.151 | 9.185 | 9.148 |
| 0.75 | 9.220 | 9.191 | 9.252 | 9.173 | 9.225 | 9.199 | 9.245 | 9.243 | 9.224 | 9.253 |
| 0.80 | 9.199 | 9.160 | 9.221 | 9.203 | 9.187 | 9.205 | 9.209 | 9.254 | 9.243 | 9.232 |
| 0.85 | 9.166 | 9.237 | 9.180 | 9.150 | 9.189 | 9.218 | 9.255 | 9.199 | 9.228 | 9.266 |
| 0.90 | 9.204 | 9.199 | 9.170 | 9.214 | 9.228 | 9.168 | 9.149 | 9.259 | 9.225 | 9.125 |

## VII. Conclusion

In this paper, an algorithm based on learning automata algorithm is proposed to solve the maximum clique in a stochastic graph. Based on the application of real networks, it is assumed that the probability distribution of the vertex weight is unknown. Moreover, in this paper, the stochastic maximum clique was introduced. According to the simulation results, the number of samples taken by the proposed algorithm is less than the standard sampling method for constructing the maximum clique in stochastic graph.

### References


[1] P. Martins, "Cliques with maximum/minimum edge neighborhood and neighborhood density," *Computers & Operations Research*, vol. 39, no. 3, pp. 594–608, 2012.

[2] Q. Wu, J.-K. Hao, and F. Glover, "Multi-neighborhood tabu search for the maximum weight clique problem," *Annals of Operations Research*, vol. 196, no. 1, pp. 611–634, 2012.

[3] D. Manrique, A. Rodríguez-Patón, and P. Sosík, "On the scalability of biocomputing algorithms: The case of the maximum clique problem," *Theoretical Computer Science*, vol. 412, no. 51, pp. 7075–7086, 2011.

[4] M. Brunato and R. Battiti, "R-EVO: A Reactive Evolutionary Algorithm for the Maximum Clique Problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 770 –782, 2011.

[5] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, vol. 40, no. 4, pp. 85–103, 1972.

[6] D. Duan, Y. Li, R. Li, and Z. Lu, "Incremental K-clique clustering in dynamic social networks," *Artificial Intelligence Review*, vol. 38, no. 2, pp. 129–147, 2012.

[7] S. Mimaroglu and M. Yagci, "CLICOM: Cliques for combining multiple clusterings," *Expert Systems With Applications*, vol. 39, no. 2, pp. 1889–1901, 2011.

[8] L. Wang, R. Wei, Y. Lin, and B. Wang, "A clique base node scheduling method for wireless sensor networks," *Journal of Network and Computer Applications*, vol. 33, no. 4, pp. 383–396, Jul. 2010.

[9] J. Pattillo, N. Youssef, and S. Butenko, "Clique Relaxation Models in Social Network Analysis," in *Handbook of Optimization in Complex Networks*, vol. 58, 2012, pp. 143–162.

[10] S. Wasserman, *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[11] M. Soleimani-Pouri, A. Rezvanian, and M. R. Meybodi, "Solving maximum clique problem in stochastic graphs using learning automata," in *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN),*, 2012, pp. 115–119.

[12] C. Solnon and S. Fenet, "A study of ACO capabilities for solving the maximum clique problem," *Journal of Heuristics*, vol. 12, no. 3, pp. 155–180, 2006.

[13] B.-D. Zhou, H.-L. Yao, M.-H. Shi, Q. Yue, and H. Wang, "An new immune genetic algorithm based on uniform design sampling," *Knowledge and Information Systems*, vol. 31, no. 2, pp. 389–403, 2012.

[14] R. Battiti and M. Protasi, "Reactive local search for the maximum clique problem 1," *Algorithmica*, vol. 29, no. 4, pp. 610–637, 2001.

[15] E. Tomita and T. Kameda, "An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments," *Journal of Global Optimization*, vol. 37, no. 1, pp. 95–111, 2007.

[16] J. Konc and D. Janezic, "An improved branch and bound algorithm for the maximum clique problem," *proteins*, vol. 4, p. 5, 2007.

[17] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Printice-Hall, 1989.

[18] A. Rezvanian and M. R. Meybodi, "Tracking Extrema in Dynamic Environments Using a Learning Automata-Based Immune Algorithm," in *Grid and Distributed Computing, Control and Automation*, vol. 121, Springer Berlin Heidelberg, 2010, pp. 216–225.

[19] A. Rezvanian and M. R. Meybodi, "LACAIS: Learning Automata based Cooperative Artificial Immune System for Function Optimization," in *Contemporary Computing*, vol. 94, Springer Berlin Heidelberg, 2010, pp. 64–75.

[20] A. Rezvanian and M. R. Meybodi, "An adaptive mutation operator for artificial immune network using learning automata in dynamic environments," in *Proceedings of the 2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, 2010, pp. 479–483.

[21] H. Beigy and M. R. Meybodi, "Utilizing distributed learning automata to solve stochastic shortest path problems," *International Journal of Uncertainty Fuzziness And Knowledge Based Systems*, vol. 14, no. 5, p. 591, 2006.

[22] J. A. Torkestani and M. R. Meybodi, "An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata," *Computer Networks*, vol. 54, no. 5, pp. 826–843, 2009.

[23] R. Forsati and M. R. Meybodi, "Effective page recommendation algorithms based on distributed learning automata and weighted association rules," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1316–1330, 2010.

[24] J. Akbari Torkestani and M. R. Meybodi, "A learning automata-based heuristic algorithm for solving the minimum spanning tree problem in stochastic graphs," *The Journal of Supercomputing*, vol. 59, no. 2, pp. 1035–1054, 2012.

[25] J. Akbari Torkestani and M. R. Meybodi, "Finding minimum weight connected dominating set in stochastic graph based on learning automata," *Information Sciences*, vol. 200, no. 1, pp. 57–77, 2012.

[26] J. A. Torkestani and M. R. Meybodi, "Learning automata-based algorithms for solving stochastic minimum spanning tree problem," *Applied Soft Computing*, vol. 11, no. 6, pp. 4064–4077, 2011.

[27] "AUT Soft Computing Resources," 2013. [Online]. Available: http://ceit.aut.ac.ir/softlab/Resources.html.

## Author Biographies

**Mohammad Soleimani-Pouri** received his B.S. degree in Computer Engineering (Software Engineering) from Islamic Azad University of Hamedan, Iran in 2010. Currently he is a M.S. student in Computer Engineering (Artificial Intelligence) at Department of Electrical, Computer & Biomedical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran. His research interests cover Social network analysis, Complex networks, Learning system and the related practical application in Graph Theory.

**Alireza Rezvanian** received his B.Sc. degree in Computer Engineering from Bu-Ali Sina University of Hamedan, Iran in 2007 and his masters in Computer Engineering from Islamic Azad University of Qazvin in 2010. Currently he is a Ph.D. student in Computer Engineering at the Computer Engineering & Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. His research activities include soft computing, evolutionary computation, complex networks, social networks, signal processing, and learning systems.

**Mohammad Reza Meybodi** received the B.S. and M.S. degrees in Economics from the Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degrees from the Oklahoma University USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include, channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.