# Solving Steiner Connected Dominating Set Using Distributed Learning Automata

Parastoo Jamehshourani
Computer Engineering Department
Islamic Azad University
Arak, Iran
p.jameshourani@yahoo.com

Mohammad Reza Meybodi
Computer Engineering Department
Amirkabir University of Technology
Tehran, Iran
mmeybodi@aut.ac.ir

Javad Akbari Torkestani
Computer Engineering Department
Islamic Azad University
Arak, Iran
j-akbari@iau-arak.ac.ir

***Abstract*— The Steiner Connected Dominating Set is a generalization of connected dominating set in which only one subset of the incoming graph nodes are dominated. This problem is considered to be one of the NP-hard problems in general graphs and even in Unit Disk Graphs. Various algorithms have been proposed for this problem. This paper proposes an algorithm based on Distributed Learning Automata to solve the problem. The proposed algorithm is compared with those by Wu, Guha and Muhammad. Results from the random graphs indicate that the proposed algorithm produces better results.**

***Keywords- Steiner Connected Dominating Set, Learning Automata, Distributed Learning Automata, Heuristic Algorithms***

## I. INTRODUCTION

The problem of dominating set (DS) in graph G(V,E) with vertices set of V and edges set of E is a subset of vertices like $S \subseteq V$ in a way that if there is not any vertex in this set, there should be at least on in the proximity of one of the nodes of this set [1]. Set S is called the dominating set and the other nodes are called dominated. Connected Dominating Set (CDS) is a dominating set whose vertices are connected to each other [1]. Steiner Connected Dominating Set (SCDS) problem [2] is a generalization of connected dominating set in which only one subset of the incoming graph nodes like $R \subseteq V$ are dominated. This problem was first introduced by Guha and Khuller. It is considered as one of the NP-hard problems in general graphs and even in Unit Disk Graphs (UDG). Unit Disk Graphs are crossover graphs of same-size circles on the disk [3]. One of the most important applications of SCDS problem is the creation of Virtual Backbone in mobile ad hoc networks [4].

In order to solve this problem, Guha and Khuller [2] introduced a greedy algorithm. The proposed algorithm uses Set Covering definition to find a dominating set and then applies Steiner Tree Approximation Algorithm to connect the newly-found DS. The approximation rate for the found set is at most $(c+1)H(\delta)+c-1)\times Opt$.

In this relationship, δ is the value of the biggest subset of R, c is the approximation rate of Steiner Tree (currently $c \simeq 1.55$ [5]) and H is the harmonic function. Wu et al [4] introduced an algorithm based on Maximal Independent set (MIS) [6]. This algorithm first creates a MIS like I on the nodes of set R and then applies Steiner Tree Algorithm on the vertices of this set. Since set I, is a MIS, all nodes of set R become dominated. Muhammad [7] offered a two-phase algorithm.

In the first phase, a MIS is computed for the vertices of set R and in phase two, a Steiner Tree is used to connect the found MIS. The algorithm proposed by this article is compared with those by Wu, Mohammad and Guha. Results indicate that the proposed algorithm produces better results in terms of the size of the created SCDS. However, in terms of the execution time, the mentioned algorithms are of better efficiency. The rest of the article is organized as follows: section II reviews Learning Automata and Distributed Learning Automata. Section III explains the proposed algorithm based on Learning Automata. In section IV, the proposed algorithm is compared with those by Wu, Muhammad and Guha while section V concludes the article.

## II. LEARNING AUTOMATA

Learning Automata [8] is an abstract model with limited number of action and probability of choosing action. Each chosen action is evaluated by a probable environment and a response is sent back to the environment. Learning Automata uses this response to choose an action for its next level [9]. Distributed Learning Automata (DLA) [10] is a network of learning automata that cooperate to solve a problem. The number of action of automata in a DLA is equal to the number of the learning automata attached to it. Officially, DLA can be defined with graph G = (V, E) in which V is the set of learning automata and $E \subset V \times V$ is the set of graph edges. Edge (i, j) demonstrates action j by the learning automata $LA_i$. Learning automata $LA_j$ will be active when action j of the learning automata is chosen.

An automaton with variable action set [11] is one that the number of its action is variable in every moment. Such automation chooses its actions in moment n only from a non-empty subset V (n) of actions which are called active actions. This choice is conducted by an extrinsic element and in a random manner. Automata works as follows: in order to choose an action in time n, learning automata computes first the sum of probability of its active actions

**978-1-61284-144-1/11/$26.00 ©2011IEEE**

K(n) and then vector $\hat{p}(n)$ according to the following relationship:

$$\hat{p}_i(n) = \text{prob}\,[\alpha(n) | V(n) \text{is set of active actions}, \alpha_i \in V(n)] = \frac{\hat{P}_i(n)}{K(n)} \quad (1)$$

Then, the automaton randomly chooses an action from its active actions in accordance with probability vector $\hat{p}(n)$ and applies it to the environment. If the chosen action is $\alpha_i$, the automata updates probability vector $\hat{p}(n)$ of its actions as follows after receiving the response of the environment.
If the environment response is desirable:

$$\hat{p}_i(n+1) = \hat{p}_i(n) + a(1 - \hat{p}_i(n)) \quad \alpha(n) = \alpha_i \quad (2)$$
$$\hat{p}_j(n+1) = (1-a)\,\hat{p}_j(n) \quad \alpha(n) = \alpha_j, \forall j \; j \neq i$$

If the environment response is not desirable:

$$\hat{p}_i(n+1) = (1-b)\,\hat{p}_i(n) \quad \alpha(n) = \alpha_i \quad (3)$$
$$\hat{P}_j(n+1) = \frac{b}{r-1} + (1-b)\hat{P}_j(n) \quad \alpha(n) = \alpha_j \quad \forall j, j \neq i$$

Then the automata updates actions probability vector $\hat{p}(n)$ as below:

$$P_j(n+1) = P_j(n+1) \quad \forall j, \alpha_j \in V(n) \quad (4)$$
$$P_j(n+1) = \hat{p}_j(n+1)K(n) \quad \forall j, \quad \alpha_j \in V(n)$$

## III. PROPOSED DISTRIBUTED LAERNING AUTOMATA ALGORITHM (PDLA)

In the proposed algorithm (PDLA), a network of learning automata corresponding to incoming graph is created and a learning automata is assigned to each node of the graph. At the beginning, the list of actions of each automata, the set of its neighbor nodes and the probability of each action are equal with other actions and is $\frac{1}{k}$ in which, variable k is the number of neighboring nodes to each automata. In this algorithm, learning automata are considered as $L_{RI}$ type. The steps of the proposed algorithm are described as follows:

**Step1:** The initial automata is assigned to the starter node. The ID of this node is added to Path list. This automata compare its action list with set R, if finds common node, that element will be eliminated from the set. This automata randomly chooses one of its actions as the ID of the next automata. The ID of selected action is added to Path list.

**Step2:** The activated automata compare its action list with set R, if finds common node that element delete from the set. This node randomly chooses one of its actions based on Check procedure. The ID of the chosen node will be added to the Path list.

**Step3:** the activation of automata (step2) continues until either set R is empty or all the automata are activate. By the end of this step, a spanning tree is created on the receiving nodes.

**Step4:** this tree prunes its leaves and what's left beneath is a SCDS. In every repetition, the expense underneath the tree is used to evaluate the actions chosen by the automata. If the size of the obtained SCDS is less than the best existing one, the activated automata reward the chosen actions; otherwise, they penalize them.

```
Algorithm PDLA
Input: Graph G (V, E)
Output: The number of nodes needs be chosen for Steiner connected dominating set
Begin

Initialize probability vector of each automaton
Repeat

 Current Node = Initial Node
 Path=Path + Current Node

 If (Current Node or its action list elements is (are) in set R)
Delete it from set R

 While (set R is not empty)
            Next Node = GetNextNode (Current Node)
            If (If the selected node does not create the
loop)
                Current Node = Next Node
                Path=Path +  Current Node
                 If (Current Node's action list have common
                node in set R)
                                Delete the action from set R
            Else
                Disable selected action
                Next Node = GetNextNode (Current Node)

End While

Call Pruning procedure to prune the created tree
Compute the cardinality of SCDS

If (the cardinality of the current SCDS < cardinality of the
best SCDS)
            All selected automata rewards their chosen
            actions
            Best SCDS = Current SCDS
Enable all the disabled action

Until (stop condition)
```

Figure 1- pseudo code of PDLA

**Step5:** forming and pruning the tree continues until multiplied value of the probabilities of chosen actions is bigger or equal to that of the threshold or the number of constructed SCDS exceeds a pre-specified threshold.

Check procedure takes place this way: current automata X chooses one of its actions like Y. However, in order to prevent the formation of cycle as well as to provide contexts in which a node (automata) is presented on the tree with more than one edge (action), the actions chosen by previous automata need to be examined. Thus, if node Y exists in the previous path, the current automata cannot choose action Y due to the formation of cycle and therefore, it deactivates it in its action list and randomly chooses another one.

## IV. EVALUATION

Proposed algorithm PDLA is compared with three algorithms introduced in this field i.e. those by Wu, Guha and Muhammad. Results of this comparison are demonstrated in the following diagrams.

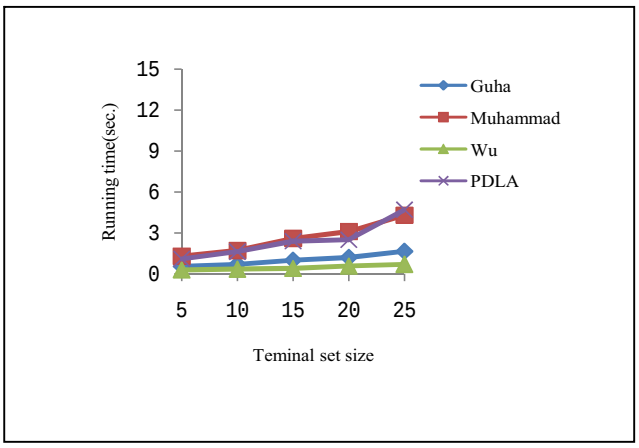


Figure 2- The size of SCDS on stochastic graph1

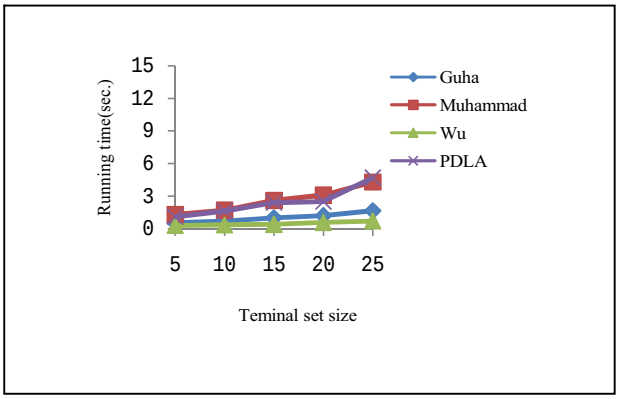


Figure 3- Time taken by different algorithm on stochastic graph1

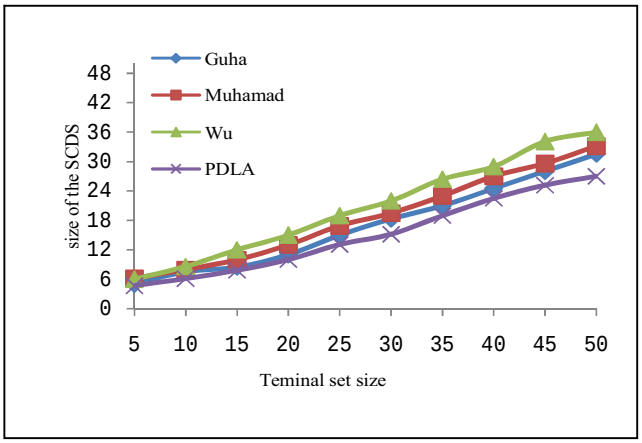


Figure 4- The size of SCDS on stochastic graph 2

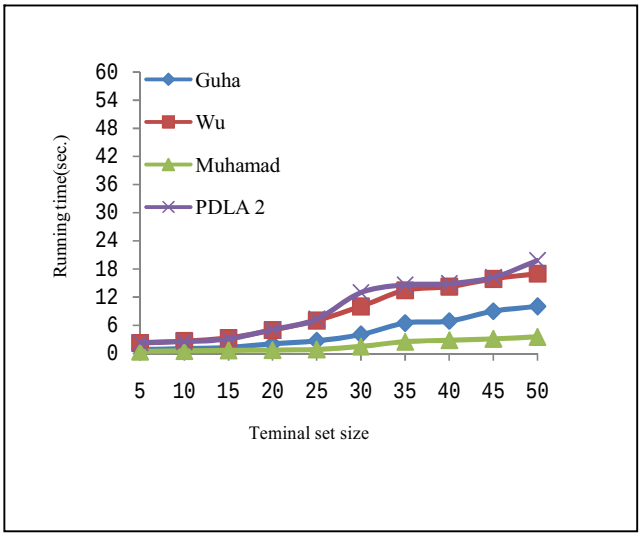


Figure 5- Time taken by different algorithm on stochastic graph 2

These algorithms are tested on random graphs. Parameters reward and penalty are considered 0.2 and 0, respectfully. The criteria for evaluating the algorithms were the size of the created Steiner Connected Dominating Set and the algorithm execution time.

Diagrams show that the proposed algorithm is more desirable than the existing ones in terms of the size of the created set. In terms of the execution time, however, algorithms by Guha and Wu yield better results.

## V. CONCLUSION

This paper introduced an algorithm based on learning automata to solve Steiner Connected Dominating Set problem. The proposed algorithm was compared with those by Wu, Guha and Muhammad. Results indicate that the proposed algorithm produces better results in terms of the size of the created set than these three algorithms.

## REFERENCES


[1] P. Santi, Topology Control in wireless Ad hoc and sensor networks, Wiley, 2006.

[2] S. Guha and S. Khuller, "Approximation algorithms for Connected Dominating Sets," *Algorithmica*, 1998, pp. 374-387.

[3] B.N. Clark, C.J. Colbourn and D.S. Johnson, "Unit Disc Graphs**",** Discrete Mathematics, 1990, pp. 165-177.

[4] Y.Wu, Y. Xu, G. Chen and K. Wang, "On the construction of virtual multicast backbone for wireless ad-hoc networks," *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004, pp. 294-303.

[5] M. Min, et al., "Improving Construction for Connected Dominating Set with Steiner Tree in Wireless Sensor Networks", *Journal of Global Optimization*, 2006, pp. 111-119.

[6] J. Abello, "Finding independent sets in a graph using continuous multivariable polynomial formulations", *Journal of Global Optimization*, 2001, pp.111-137.

[7] R. B. Muhammad, "Distributed Steiner Tree Algorithm and its Application in Ad-hoc Wireless Networks", *in Proceedings of th 2006 International Conference on Wireless Networks*, 2006, pp.173-178.

[8] K.s. Narendra and M.A.L. Tahtahchar, Learning Automata: An Introduction, Prentice Hall, 1989.

[9] C. Unsal, P. Kachroo and J.S. Bay, "Multiple Stochastic Learning Automata for Vehicle Path Control in an Automated Highway System", IEEE Transactions on Systems, Man and Cybernetics, 1999, chapter 3.

[10] M. R. Meybodi and H. Beigy, "*Solving Stochastic Shortest Path Problem.Using Monte Carlo Sampling* Method: A Distributed Learning Automata Approach", *in Proceedings of the sixth International Conference on Neural Networks and Soft Computing*, 2003, pp. 626-632.

[11] M.A.L. Tahtahchar and Harita, "Learning automata with changing number of actions", In IEEE Transactions on system, man and cybernetics, 1987.