https://doi.org/10.31436/iiumej.v21i1.1088

USING MODIFICATION OF PRIM'S ALGORITHM AND GNU OCTAVE AND TO SOLVE THE MULTIPERIODS INSTALLATION PROBLEM

WAMILIANA[1*], MUSTOFA USMAN[1], WARSONO[1], WARSITO[2] AND JAMAL IBRAHIM DAOUD[3]

[1]Department of Mathematics, FMIPA Universitas Lampung, Bandarlampung, Indonesia
[2]Department of Physics, FMIPA Universitas Lampung, Bandarlampung, Indonesia
[3]Department of Science in Engineering, Faulty of Engineering,
International Islamic University Malaysia,
P.O. Box 10, 50728 Kuala Lumpur, Malaysia.

[*]Corresponding author: wamiliana.1963@fmipa.unila.ac.id

(Received: 19th February 2019; Accepted: 28th November 2019; Published on-line: 20th January 2020)

ABSTRACT: The Minimum Spanning Tree (MST) is one of the famous problems that is used mostly as the backbone in many network design problems. Given a graph G(V,E), where V is the set of vertices and E is the set of edges connecting vertices in V, and for every edge e_{ij} there is an associated weight $c_{ij} \geq 0$. The Multi Period Degree Constrained Minimum Spanning Tree (MPDCMST) is a problem of finding an MST while also considering the degree constrained on every vertex, and satisfying vertices installation requirement on every period. Two algorithms (WWM1 and WWM2) are proposed for solving this problem. GNU OCTAVE is used for coding and visualization. GNU is a recursive acronym for "**G**NU's **N**ot **U**nix*!"*, and that name is chosen because it is like Unix but differs from Unix because it is free and contains no Unix code. Those algorithms were implemented using 300 randomly generated problems. Moreover, we compare WWM1 and WWM2 algorithms using existing data from the literature and the results show that WWM2 is the best.

ABSTRAK*:* Minimum Spanning Tree (MST) merupakan salah satu masalah mahsyur yang banyak digunakan sebagai tulang belakang kepada masalah banyak rekaan jaringan. Menerusi graf G(V,E), di mana V adalah himpunan titik dan E adalah himpunan garis yang menghubungkan titik-titik dalam V, dan bagi setiap garis eij terdapat berat berhubung cij ≥0, Multi-period Degree Constrained Minimum Spanning Tree (MPDCMST) merupakan masalah dalam menentukan MST, pada masa sama turut menimbangkan kekangan pada setiap titik vertek, dan memenuhi syarat keperluan pemasangan pada setiap detik. Dua algoritma (WWM1 dan WWM2) dicadangkan bagi menyelesaikan masalah ini. GNU OCTAVE digunakan bagi pengaturcaraan dan visualisasi. GNU merupakan suatu singkatan kepada "GNU's Not Unix", dan nama tersebut dipilih karena ianya seperti Unix, tetapi berbeza dari Unix kerana ia percuma dan tidak mempunyai kod Unix. Algoritma tersebut dilaksana dengan menggunakan 300 masalah terhasil secara rawak. Tambahan, algoritma WWM1 dan WWM2 dibandingkan dengan kajian terdahulu dan hasil kajian menunjukkan WWM2 adalah terbaik.

KEYWORDS: *multi-period; degree constrained; minimum spanning tree; Prims' algorithms; GNU OCTAVE*

1. INTRODUCTION

In most network design problems, The Minimum Spanning Tree (MST) is usually used as the backbone. If we add degree restriction on the vertices (can represent cities,

https://doi.org/10.31436/iiumej.v21i1.1088

stations, etc.) of the graph (represents the network), the problem becomes a Degree Constrained Minimum Spanning Tree (DCMST) problem. Moreover, if we restrict and divide the stages or periods of the network's installation, the problem emerges as a Multi Period Degree Constrained Minimum Spanning Tree (MPDCMST) problem. The later constraint usually occurs because of the fund limitation for installing (connecting) the network. This problem arises when we want to design a network that requires that every vertex (node) restricts the number of connections/interfaces whilst also considering a set of vertices that have to be connected/ installed in a certain period due to the fund restriction (weather, etc.)

In this paper, we organize the discussion as follows: in Section, 2 we give the history of the problem; in Section 3, we discuss how to apply the algorithms developed using GNU Octave and implement them; in Section 4, we discuss the results; followed by a conclusion.

2. THE CONSTRAINED MINIMUM SPANNING TREE PROBLEM

It Because of its specific structure and use in many network design problems, the Minimum spanning tree problem has been studied extensively and a variety of fast algorithms have been developed. An efficient and fast minimum spanning tree algorithm that requires computational time nearly linear in the number of edges had been developed [1]. The MST problem is one of the classical problems where the objective is to construct a minimum cost/weight network. This problem usually arises in network design applications that must satisfy other graph parameters such as: degree, distance, diameter, connectivity, flow, etc. For instance, in a transportation network, a distance restriction on the flow commodities could represent the maximum distance allowed for delivery.

The Degree Constrained Minimum Spanning Tree (DCMST) problem is a Minimum Spanning Tree with a degree restriction on every vertex. This problem arises when designing networks where the degree restriction represents the number of allowable links on that vertex, i.e. the handling capacity of each of the vertices imposes a restriction on the number of edges (or wires/roads) that can be connected to a vertex. For example, the application of The DCMST is present in designing the road system, where the set of roads must connect a collection of suburbs/towns, but there is a restriction that no more than a certain number of roads (example: four roads) are allowed to meet at an intersection.

There are lots of investigations regarding the DCMST problem. This problem is considered to be an NP-Complete problem. An NP-complete problem is any of a class of computational problems for which no efficient solution algorithm has been found. Because of its NP-completeness [2], heuristic methods have dominated such as: the greedy algorithm based on Prim's and Kruskal's algorithm by [3], the Genetic Algorithm by Zhou and Gen[4], the Iterative Refinement by Deo and Kumar [5], the Simulated Annealing by Krishnamoorthy et al [6], the Modified Penalty by Wamiliana [7], and the Tabu Search by Caccetta and Wamiliana [8], Wamiliana and Caccetta [9-10].

In real situations, connecting all components in a network requires a certain time and process in order to be completed. The time of completion can vary depending on the need and priority of the network itself. The Multi Period Degree Constrained Minimum Spanning Tree (MPDCMST) problem was introduced in 2002 by Kawatra [11] and it proposed the hybrid of branch exchange and Lagrangean relaxation method to solve the problem on directed graph with vertex order ranging from 40 to 100. By modifying the problem in using the undirected graph, some algorithms based on Kruskal's and Prim's

algorithms were developed. WADR1 and WADR2 algorithms were developed by doing some modifications on Kruskal's algorithm [12], while WADR3 and WADR4 are the algorithms developed as varians of WADR1 and WADR2 [13]. Motivated by the connectivity property on the process of finding MST by Prim's algorithm, some algorithms were developed by Wamiliana et al. [14-16]. Wamiliana et al. [17] illustrated why the quality of the solution also depends on the number of vertices in HVT_i (the set of vertices that must be installed/connected in i^{th} period or before). The comparison of some algorithms developed and implemented on undirected graphs was given in Wamiliana et al [18] especially on the process of installation/connection of vertices in HVT_i.

3. GNU OCTAVE, MPDCMST AND IMPLEMENTATION

3.1 GNU OCTAVE

GNU Octave is a free software that runs on GNU/Linux, macOS, BSD, and windows. This software originally was intended as a companion to a chemical reactor design course, but the one who first developed it is John W. Eaton [19]. Since this is free software, users are encouraged to modify and develop this software and are free to distribute it as well.

The difference in using GNU Octave as opposed to other language programming (such as Java), is that GNU Octave is more rigid. Certain rules must be followed (such as converting to matrix form or others). Java is more flexible to development of the program. For example, the data used for implementation represent the edges of the complete graph for certain vertex orders. For vertex order 10, the number of edges is 45, which is obtained from the formula $n(E) = \frac{n(n-1)}{2}$, n =10. Before putting in GNU Octave, we have to modify this equation to be $2n(E)) = n^2 - n$. This is a quadratic equation and to find n, $n = \frac{1+\sqrt{1+8n(E)}}{2}$ is used. Thus, to read the data, the source code for reading the data (vertex order) is as follows (let n(E) = Un):

```
function vertex = check_vertex(matrix)
bykedge=length(matrix);%bykedge=Un
        c=2*bykedge;
        n=(1+sqrt(1+(4*c)))/2;
        vertex=n;
      end
```

In GNU Octave, the graph was represented using a matrix, therefore a source code was designed so that when a problem is entered as a data problem, then the program would automatically detect the data and arrange it as two matrices, one as the matrix for original vertices (X) and the other for terminal vertices (Y). For example, if the data of a complete graph with order ten (consists of 45 edges) is entered, then the program automatically defines X and Y as follows:

X =	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	3
	3	3	3	3	3	3	4	4	4
	4	4	4	5	5	5	5	5	6
	6	6	6	7	7	7	8	8	9
Y =	2	3	4	5	6	7	8	9	10
	3	4	5	6	7	8	9	10	4

5	6	7	8	9	10	5	6	7
8	9	10	6	7	8	9	10	7
8	9	10	8	9	10	9	10	10

After the X and Y are generated, the value of the weight will be put to connect X and Y in the form of a symmetrical matrix K, as follows (example for datafile 10.dat):

K =

0	115	74	42	955	712	660	697	450	806
115	0	391	504	597	537	452	709	452	784
74	391	0	939	472	372	272	250	644	354
42	504	939	0	212	515	132	248	603	642
955	597	472	212	0	935	70	715	328	24
712	537	372	515	935	0	426	987	72	227
660	452	272	132	70	426	0	144	814	82
697	709	250	248	715	987	144	0	740	703
450	452	644	603	328	72	814	740	0	886
806	784	354	642	24	227	82	703	886	0

For implementation, the same data used as in [12, 14, 16]. The data are generated randomly with the edge weight varying from 1 to 1000. For every vertex order, 30 problems were generated.

3.2 WWM1 and WWM2 Algorithms

We developed two algorithms to solve the MPDCMST based on Prim's algorithm and modified them to satisfy the given constraints. The reason for choosing Prim's algorithm was that Prim's maintains the connectivity property during the connection/installation process. In these algorithms, we used the terminologies HVT_i as the set of vertices that must be installed/connected on i^{th} period or before, and $MAXVT_i$ as the maximum number of vertices that can be connected on the i^{th} period. The reason for using HVT_i was that in reality, it is possible to add some requirements that some components in the network be connected early due to public needs or other reasons. For example, in designing a fresh water pipe or electricity network, there is a requirement that some buildings (hospitals, police stations, government buildings, etc.) must be installed/connected within a certain period or earlier in the network.

3.2.1 WWM1 Algorithm

The WWM1 algorithm starts by setting vertex 1 as the root and puts it in set V. V is the set of vertices in the network. Initially, V only contains vertex 1 as the central vertex, V ={1}, and no edges in T, T ={}. Then, the algorithm checks the nearest vertices in HVT_i to be connected/installed in V and the corresponding edges to the network (T). The algorithm will continue connecting/installing those vertices in HVT_i as long as the connection neither violates the degree restriction nor constitutes a cycle. If degree violation occurs, then edge exchange will be performed while also maintaining the connectivity of vertices in HVT_i. Next, the algorithm will check the $MAXVT_i$, and connect the edges with the smallest edge weight to T, as long as the connection neither violates the degree constraint nor constitutes a cycle. If the number of vertices connected on that period is already the same as $MAXVT_i$, the algorithm will continue to the next period, and the process is similar to the previous period until all vertices are connected/installed. To illustrate the WWM1 algorithm we used one problem with vertex

order 10 in the data (datafile.22 dat). The weight of the problem is given in Table 1. The data represents a complete graph of order 10, and the graph is illustrated in the Fig. 1. Moreover, we use the same set of HVT_i as in [12-17] as given in Table 2, and also use $MaXVT_i$ as the floor function of $\frac{n-1}{3}$, $MaxVT_i = \left\lfloor \frac{n-1}{3} \right\rfloor$.

Table 1: datafile.22.dat (graph with order 10)

Edge	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}	e_{17}	e_{18}	e_{19}	$e_{1,10}$	e_{23}	e_{24}	e_{25}	e_{26}	e_{27}	e_{28}
Weight	740	572	447	835	427	807	362	832	120	221	109	276	741	978	352
Edge	e_{29}	$e_{2,10}$	e_{34}	e_{35}	e_{36}	e_{37}	e_{38}	e_{39}	$e_{3,10}$	e_{45}	e_{46}	e_{47}	e_{48}	e_{49}	$e_{4,10}$
Weight	368	403	505	921	757	884	369	886	545	639	253	750	251	187	857
Edge	e_{56}	e_{57}	e_{58}	e_{59}	$e_{5,10}$	e_{67}	e_{68}	e_{69}	$e_{6,10}$	e_{78}	e_{79}	$e_{7,10}$	e_{89}	$e_{8,10}$	$e_{9,10}$
Weight	807	926	781	605	112	559	411	473	743	882	693	851	509	434	828

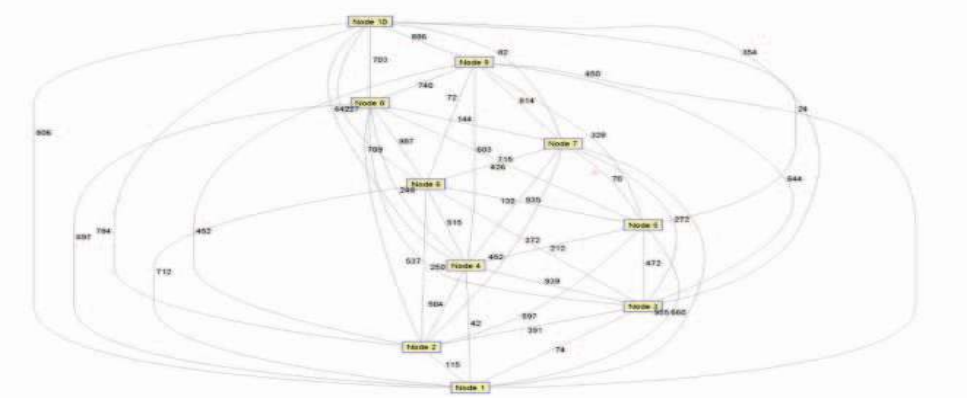

Fig. 1: The graph of datafile.22.dat.

Table 2: Element of HVT_i for every period

n	HVT_1	HVT_2	HVT_3
10	2	3	4
20	2	3	4
30	2,3	4,5	6,7
40	2,3,4	5,6,7	8,9,10
50	2,3,4,5	6,7,8,9	10,11,12,13
60	2,3,4,5,6	7,8,9,10,11	12,13,14,15
70	2,3,4,5,6,7	8,9,10,11,12,13	14,15,16,17,18,19
80	2,3,4,5,6,7,8	9,10,11,12,13,14,15	16,17,18,19,20,21,22
90	2,3,4,5,6,7,8	9,10,11,12,13,14,15	16,17,18,19,20,21,22
100	2,3,4,5,6,7,8,9	10,11,12,13,14,15,16,17	18,19,20,21,22,23,24,25

For the first period, the algorithm checks the vertices in HVT_1= {2}. Vertex 2 is the only vertex in HVT_1 then the algorithm connects vertex 2 to T using edge e_{12} with weight 740. V and T are updated to V= {1,2} and T ={$e_{1\text{-}2}$}. Next, the algorithm checks the difference of $MAXVT_1 = \left\lfloor \frac{n-1}{3} \right\rfloor$= 3 and the number of vertices in HVT_1. $|HVT_1|$ is the number of vertices in HVT_1, $|HVT_1|$ =1. Since the difference of $MAXVT_1$ and $|HVT_1|$ is 2, then it is possible to add two more vertices. Therefore, the algorithm searches for the next smallest edge with the vertices in the network which is $e_{2\text{-}4}$ with weight 109. Adding $e_{2\text{-}4}$ to

T neither creates a cycle on T nor violates degree restriction on the vertices in V. Then, e_{2-4} is added to T and vertex 4 to V. V and T are again updated to V= {1,2,4} and T = {e_{1-2}, e_{2-4}}. Since vertex 4 was already added to V, then only one more vertex can be added, and e_{1-10} is the next smallest edge with weight 120. Adding e_{1-10} neither creates a cycle in T nor violates degree restriction on vertices in V. Therefore e_{1-10} was added to T and vertex 10 to V. V and T are updated to V= {1,2,4,10} and T = {e_{1-2}, e_{2-4}, e_{1-10}}. Since $MAXVT_1$ is achieved, the first period is finished. The following figure shows the network after the first period is finished.

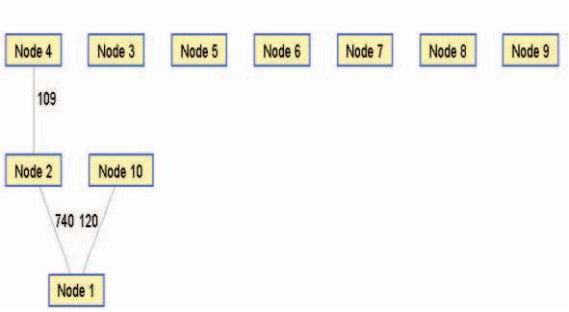

Fig. 2: The network after the first period of WWM1 algorithm is done.

On the second period, HVT_2 ={3}, therefore the algorithm connects vertex 3 to the network using the smallest edge of incidence with vertex 3; edge e_{2-3} is the smallest. Adding e_{2-3} to T neither violates the degree condition nor constitutes a cycle, therefore e_{2-3} is added to T and vertex 3 to V. V and T are updated to V= {1,2,4,10,3} and T = {e_{1-2}, e_{2-4}, e_{1-10}, e_{2-3}}. Then, the algorithm checks $MAXVT_2 - |HVT_2|$ which is 2. Therefore, in this period, we can add two more edges to T and two more vertices in V. The next smallest edge of incidence with the vertices in V is e_{5-10} with a weight of 112. Adding e_{5-10} to T neither violates the degree condition nor constitutes a cycle, therefore e_{5-10} is added to T and vertex 5 to V. V and T are updated to V= {1,2,4,10,3,5} and T = {e_{1-2}, e_{2-4}, e_{1-10}, e_{2-3}, e_{5-10}}. Next, the algorithm searches the smallest edge of incidence with the vertices in V which is e_{4-9} with a weight of 187. Adding e_{4-9} to T neither violates the degree condition nor constitutes a cycle, therefore e_{4-9} is added to T and vertex 9 to V. V and T are updated to V= {1,2,4,10,3,5,9} and T = {e_{1-2}, e_{2-4}, e_{1-10}, e_{2-3}, e_{5-10}, e_{4-9} }. This is the end of the second period, and the network is illustrated in Fig. 3.

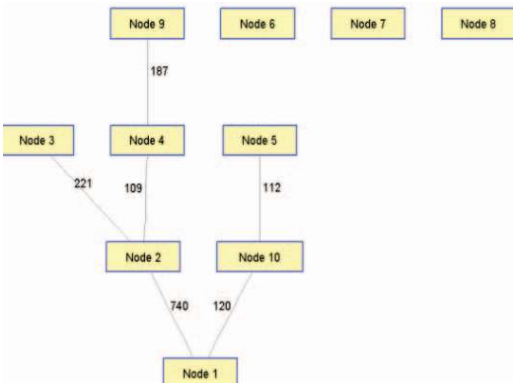

Fig. 3: The network after the second period of WWM1 algorithm is done.

Now, there are seven vertices already in the network, which are vertex 1, 2, 10, 4, 3, 5, and 9; and there are three more vertices that need to be connected. Those vertices are vertex 6, 7, and 8. Please note here that vertex is 4 already in the network (installed on the

https://doi.org/10.31436/iiumej.v21i1.1088

first period). After the second period was done, vertex 3 (the member of vertices in HVT_2) was in the network, as well as vertex 4 (an element in HVT_3). Therefore, the algorithm is just searching for the smallest edges of incidence to the uninstalled vertices and to connect them to the network. Three edges that satisfy that condition and also neither create a cycle nor violate the degree condition (maximum degree is 3 for every vertex) are e_{4-8}, e_{6-8}, e_{6-7} with weight 251, 411, and 559 respectively. V and T are updated to V= {1,2,4,10,3,5,9,8,6,7} and T = {e_{1-2}, e_{2-4}, e_{1-10}, e_{2-3}, e_{5-10}, e_{4-9}, e_{4-8}, e_{6-8}, e_{6-7}}. All vertices are then in set V, so the algorithm stops. Figure 4 represents the network when the third period is finished.

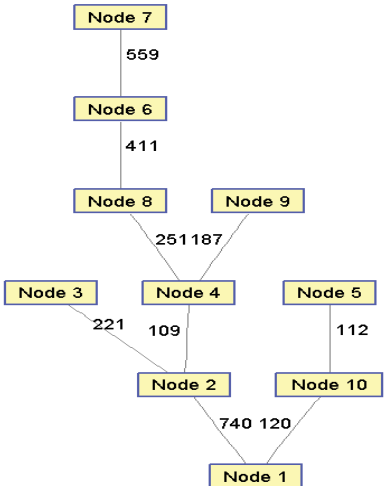

Fig. 4: The network when the third period is finished.

Table 3 gives the information about the period when the edges are installed/connected, the weight, and the total weight after the connection are done.

Table 3: The period when the vertices were connected in WWM1 algorithm

Period	From vertex	To vertex	Weight
1	1	2	740
1	1	10	120
1	2	4	109
2	2	3	221
2	10	5	112
2	4	9	187
3	4	8	251
3	8	6	411
3	6	7	559
Total			**2710**

3.2.2 WWM2 Algorithm

The WWM2 algorithm is similar with the WWM1 algorithm, except in the WWM2, the process of connecting the vertices in HVT_i is more flexible. Those vertices in HVT_i

can be installed in the beginning of the period or at the end, as long as the installation/connection process is still within that period or before. In the first period of the WWM2 algorithm, vertex 2 is not installed first, but the algorithm searches for possible smallest edges among those in the data that connect to vertex 1. This is possible because $MAXVT_1 = 3$ and the number of $HVT_1 = |HVT_1| = 1$ (only consist of vertex 2). Therefore, it is possible to add two more vertices (but not vertex 2) in the network. Therefore vertex (node) 10 was connected first, followed by vertex 5, and finally, because vertex 2 must be installed/connected in the first period, then vertex 2 is installed. Figure 5 shows the network after the first period of the WWM2 algorithm is done.

Fig. 5: The network after the first period of WWM2 algorithm is done.

For the second period of the WWM2 Algorithm, the vertex in HVT_2 is vertex 3. However, the algorithm first checks the difference between $MAXVT_2$ and the number of HVT_2. Since $MAXVT_2 - |HVT_2| = 2$, then it is possible to add the other two vertices (besides vertex 3). However, if vertex 3's incidence with the smallest edge that connected with the vertices already in the network during the first period, then vertex 3 will be installed/connected first on the second period (of course, without violating the degree requirement on the vertices that are already in the network). Since the smallest edge on the second period is edge e_{24} then that edge is connected first on the second period, followed by edge e_{49}. After edge e_{49} was added in the network, then there are no more choices except adding the smallest edge of incidence with vertex 3, because vertex 3 must be installed/connected on the second period or before. Figure 6 shows the network after the second period of the WWM2 algorithm is done.

Similar with the WWM1 algorithm, on the third period of the WWM2 there are 7 vertices already in the network and there are three more vertices still uninstalled. Those vertices are vertex 6, 7, and 8. The difference between WWM1 and WWM2 is that vertex 4 in WWM1 is installed/connected on the first period, while in WWM2, vertex 4 is connected on the second period. The next three smallest are e_{48}, e_{28}, and e_{210}. There are no degree restrictions and cycle occurrences by adding e_{48}, therefore vertex 8 is connected. However, there is a problem with adding e_{28}. The degree of vertex 2 is already 3, then adding e_{28} will violate the degree restriction, therefore e_{28} is omitted, and not connected. A different reason is applied to e_{210}. Adding e_{210} will create a cycle, therefore e_{210} is also omitted. Now, the algorithm is just searching for the next two smallest edges of incidence to the uninstalled vertices to connect them to the network (as long as the connection neither violates the degree restriction nor creates a cycle). The next two smallest are e_{68} and e_{67}. Figure 7 shows the network after the third period is finished.

https://doi.org/10.31436/iiumej.v21i1.1088

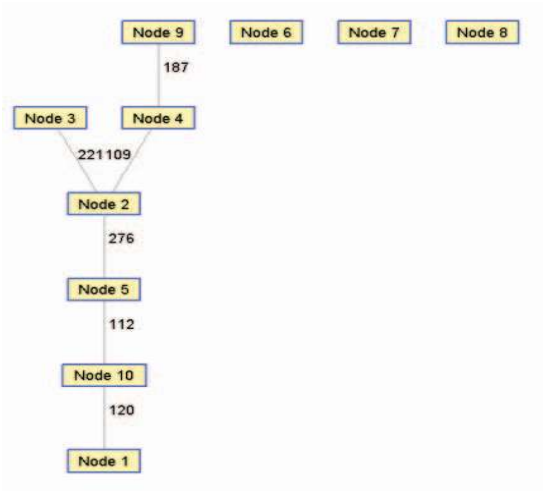

Fig. 6: The network after the second period of WWM2 algorithm is done.

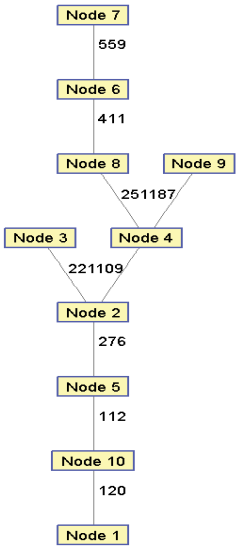

Fig. 7: The network after the second period of WWM2 algorithm is done.

Table 4: The period when the vertices were connected in the WWM2 algorithm

Period	From vertex	To Vertex	Weight
1	1	10	120
1	10	5	112
1	5	2	276
2	2	4	109
2	4	9	187
2	2	3	221
3	4	8	251
3	8	6	411
3	6	7	559
Total			**2246**

Table 4 gives the information about the period when the edges are installed/connected, the weight, and the total weight after the WWM2 Algorithm finishes the process.

https://doi.org/10.31436/iiumej.v21i1.1088

4. RESULTS AND DISCUSSION

The WWWM1 and WWM2 algorithms were compared against WADR5, WAC1 and WAC2 algorithms because those three algorithms are developed based on the same algorithm (Prim's algorithm). Moreover, those algorithms were implemented using the same data, same set of HVT_i,, $MAXVT_i$ and the number of periods. The data used for implementation was random problems generated using uniform distribution with weight ranging from 1 to 1000 (integer). For every vertex order, there are 30 problems generated and the vertex order used are 10 to 100 in increments of 10. Therefore, there are a total of 300 problems implemented. The solution taken is the average solution from 30 problems for every vertex order. The following table shows the result:

Table 5: The Comparative Solutions

Vertex order	MST	DCMST	WWM1	WWM2	WADR5	WAC1	WAC2
10	1129.43	1178.8	1498.6	1286.6	1341.93	1495.1	1359.93
20	1196.1	1299	1804.97	1428.57	1557.63	1790.37	1437.5
30	1177.43	1319.53	2039.97	1490.03	1755.40	2018.9	1516.43
40	1151.23	1286.3	2117.1	1440.03	1719.27	2079.73	1455.3
50	1223.43	1356.47	2427.27	1566.6	1844.10	2381	1603.7
60	1175.57	1320.73	2389.87	1573.57	1851.20	2364.4	1639.53
70	1242.1	1410.03	2558.5	1612.27	1963.83	2520.2	1671.9
80	1236.83	1410.23	2579.4	1675.2	1942.40	2547.8	1722.23
90	1248	1404.93	2618.43	1613.23	1941.33	2588.07	1649.33
100	1234.1	1370.8	2564.07	1567.53	1992.40	2535.2	1597.63
Average	**1201.42**	**1335.68**	**2259.82**	**1525.36**	**1790.95**	**2232.08**	**1565.35**

From Table 5 we can see the average solutions for MST, DCMST, WWM1, WWM2, WADR5, WAC1, and WAC2. For the algorithms developed for solving MPDCMST (WWM1, WWM2, WADR5, WAC1, and WAC2) the best performance is gained by the WWM2 algorithm, which is slightly close to the performance of WAC2, while the WWM1 and WAC1 are the two worse algorithms compared in this study. WWM1 and WAC1 are two algorithms developed based on the Modified Prim's algorithm where the priority vertices in the set HVT_i must be installed/connected as soon as possible, while WWM2 is a modification of WWM1, and WAC2 is a modification of WAC1. The modification made for those two algorithms is the same: relaxing the process of connecting the priority vertices in HVT_i whilst also maintaining those vertices to be connected in the i^{th} period or before. WADR5 is an algorithm developed based on Modified Prim's algorithm where the algorithm searches two smallest edges for being considered to be connected in the network, except the last one in the period. In this study, we compared the ratio of the algorithms (WWM1, WWM2, WADR5, WAC1, and WAC2) against their lower bound (DCMST). The ratio is $\frac{H-LB}{LB}$, where H is the heuristic (algorithm) and LB is the lower bound (DCMST). Table 6 below shows the ratio of $\frac{H-LB}{LB}$.

https://doi.org/10.31436/iiumej.v21i1.1088

Table 6: The value of ratio $\frac{H-LB}{LB}$ of the algorithms against the lower bound

Vertex order	MST	DCMST	WWM1	WWM2	WADR5	WAC1	WAC2	$\frac{WWM1-DCMST}{DCMST}$	$\frac{WWM2-DCMST}{DCMST}$	$\frac{WADR5-DCMST}{DCMST}$	$\frac{WAC1-DCMST}{DCMST}$	$\frac{WAC2-DCMST}{DCMST}$
10	1129.43	1178.8	1498.6	1286.6	1341.93	1495.1	1359.93	27.129%	9.145%	13.839%	26.832%	15.366%
20	1196.1	1299	1804.97	1428.57	1557.63	1790.37	1437.5	38.950%	9.974%	19.910%	37.827%	10.662%
30	1177.43	1319.53	2039.97	1490.03	1755.40	2018.9	1516.43	54.598%	12.921%	33.032%	53.001%	14.922%
40	1151.23	1286.3	2117.1	1440.03	1719.27	2079.73	1455.3	64.588%	11.952%	33.660%	61.683%	13.138%
50	1223.43	1356.47	2427.27	1566.6	1844.10	2381	1603.7	78.940%	15.491%	35.949%	75.530%	18.226%
60	1175.57	1320.73	2389.87	1573.57	1851.20	2364.4	1639.53	80.950%	19.143%	40.165%	79.022%	24.138%
70	1242.1	1410.03	2558.5	1612.27	1963.83	2520.2	1671.9	81.450%	14.342%	39.276%	78.733%	18.572%
80	1236.83	1410.23	2579.4	1675.2	1942.40	2547.8	1722.23	82.906%	18.789%	37.736%	80.665%	22.124%
90	1248	1404.93	2618.43	1613.23	1941.33	2588.07	1649.33	86.374%	14.826%	38.180%	84.213%	17.396%
100	1234.1	1370.8	2564.07	1567.53	1992.40	2535.2	1597.63	87.049%	14.352%	45.346%	84.943%	16.548%
			Average					68.293%	14.094%	33.709%	66.245%	17.109%

From Table 6 we see that average ratio of the WWM1 and WAC1 algorithms against the lower bound are above 50%. These values show that the performance of the algorithm is not good. The performance of WWM2 and WAC2 are below 20%, while the ratio of WADR5 is around 34%. The smaller the value of the ratio, the better the performance. The closer the solution of the algorithm to the solution of the DCMST, the better the algorithm. Figure 8 shows the performance of the algorithm.

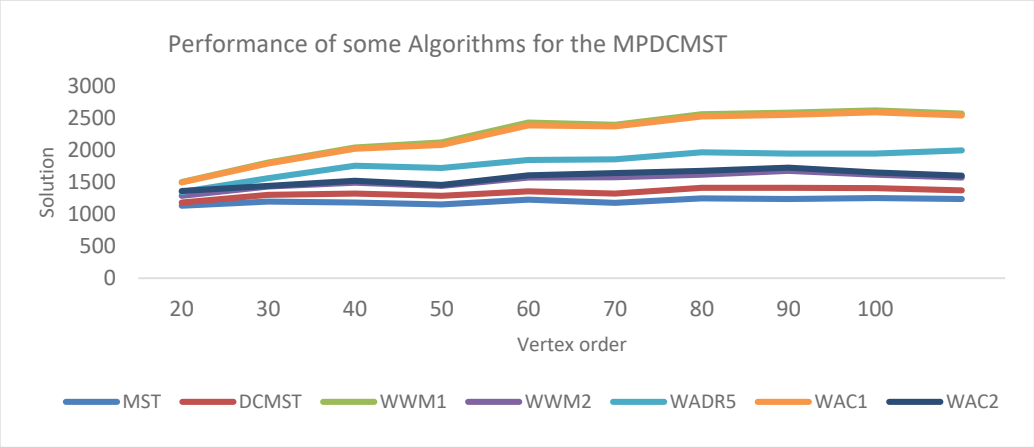

Fig. 8: Performance of WWM1, WWM2, WADR5, WAC1, and WAC2 algorithms.

5. CONCLUSIONS

From the results and discussion we can see that among WWM1, WWM2, WADR5, WAC1, and WAC2 algorithms, WWM2 performs the best, followed by WAC2 algorithm. These two algorithms have similar approaches: relaxing the time of installation for vertices in HVT_i or adding flexibility to the algorithm. Accepting flexibility in the process of connecting/installing the vertices without violating the rules gives a better solution rather than sticking with the schedule of connecting the vertices in a certain order.

ACKNOWLEDGEMENT

This research work was supported by Research Grant No: 582/ UN26.21/KU/2017. The authors would also like to acknowledge the contributions and financial support from the Directorate General of Higher Education, Ministry of Research, Technology and Higher Education, Republic of Indonesia.

REFERENCES

[1] Gabow H.N and R.E. Tarjan.(1984) Efficient algorithms for a family of matroid intersection problems. Journal of Algorithms, 5:80-131.
[2] Garey, M.R.,and Johnson, D.S.(1979) Computers and Intractibility, A Guide to the Theory of NP-Completeness. Freemann, San Francisco.
[3] Narula, S.C., and C. A.Ho.(1980) Degree-Constrained Minimum Spanning Tree. Computer and Operation Research, 7:239-249.
[4] Krishnamoorthy, M.,A.T. Ernst and Y. M Sharaila.(2001) Comparison of Algorithms for the Degree Constrained Minimum Spanning Tree. Journal of Heuristics, 7(6): 587-611.
[5] Deo N. and N. Kumar.(1997) Computation of Constrained Spanning Trees: A Unified Approach. Network Optimization (Lecture Notes in Economics and Mathematical Systems. Editor : Panos M. Pardalos, et al. ,Springer-Verlag, Berlin, Germany: 194 – 220.
[6] Zhou, G. and M Gen. (1997) A Note on Genetics Algorithms for Degree- Constrained Spanning Tree Problems. Networks, Vol. 30: 91 – 95.
[7] Wamiliana. (2004) Solving the Degree Constrained Minimum Spanning Tree Using Tabu and Penalty Method. Jurnal Teknik Industri:1-9.
[8] Caccetta L. and Wamiliana.(2001) Heuristics Algorithms for the Degree Constrained Minimum Spanning Tree Problems.Proceeding of the International Congress on Modelling and Simulation (MODSIM),Canberra, Editors: F. Ghassemi et.al:2161-2166.
[9] Wamiliana and L. Caccetta. (2003)Tabu search Based Heuristics for the Degree Constrained Minimum Spanning Tree Problem.Proceeding of South East Asia Mathematical Society:133-140.
[10] Wamiliana and L. Caccetta.(2012) The Modified CW1 Algorithm for The Degree Restricted Minimum Spanning Tree Problem, Proceeding of International Conference on Engineering and Technology Development, Bandarlampung 20-21 June:36-39.
[11] Kawatra R. (2002)A multi period degree constrained Minimum Spanning Tree Problem, European Journal of Operational Research,143: 53 – 63.
[12] [Wamiliana, D. Sakethi, and R. Yuniarti,(2010) Computational Aspect of WADR1 and WADR2 Algorithms for The Multi Period Degree Constrained Minimum Spanning Tree Problem. Proceeding SNMAP, Bandar lampung 8 – 9 December:208 – 214.
[13] Wamiliana, Amanto, and M. Usman.(2013) Comparative Analysis for The Multi Period Degree Constrained Minimum Spanning Tree Problem. Proceeding The International Conference on Engineering and Technology Development (ICETD):39 – 43.
[14]]Wamiliana, F. A.M. Elfaki, M. Usman, and M. Azram. (2015) Some Greedy Based Algorithms for Multi Periods Degree Constrained Minimum Spanning Tree Problem. ARPN Journal of Engineering and Applied Sciences, 2015: 10 (21): 10147 – 10152.
[15] Wamiliana, M. Usman, D. Sakethi, R. Yuniarti, and A. Cucus.(2015) 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
[16] Wamiliana, Asmiati, M. Usman, A. Hijriani, and W. C. Hastono. (2018) Comparative Analysis of Some Modified Prim's Algorithms to Solve the Multiperiod Degree Constrained Minimum Spanning Tree Problem. Indian Journal of Science and Technology,11(11):1-6.

[17] Wamiliana, Warsono, Asmiati, A. Hijriani, and W. C. Hastono.(2018) Different Time Installation Effect on The Quality Of The Solution For The Multiperiod Installation Problem Using Modified Prim's Algorithm. Far East Journal of Electronics and Communications, 18(2): 291-300.
[18] GNU Octave. Available: https://www.gnu.org/software/octave.