Using Learning Automata to Solving Degree-constrained Minimum Spanning Tree Problem

Leila Safari and Azizallah Rahmati

Department of Computer, Kangavar Branch, Islamic Azad, University Kangavar, Iran.

Abstract: The Degree Constrained Minimum Spanning Tree on a graph is the problem of generating a minimum spanning tree with constraints on the number of arcs that can be incident to vertices of the graph. This problem arises naturally in communication networks where the degree of a vertex represents the number of line interfaces available at a center. Since this problem belongs to NP-hard problems, many approximate algorithms have been designed for solving it. This paper proposes an algorithm based on learning automata for solving Degree Constrained Minimum Spanning Tree Problem. Using simulation it has been shown that with proper selection of parameters of the learning automata, the proposed algorithm is superior to other existing algorithms.

Key words: degree constrained minimum spanning tree; learning automata; NP-hard.

INTRODUCTION

The problem of finding a Degree Constrained Minimum Spanning Tree (DCMST) of a graph is a well-studied NP-hard problem and important in the design of telecommunication networks, design of networks for computer communication, design of integrated circuits, energy networks, transportation, logistics, sewage networks, and plumbing for maximum network reliability, and optimality such as rerouting of traffic in case of vertex failures, and improve the network performance by distributing the traffic across many vertices (Krishnamoorthy, 2001). There might be constraints imposed on the design such as the number of vertices in a subtree, degree-constraints on vertices, flow and capacity constraints on any edge or vertex, and type of services available on the edge or vertex.

The Degree Constrained Minimum Spanning Tree problem is obtained by the modification of the MST problem from a given connected, edge weighted, undirected graph $G$, such that no vertex of the spanning tree has degree greater than $d$. It is because when $d = 2$, the MST meeting the constraint will take the form of a path. This is the path of least total weight which includes every vertex in the graph. In other words, this is a Hamiltonian path. Hence an algorithm which solves the DCMST problem also solves the Hamiltonian path problem, which is NP-complete. Therefore, the DCMST is NP-hard problem. The DCMST is in $P$ if $d = |V| - 1$, whereby $|V|$ is the number of vertices. When $d = |V| - 1$ there is no degree constraint and this is equal to MST problem that could be solved using a polynomial amount of computation time. Because of its complexity, we can apply exact optimization algorithms only to small instances of them. For larger instances, we turn to approximate algorithms. One type of approximate algorithms that has not been previously used to solve the DCMST problem is the learning automata. Learning automata is a metaheuristic approach for solving hard combinatorial optimization problems and have been used to solve minimum spanning tree problem in Stochastic Graphs (Akbari, 2009) minimum cost path problems and problems that can be reduced to a kind of shortest path problems (Beigy, 2006; Misra, 2006).

The d-MST problem was first studied by Deo and Hakimi in (1968). Since computing a d-MST is NP-hard for every $d$ in the range $2 \leq d \leq |V| - 2$, a few heuristics had been introduced to solve the d-MST problem such as ant colony optimization (Bau, 1968; Bui, 2006), branch and bound (Krishnamoorthy, 2001), evolutionary algorithms (Krishnamoorthy, 2001; Soak, 2004; Raidl, 2000), genetic algorithms (Krishnamoorthy, 2001; Raidl, 2000; Zhou, 1997) Lagrangean relaxation (Krishnamoorthy, 2001; Volgenant, 1989), parallel algorithms (Mao, 1999), problem space search (Krishnamoorthy, 2001), simulated annealing (Krishnamoorthy, 2001) and variable neighborhood search (Ribeiro, 2002). Soak, Corne and Ahn (Soak, 2004) have proposed a new encoding based on tree construction rule for DCMST to be used with EAs. The d-MST problem has also been studied for the complete graphs of points in a plane where edge costs are the Euclidean distance between these points coordinate. Euclidean problems are relatively simple to solve (Krishnamoorthy, 2001).
For these Euclidean problems there always exists a MST with degree no more than five and it is also being described in (Monma, 1992). Using exact algorithms such as branch and bound and Lagrangean relaxation as described by (Krishnamoorthy et al. 2001), one can find optimal solutions even for large problem instances including several hundred vertices in polynomial time. This showed that there exist effective polynomial-time heuristics for finding DCMST in the plane. In practice, the costs associated with the graph’s edge are arbitrary and need not satisfy the triangle inequality (Knowles, 2000).

Among the approaches that were used for our comparison study were Prüfer-coded evolutionary algorithm (F-EA) (Krishnamoorthy, 2001), problem search space (PSS) (Krishnamoorthy, 2001), simulated annealing (SA) (Krishnamoorthy, 2001), branch and bound (B&B) (Krishnamoorthy, 2001), Knowles and Corne’s evolutionary algorithm (K-EA) (Knowles, 2000), weight-coded evolutionary algorithm (W-EA) (Raidl, 2000), edge-set representation evolutionary algorithm (S-EA) (Raidl, 2000) and ant-based algorithm (AB) (Bui, 2006). (Krishnamoorthy et al., 2001) presented F-EA, PSS, SA and B&B for the DCMST problem. F-EA employs Prüfer coding based evolutionary algorithm (EA) and uses standard single point crossover. PSS is metaheuristic which combines a simple constructive heuristic with a genetic algorithm. They use Prim’s based heuristic to construct randomized minimum spanning tree. The SA approach provides a means to escape local optima by allowing some downhill moves during stochastic hill climbing in hopes of finding a global optimum. B&B is in general an exact technique. (Knowles and Corne 2000) described another EA for the DCMST problem. In their algorithm, chromosomes are sequences of integer values that influence the order on which edge vertices to connect to form the growing spanning tree. Raidl and Julstrom (Raidl, 2000) presented W-EA for the DCMST problem. In this W-EA approach, a feasible spanning tree is represented by a string of numerical weights associated with the vertices. (Raidl, 2000) also presented S-EA for the DCMST problem. In this S-EA approach, spanning trees in EA is represented directly as sets of their edges. Bui T. N. and Znecic C. M. (Bui, 2006) presented AB for the DCMST problem. In their algorithm they use cumulative pheromone levels to determine candidate sets of edges from which degree-constrained spanning trees are built.

However, in this paper we present an algorithm based on learning automata for solving degree-constrained minimum spanning tree problem.

The rest of the paper is organized as follows. The learning automata and degree constrained minimum spanning tree are described in section II. In section III, the proposed learning automata based algorithm are presented. Section IV compares the performance of our algorithm against existing algorithm. Section V concludes the paper.

**Learning Automata and Degree Constrained Minimum Spanning Tree:**

**Learning Automata:***

A learning automaton (Akbari, 2009; Beigy, 2006) is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. 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 \( \langle \mathcal{I}, \mathcal{A}, T \rangle \), where \( \mathcal{I} \) is the set of inputs, \( \mathcal{A} \) 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 \( p \) is updated. Let \( ai(k) \) be the action chosen by the automaton at instant \( k \).

\[
P_j(n+1)=p_j(n)+a[1-p_j(n)] \quad j=i \tag{1}
\]

\[
P_j(n+1)=(1-a)p_j(n) \quad \forall j \neq i
\]

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

\[
P_j(n+1)=(1-b)p_j(n) \quad j=i \tag{2}
\]

\[
P_j(n+1)=(b/r-1)+(1-b)p_j(n) \quad \forall j \neq i
\]

When the taken action is penalized by the environment (i.e., \( \beta(n) = 1 \)). \( r \) is the number of actions 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 Equation (1) and (2) are called linear reward-penalty \( (L \ R-P) \) algorithm, if \( a(k) >> b(k) \) the given equations are called linear reward- \( \varepsilon \) penalty \( (L \ R-\varepsilon P) \), and finally if \( b(k) = 0 \) they are called linear reward-inaction \( (L \ R-I) \).
the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment. In the DCMST algorithm presented in this paper, each learning automaton uses a linear reward-inaction learning algorithm to update its action probability vector.

**Degree Constrained Minimum Spanning Tree:**

The degree constrained minimum spanning tree (DCMST) problem can be stated as follows: Let graph \( G = (V, E, W) \) be a connected weighted undirected graph, where \( V = \{v_1, v_2, \ldots, v_n\} \) is a finite set of vertices, and \( E = \{e_{ij} \mid i \in V, j \in V, i \neq j\} \) is a finite set of edges representing connections between these vertices. Each edge has a nonnegative real number denoted by \( W = \{w_1, w_2, \ldots, w_{|E|}\} \), representing weight or cost. Note that in a complete graph having \(|V|\) vertices, the number of edges, \(|E|\), is \(|V|(|V| - 1)/2\), and the number of spanning trees is \(|V|^{(V|-2)}\). A spanning tree always consists of \(|V| - 1\) edges. Any subgraph of \( G \) can be described using a vector \( x = (x_1, x_2, \ldots, x_m) \) where each \( x_i \) is a binary decision variable defined as:

\[
x_i = 1 \quad \text{if edge } e_{ij} \text{ is part of the subgraph} \\
x_i = 0 \quad \text{otherwise}
\]

Let \( S \) be a subgraph of \( G \). \( S \) is said to be a spanning tree in \( G \) if \( S \): (a) contains all the vertices of \( G \) and the vertices can be in non-order form; (b) is connected, and graph contains no cycles.

Now let \( T \) be the set of all spanning trees corresponding to the simple graph \( G \). In the MST problem, if we assume that there is a degree constraint on each vertex such that the degree value \( d_j \) of vertex \( j \) is at most a given constant value \( d \), the number of edges incident to each vertex is constrained. Then the problem is denoted as a DCMST and can be formulated as follows:

\[
\min \{ z(x) = \sum_{j \in V} w_{ij} \mid j \in V, d_j \leq d, x \in T \}
\]

**The Proposed Dcmst Algorithm:**

In this section, we propose learning automata-based algorithms to solve the degree-constrained minimum spanning tree problem which focus on finding the spanning tree with the minimum expected weight in a degree constrained graph.

In proposed algorithm to each vertex of the graph one learning automata is allocated, this proposed algorithm is an iterative algorithm, at each step edges of constrained minimum spanning tree has been chosen by learning automata by randomly. The algorithm will be iterated until the weight of the constructed tree is less than the define amount or the number of iterates are more than the critical amount.

The presented algorithm has this potentiality that in every situation in which threatening the characteristics of being a tree, spanning and degree constrained, stop the procedure of performing that algorithm in that step. So during the proceeding steps it is guaranteed that the answer is a spanning tree with a constrained degree. So, only the condition of minimum of the constrained spanning tree after the algorithm constructed will be examined, this feature cases the algorithm to find the best answer in an appropriate time. The constrained spanning tree which is selected just before the algorithm stops is the constrained spanning tree with the minimum expected weight among all the constrained spanning trees of the graph. Figure 1 shows the pseudo code of the proposed algorithm.

**Algorithm : Degree-constrained Minimum Spanning Tree Problem:**

1. Input: Graph \( G=\langle V, E, d \rangle \)
2. Output: The Degree Constrained Minimum Spanning Tree
3. Assumptions
4. Assign a learning automaton \( A_i \) to each vertex \( V_i \)
5. Begin Algorithm
6. \( K = 0 \)
7. Repeat
8. \( W = 0 \), \( t = 0 \)
9. The first automaton is randomly selected, denoted as \( A_1 \) and activated
10. Repeat
11. If \( A_i \) has no possible actions Then
12. Path induced by activated automata is traced back for finding an automaton with available actions The found learning automaton is denoted as \( A_i \)
13. End If
14. If(d<degree of Ai)
15. Automaton Ai prunes its action set
16. Update action probability vectors
17. End If
18. Automaton Ai chooses one of its actions
19. W\rightarrow w + \omega(A_i,A_j))
20. t\rightarrow t + e(A_i,A_j)
21. Each automaton prunes its action set to avoid the loop
22. Automaton Aj is activated
23. Set Ai to Aj
24. Until |t|\geq|V|-1
25. If(w<minimum weight of the selected constrained spanning tree)
26. Reward the selected action of automata
27. Else
28. Penalize the selected action of automata
29. End if
30. K\rightarrow k + 1
31. Enable all the disabled actions
32. Until(probability of selecting constrained path are greater than a pre specified threshold or stage number k are greater than another pre specified threshold)
33. End Algorithm

Fig. 1: learning automata-based algorithm to solve the DCMST problem.

Empirical Results:
The proposed algorithm(DeLA) was tested on different data set containing difficult d-MST instances: structured hard (SHRD) graph (Krishnamoorthy, 2001). Learning automata was performed with the following strategy parameters: a=0.08. The solution quality is measured by the relative difference between the final objective value \( C \) obtained by a specific approach and the objective value \( C_{d-Prim} \) of the solution found by the d-Prim heuristic where the first vertex is used as starting point in percent. This measure is called quality gain:

\[
\text{quality gain} = \left( \frac{C_{d-Prim} - C}{C_{d-Prim}} \right) \times 100\%
\]

In other words, we use d-Prim as a reference algorithm and calculate relative quality improvements for the other approaches; where larger values indicate better results.

Table 1 shows results for the SHRD data set after 50 times frequency. The numbers of vertices are in the range 15, 20, 25, and 30 where the maximum degree was set to 3, 4 and 5. The results for GA-F, GA-P, PSS, SA and AB are adopted from (Bau, 2005; Bui, 2006; Raidl, 2000) and printed for comparison purposes only. Besides average gains, the gains of the average CPU-times in seconds are reported in Table 1.

Table 1: Average results on structured hard instances of the DCMST problem.

<table>
<thead>
<tr>
<th>Problem</th>
<th>GA-F</th>
<th>GA-P</th>
<th>PSS</th>
<th>SA</th>
<th>AB</th>
<th>DeLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHRD 15</td>
<td>3</td>
<td>5.33</td>
<td>0.55</td>
<td>3.61</td>
<td>3.76</td>
<td>1.72</td>
</tr>
<tr>
<td>SHRD 15</td>
<td>4</td>
<td>4.62</td>
<td>0.41</td>
<td>16.86</td>
<td>1.20</td>
<td>2.08</td>
</tr>
<tr>
<td>SHRD 15</td>
<td>5</td>
<td>6.19</td>
<td>0.36</td>
<td>11.80</td>
<td>1.20</td>
<td>0.00</td>
</tr>
<tr>
<td>SHRD 20</td>
<td>3</td>
<td>0.00</td>
<td>1.44</td>
<td>6.70</td>
<td>2.23</td>
<td>0.45</td>
</tr>
<tr>
<td>SHRD 20</td>
<td>4</td>
<td>0.24</td>
<td>0.73</td>
<td>6.72</td>
<td>2.21</td>
<td>0.00</td>
</tr>
<tr>
<td>SHRD 20</td>
<td>5</td>
<td>1.58</td>
<td>0.88</td>
<td>6.80</td>
<td>2.20</td>
<td>0.47</td>
</tr>
<tr>
<td>SHRD 25</td>
<td>3</td>
<td>2.74</td>
<td>2.57</td>
<td>2.35</td>
<td>3.73</td>
<td>0.00</td>
</tr>
<tr>
<td>SHRD 25</td>
<td>4</td>
<td>2.09</td>
<td>1.83</td>
<td>5.58</td>
<td>3.68</td>
<td>0.00</td>
</tr>
<tr>
<td>SHRD 25</td>
<td>5</td>
<td>3.18</td>
<td>1.74</td>
<td>2.58</td>
<td>3.64</td>
<td>1.69</td>
</tr>
<tr>
<td>SHRD 30</td>
<td>3</td>
<td>5.37</td>
<td>2.77</td>
<td>5.37</td>
<td>5.60</td>
<td>0.00</td>
</tr>
<tr>
<td>SHRD 30</td>
<td>4</td>
<td>3.67</td>
<td>3.62</td>
<td>7.59</td>
<td>5.52</td>
<td>0.00</td>
</tr>
<tr>
<td>SHRD 30</td>
<td>5</td>
<td>2.69</td>
<td>2.93</td>
<td>4.78</td>
<td>5.51</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Conclusions:
We presented learning automata-based algorithm to solve the degree-constrained minimum spanning tree problem. Performance studies have revealed that proposed algorithm is competitive with a number of other metaheuristic approaches. High-quality solutions are typically obtained within few seconds. It seems possible to adapt the proposed representation together with its operators also to other network optimization problems. The diameter constrained MST problem and the capacitated MST problem, are examples which we currently investigate.

REFERENCES


