Far East Journal of Electronics and Communications

http://www.pphmj.com
http://dx.doi.org/10.17654/EC018020291
Volume 18, Number 2, 2018, Pages 291-300 **ISSN: 0973-7006**

DIFFERENT TIME INSTALLATION EFFECT ON THE QUALITY OF THE SOLUTION FOR THE MULTIPERIOD INSTALLATION PROBLEM USING MODIFIED PRIM'S ALGORITHM

Wamiliana[1,*], Warsono[1], Asmiati[1], Astria Hijriani[2] and Wibi Cahyo Hastono[2]

[1]Department of Mathematics
Universitas Lampung
Bandar Lampung, Indonesia

[2]Department of Computer Science
Universitas Lampung
Bandar Lampung, Indonesia

Abstract

In most of network design problems, the minimum spanning tree (MST) is usually used as the backbone. If we add degree restriction on its vertices (can represent cities, stations, etc.) of the graph (represents the network), then the problem becomes the degree constrained minimum spanning tree (DCMST) problem. However, to do the installation or connecting the network, it is possible that the process must be done into some stages or periods. That situation occurs because of the weather constraint, fund constraint, etc. By restricting and dividing the stages or periods of the network's installation, the problem emerges as the multiperiod degree constrained minimum spanning tree (MPDCMST) problem or multiperiod installation

Received: September 19, 2017; Accepted: November 2, 2017

Keywords and phrases: minimum spanning tree, degree constrained, installation, period.

*Corresponding author

problem. We develop two algorithms based on modified Prim's algorithms (WAC1 and WAC2) to solve the MPDCMST problem, and show and compare the different time installation effect on quality of the solution by implementing and comparing those algorithms using 300 generate problems.

1. Introduction

In optimization problem, the concept of graph theory plays an important role, especially in network design. A graph can be considered as a network where the vertices represent cities, terminals, computers, stations, and so on, while the edges represent roads, links, routes, train tracks, canals, and so on. Given a weighted graph $G(V, E)$, V is the set of vertices and E is the set of edges connecting the vertices in V; $c_{ij} \geq 0$ is the weight of edge e_{ij}. The multiperiod degree constrained minimum spanning tree (MPDCMST) is a problem of finding a minimum weight/cost of a certain graph (network), which has to maintain degree requirements on every vertex and satisfy period installation of certain vertices. The degree restriction usually relates with the reliability or interconnection on the network and the period requirement occurs mostly because of the fund, weather or others. In the next section, we give a brief discussion about the problem and its backbone. In Section 3, we discuss about the algorithms developed. The results and discussion are given in Section 4, and Section 5 concludes the paper.

2. Multiperiod Degree Constrained Minimum Spanning Tree and its Backbone

A. Minimum spanning tree

In network design problem, the minimum spanning tree is one of the classical problems that commonly arises as the backbone to represent the problem. There are some algorithms developed to solve the MST, but the two well known are Kruskal's algorithm [1] and Prim's algorithm [2]. The main difference of these two algorithms lies in the connectivity property on the process of finding MST. In Kruskal's algorithm, it is possible to get forest

during the process (violates the connectivity property), while in Prim's algorithm is not.

B. Degree constrained minimum spanning tree problem

The degree constrained minimum spanning tree problem is a problem of finding an MST while also maintaining the degree requirement on the vertices. Since this problem is considered as NP-complete [3], heuristics more preferred compared to exact methods. Some of the heuristics include greedy algorithm based on Prim's and Kruskal's algorithms [4], genetic algorithm [5], iterative refinement [6], simulated annealing [7], modified penalty [8], and Tabu search [9-11]. Some exact methods already investigated include the branch and bound algorithm based on an edge exchange analysis [12], Lagrangian relaxation [13, 14], and branch and cut [15].

C. Multiperiod degree constrained minimum spanning tree

The multiperiod degree constrained minimum spanning tree is a problem that satisfies DCMST with additional constraint that the vertices in the network must be installed/connected on specific stage/period. The later constraint usually occurs because of the fund limitation, weather condition or others. This problem was introduced in 2002 and was solved using hybrid of branch exchange and Lagrangian relaxation, implemented on vertex order ranging from 40 to 100 and used 10 years planning horizon [16]. By using one year horizon, some algorithm based on Kruskal's and Prim's algorithms was developed [17-21], implemented, tested, and compared.

3. The WAC1 and WAC2 Algorithms

These two algorithms are developed using modified Prim's algorithm. We prefer using Prim's algorithm to Kruskal's algorithm because Prim's algorithm maintains connectivity property during installation/connecting process and this is more applicable in real-life problems. These two algorithms also have the same initiation: use vertex 1 as the central vertex or root, and set $T = \varnothing$. Moreover, we apply the same HVT_i as in [21], HVT_i

is the set of vertices that must be installed/connected on *i*th period, and MAXVT_i is the maximum number of vertices that can be installed/ connected on the *i*th period; $\text{MAXVT}_i = \left\lfloor \frac{(n-1)}{3} \right\rfloor$, n is the order of the vertices.

A. WAC1 algorithm

This algorithm starts by finding the smallest edge that connects the vertices in HVT_i, $i = 1$ to the central vertex, and continuing the process until all vertices in HVT_1 already connected/installed. All the edges already connected are stored in *T* and the vertices in *V*. Next, the algorithm will check MAXVT_i, $i = 1$. If $\text{MAXVT}_1 - |\text{HVT}_1| > 0$, then the algorithm will find the nearest edges that connect the vertices in *V* with the vertices not installed yet. If connecting the edges neither violates the degree restriction nor constitutes a cycle, then the edge is connected to the network (*T*) and the related vertex to *V*. Continue with similar pattern until $\text{MAXVT}_1 - |\text{HVT}_1| = 0$. The second and third periods $(i = 2, 3)$ are similar to the first period.

B. WAC2 algorithm

This algorithm is similar to WAC1 algorithm. The main difference of WAC1 and WAC2 algorithms lies in the process of connecting the edges in the HVT_i. In WAC1 algorithm, we find the smallest edges that connect to vertices in HVT_i and then install/connect the vertices in HVT_i first in every *i*th period before investigating other edges, while in WAC2 algorithm we find the smallest edges in the network to be installed/connected first, and these edges may not be connected with vertices in HVT_i. However, all the vertices in HVT_i must be installed/connected when the *i*th period is finished. We give the following illustration to explain the process of both the algorithms:

$$\text{HVT}_1 = \{2\},\ \text{HVT}_2 = \{3\},\ \text{HVT}_4 = \{4\}.$$

Table 1. The weight of file 22.dat for 10 vertices

From	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2
to	2	3	4	5	6	7	8	9	10	3	4	5	6	7	8
Weight	740	572	447	835	427	807	362	832	120	221	109	276	741	978	352
From	2	2	3	3	3	3	3	3	3	4	4	4	4	4	4
to	9	10	4	5	6	7	8	9	10	5	6	7	8	9	10
Weight	368	403	505	921	757	884	369	886	545	639	253	750	251	187	857
From	5	5	5	5	5	6	6	6	6	7	7	7	8	8	9
to	6	7	8	9	10	7	8	9	10	8	9	10	9	10	10
Weight	807	926	781	605	112	559	411	473	743	882	693	851	509	434	828

For WAC1, the first edge installed/connected is $e_{12} = 740$ because vertex 2 is on HVT_1 and e_{12} is stored in *T*. Next, we check MAXVT_1 and since $\text{MAXVT}_1 = 3$ and we already connect vertex 2, we still have two more vertices to be connected. Edge $e_{24} = 109$ is the smallest among the edges considered (the edges that connect to vertex 1 or 2). Thus, we add vertex 4 to *V* and store e_{24} in *T*. Now, we still have one more vertex to be installed in the first period. The smallest edge that connects to vertices in HVT_1 is $e_{1,10} = 120,$ therefore, we add vertex 10 to *V* and store $e_{1,10}$ in *T*. Since MAXVT_1 is already satisfied, we finish the first period and get the following figure:

Figure 1. The end of first period of WAC1.

Note that the area connecting every pair of vertices represents the weight of the edge. The bigger the area, the bigger the weight. For example, the weight of edge e_{12} is bigger than e_{24}. Continuing with similar step for second and third periods, we get the following figures:

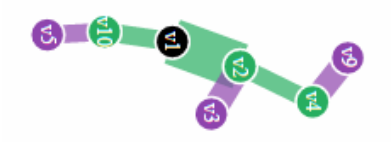

Figure 2. The end of second period.

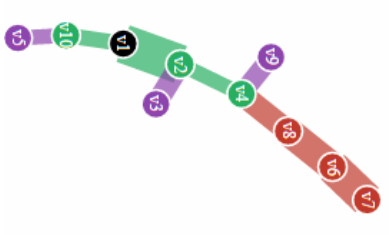

Figure 3. The end of third period of WAC1.

The following is the table of the edges in the network after installation.

Table 2. Edges on the network after installation using WAC1 algorithm

No.	Period	From	to	Weight
1	1	1	2	740
2	1	2	4	109
3	1	1	10	120
4	2	2	3	221
5	2	10	5	112
6	2	4	9	187
7	3	4	8	251
8	3	8	6	411
9	3	6	7	559
Total weight				2710

For WAC2, the initiation is the same with WAC1. Since we investigate the smallest edges first, we find $e_{1,10}$. Next, add vertex 10 to V and store $e_{1,10}$ in T. Now, the smallest edge that connects the vertices in T is $e_{2,10}$, and $\text{HVT}_1 = \{2\}$. Since $\text{MAXVT}_1 = 3$ and the number of vertices installed so far is only one vertex, so we still have two vertices to be installed,

including vertex 2. Then we add vertex 2 to V and store $e_{2,10}$ in T. But, vertex 2 is in HVT_1, therefore, we find the next smallest edge which is e_{24} and finish the first period. The second and third periods are similar to the first period. The following table shows the edges on the network after applying WAC2 algorithm:

Table 3. Edges on the network after installation using WAC2 algorithm

No.	Period	From	to	Weight
1	1	1	10	120
2	1	10	2	403
3	1	2	4	109
4	2	10	5	112
5	2	4	9	187
6	2	2	3	221
7	3	4	8	251
8	3	8	6	411
9	3	6	7	559
Total weight				2373

4. Results and Discussion

For the degree condition, we restrict our implementation only for degree bound 3. We chose this bound, since our early computational work revealed that for degree bound greater than 3, the MST is usually feasible and hence optimal. We provide results on 300 random problems generated as follows:

- Number of vertices ranges from 10 to 500 with an increment of 10 for up to 100 vertices.
- The edge weights are generated randomly from uniform distribution from 1 to 1000.
- For every vertex order, there are 30 problems generated. The following table shows the result.

Table 4. The average solution of the algorithm

n	Average of the solution			
	MST	DCMST	WAC1	WAC2
10	1129,43	1178,8	1495,1	1359,93
20	1196,1	1299	1790,37	1437,5
30	1177,43	1319,53	2018,9	1516,43
40	1151,23	1286,3	2079,73	1455,3
50	1223,43	1356,47	2381	1603,7
60	1175,57	1320,73	2364,4	1639,53
70	1242,1	1410,03	2520,2	1671,9
80	1236,83	1410,23	2547,8	1722,23
90	1248	1404,93	2588,07	1649,33
100	1234,1	1370,8	2535,2	1597,63

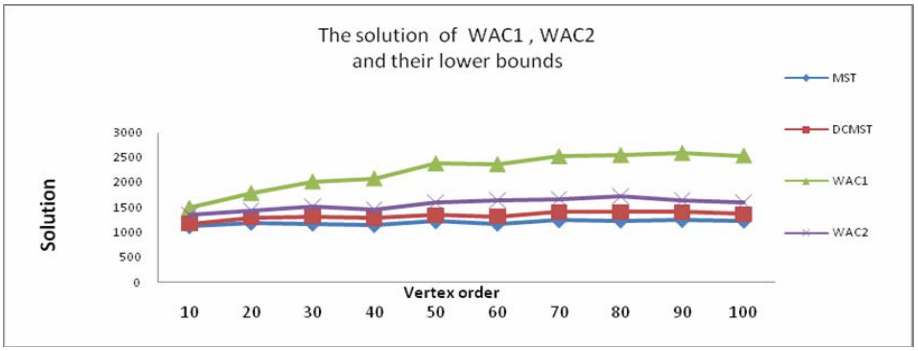

Figure 4. The average solutions of MST, DCMST, WAC1 and WAC2.

5. Conclusion

In the above discussion, we have shown that relaxing the time of connecting/installing the vertices in HVT_i gave better results. Therefore, during the installation/connection process, it should be advisable not to install the vertices on HVT_i before investigating the smallest possible edges to be installed. However, when the *i*th period is ended, the vertices in HVT_i are already installed/connected.

Acknowledgement

The authors would like to acknowledge the contributions and financial support from the Directorate General of Higher Education, The Ministry of Research, Technology and Higher Education, The Republic of Indonesia. This research was supported under research grant no.: 1636/UN26.21/KU/ 2017.

References

[1] J. B. Kruskal, On the shortest spanning tree of a graph and the travelling salesman problem, Proc. Amer. Math. Soc. 7 (1956), 48-50.

[2] R. C. Prim, Shortest connection networks and some generalizations, Bell Syst. Tech. Journal 36 (1957), 1389-1401.

[3] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W. H. Freeman & Co., San Francisco, 1979.

[4] S. C. Narula and Cesar A. Ho, Degree-constrained minimum spanning tree, Computer and Operation Research 7 (1980), 239-249.

[5] M. Krishnamoorthy, A. T. Ernst and Yazid M. Sharaila, Comparison of algorithms for the degree constrained minimum spanning tree, Journal of Heuristics 7(6) (2001), 587-611.

[6] N. Deo and Nishit Kumar, Computation of constrained spanning trees: a unified approach, Network Optimization, Lecture Notes in Economics and Mathematical Systems, Panos M. Pardalos, Donald W. Hearn and William W. Hager, eds., Springer-Verlag, Berlin, Germany, 1997, pp. 194-220.

[7] G. Zhou and Mitsuo Gen, A note on genetics algorithms for degree-constrained spanning tree problems, Networks 30 (1997), 91-95.

[8] Wamiliana, Solving the degree constrained minimum spanning tree using tabu and penalty method, Jurnal Teknik Industri 6(1) (2004), 1-9.

[9] L. Caccetta and Wamiliana, Heuristics algorithms for the degree constrained minimum spanning tree problems, F. Ghassemi, D. Post, M. Sivapalan and R. Vertessy, eds., Proceeding of the International Congress on Modelling and Simulation (MODSIM), Canberra, 2001, pp. 2161-2166.

[10] Wamiliana and Caccetta, Tabu search based heuristics for the degree constrained minimum spanning tree problem, Proceeding of South East Asia Mathematical Society, 2003, pp. 133-140.

[11] Wamiliana and L. Caccetta, The modified CW1 algorithm for the degree restricted minimum spanning tree problem, Proceeding of International Conference on Engineering and Technology Development, Bandar Lampung, 20-21 June 2012, pp. 36-39.

[12] M. Savelsbergh and Ton Volgenant, Edge exchange in the degree-constrained minimum spanning tree, Computer and Operation Research 12 (1985), 341-348.

[13] B. Gavish, Topological design of centralized computer networks formulations and algorithms, Networks 12 (1982), 355-377.

[14] A. Volgenant, A Lagrangian approach to the degree-constrained minimum spanning tree problem, European J. Oper. Res. 39 (1989), 325-331.

[15] L. Caccetta and S. P. Hill, A branch and cut method for the degree constrained minimum spanning tree problem, Networks 37 (2001), 74-83.

[16] R. Kawatra, A multi period degree constrained minimum spanning tree problem, European J. Oper. Res. 143 (2002), 53-63.

[17] Wamiliana, Dwi Sakethi and Restu Yuniarti, Computational aspect of WADR1 and WADR2 algorithms for the multi period degree constrained minimum spanning tree problem, Proceeding SNMAP, Bandar Lampung, 8-9 December 2010, pp. 208-214.

[18] A. Junaidi, Wamiliana, Dwi Sakethi and Edy Tri Baskoro, Computational aspects of greedy algorithm for solving the multi period degree constrained minimum spanning tree problem, J. Sains MIPA 14(1) (2008), 1-6.

[19] Wamiliana, Amanto and Mustofa Usman, Comparative analysis for the multi period degree constrained minimum spanning tree problem, Proceeding of the International Conference on Engineering and Technology Development (ICETD), 2013, pp. 39-43.

[20] Wamiliana, Faiz A. M. Elfaki, Mustofa Usman and M. Azram, Some greedy based algorithms for multi periods degree constrained minimum spanning tree problem, ARPN Journal of Engineering and Applied Sciences 10(21) (2015), 10147-10152.

[21] Wamiliana, Mustofa Usman, Dwi Sakethi, Restu Yuniarti and Ahmad Cucus, The hybrid of depth first search technique and Kruskal's algorithm for solving the multiperiod degree constrained minimum spanning tree problem, The 4th International Conference on Interactive Digital Media (ICIDM), IEEE Explore, Dec. 2015.