

## The Use of Probability and Edge Analysis to Solve the Multi-Period Degree Constrained Minimum Spanning Tree Problem

Wamiliana<sup>\*</sup>, Akmal Junaidi<sup>2</sup>, Mohammad Danil Hendry Gamal<sup>3</sup>, Taqwan Thamrin<sup>4</sup>

<sup>1</sup>Department of Mathematics, Faculty of Mathematics and Natural Science, Universitas Lampung, Bandar Lampung, 35145, Indonesia

<sup>2</sup>Department of Computer Science, Faculty of Mathematics and Natural Science, Universitas Lampung, Bandar Lampung, 35145, Indonesia

<sup>3</sup>Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Riau, Pekanbaru, 28293, Indonesia

<sup>4</sup>Faculty of Computer Science, Universitas Bandar Lampung, Bandar Lampung, 35145, Indonesia

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

### Abstract

The goal of the Multiperiod Degree Constrained Minimum Spanning Tree (MPDCMST) problem is to determine the smallest weight-spanning tree that satisfies the vertex installation criterion for each period and maintains the degree requirement in each vertex. This issue emerges as a network connection problem. The degree requirement indicates the reliability of each vertex, and the vertex connection/installation requirement denotes the priority vertices that must be inserted in the network within a specific time frame. The installation is split up into multiple phases/stages. This is because of various considerations such as severe weather, budgetary limitations, etc. In this research, two algorithms for solving the MPDCMST using probability hybridized with Prim's modification, and edge analysis are proposed. The algorithms are implemented on the undirected complete graph of orders 10 to 100. The solutions are compared with some heuristics which are already in the literature. The results show that the proposed algorithms perform better.

### Keywords

Multi-Period, Degree Constrained, Minimum Spanning Tree, Probability Factor, Edge Analysis

Received: 15 June 2024, Accepted: 28 September 2024

<https://doi.org/10.26554/sti.2024.9.4.999-1008>

## 1. INTRODUCTION

Undoubtedly, graph theory has proven to be one of mathematical disciplines that has been embraced in numerous practical applications. Graph theory is also distinguished by having precise dates of origin (Vasudev, 2006). A graph serves as a means to depict discrete entities and the association linking them. For a set  $V = \{v_1, v_2, \dots, v_n\}$  of vertices, where  $V$  is not empty, and a set of edges  $E$ , represented as  $e_{ij}$ ,  $i, j \in V$ , a graph  $G(V, E)$  is a structure consisting of an ordered pair of  $V$  and  $E$ . An object can be symbolized by a vertex, while the connection between objects can be denoted by an edge. The vertices can represent various entities like cities, stations, buildings, computers, people/organizations, and more. Meanwhile, the edges can symbolize roads, train tracks, pipes, relationships/connections, and similar elements. Moreover, the graph creation process allows for adaptability, enabling the depiction of edges without strict adherence to specific rules; they can be drawn in curved or straight lines.

There are a lot of uses of graph theory, for example, graph colouring is used to model practical scheduling applications (Thevenin et al., 2018; Thadani et al., 2022), Kawakura and

Shibasaki (2018) and Kannimuthu et al. (2020) use that concept in agriculture; a leaf-labelled tree is used to represent a phylogenetic tree by Huson and Bryant (2006) and Brandes and Cornelsen (2009); a combined tree is used to represent DNA by Mathur and Adlakha (2016); Hamiltonian circuits with relation in network design is explored by Hsu and Lin (2009); cycle graph, complete graph, and minimum spanning tree concepts are used in data security by Al Etaiwi (2014) in creating complex cipher; expander and extremal graph are used by Priyadarsini (2015) to create cipher for data security, while bipartite and corona graph are used by Ni et al. (2021) to do encryption in cryptography, and many more.

Given a connected weighted graph  $G(V, E)$ , where the weight is nonnegative, one of the classical problems is finding a minimum spanning tree (MST) of that graph. One of the fundamental concepts in graph theory and optimization is MST. This concept is commonly used to solve various real-world problems. It is a valuable concept with a variety of applications in diverse fields. It helps to optimize resource usage, connectivity, and efficiency while maintaining a connected and acyclic structure. There are numerous applications of the MST, such as constructing power grids for communication networks. To

determine the MST, three well-known techniques are available: Boruvka's algorithm which was introduced by [Boruvka \(1926\)](#), Prim's algorithm developed by [Prim \(1957\)](#), and Kruskal's algorithm by [Kruskal \(1956\)](#). The MST can identify clusters or groups of data points close to each other in data mining; design optimal electrical circuits with minimum total wire length, and so on.

The Degree constrained Minimum Spanning Tree (DCMST) Problem is a variation of the classic MST problem, which involves finding a tree that spans all the vertices in a given graph while satisfying a certain degree restriction on each vertex. In other words, it seeks to connect all vertices with the fewest total edge weight, while ensuring that each vertex adheres to a specific degree requirement. Because of the added constraints, solving the DCMST issue may be more difficult than solving the regular MST problem. The DCMST occurs during the designing of networks, where the degree limitation symbolizes the permissible quantity of connections for each vertex. In essence, the operational capacity of each vertex imposes a constraint on the count of links (such as wires or roads) that can link to a vertex. For instance, the use of the DCMST concept is relevant in the design of road systems. In this context, a set of roads needs to interconnect various suburbs or towns, while adhering to a condition that no more than a specified number of roads (e.g., four roads) are permissible to meet at a junction ([Wamiliana et al., 2020](#)).

To overcome this problem, many algorithms and techniques have been investigated, and they often involve a combination of graph theory, optimization, and heuristics to find feasible solutions that meet the degree restrictions while minimizing the total edge weights. Some of the algorithms that have already been proposed include Iterative refinement ([Deo and Kumar, 1997](#)), Branch and cut ([Caccetta and Hill, 2001](#)), Simulated Annealing ([Krishnamoorthy et al., 2001](#)), Tabu search ([Caccetta and Wamiliana, 2001](#)), Genetic Algorithm ([Zhou and Gen, 1997](#)), Ant colony ([Adasme and Firoozabadi, 2020](#)), Modified Branch and Bound and the idea of Lin Kernighan algorithm (one of the TSP heuristic approach that is frequently employed) ([Thiessen et al., 2020](#)), and many more.

A Multi-Period Degree Constrained Minimum Spanning Tree (MPDCMST) is a specialized variant of the Minimum Spanning Tree (MST) problem that considers degree constraints and spans multiple periods or time intervals. This problem arises in various applications, including network design, transportation planning, and resource allocation, where the goal is to find a spanning tree that satisfies both edge weight constraints and vertex degree constraints over multiple periods. In real situations, establishing connections between all elements within a network needs a specific duration and sequence of steps for successful implementation. The time frame for completion is subject to fluctuations based on the network's requirements and its relative importance. The MPDCMST adds the dimension of time intervals or periods. This means that the tree needs to satisfy degree constraints for each vertex in each period while minimizing the sum of edge weights over

all periods. This problem is more complex than the classical MST or the DCMST because of the added period constraint. The period constraint is added due to some conditions such as harsh weather, limitation of funds, and so on.

Solving the MPDCMST problem typically involves finding a set of trees, for each period, that collectively satisfy the degree constraints and minimize the sum of edge weights across all periods. This problem can have applications in various domains, such as designing communication networks that have varying demands over time or optimizing transportation routes considering varying traffic conditions. The problem might be tackled using a combination of techniques from graph theory, optimization, and algorithms for spanning trees, considering both the spatial and temporal dimensions of the problem.

[Kawatra \(2002\)](#) introduced the MPDCMST problem and implemented it on the directed graph and used a combination of branch exchange and Lagrangean relaxation. The order of the graphs ranges from 40 to 100 with a central vertex set at vertex 1. The investigation spanned a 10-year planning horizon. [Junaidi et al. \(2008\)](#) investigated the MPDCMST by developing two algorithms by modifying Kruskal's algorithm and tested the method using a variety of problems sourced from the TSPLIB. [Wamiliana et al \(2015a\)](#) developed four algorithms by modifying Kruskal's algorithms. The results show that in every period, the priority vertex is treated similarly as other vertices, and if at the end of the period that vertex is still uninstalled/ not connected, forcing it to be connected is better rather than connecting that vertex in the beginning of the period; while [Wamiliana et al. \(2015b\)](#) show that adopting Depth First Search Technique will improve the quality of the solution. [Wamiliana et al. \(2018\)](#) proposed WAC1, WAC2, and WAC3 algorithms. Those algorithms are based on Prim's modification algorithm, and implemented on undirected complete graphs of orders ranging from 10 to 100 increasing in increments of 10, where the weights of the edges are nonnegative with values generated uniformly from 1 – 1000. Prim's algorithm was modified because Prim's algorithm maintains connectivity, moreover, those algorithms adopted a one-year timeframe consisting of three periods. This modification of the MPDCMST approach aimed to reflect the real-world situation in Indonesia, where project funding is often allocated over three discrete terms. [Wamiliana et al. \(2020\)](#) use a similar reason when proposing WWM1 and WWM2 algorithms using GNU Octave. Even though Prim's algorithm is used more because it maintains connectedness, the modification of Kruskal's algorithm has also been investigated among others ([Junaidi et al., 2008](#); [Wamiliana et al., 2015a, 2018](#)).

This paper is organized as follows: In Section 1 the background, problem statement, and literature review are given. Section 2 is Methodology. In this section, the proposed algorithm for solving the MPDCMST is discussed. Section 3 is the Result, where the implementation of the proposed algorithm is given, followed by a conclusion in Section 4.



In WGAT1 and WGAT2 algorithms, we add probability for every edge incident to the vertices in  $HVT_s$ .  $HVT_s$  represent the collections of vertices that need to be connected on the  $s$  stage or period.  $MaxVT_s$  is the maximum number of vertices that can be linked in the  $s$  period, where  $s = 1, 2, 3$  (i.e., the number of periods needed until all vertices are connected is 3).  $MaxVT_s$  is  $\lfloor \frac{n}{3} \rfloor$ . The same set of elements for  $HVT_s$  as used in WAC's algorithms is chosen as shown in Table 1.

Table 1 shows the set of vertices that must be connected in every period. For example, for vertex order 30, the vertices that must be connected in the first period are vertices 2 and 3; in the second period, the vertices that must be connected are vertices 4, 5; and in the third period, the vertices that must be connected are vertices 6 and 7. Note that the vertices 4 or 5 have the probability to be connected in the first period. This situation occurs when  $MaxVT_1 - |HVT_1| > 0$  and all vertices in  $HVT_1$  are already stored in  $V$ , thus the algorithm chooses the smallest weighted edge that satisfies the constraints (not constitute cycle and not violated degree restriction). This is similar to vertices 6, and 7, those vertices may be already connected in the first or second period.

Like in the WAC's algorithms, in the WGAT1 and WGAT2, before starting the algorithms, vertex 1 is set as the central vertex and stored in a set of vertices  $V, V = \{1\}$ .  $T$  is empty ( $\emptyset$ ), and the maximum period is 3. The algorithms start by calculating the probability  $p$  of edges incident to vertices in  $HVT_1$ . Then, for those edges, a random number  $q$  is assigned ( $0 < q < 1$ ). If  $p \geq q$ , do edge analysis (check the smallest edge among the edges incident to  $HVT_s$ , and has the smallest path to the central vertex). If all edges satisfy the edge analysis condition, then choose the smallest edge, store it to  $T$  and the corresponding incident vertex to  $V$ . Otherwise, search for other smallest edges that are not incident to the vertices in  $HVT_s$  and choose the smallest edge. Note that the number of edges that can be selected in this condition is  $MaxVT_s - |HVT_s|$ . Next, check the degree and cycle conditions. If the edge does not violate both conditions then do the same steps until the first stage finishes. The algorithm does a similar process in the second and third stages, except in the third stage the algorithm checks if  $|T| = n - 1$  to ensure that the spanning tree is obtained.

Figure 1 is the flowchart of the proposed algorithm. The main difference between the algorithms proposed with those in the literature is in adopting the probability and edge analysis. That flowchart is applied for WGAT1 and WGAT2. The difference between WGAT1 and WGAT2 is in the implementation. In WGAT2, the algorithm is running 30 times for every problem on every vertex order (i.e., the algorithm runs 900 times for every order). For every problem, the best value is recorded, and then take the average of the best solution. The algorithms are written in Java.

This snippet code is written in Java to connect/install an edge to the network on a certain period.

```
Edge shortestEdge = getShortestEdge(period);
int idTarget = shortestEdge.target;
Vertex targetVertex = getVertexById(idTarget);
if(addVertex(targetVertex)) {
    nodes.add(new Node(targetVertex.id, period.group));
    shortestEdge.group = period.group;
    this.edges.add(shortestEdge);
    getVertexById(shortestEdge.source).link++;
    getVertexById(shortestEdge.target).link++;
}
```

The algorithm also checks whether a circuit has been constituted, and remove the considered edge which causes the circuit to occur, as shown in the following:

```
private boolean addVertex(Vertex vertex) {
    //Checking if circuit
    for(int i=0; i<this.linkedVertices.size();i++){
        if(vertex.id == this.linkedVertices.get(i).id) {
            return false;
        }
    }
}
```

### 3. RESULTS AND DISCUSSION

The WGAT1 and WGAT2 algorithms are implemented on complete graphs. Those graphs have orders ranging from 10 to 100, increasing in increments of 10. There are 30 problems generated for every vertex order (the problems are named as dat.1, dat.2, ..., dat.30). The WGAT1 and WGAT2 are compared with the WAC1, WAC2, and WAC3. Table 2 shows the results for order 10, 20, 30, and 40, Table 3 shows the results for order 50 to order 70, and Table 3 shows the results for order 80 to order 100

From Table 2 and Figure 2, it can be seen that for order graph of order 10, all algorithms give the same solution on nine problems (problems 1.dat, 2.dat, 6.dat, 9.dat, 16.dat, 17.dat, 18.dat, 20.dat, 23.dat), while for other problems, the WGAT2 has the smallest solution on 19 problems (3.dat, 4.dat, 5.dat, 7.dat, 8.dat, 10.dat, 11.dat, 12.dat, 13.dat, 14.dat, 15.dat, 21.dat, 22.dat, 24.dat, 25.dat, 26.dat, 28.dat, 29.dat, and 30.dat), where among those 10 problems the WAC2 shares the same solutions on 14 problems (3.dat, 4.dat, 5.dat, 8.dat, 10.dat, 12.dat, 13.dat, 15.dat, 21.dat, 22.dat, 25.dat, 28.dat, 29.dat, and 30.dat) and the WGAT1 shares the same solutions on 10 problems (3.dat, 5.dat, 12.dat, 13.dat, 15.dat, 21.dat, 26.dat, 28.dat, 29.dat, and 30.dat), and the WAC1 share the same solution in one problem (11.dat). The WAC2 has the smallest solutions on 19.dat and 27.dat.

Figure 3 shows the results for vertex order 20. From this figure, the WGAT2 has the smallest solutions on 29 problems except problem number 16 (16.dat), where the smallest solution is gained by the WAC2. The WAC1 and WAC3 have the same solutions as the WGAT2 only in one problem(2.dat), the WAC2 and WGAT1 share the same solutions as the WGAT2 on 19 problems (1.dat, 3.dat, 4.dat, 5.dat, 6.dat., 7.dat, 8.dat, 9.dat, 15.dat, 17.dat, 19.dat, 20.dat, 21.dat, 22.dat, 23.dat, 24.dat, 26.dat, 27.dat, and 28.dat).

From Table 2 and Figure 4 it can be seen that for graph of order 30, the solutions of the WGAT2 are the smallest



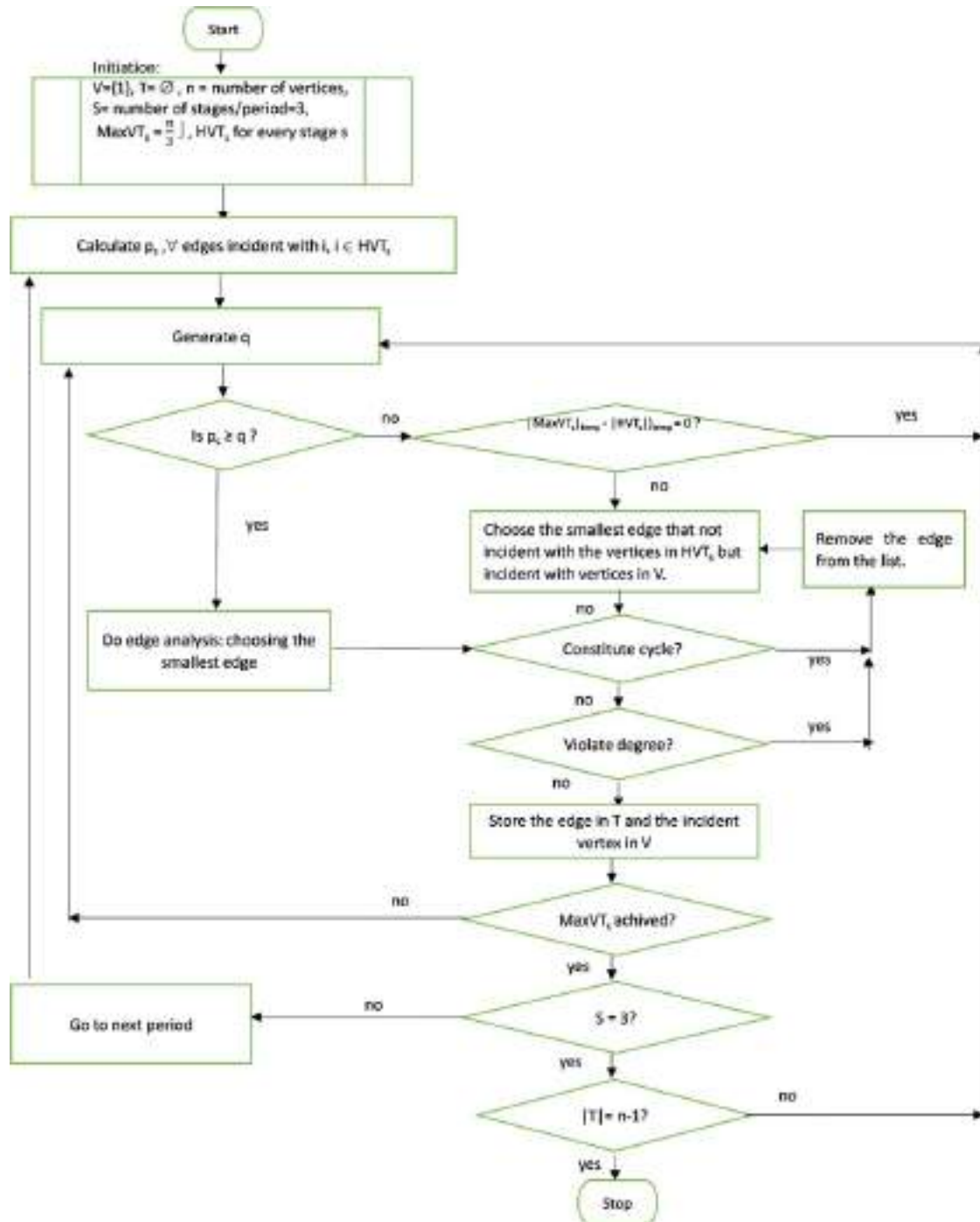


Figure 1. The Flowchart of the Algorithm

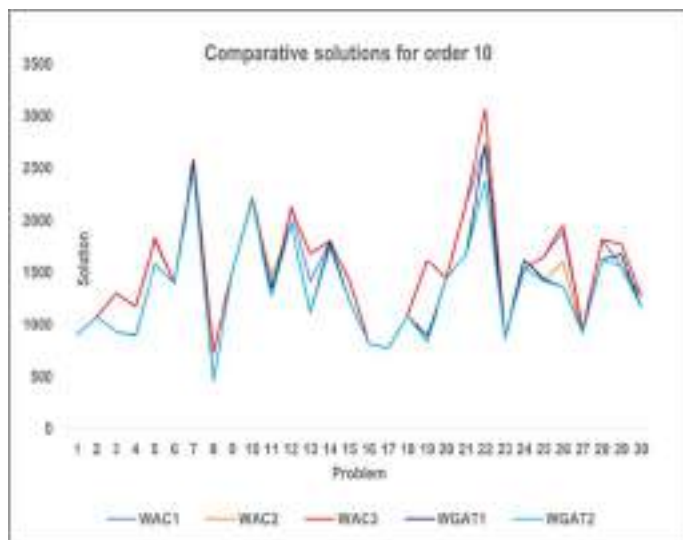
on 25 problems (1.dat, 2.dat, 3.dat, 4.dat, 5.dat, 6.dat, 7.dat, 8.dat, 9.dat, 10.dat, 11.dat, 14.dat, 15.dat, 16.dat, 17.dat, 18.dat, 19.dat, 23.dat, 24.dat, 25.dat, 26.dat, 27.dat, 28.dat, 29.dat, and 30.dat). Among those 25 problems, the WAC2 and WGAT1 have the same solutions on 13 problems (4.dat, 5.dat, 8.dat, 9.dat, 10.dat, 11.dat, 15.dat, 16.dat, 17.dat, 18.dat, 19.dat, 28.dat, and 29.dat). Out of 30 problems, the WAC1

and WAC3 have the smallest solutions on two problems (12.dat, and 22.dat), while the WAC2 has the smallest solution on three problems out of 30 problems (13.dat, 20.dat, and 21.dat).

For vertex order 40, the solutions of WGAT2 are the smallest on 29 problems out of 30 problems, except one problem (28.dat) where WAC2 performs better. Among those 29 problems, the WAC2 and WGAT1 have the same solutions on 6

**Table 4.** The Results for Order 80 to Order 100

Problem	Orde 80						Orde 90						Orde 100					
	WAC 1	WAC 2	WAC 3	WGA T1	WGA T2	WAC 1	WAC 2	WAC 3	WGA T1	WGA T2	WAC 1	WAC 2	WAC 3	WGA T1	WGA T2			
1.dat	2239	1570	2239	1570	1473	2334	1789	2450	1794	1770	2361	1574	2433	1555	1484			
2.dat	2611	2192	2673	2214	1719	2500	1932	2674	2027	1883	2887	1620	2890	1615	1555			
3.dat	3088	1873	3207	1864	1759	2466	1599	2531	1599	1599	1935	1474	1935	1474	1857			
4.dat	2463	1515	2463	1515	1475	2608	1762	2690	1804	1607	2005	1405	2074	1425	1874			
5.dat	2868	1827	3004	1827	1808	3206	1693	3290	1728	1537	1820	1601	1926	1647	1535			
6.dat	2813	1540	3551	1542	1542	2705	1694	2793	1709	1653	2907	1553	3041	1553	1534			
7.dat	3099	1959	3187	1959	1959	2415	1557	2415	1651	1424	2397	1626	2531	1626	1529			
8.dat	1878	1482	1873	1487	1458	2277	1443	2277	1443	1443	2846	1533	2846	1533	1553			
9.dat	3241	1603	3251	1603	1485	2428	1920	2697	1920	1627	2339	1536	2487	1536	1536			
10.dat	3051	1964	3821	1964	1839	2271	1628	2271	1638	1429	2529	1854	2779	1854	1693			
11.dat	2160	1653	2169	1702	1672	3136	1767	3266	1767	1767	2536	1629	2536	1629	1340			
12.dat	1692	1324	1705	1324	1324	2537	1731	2586	1731	1375	2650	1481	3019	1481	1457			
13.dat	2560	1588	3060	1588	1502	2580	1437	2580	1443	1341	2341	1567	2452	1563	1551			
14.dat	1892	1421	1892	1470	1453	2299	1450	2321	1450	1403	3112	1880	3723	1811	1810			
15.dat	2642	1918	2733	1918	1915	2542	1701	3016	1707	1375	3209	1834	3434	1835	1610			
16.dat	1800	1434	1934	1434	1401	2646	1310	2692	1310	1271	2023	1410	2143	1410	1306			
17.dat	2461	2050	2491	2053	1709	2574	1883	2582	1879	1857	2412	1654	2554	1654	1509			
18.dat	2479	1718	2493	1746	1735	1980	1341	2184	1341	1341	2770	1686	2770	1701	1544			
19.dat	2964	1560	2964	1560	1560	2569	1791	2677	1791	1705	3137	1476	3212	1476	1428			
20.dat	3166	1818	3110	1818	1608	3838	1716	4126	1716	1609	2216	1589	2374	1621	1487			
21.dat	2834	1857	3094	1857	1695	2516	1674	2581	1678	1574	2121	1631	2154	1651	1518			
22.dat	2589	1626	3071	1819	1551	2739	1874	2792	1874	1750	2629	1497	2771	1497	1288			
23.dat	2700	2236	2858	2211	1615	2099	1263	2099	1308	1227	2844	1854	3014	1863	1727			
24.dat	2271	1955	2317	2049	1626	2271	1670	2514	1704	1630	3192	1530	3473	1533	1468			
25.dat	2247	1813	2294	1813	1568	1722	1098	1750	1098	1098	2429	1413	2528	1446	1387			
26.dat	2548	1782	2874	1794	1523	3520	1640	3882	1639	1501	2333	1448	2722	1448	1398			
27.dat	2681	2032	3020	2045	1769	2834	1638	2905	1643	1604	2085	1405	2238	1405	1455			
28.dat	2718	1449	2561	1503	1459	2908	1799	3126	1799	1616	2481	1662	2487	1720	1628			
29.dat	2220	1504	2876	1540	1344	2590	2013	2761	1954	1694	3279	1853	3329	1853	1821			
30.dat	2459	1404	2459	1467	1421	2532	1667	2863	1667	1592	2231	1634	2231	1636	1510			
Average	2548	1722	2708	1742	1599	2588	1649	2713	1660	1543	2535	1598	2670	1602	1511			



**Figure 2.** The Comparative Solutions for Order 10



**Figure 3.** The Comparative Solutions for Order 20

problems (6.dat, 12.dat, 16.dat, 18.dat, 22.dat, 30.dat).

The performance of the algorithms on vertex order 50 are shown on Figure 6. For vertex order 50, out of 30 problems, the solutions of WGAT2 algorithm are the smallest on 28 problems (among those WGAT1 share the same solutions in nine problems (2.dat, 4.dat, 5.dat, 10.dat, 17.dat, 20.dat, 21.dat, 26.dat, and 30.dat), while WAC2 also share the same solution with WGAT2 on 8 problems (2.dat, 4.dat, 5.dat, 17.dat, 20.dat, 21.dat, 26.dat, and 30.dat). Out of 30 problems, the WAC2 solutions are the smallest on two problems (3.dat and 11.dat).

Figure 7 shows the performance of the algorithms on vertex order 60. The solutions of the WGAT2 algorithm are the smallest on 26 problems, where three problems (10.dat,

16.dat, and 27.dat) out of those 26 problems, the solutions of the WAC2 and WGAT1 algorithms also the same with the WGAT2 algorithm. The smallest solutions of the rest four problems are gained by the WAC2 algorithm (4.dat, 15.dat, 28.dat, and 30.dat). Table 4 below shows the results for order 80 to order 100.

The results for vertex order 70 and 80 can be seen on Table Figure 5. For vertex order 70, the solutions of the WGAT2 algorithm on 26 problems out of 30 problems are the smallest, where six problems (1.dat, 6.dat, 9.dat, 15.dat, 26.dat, 29.dat) of those 26 problems, the solutions of the WGAT1 also the same with the WGAT2 algorithms, and five problems (1.dat, 6.dat, 9.dat, 15.dat, 26.dat) the solutions of the WAC2 algo-

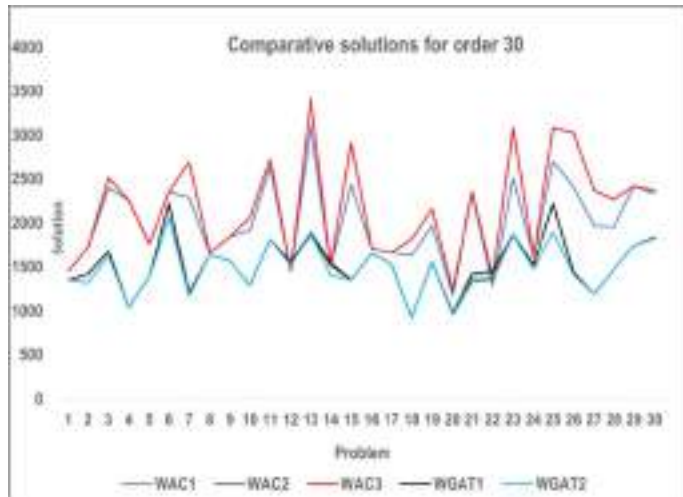


Figure 4. The Comparative Solutions for Order 30

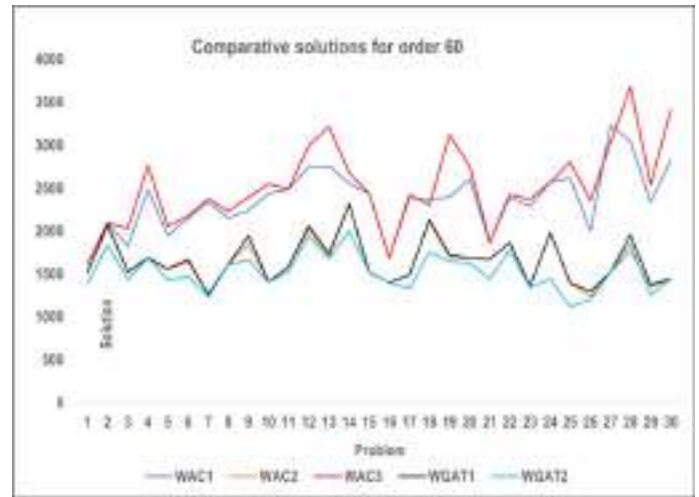


Figure 7. The Comparative Solutions for Order 60

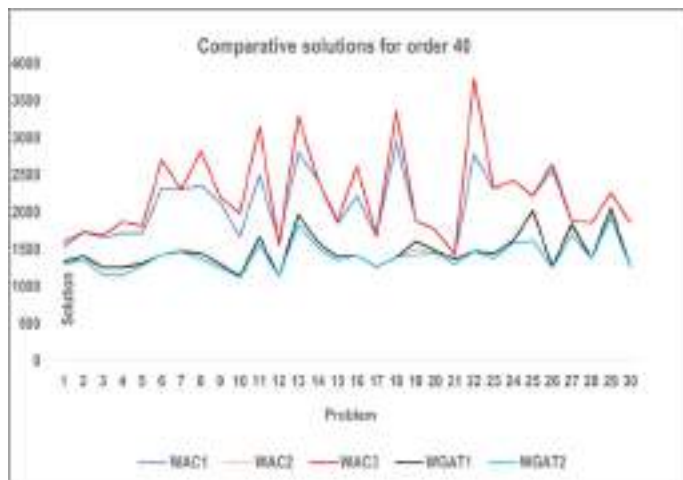


Figure 5. The Comparative Solutions for Order 40



Figure 8. The Comparative Solutions for Order 70

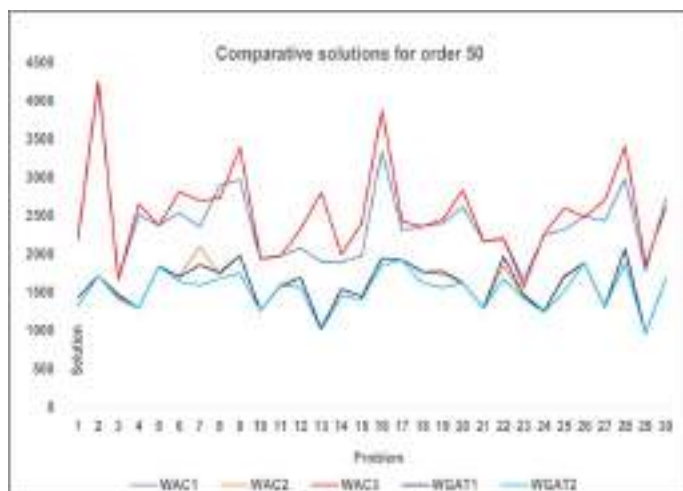


Figure 6. The Comparative Solutions for Order 50

Table 5. The Comparison of the Average Solutions of WAC1, WAC2, WAC3, WGAT1 and WGAT2 for Every Vertex Order

Order	WAC1	WAC2	WAC3	WGAT1	WGAT2
10	1495	1360	1526	1369	1340
20	1790	1438	1814	1443	1417
30	2017	1516	2164	1526	1491
40	2080	1455	2233	1467	1403
50	2381	1604	2533	1607	1532
60	2364	1640	2516	1660	1543
70	2520	1672	2655	1679	1582
80	2548	1722	2708	1742	1599
90	2588	1649	2713	1660	1543
100	2535	1598	2670	1602	1511

gorithm also the same as the WGAT2. The smallest solutions of the rest four problems are gained by the WAC2 algorithm (2.dat, 8.dat, 10.dat, 19.dat).

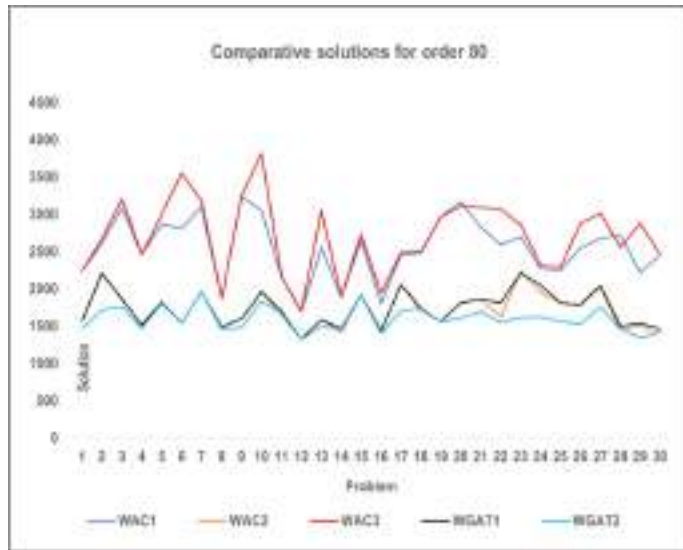


Figure 9. The Comparative Solutions for Order 80

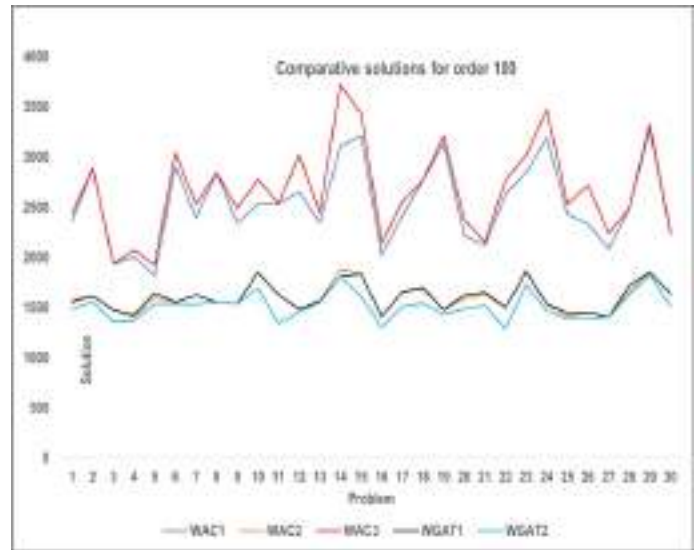


Figure 11. The Comparative Solutions for Order 100

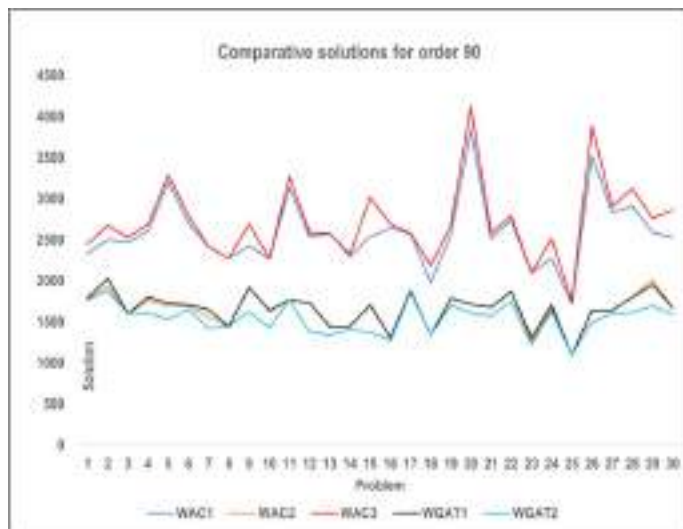


Figure 10. The Comparative Solutions for Order 90

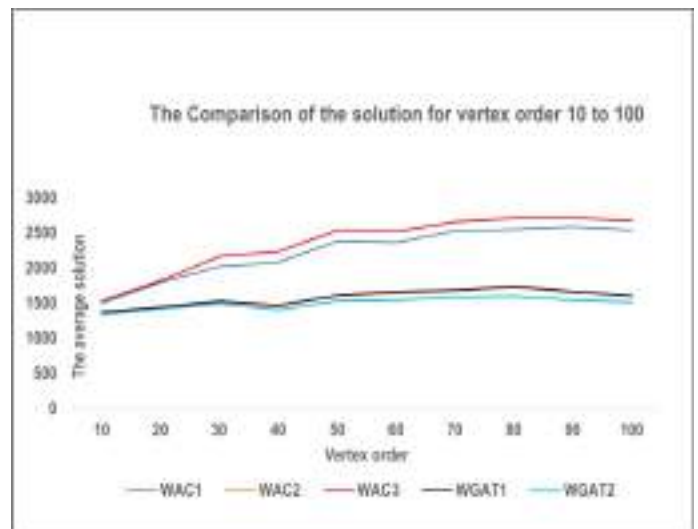


Figure 12. The Comparison of the Average Solutions for Vertex Order 10 to 100

For vertex order 80, the solutions of the WGAT2 algorithms on 22 problems are the smallest, where one problem (7.dat) out of those 22 problems, the solutions of the WAC2 and WGAT1 also the same as the WGAT2 algorithm. The smallest solutions of the rest 8 problems are gained by the WAC2 algorithm (6.dat, 11.dat, 12.dat, 14.dat, 18.dat, 19.dat, 28.dat, and 30.dat).

The results for vertex order 90 and 100 can be seen on Figure 6. For vertex order 90, the solutions of the WGAT2 algorithm on all 30 problems are the smallest, where five problems (3.dat, 8.dat, 11.dat, 18.dat, and 25.dat) out of 30 problems, the solutions of WAC2 and WGAT1 also the same as the WGAT2 algorithm.

For vertex order 100, the solutions of the WGAT2 algorithm on all 30 problems are the smallest, where 3 problems

(8.dat, 9.dat, and 27.dat) out of 30 problems, the solutions of WAC2 and WGAT1 are also the same as the WGAT2 algorithm.

From the discussion and the figures above, we can see that in general, the performance of the WGAT2 algorithm is the best among all the algorithms compared. The comparison of the average solutions for every order is shown in Table 5.

Table 5 shows that the performance of WAC2, WGAT1 and WGAT2 surpasses WAC1, and WAC3. The WAC2 performs better than WGAT1, but cannot surpasses the WGAT2 algorithm. In overall, the performance of WGAT2 is the best.

Figure 12 compares the solutions of the WAC1, WAC2, and WAC3. WGAT1, and WGAT2 algorithms using a line chart. It also can be seen that the WGAT2 is the best among



others. Introducing probability for edges incident to vertices in HVTs gives a chance for the algorithm seeking for better solution. The average solutions of WGAT2 are the smallest among others.

#### 4. CONCLUSIONS

According to the preceding discussion, we can conclude that in general, the performance of the WAC2, WGAT1, and WGAT2 better than that of the WAC1, and WAC3. Moreover, the WGAT2 performs the best among others, where in every graph order, the solutions of WGAT2 surpasses all algorithms in almost all problems. It is also can be seen that from the figures above that the graph of WGAT2 algorithm always in the bottom part (the minimum). Thus, assigning probability for the vertices in the set HVTs gives the algorithm searches more flexible rather than just choosing the smallest possible edge weight as usually done in greedy algorithm. The edge analysis also plays important process where this process assures that the chosen edge contributes smaller path length from the central vertex. Moreover, implement the algorithm to the same problem more than once also give better solution. This can be done because in every implementation the value of generated  $q$  may be differ. Until now, there is no research about the MPDCMST incorporated with machine learning. Incorporating machine learning techniques to adjust constraints based on historical data may enhance the robustness of solutions.

#### 5. ACKNOWLEDGMENT

This project is being funded by a research grant from the Directorate of Higher Education of the Republic of Indonesia, under contract number. 057/E5/PG.02.00.PL/2024, and subcontract no. 574/UN26.21/PN/2024. Kindly accept our sincere gratitude for your support.

#### REFERENCES

- Adasme, P. and A. D. Firoozabadi (2020). Degree-Constrained-Minimum Spanning Tree Problem. *Complexity*, **2020**(1); 1–25
- Al Etaiwi, W. M. (2014). Encryption Algorithm Using Graph Theory. *Journal of Scientific Research and Reports*, **3**(19); 2519–2527
- Boruvka, O. (1926). O Jistém Problému Minimálním. *Prace Moravske Prirodovedecke Spolecnosti*, **3**(3); 37–58
- Brandes, U. and Cornelsen (2009). Phylogenetic Graph Models Beyond Trees. *Discrete Applied Mathematics*, **157**(10); 2361–2369
- Caccetta, L. and S. P. Hill (2001). A Branch and Cut Method for the Degree Constrained Minimum Spanning Tree Problem. *Networks*, **37**; 74–83
- Caccetta, L. and Wamiliana (2001). Heuristics Algorithms for the Degree Constrained Minimum Spanning Tree Problems. In F. Ghassemi, D. Post, M. Sivapalan, and R. Vertessy, editors, *Proceeding of the International Congress on Modelling and Simulation (MODSIM)*. Canberra, pages 2161–2166
- Deo, N. and N. Kumar (1997). Computation of Constrained Spanning Trees: A Unified Approach. In P. M. Pardalos, D. W. Hearn, and W. W. Hager, editors, *Network Optimization*, Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, pages 194–220
- Hsu, L. H. and C. K. Lin (2009). *Graph Theory and Interconnection Network*. Taylor and Francis Group, LLC, New York
- Huson, D. H. and D. Bryant (2006). Application of Phylogenetic Networks in Evolutionary Studies. *Molecular Biology and Evolution*, **23**(2); 254–267
- Junaidi, S., Wamiliana, D. Sakethi, and E. T. Baskoro (2008). Computational Aspects of Greedy Algorithm for Solving the Multi Period Degree Constrained Minimum Spanning Tree Problem. *Jurnal Sains MIPA*, **14**(1); 1–6
- Kannimuthu, S., D. Bhanu, and K. S. Bhuvaneshwari (2020). A Novel Approach for Agricultural Decision Making Using Graph Coloring. *SN Applied Sciences*, **2**(1); 31
- Kawakura, S. and R. Shibasaki (2018). Grouping Method Using Graph Theory for Agricultural Workers Engaging in Manual Tasks. *Journal of Advanced Agricultural Technologies*, **5**(3); 173–181
- Kawatra, R. (2002). A Multi Period Degree Constrained Minimum Spanning Tree Problem. *European Journal of Operational Research*, **143**(1); 53–63
- 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
- Kruskal, J. B. (1956). On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceeding of the American Mathematical Society*, **7**(1); 48–50
- Mathur, R. and N. Adlakha (2016). A Graph Theoretic Model for Prediction of Reticulation Events and Phylogenetic Networks for DNA Sequences. *Egyptian Journal of Basic and Applied Sciences*, **3**(3); 263–271
- Ni, B., R. Qazi, S. U. Rehman, and G. Farid (2021). Some Graph-Based Encryption Schemes. *Journal of Mathematics*, **2021**(1); 6614172
- Prim, R. C. (1957). Shortest Connection Networks and Some Generalizations. *The Bell System Technical Journal*, **36**(6); 1389–1401
- Priyadarsini, P. L. K. (2015). A Survey on Some Applications of Graph Theory in Cryptography. *Journal of Discrete Mathematical Sciences and Cryptography*, **18**(3); 209–217
- Thadani, S., B. Seema, and S. Anand (2022). Applications of Graph Coloring in Various Fields. *Materials Today: Proceedings*, **66**(Part 8); 3498–3501
- Thevenin, S., Z. Nicolas, and P. Jean-Yves (2018). Graph Multi-Coloring for a Job Scheduling Application. *Discrete Applied Mathematics*, **234**; 218–235
- Thiessen, M., L. Quesada, and K. N. Brown (2020). Improving a Branch-and-Bound Approach for the Degree-Constrained Minimum Spanning Tree Problem with LKH. In E. Hebrard and N. Musliu, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research. CPAIOR 2020*,

- Lecture Notes in Computer Science*, volume 12296. Springer International Publishing, Cham, pages 447–456
- Vasudev, C. (2006). *Graph Theory with Applications*. New Age International
- 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
- Wamiliana, M. Usman, F. Elfaki, and M. Azram (2015a). Some Greedy Based Algorithms for Multi Periods Degree Constrained Minimum Spanning Tree Problem. *ARPN Journal of Engineering and Applied Sciences*, **10**(21); 10147–10152
- Wamiliana, M. Usman, D. Sakethi, and A. Cucus (2015b). The Hybrid of Depth First Search Technique and Kruskal's Algorithm for Solving the Multiperiod Degree Constrained Minimum Spanning Tree Problem. In *IEEE Proceeding on 4th International Conference on Interactive Digital Media (ICIDM)*. pages 1–4
- Wamiliana, M. Usman, W. Warsito, W. Warsono, and J. I. Daoud (2020). Using Modification of Prim's Algorithm and GNU Octave to Solve the Multiperiods Installation Problem. *IJUM Engineering Journal*, **21**(1); 100–112
- Zhou, G. and M. Gen (1997). A Note on Genetics Algorithms for Degree Constrained Spanning Tree Problems. *Network*, **30**; 91–95