IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661, ISBN: 2278-8727 Volume 6, Issue 4 (Sep-Oct. 2012), PP 06-10
www.iosrjournals.org

An Improved Ant-Based Algorithm for Minimum Degree Spanning Tree Problems

Md. Niaz Imtiaz[1], Md.Akkas Ali[2]
[1, 2]Lecturer, Dept. of CSE, University of Information Technology & Sciences (UITS), Baridhara, Dhaka-1212, Bangladesh

Abstract : *A spanning tree of à connected graph is à sub graph, with least number of edges that still spans. The problem of finding degree constraint spanning tree is known to be NP-hard. In this paper we discuss an Ant-Based algorithm for finding minimum degree spanning trees and give improvement of the algorithm. We also show comparisons among the three algorithms and find the best improved Ant-Based algorithm. Extensive experimental results show that our improved algorithm performs very well against other algorithms on a set of 50 problem instances.*
Keywords - *Ant algorithm, Graph algorithms, Heuristic methods, minimum degree spanning tree,*

I. INTRODUCTION

This paper describes Ant-Based algorithms for minimum degree spanning tree (AB-MDST) of unweighted connected graph. This is an interesting, real-world problem that seems well suited to an ant algorithm approach. The AB-MDST problem entails finding a spanning tree such that the maximum degree of a vertex in the tree becomes minimum [2]. This concept is useful in the design of telecommunication networks, design of networks for computer communication, design of integrated circuits, energy networks, transportation, logistics, and sewage networks [3]. For instance, switches in an actual communication network will have limited number of connections available. Transportation systems must place a limit on the number of roads meeting in one place.

The problem of finding degree constraint spanning tree is NP-hard [9]. Therefore, heuristics are often used to find good solutions in a reasonable amount of time [10]. We have used one type of heuristic called Ant Colony Optimization (ACO) [10]. Here, artificial ants move based on local information and pheromone levels. Our algorithm uses cumulative pheromone levels to determine candidate set of edges from which minimum degree spanning trees are built [7]. In this paper, we compare 3 algorithms - AB-MDST without local search and without degree constraint, AB-MDST with local search but without degree constraint and the last one AB-MDST with local search and with degree constraint. Extensive experimental results show that AB-MDST with local search and with degree constraint performs very well against other algorithms.

The rest of the paper is organized as follow. In section 2 Our Ant-Based algorithm and two improved versions of that algorithm are described. Section 3 compares the performances of the three algorithms. The conclusion is given in section 4.

II. Methodology

2.1 Initialization

Initially one ant is assigned to each vertex of the graph. In the next step, pheromone is assigned to each edge using the formula P[i][j]=(M-d[j])+(M-m)/3, where P[i][j] is pheromone level of edge (i,j), d[j] is degree of j vertex, M is maximum degree of the input Graph and m is minimum degree of the input Graph [6]. Note that if a vertex has smaller degree, the edges connected to that vertex have higher pheromone levels.

2.2 Exploration

In each step each ant moves along one of the adjacent edge of its present vertex and after moving, the pheromone level of that edge is enhanced using the formula P[i][j]=(M-d[j])+(M-m)/3. This event occurs until all the ants visit all the vertices [10]. In the tree construction section, a set C of candidate edges based on pheromone levels is identified. From C a spanning tree T is constructed. After constructing spanning tree the maximum degree cost (T) of the spanning tree T is calculated. Next, cost (T) is compared with cost (B), where cost (B) is the previous best tree (whose maximum degree is minimum) [5]. If cost (T) is smaller than cost (B), T is the present best tree and cost (T) is assigned to cost (B), because we always try to find the best tree whose degree is minimum. The pheromone level for edges in the best tree B is then enhanced. This entire event is repeated until the stopping criteria met.

2.3 Ant Movement

Let, an ant α is at vertex i. In case of AB-MDST without local search, an edge (i,j) is selected randomly, where j is an adjacent vertex of i. In case of AB-MDST with local search, all the adjacent edges of vertex i are considered, then the edge with highest pheromone level is selected. Note that, among all the adjacent edges the edge (i,j) has highest pheromone level if the degree of vertex j is minimum [4]. For this reason in case of AB-MDST with local search, better result is found than in the case of AB-MDST without local search. After an edge (i,j) is selected, ant α moves from vertex i to vertex j and the pheromone level of the edge (i,j) is enhanced and vertex j is then marked as visited .

2.4 Tree Construction

After the ants have completed their movements and the pheromone levels of all the edges are updated, we are ready to identify the edges from which to construct a spanning tree. To identify a set of candidate edges, we first sort the edges in the graph in the order of descending pheromone level. The top candidate edges from the sorted list are selected to form candidate set C. During constructing tree, edges are taken one by one from C maintaining the order [9]. Let (i,j) be next candidate edge and i and j are not connected in T. In case of AB-MDST without degree constraint, the edge (i,j) is removed from C and added to tree T if this would create no loop. In case of AB-MDST with degree constraint, while adding (i,j) edge to tree T another checking is needed. If after adding (i,j), degree of i or degree of j exceeds given parameter k and number of skipped edges(skippedEdge) C is smaller than E-V, the edge, (i,j) is skipped and added to skippedEdge rather than adding it to tree T and each time C is increased by one [8]. This event continues until the entire tree is constructed.

2.5 Stopping Criteria

The algorithm stops if one of the following two conditions is satisfied: 1. there is no improvement found in 1,000 consecutive cycles, or 2. It has run for 5,000 cycles. When the algorithm stops, the current best tree is returned.

```
AB-MDST (G= (V, E))
//initialization
assign one ant to each vertex of the Graph.
initialize pheromone level of each edge using the formula P[i][j]=(M-d[j])+(M-m)/3.
 // P[i][j] is pheromone level of edge (i,j)
// d[j] is degree of vertex j
                                                    // M is maximum degree of the input Graph
                                                    // m is minimum degree of the input Graph
B ← Ø
cost (B) ← Ø
while stopping criteria not met   // loop will continue when counter<5000 and notImproved <1000
        for each vertex
                for each ant
                        move α along one edge
                        update pheromone level of the edge using same formula
                end-for
        end-for
        // Tree construction stage
identify a set C of candidate edges using pheromone levels
        while |T| < n-1
                construct spanning tree T from C
        end-while
        count the maximum degree cost (T) of the spanning tree T
        if cost (T) < cost (B)
                B ← T
                cost (B) ← cost (T)
                enhance pheromone level for edges in the best tree B
        end-if
end-while
return the best tree found B
```

Figure 1: Ant Based Minimum Degree Spanning Tree Algorithm

Move (α,i)   // ant α is at vertex i

```
select an adjacent edge (i,j)  // j is an adjacent vertex of i vertex
if vertex j is unvisited
        update pheromone level of edge (i,j)
        move α from vertex i to vertex j
        mark  j  visited
        break
else
        α remains at vertex i
end-if
```

Figure 2: One Step in the ant movement algorithm (AB-MDST without local search)

```
ConstructTree (G=(V,E))
sort all the edges by pheromone level in descending order
C ⟵ top candidate edges ( highest pheromone levels )
T ⟵ Ø

while |T| < n-1

        let (i,j) be next candidate edge
        if i and j not connected in T
                if adding (i,j) to T would create no loop
                        remove (i,j) from C
                        add (i,j) edge to tree T
                end-if
        end-if
end-while
return  T
```

Figure 3: Tree construction (AB-MDST without degree constraint)

```
Move (α,i)  // ant α is at vertex i
find all the adjacent edges of verrex i
count the number of adjacent edges
if count = 0
        ant α remains at vertex i
else if count =1
        select the adjacent edge (i,j)
else
        find the adjacent edge (i,j) whose pheromone level is maximum
        select the adjacent  edge (i,j)
end-if
if vertex j is unvisited
        update pheromone level of edge (i,j)
        move α from vertex i to vertex j
        mark  j  visited
        break
else
        α remains at vertex i
end-if
```

Figure 4: One step in the ant movement algorithm (AB-MDST with local search)

```
ConstructTree (G=(V,E),k)              // k is degree constraint
sort all the edges by pheromone level in descending order
C ⟵ top candidate edges ( highest pheromone levels )
T ⟵ Ø

while |T| < n-1

        let (i,j) be next candidate edge
        if i and j not connected in T
                if degree [i] > k or  degree [j] >k
                        and  C < E-V      // C is number of skipped edges
                                add (i,j) to skippededge
```

increase C by one
else
if adding (i,j) to T would create no loop
remove (i,j) from C
add (i,j) to tree T
end-if
end-if
end-if
end-while
return T

Figure 5: Tree construction (AB-MDST with degree constraint)

III. Experimental Results

Our algorithm and the two improved versions of the algorithm are run on a set of 50 complete graphs ranging from 10 to 200 vertices. The algorithms were implemented in C and run on a 1.80 Ghz Pentium Dual-Core with 2 GB of RAM running the Windows 7 operating system. “Table 1” shows the final results. For all of the tables below, first column represents the data set number from 1 to 50. Second column and third column represent the input graph (number of vertices V and number of edges respectively E) for each of the data set. Fourth and fifth column represent the result for the algorithm AB-MDST without local search and without degree constraint. Sixth and seventh column represent the result for the algorithm AB-MDST with local search but without degree constraint and Eighth and ninth column represent the result for the algorithm AB-MDST with local search and with degree constraint. For each of the three algorithms the Degree column shows the maximum degree of the constructed tree and the Time column shows execution time in seconds for each of the input graphs. From the table we see that for most of the input data sets, AB-MDST with local search but without degree constraint gives better result than AB-MDST without local search and without degree constraint. AB-MDST with local search and with degree constraint gives much better result than both AB-MDST with local search but without degree constraint and AB-MDST without local search and without degree constraint for both Degree and Time.

Table 1: Experimental Results

	Input Graph		Output					
			Algorithm 1		Algorithm 2		Algorithm 3	
Dataset	V	E	Degree	Time	Degree	Time	Degree	Time
Data1	10	21	3	0	3	0	2	0
Data2	10	22	3	0.016	3	0.016	2	0
Data3	10	24	2	0	2	0	2	0
Data4	10	25	2	0	3	0.02	2	0
Data5	10	26	3	0	3	0	2	0
Data6	10	27	3	0.016	3	0	2	0
Data7	10	34	2	0	3	0	2	0
Data8	25	69	3	0.078	3	0.047	2	0
Data9	25	70	3	0.078	3	0.062	2	0
Data10	28	75	4	0.094	3	0.078	2	0
Data11	25	72	3	0.078	3	0.063	2	0.094
Data12	25	90	3	0.078	3	0.078	2	0.01
Data13	25	71	3	0.094	3	0.078	2	0
Data14	43	63	4	0.25	5	0.203	3	0.203
Data15	45	85	3	0.297	2	0	3	0.171
Data16	50	123	5	0.486	3	0.359	3	0.235
Data17	50	145	3	0.453	4	0.375	3	0.234
Data18	60	166	4	0.719	2	0	3	0.359
Data19	50	157	4	0.437	4	0.321	3	0.282
Data20	50	183	4	0.453	4	0.328	3	0.265
Data21	50	491	4	0.625	5	0.485	3	0.359
Data22	50	582	3	0.593	3	0.469	3	0.344
Data23	50	171	4	0.437	3	0.359	3	0.235
Data24	75	196	4	1.359	4	0.984	3	0.703
Data25	75	215	5	1.531	5	1.547	3	0.781
Data26	75	256	4	1.36	4	1.265	3	0.703
Data27	75	202	4	1.375	4	0.906	3	0.672

Data28	75	266	5	1.547	5	1.094	3	0.797
Data29	100	297	5	2.922	5	2.828	3	1.532
Data30	100	324	4	3.125	5	2.578	3	1.578
Data31	100	334	4	4.563	5	3.734	3	2.047
Data32	100	314	3	3.062	6	4.969	3	1.547
Data33	100	394	4	3.862	4	2.812	3	1.781
Data34	100	261	4	5.718	5	3.218	3	1.844
Data35	100	271	5	3.156	4	2.125	3	1.547
Data36	100	451	5	4.032	4	2.484	3	2.031
Data37	100	742	5	4.438	5	3.906	3	2.266
Data38	100	922	4	4.938	4	3.406	3	2.531
Data39	150	481	5	11.609	4	7.841	3	7.11
Data40	150	473	5	10.531	5	9.062	3	5.5
Data41	150	402	4	12	5	9.297	4	5.986
Data42	100	334	4	2.875	6	2.782	4	1.47
Data43	150	453	5	19.687	5	8.653	4	6.656
Data44	150	1064	5	16.204	4	14.468	4	7.86
Data45	200	514	5	21.828	4	18.672	4	10.797
Data46	200	654	5	21.469	6	16.016	4	11.516
Data47	200	644	5	24.047	5	15.972	4	12.422
Data48	200	664	5	22.89	8	39.281	4	11.437
Data49	200	519	5	43.188	4	18.266	4	13.67
Data50	200	701	4	26.188	5	18.375	4	15.61

Algorithm 1: AB-MDST without local search and without degree constraint
Algorithm 2: AB-MDST with local search but without degree constraint
Algorithm 3: AB-MDST with local search and with degree constraint

IV. CONCLUSION

In this paper we discussed an Ant-Based algorithm- AB-MDST without local search and without degree constraint (algorithm 1) to find minimum spanning degree spanning trees from different input graphs and gave two improved versions of the algorithm named AB-MDST with local search but without degree constraint (algorithm 2) and AB-MDST with local search and with degree constraint (algorithm 3). The experimental results show that for both parameters (Degree and Time); algorithm 3 gives much better result than the other two algorithms.

REFERENCES

[1]. Dorigo, M., V. Maniezzo, and A. Colorni, “Ant System: Optimization by a Colony of Cooperating Agents,” IEEE Trans. on Systems, Man, and Cybernetics - Part B, 26(1), Feb. 1996, pp. 29–41.

[2]. Volgenant, A., “A Lagrangean Approach to the Degree-Constrained Minimum Spanning Tree Problem,” European Journal of Operational Research, 39, 1989, pp. 325–331.

[3]. Eila Safari and Azizallah Rahmati, “Using Learning Automata to Solving Degree-constrained Minimum Spanning Tree Problem”, Australian Journal of Basic and Applied Sciences, 5(6): 337-341, 2011, ISSN 1991-8178

[4]. Minh N. Doan,” An Effective Ant-Based Algorithm for the Degree-Constrained Minimum Spanning Tree Problem”, 1-4244-1340-0/07$25.00_c 2007 IEEE

[5]. YOON-TECK BAU, CHIN-KUAN HO AND HONG-TAT EWE, “Ant Colony Optimization Approaches to the Degree-constrained Minimum Spanning Tree Problem”, JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 24, 1081-1094 (2008)

[6]. Thang N. Bui and Catherine M. Zrncic, “An Ant-Based Algorithm for Finding Degree-Constrained Minimum Spanning Tree”, GECCO’06, July 8–12, 2006, Seattle, Washington, USA.

[7]. Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar, “A Push-Relabel Algorithm for Approximating Degree Bounded MSTs”

[8]. Savelsbergh, M. and T. Volgenant, “Edge Exchanges in the Degree-Constrained Minimum Spanning Tree Problem,” Computers and Operations Research, 12(4), 1985, pp. 341–348.

[9]. Marck De Berg, Marc Van Kreveld and Mark Overmars, “Computational Geometry Algorithms and Applications”.

[10]. Marco Dorigo, Mauro Birattari, and Thomas Stutzle, “Ant Colony Optimization-Artificial Ants as a Computational Intelligence Technique”.