Agustina Pradjaningsih

Pipeline Network Optimization Using MST Algorithms in Argopuro Housing-Jember_Agustina et al

Pipeline Network Optimization Using Minimum Spanning Tree Algorithms in Argopuro Housing-Jember

Document Details

Submission ID
trn:oid:::3618:127163709

Submission Date
Jan 30, 2026, 10:51 AM GMT+7

Download Date
Jan 30, 2026, 10:56 AM GMT+7

File Name
Pipeline Network Optimization Using MST Algorithms in Argopuro Housing-Jember_Agustina et al.pdf

File Size
1.0 MB

12 Pages

5,377 Words

31,026 Characters

19% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

64 Not Cited or Quoted 19%
Matches with neither in-text citation nor quotation marks

3 Missing Quotations 1%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

17% Internet sources
11% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **64** Not Cited or Quoted 19%
  Matches with neither in-text citation nor quotation marks
- **3** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

17% Internet sources
11% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	ghost2.altcademy.com 3%
2	Internet	jurnal.uinsu.ac.id 3%
3	Internet	algocademy.com 2%
4	Internet	repository.usd.ac.id 1%
5	Internet	www.geeksforgeeks.org <1%
6	Publication	Bruce Boldon, Narsingh Deo, Nishit Kumar. "Minimum-weight degree-constraine... <1%
7	Internet	iapress.org <1%
8	Publication	Ahmed, Imtiaz. "Unsupervised Anomaly Detection of High Dimensional Data with... <1%
9	Internet	devel.isa-afp.org <1%
10	Internet	www.simplilearn.com <1%

11	Publication	Sulabh Bansal, Aprna Tripathi, Shilpa Srivastava, Prem Prakash Vuppuluri. "Natur... <1%
12	Publication	"Business and Consumer Analytics: New Ideas", Springer Science and Business M... <1%
13	Internet	c.coek.info <1%
14	Internet	dtale.readthedocs.io <1%
15	Internet	www.tutorialspoint.com <1%
16	Internet	cahaya-ic.com <1%
17	Internet	dokumen.pub <1%
18	Publication	A Arkundato, M Hasan, E Purwandari, A Pramutadi, F Aziz. "Temperature depend... <1%
19	Publication	Lovro Šubelj. "Computing Well-Balanced Spanning Trees of Unweighted Network... <1%
20	Internet	ijece.iaescore.com <1%
21	Publication	Siti Hariati Hastuti, Alissa Chintyana, Hanipar Mahyulis Sastriana. "Evaluating Ker... <1%
22	Publication	Clyde Monma, Michael Paterson, Subhash Suri, Frances Yao. "Computing euclidea... <1%
23	Internet	core.ac.uk <1%
24	Internet	dmj.one <1%

25 Internet
ejournal.isha.or.id <1%

26 Internet
ijeecs.iaescore.com <1%

27 Publication
Rene Markovič, Marko Gosak, Vladimir Grubelnik, Marko Marhl, Peter Virtič. "Dat... <1%

28 Internet
climbtheladder.com <1%

29 Internet
jatit.org <1%

30 Internet
livecodebench-code-generation-samples.hf.space <1%

31 Internet
plj.ac.id <1%

32 Internet
propulsiontechjournal.com <1%

33 Internet
tau.edu.ng <1%

34 Internet
www.chenq.ecei.tohoku.ac.jp <1%

35 Internet
www.mdpi.com <1%

36 Internet
www.mycplus.com <1%

37 Publication
Hou Xiaolan, Wu Muqing, Xu Weiyao. "A Controller Placement Algorithm Based o... <1%

38 Publication
Lecture Notes in Computer Science, 2015. <1%

39 Publication

Mouna Ktari, Raïda Ktari, Yassine Khemakhem, Ahmed Hadj Kacem. "iAgent fault-... <1%

40 Publication

Cinel, Sertaç. "Sequential and Parallel Heuristic Algorithms for the Rectilinear Stei... <1%

Zero : Jurnal Sains, Matematika, dan Terapan
E-ISSN : 2580-5754; P-ISSN : 2580-569X
Volume 9, Number 2, 2025
DOI: 10.30829/zero.v9i2.25
Page:

Pipeline Network Optimization Using Minimum Spanning Tree Algorithms in Argopuro Housing-Jember

[1] **Agustina Pradjaningsih** Please include the active ORCID iD link of each author
Mathematics Department, FMIPA, Universitas Jember, Jember, Indonesia
[2] **Audy Putri Lestari**
Mathematics Department, FMIPA, Universitas Jember, Jember, Indonesia
[3] **Firda Fadri**
Mathematics Department, FMIPA, Universitas Jember, Jember, Indonesia
[4] **Apriani Soepardi**
Industrial Engineering Department, Universitas Pembangunan Nasional Veteran Yogyakarta, Yogyakarta, Indonesia

Article Info

Article history:

Received mm dd, yyyy
Revised mm dd, yyyy
Accepted mm dd, yyyy

Keywords:

Minimum Spanning Tree (MST);
Prim's Algorithm;
Solin's Algorithm;
Kruskal's Algorithm;
Borůvka's Algorithm

ABSTRACT

Infrastructure development required high investment; therefore, cost efficiency was essential to support sustainable development. This study aimed to optimize the clean water pipeline network in Argopuro Housing, Jember, managed by PERUMDAM Tirta Pandalungan, using Minimum Spanning Tree (MST) algorithms. The pipeline network was modeled as a weighted graph, where nodes represented junctions and edges represented pipe segments with length weights. Four MST algorithms—Prim's, Kruskal's, Sollin's, and Borůvka's—were implemented to determine the optimal network configuration and to compare computational performance. The results showed that all algorithms produced the same optimal pipeline length of 31.000163 km, achieving a 40.24% reduction compared to the existing network. However, execution times differed, with Borůvka's algorithm demonstrating the fastest performance. These findings indicated that MST-based optimization effectively reduced infrastructure costs and could support efficient planning of water distribution networks in similar urban areas.

Corresponding Author:

Agustina Pradjaningsih,
Mathematics Department
Universitas Jember, Jember, Indonesia
Email: agustina.fmipa@unej.ac.id

1. INTRODUCTION

Infrastructure network planning plays a crucial role in supporting sustainable development, particularly in essential public services such as electricity distribution, telecommunications, transportation, and clean water supply. Inefficient network design can lead to excessive construction and maintenance costs, which reduce long-term system sustainability. Consequently, mathematical optimization approaches have been widely applied to support cost-efficient infrastructure planning by identifying optimal network configurations that ensure full connectivity with minimal total cost [1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11].

One of the most widely used mathematical models for network optimization is the Minimum Spanning Tree (MST), which aims to connect all nodes in a network with the minimum possible total edge weight. Infrastructure networks can be modeled as graphs, where nodes represent connection points and edges represent physical links between nodes. In pipeline network applications, the graph is typically undirected and weighted, with edge weights corresponding to pipe lengths or installation costs [12],[13],[14],[15],[16],[17],[18],[19],[20]. Due to its simplicity and optimality guarantees, MST has been extensively applied in water distribution networks, transportation planning, and utility infrastructure optimization.

Several algorithms have been developed to solve MST problems, including Prim's, Kruskal's, Sollin's, and Borůvka's algorithms [21],[22],[23],[24],[25],[26],[27],[28],[29]. Although these algorithms theoretically produce the same optimal spanning tree for a given weighted graph, they differ in computational mechanisms, execution time, and implementation complexity. Recent studies have emphasized the importance of evaluating not only the optimality of the resulting network but also the computational performance of different MST algorithms when applied to real-world infrastructure systems. However, many existing studies focus on simulated or abstract networks, while comparative implementations on real urban pipeline networks remain limited, particularly in developing-country contexts.

In Indonesia, clean water distribution systems are managed by regional water utilities (Perusahaan Daerah Air Minum/PDAM), where pipeline network efficiency directly affects service reliability and operational costs. The clean water pipeline network in Sumbersari Subdistrict, Jember, managed by PERUMDAM Tirta Pandalungan, is known to contain cycles, indicating that the existing network configuration is not optimal in terms of total pipe length. Despite this condition, no prior study has systematically optimized this network using MST-based methods while simultaneously comparing the computational performance of multiple MST algorithms. This gap highlights the need for applied research that bridges mathematical optimization theory and practical infrastructure planning.

The novelty of this study lies in the comparative application of four MST algorithms—Prim's, Kruskal's, Sollin's, and Borůvka's—on a real-world clean water pipeline network, with an emphasis on both optimization effectiveness and computational efficiency. This research provides practical insights into algorithm selection for infrastructure planning using readily implementable tools.

Therefore, this study aims to optimize the clean water pipeline network in Argopuro Housing, Sumbersari District, Jember, using Minimum Spanning Tree algorithms. Specifically, this research compares the performance of Prim's, Kruskal's, Sollin's, and Borůvka's algorithms in terms of optimal pipeline length and execution time to support efficient, practical decision-making in water distribution network planning.

2. RESEARCH METHOD

2.1 Research Design

This study employed a quantitative, computational research design, which was well-suited for analyzing and optimizing infrastructure networks using graph-based mathematical algorithms. The research focused on applying Minimum Spanning Tree (MST) algorithms to determine the most cost-efficient configuration of a clean water pipeline network. A computational approach was adopted to evaluate algorithmic performance in terms of optimal network length and execution time under identical data conditions. This design enabled objective comparison and ensured reproducibility of the results.

2.2 Data Source and Variables

The data used in this study were secondary data obtained from PERUMDAM Tirta Pandalungan Jember, specifically the clean water pipeline network for Argopuro Housing in Sumbersari District. The dataset consisted of information on pipeline junctions and pipe connections forming the distribution network.

In this study, the pipeline network was modeled as a weighted undirected graph $G = (V, E)$, where: V represented the set of nodes corresponding to pipeline junctions or intersection points, and E represented the set of edges corresponding to pipeline segments connecting two nodes [16],[18],[19],[20].

The weight associated with each edge was the physical length of the pipe segment, measured in kilometers (km). The primary variable of interest was the total pipeline length, while the secondary variable was the algorithm execution time (in seconds).

2.3 Analytical Methods and Algorithms

The analytical framework of this study was based on graph theory and Minimum Spanning Tree optimization. The initial pipeline network was first constructed as a weighted undirected graph. Subsequently, four MST algorithms—Prim's, Kruskal's, Sollin's, and Borůvka's algorithms—were applied to the same graph to ensure consistency in comparison.

Prim's algorithm was implemented by iteratively adding the minimum-weight edge connected to the growing tree without creating cycles. According to [24],[26],[29],[30], Prim's algorithm finds the MST by connecting nodes

by considering the distance with minimum weight, which will produce a single tree. Prim's algorithm begins by initiating a graph $G = (V, E)$ where the set V of nodes and the set E of edges are $\emptyset$ and continues:

(i) Select an initial vertex and insert it into the graph V.
(ii) The minimum weighted edge (u, v) with the closest distance to the starting nodes is selected.
(iii) Select another minimum weight edge (u', v') where $u \in V$ and $v \notin V$ without causing the tree to cycle.
(iv) The above steps are repeated until $|E| = n - 1$ satisfied, where n is the number of nodes.

Kruskal's algorithm was executed by sorting all edges in ascending order and selecting the smallest edge that did not form a cycle. Kruskal's algorithm works by adding the edge with the smallest weight. The MST generated by Kruskal will not have a cycle because the added edge is an edge that does not result in the formation of a cycle in the tree [23],[26],[29]. In a graph $G = (V, E)$ the edges in the formation of MST with Kruskal's algorithm are sorted by their weights, and $w(e_1) \leq w(e_2) \leq \cdots \leq (w(e_m)$ then proceed as follows:

(i) $\forall v \in V$ a separate set will be created $\forall v \in V, C(v) = \{v\}$.
(ii) Select the edge with the smallest weight that does not cause a cycle $C(u) \neq C(v)$, then add the edge to the MST.
(iii) Components are merged until all nodes are connected in the MST.
(iv) Repeat $|E| = n - 1$ until it is satisfied.

Sollin's algorithm was applied by identifying and removing the largest-weight edges that formed cycles while maintaining network connectivity. . Based on the explanation from [28], Sollin's algorithm will produce an MST with no cycle so that the resulting tree has the smallest weight. This algorithm runs by removing the most significant edge, the existence of which results in the formation of a cycle. The edges in a graph $G(V, E)$ are sorted by their weights $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_m)$ and proceed with the following algorithm:

(i) The sorted edges will be checked one by one from the largest to see whether they have a cycle.
(ii) If the edge is detected to have a cycle, then the edge will be deleted, provided that the deleted edge does not cause the nodes in the tree to become unconnected. Meanwhile, if the edge does not have a cycle, then the edge will be left and included in the MST.
(iii) The steps are repeated until the total edges in the MST meets $|E| = n - 1$.

Borůvka's algorithm was executed by initially treating each node as an independent component and repeatedly merging components by selecting the minimum outgoing edge from each node. According to [21],[23],[24],[25],[26] the explanation of Boruvka's algorithm works in an iterative way, where each node will choose the closest node connected by the edge with the smallest weight, so that a tree will be formed. The trees will be merged into the set of MST(T), then:

(i) $\forall$ node $u \in V$, the node selects the edge (u, v) with the smallest weight to connect (u) to another node (v).
(ii) The selected edges are added to the set T.
(iii) Merge the vertices (u) and (v) connected by edges T into one main tree.
(iv) Repeat the process until there is only one tree that includes all the vertices.

For each algorithm, two performance indicators were evaluated: (1) the total length of the resulting MST, and (2) the execution time required to obtain the solution. All algorithms were applied under identical conditions to ensure objective comparison.

2.4 Software and Tools

All computations and graph constructions were performed using the Python programming language. Python was selected for its flexibility, computational efficiency, and the availability of graph-processing libraries. The pipeline network graph was constructed programmatically, and each MST algorithm was implemented using standard algorithmic procedures. Execution time was recorded automatically during the computational process to evaluate algorithm efficiency. The following Python pseudocode describes the fourth approach to the algorithm [30].

```
# Prim's Algorithm
def prim(G, start_node):
    E = []
    V = set([start_node])
    edges = [(weight, start_node, v) for v, weight in G[start_node].items()]
    heapq.heapify(edges)
    while edges and len(V) < len(G):
        weight, u, v = heapq.heappop(edges)
        if v not in V:
            V.add(v)
            E.append((u, v, weight))
            for next_v, next_weight in G[v].items():
```

```
                if next_v not in V:
                    heapq.heappush(edges, (next_weight, v, next_v))
    return E
```

```
class DisjointSet: # Disjoint Set (Union-Find) - Kruskal
    def __init__(self, vertices):
        self.parent = {v: v for v in vertices}
        self.rank = {v: 0 for v in vertices}
    def find(self, v):
        if self.parent[v] != v:
            self.parent[v] = self.find(self.parent[v])
        return self.parent[v]
    def union(self, u, v):
        root_u = self.find(u)
        root_v = self.find(v)
        if root_u == root_v:
            return False
        if self.rank[root_u] < self.rank[root_v]:
            self.parent[root_u] = root_v
        elif self.rank[root_u] > self.rank[root_v]:
            self.parent[root_v] = root_u
        else:
            self.parent[root_v] = root_u
            self.rank[root_u] += 1
        return True

def kruskal(G): # Kruskal's Algorithm
    edges = []
    for u in G:
        for v, weight in G[u].items():
            if u < v:
                edges.append((weight, u, v)), edges.sort()
    ds = DisjointSet(G.keys())
    mst = []
    for weight, u, v in edges:
        if ds.union(u, v): mst.append((u, v, weight))
        if len(mst) == len(G) - 1:
            break
    return mst
```

```
# Sollin's Algorithm
def sollin(G):
    edges = sorted(G.edges(data=True), key=lambda x: x[2]
     ['weight'], reverse=True)
    removed_edges = []
    remaining_edges = list(G.edges(data=True))
    for edge in edges:
        u, v, w = edge
        def is_connected(G):
            return nx.is_connected(G)
        G.remove_edge(u, v)
        if not is_connected(G):
            G.add_edge(u, v, weight=w['weight'])
        else:
            removed_edges.append(edge)
            remaining_edges.remove(edge)
    return G, removed_edges, remaining_edges
```

```
class DisjointSet: # Disjoint Set (Union-Find) - Borůvka
    def __init__(self, vertices):
        self.parent = {v: v for v in vertices}
    def find(self, v):
        if self.parent[v] != v:
```

```
            self.parent[v] = self.find(self.parent[v])
        return self.parent[v]
    def union(self, u, v):
        root_u = self.find(u)
        root_v = self.find(v)
        if root_u != root_v:
            self.parent[root_v] = root_u
            return True
        return False
def boruvka(G): # Boruvka's Algorithm
    ds = DisjointSet(G.keys())
    num_components = len(G)
    mst = []
    while num_components > 1:
        cheapest = {}
        for u in G:
            for v, weight in G[u].items():
                set_u = ds.find(u)
                set_v = ds.find(v)
                if set_u != set_v:
                    if set_u not in cheapest or cheapest[set_u][0] > weight:
                        cheapest[set_u] = (weight, u, v)
                    if set_v not in cheapest or cheapest[set_v][0] > weight:
                        cheapest[set_v] = (weight, v, u)
        for weight, u, v in cheapest.values():
            if ds.union(u, v):
                mst.append((u, v, weight))
                num_components -= 1
    return mst
```

3. RESULT AND ANALYSIS

3.1 Characteristics of the Pipeline Network Data

The clean water pipeline network analyzed in this study was obtained from PERUMDAM Tirta Pandalungan Jember and represented the actual distribution system in Argopuro Housing, Sumbersari District. The network consisted of 505 nodes representing pipe intersections and 660 edges representing pipeline segments, as illustrated in Figure 1. Each edge was assigned a weight corresponding to the physical length of the pipe segment in kilometers, as summarized in Table 1.

Table 1. Pipeline Network Data

Edge	TA	Longitude_TA	Latitude_TA	TT	Longitude_TT	Latitude_TT	Weight
e1	v1	113.704607	-8.175343819	v2	113.7035557	-8.17527291	0.11979
e2	v2	113.7035557	-8.17527291	v5	113.7035136	-8.176235779	0.10435
e3	v1	113.704607	-8.175343819	v6	113.7045741	-8.17621289	0.09798
e4	v5	113.7035136	-8.176235779	v6	113.7045741	-8.17621289	0.11448
e5	v2	113.7035557	-8.17527291	v3	113.7027894	-8.175332624	0.08388
...	...	...	...	...	...	...	...
e656	v291	113.7042307	-8.189968641	v293	113.7039175	-8.189940214	0.03301
e657	v293	113.7039175	-8.189940214	v295	113.704148	-8.189220363	0.08235
e658	v292	113.7036793	-8.189894754	v293	113.7039175	-8.189940214	0.02172
e659	v292	113.7036793	-8.189894754	v294	113.7039031	-8.189152714	0.08484
e660	v271	113.7018405	-8.190800761	v275	113.7070947	-8.191336863	0.58158

Figure 1. Pipeline Network

3.2 Initial Graph/Network Analysis

The initial graph/network was generated in Python by mapping the pipeline data to a weighted, undirected graph. As shown in **Figure 2**, the network exhibited dense interconnections, particularly within residential clusters, resulting in overlapping paths and multiple cycles.

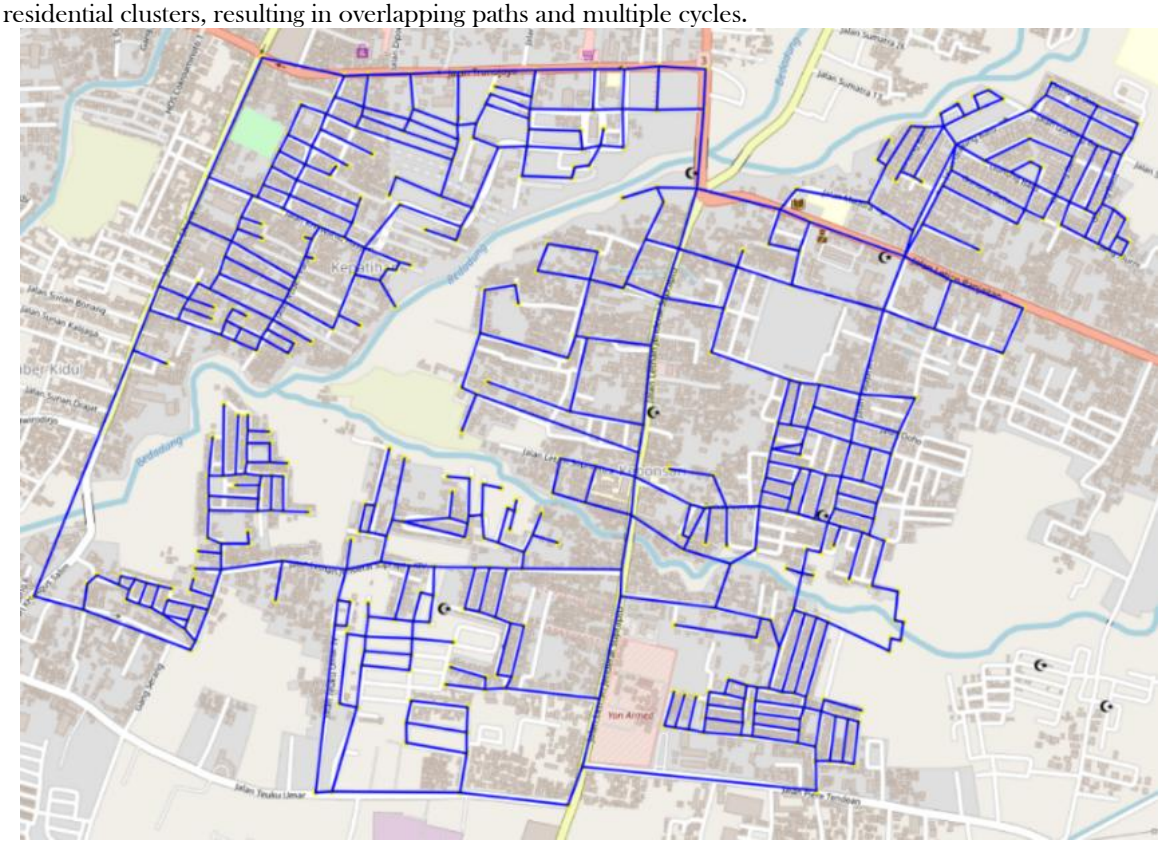

Figure 2. Initial Graph

The presence of these cycles indicated redundancy in pipeline connections, which increased the total pipeline length without improving network coverage. The total length of the initial pipeline network was 51.876 km, reflecting a non-optimal configuration. From an optimization perspective, this condition justified the application of Minimum Spanning Tree (MST) algorithms, which aim to eliminate unnecessary cycles while preserving full network connectivity. Therefore, the initial network length served as a baseline for evaluating the effectiveness of each MST algorithm in optimizing the pipeline network.

3.3 Results of MST Algorithms

3.3.1 Prim's Algorithm

The application of Prim's algorithm yielded a minimum spanning tree (MST), as illustrated in Figure 3, representing the optimal clean water pipeline network. This algorithm employed a vertex-based greedy expansion strategy, selecting edges with minimum weight iteratively to connect unvisited nodes without forming cycles. The resulting network achieved a substantial reduction in total pipeline length compared to the initial configuration, while preserving full connectivity among all pipeline junctions. In addition to its optimality in network structure, Prim's algorithm demonstrated high computational efficiency, as evidenced by its relatively short execution time. These results indicated that Prim's algorithm effectively eliminated redundant connections and provided a reliable and efficient solution for pipeline network optimization.

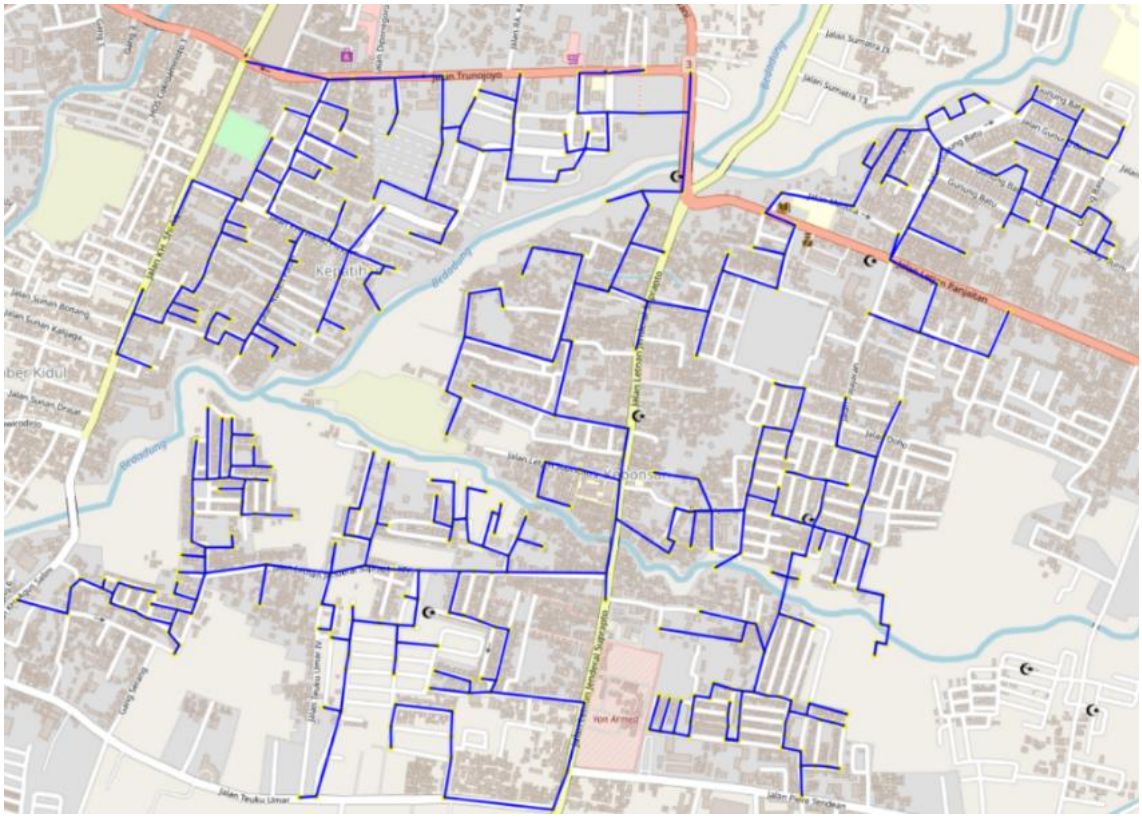

Figure 3. MST result using Prim's algorithm: Minimum spanning tree of the clean water pipeline network obtained using Prim's algorithm, illustrating the optimal network topology produced through a vertex-based greedy expansion process.

3.3.2 Kruskal's Algorithm

The application of Kruskal's algorithm produced a minimum spanning tree (MST) configuration, as illustrated in Figure 4, by selecting edges in ascending order of weight while avoiding the formation of cycles. The resulting pipeline network achieved the same optimal total length as that obtained with Prim's algorithm, thereby confirming the theoretical property that different MST algorithms yield identical optimal solutions when applied to a weighted undirected graph. Although Kruskal's algorithm took slightly longer than Prim's, it remained computationally efficient and suitable for large-scale pipeline networks. These results indicated that Kruskal's algorithm provided a reliable optimization outcome through an edge-based selection mechanism, while maintaining full network connectivity and eliminating redundant pipeline segments.

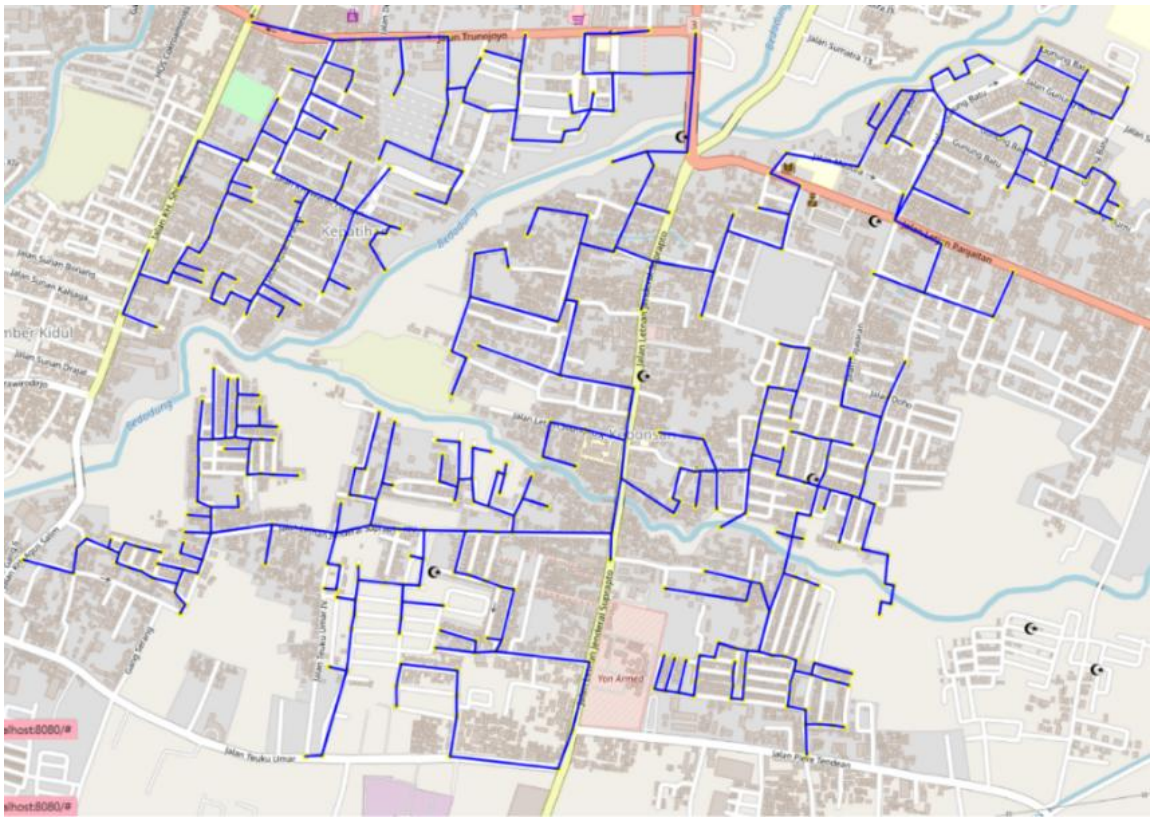

Figure 4. MST result using Kruskal's algorithm: Minimum spanning tree of the clean water pipeline network generated using Kruskal's algorithm, illustrating the optimal network topology obtained through an edge-sorting and cycle-avoidance process.

3.3.3 Sollin's Algorithm

The application of Sollin's algorithm yielded a minimum spanning tree (MST), as illustrated in Figure 5, by iteratively eliminating edges that formed cycles while preserving overall network connectivity. The resulting pipeline network achieved the same optimal total length as those obtained using the other MST algorithms, thereby confirming theoretical equivalence in terms of solution quality. However, compared to other approaches, Sollin's algorithm exhibited substantially longer execution time on the analyzed dataset. This finding indicated that although the algorithm was effective at producing an optimal spanning tree, its cycle-elimination mechanism reduced computational efficiency, potentially limiting its practical applicability for large-scale pipeline network optimization.

3.3.4 Borůvka's Algorithm

The application of Borůvka's algorithm produced a minimum spanning tree (MST) configuration, as illustrated in Figure 6, through a component-based merging process in which each connected component iteratively selected its minimum outgoing edge. The resulting pipeline network achieved the same optimal total length as those obtained using the other MST algorithms, confirming consistency in solution quality across all approaches. In terms of computational performance, Borůvka's algorithm demonstrated relatively high efficiency, with an execution time faster than Kruskal's and Sollin's algorithms but slightly slower than Prim's algorithm for the analyzed dataset. These results indicated that Borůvka's algorithm provided a balanced trade-off between computational efficiency and algorithmic robustness for pipeline network optimization.

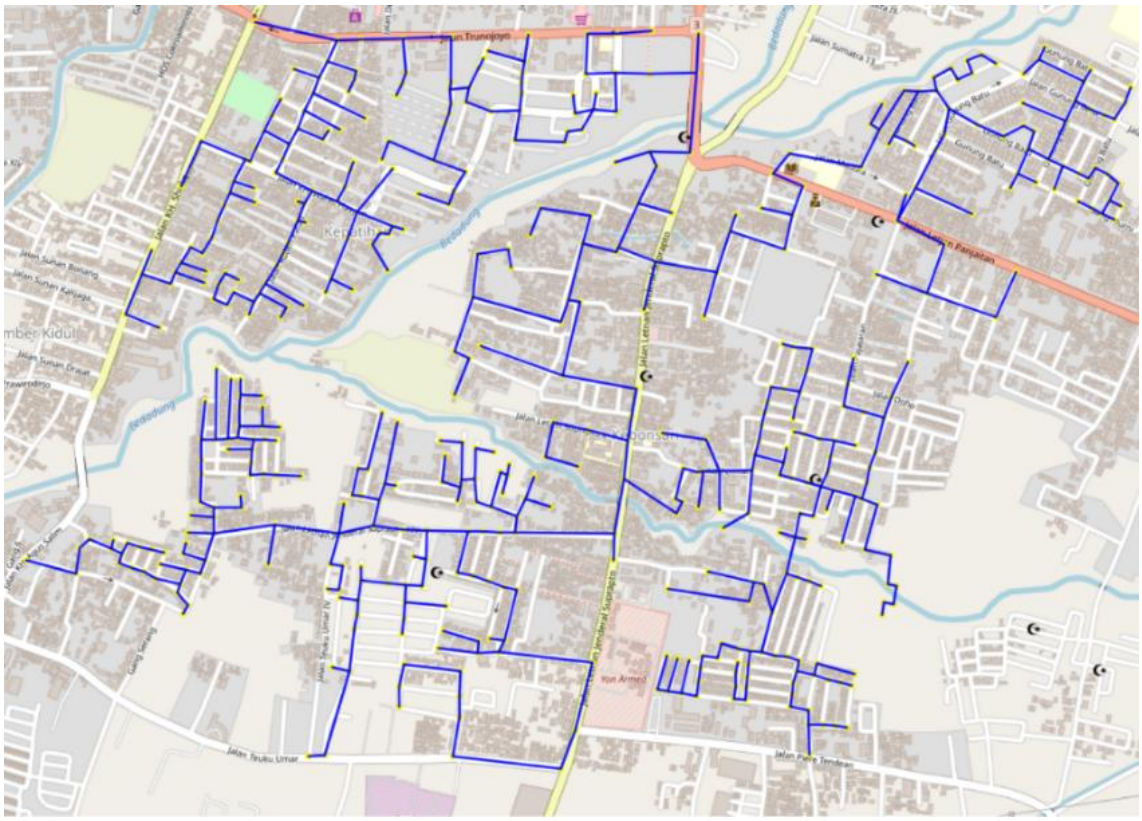

Figure 5. MST result using Sollin's algorithm: Minimum spanning tree of the clean water pipeline network generated using Sollin's algorithm, illustrating the optimal network topology obtained through an iterative cycle-elimination process.

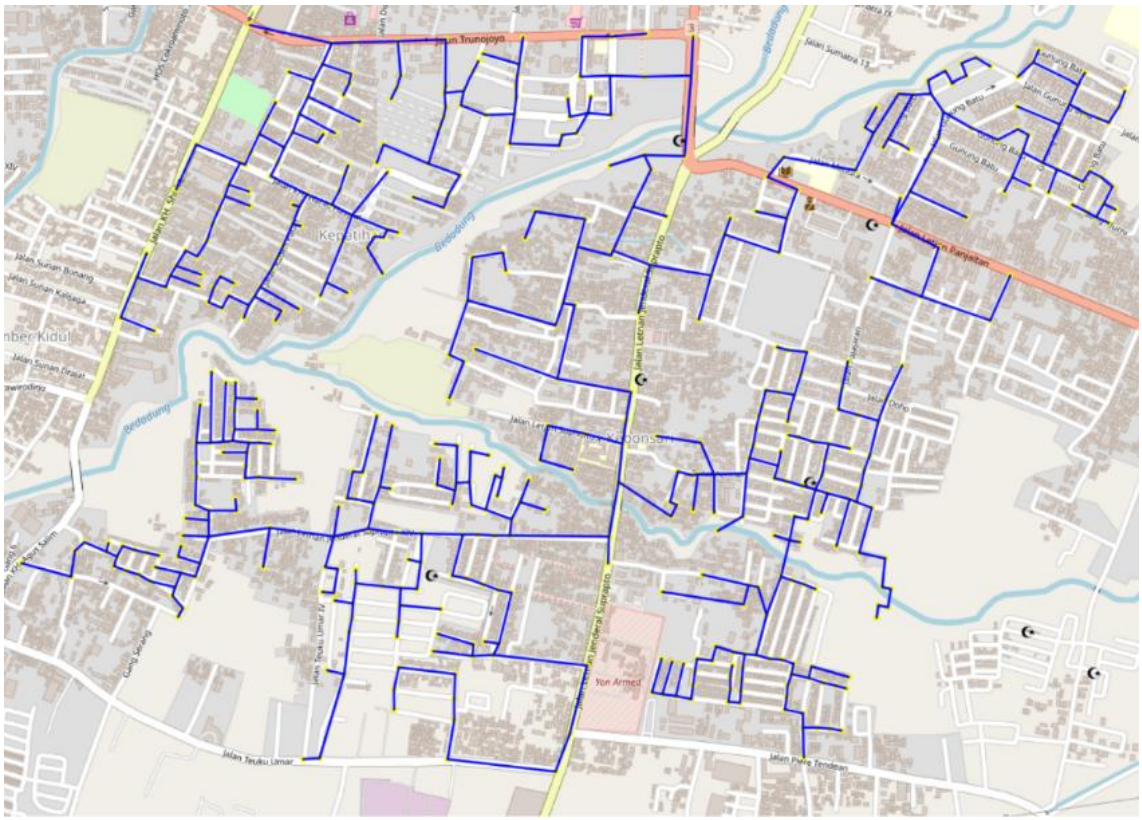

Figure 6. MST result using Borůvka's algorithm: Minimum spanning tree of the clean water pipeline network generated using Borůvka's algorithm, illustrating the convergence of component-based merging into a single optimal spanning tree.

E-ISSN : 2580-5754; P-ISSN : 2580-569X

3.4 Comparative Performance Analysis

A summary comparison of the four MST algorithms is presented in **Table 2**. All algorithms achieved the same optimal pipeline length, resulting in a **40.24% reduction** compared to the initial network. This finding confirmed that MST-based optimization was highly effective in minimizing infrastructure length without sacrificing connectivity.

Table 2. Comparison of Prim's, Sollin's, Kruskal's, and Boruvka's algorithms

Algorithm	Initial Length (km)	Length After MST (km)	Time (seconds)
Prim's algorithm	51,87616673318702	31,0001634833572	0,0008
Kruskal's algorithm			0,0012
Sollin's algorithm			0,2476
Boruvka's algorithm			0,0067

Despite identical optimization outcomes, the algorithms exhibited notable differences in execution time. Prim's algorithm demonstrated the fastest performance, followed by Kruskal's and Borůvka's algorithms, while Sollin's algorithm required substantially longer computation time. These differences can be attributed to variations in algorithmic mechanisms, particularly in edge sorting and cycle detection processes.

From a practical perspective, computational efficiency is an essential consideration for large-scale or repeated optimization tasks. Therefore, although all algorithms were mathematically equivalent in terms of solution quality, Prim's algorithm appeared to be the most suitable for practical implementation in this case.

3.5 Discussion and Implications

The results demonstrated that applying Minimum Spanning Tree (MST) algorithms to a real-world clean water pipeline network effectively reduced the total pipeline length while maintaining full network connectivity. This outcome was consistent with graph theory principles and previous infrastructure optimization studies, reinforcing the theoretical guarantee that MST algorithms yield a unique optimal solution for weighted undirected graphs. As illustrated in Figures 3–6, all algorithms produced identical network topologies in terms of node connectivity and edge selection. This visual consistency did not indicate redundancy but instead provided empirical validation of the correctness and robustness of each algorithmic implementation.

Despite producing identical optimal network configurations, the algorithms exhibited notable differences in computational performance. These variations highlighted the importance of algorithm selection in applied settings, particularly for large-scale infrastructure networks. While Sollin's algorithm was conceptually straightforward, its longer execution time limited its practical applicability. In contrast, Prim's and Borůvka's algorithms offered a more favorable balance between computational efficiency and implementation simplicity, making them more suitable for operational use by local water utility agencies.

The novelty of this study lies in the comparative evaluation of multiple MST algorithms using real operational data from an urban pipeline network, rather than simulated or hypothetical datasets. Beyond mathematical validation, the findings provided practical guidance for infrastructure planners in selecting appropriate optimization techniques based on both solution quality and computational efficiency. Nevertheless, this study was limited to pipeline length as the sole optimization criterion. Future research could extend this framework by incorporating additional factors, such as installation costs, pipe diameters, and hydraulic constraints, to develop more comprehensive and realistic network optimization models.

4. CONCLUSION

This study successfully achieved its research objectives by demonstrating the effectiveness of Minimum Spanning Tree (MST) algorithms in optimizing a real-world clean water pipeline network in Argopuro Housing, Sumbersari District, Jember. By modeling the pipeline system as a weighted undirected graph and applying four MST algorithms—Prim's, Kruskal's, Sollin's, and Borůvka's—the study confirmed that MST-based optimization could eliminate redundant network cycles while maintaining full connectivity across all pipeline junctions.

The findings showed that all examined algorithms produced an identical optimal network configuration, confirming the theoretical property of MST optimality in weighted graphs. However, differences in computational efficiency were observed, indicating that algorithm selection plays a practical role in large-scale or repeated optimization tasks. In this context, Prim's algorithm demonstrated superior computational performance, making it the most suitable option for practical implementation under similar conditions.

From a scientific perspective, this research contributed to the applied understanding of graph-based optimization by providing empirical evidence from an actual urban infrastructure network rather than a simulated dataset. In practice, the results provided a quantitative basis for local water utility agencies to redesign pipeline networks in a more cost-efficient and systematic manner, using simple, replicable computational tools.

Despite these contributions, this study was limited to pipeline length as the sole optimization criterion. Future research is recommended to incorporate additional factors, such as installation cost, pipe diameter, hydraulic performance, and demand distribution, to develop more comprehensive optimization models. Furthermore, extending the analysis to other districts or integrating MST with multi-objective optimization techniques could enhance the applicability of the proposed approach.

Overall, this study demonstrated that MST algorithms provide a robust, practical framework for infrastructure network optimization and support more efficient planning and decision-making in water distribution systems.

5. REFERENCES

[1] A. Tamber, F. Ikpotokin, and L. Okafor, "The Minimum Spanning Tree of the Nigeria Roads Network through Multiple-Roads Network System," *Nigerian Annals Of Pure And Applied Sciences*, vol. 3, no. 2, pp. 151–157, Jul. 2020, https://doi.org/10.46912/napas.170

[2] N. J. Triami, Y. Yundari, and F. Fran, "Minimum Spanning Tree Pada Jaringan Fiber Optic Di Universitas Tanjungpura," *Bimaster : Buletin Ilmiah Matematika, Statistika dan Terapannya*, vol. 9, no. 1, Jan. 2020, https://doi.org/10.26418/bbimst.v9i1.38909

[3] N. Made Ayu Ulandari, A. Amrullah, J. Junaidi, and S. Subarinah, "Implementasi Algoritma Kruskal dalam Menentukan Rute Terdekat pada Tempat Pariwisata di Daerah Lombok Tengah," *Griya Journal of Mathematics Education and Application*, vol. 1, no. 4, pp. 578–589, Dec. 2021, https://doi.org/10.29303/griya.v1i4.117

[4] R. Efendi, B. Susilo, and Y. A. Prasetyo, "Perbandingan Algoritma Boruvka dan Algoritma Sollin pada Optimasi Kebutuhan Kabel Fiber Optik Universitas Bengkulu," *JSAI (Journal Scientific and Applied Informatics)*, vol. 4, no. 2, pp. 175–181, Jul. 2021, https://doi.org/10.36085/jsai.v4i2.1623

[5] A. Łukaszewski, Ł. Nogal, and M. Januszewski, "The Application of the Modified Prim's Algorithm to Restore the Power System Using Renewable Energy Sources," *Symmetry (Basel)*, vol. 14, no. 5, p. 1012, May 2022, https://doi.org/10.3390/sym14051012

[6] K. Kusnadi, W. Gata, and F. N. Arviantino, "Aplikasi Algoritma Kruskal dan Sollin Pada Jaringan Transmisi Nasional Provinsi Sulawesi Selatan," *Metik Jurnal*, vol. 6, no. 1, pp. 8–17, Jun. 2022, https://doi.org/10.47002/metik.v6i1.260

[7] I. S. Sinaga, N. Rarasati, W. Syafmen, and G. Kholijah, "MST Dalam Perencanaan Jaringan Pipa Air Minum Dengan Perbandingan Matriks Ketetanggaan Berbobot Dan Algoritma Sollin," *FIBONACCI: Jurnal Pendidikan Matematika dan Matematika*, vol. 9, no. 2, p. 179, Dec. 2023, doi: https://doi.org/10.24853/fbc.9.2.179-196

[8] S. Syahdan, A. Efendy, and T. Pornawasari, "Penerapan Algoritma Sollin Pada Jaringan Kabel Telkom Tanjung Selor Berbantu Maple," *Jurnal Sains Benuanta*, vol. 2, no. 1, pp. 1–8, Jun. 2023, https://doi.org/10.61323/jsb.v2i1.51

[9] W. I. Ilahy, M. Ahmad, and B. P. Hartono, "Optimasi Jaringan Distribusi Air di Desa Gombolharjo Menggunakan Algoritma Prim," *Journal of Mathematics Education and Science*, vol. 6, no. 2, pp. 177–183, Oct. 2023, https://doi.org/10.32665/james.v6i2.1896

[10] H. M. Saad, A. Shdefat, A. Nawaz, A. M. El-Sherbeeny, M. A. El-Meligy, and M. R. R. Rana, "Enhancing energy balance in wireless sensor networks through optimized minimum spanning tree," *PeerJ Comput Sci*, vol. 10, p. e2269, Sep. 2024, https://doi.org/10.7717/peerj-cs.2269

[11] N. W. Wisesa and R. F. Mawardiningrum, "Analysis of the Use of the Boruvka Algorithm Method in Electricity," *Int J Sci Math Educ*, vol. 1, no. 1, pp. 29–35, Mar. 2024, https://doi.org/10.62951/ijsme.v1i1.14

[12] R. L. Graham and P. Hell, "On the History of the Minimum Spanning Tree Problem," *in Annals of the History of Computing*, vol. 7, no. 1, pp. 43-57, Jan.-March 1985, https://doi.org/1109/MAHC.1985.10011

[13] A. K. Nemani and R. K. Ahuja, "Minimum Spanning Trees," *Encyclopedia of Operations Research and Management Science* Feb. 2011, https://doi.org/10.1002/9780470400531.EORMS0816

[14] S. Pettie, "Minimum Spanning Trees", *Encyclopedia of Algorithms* (Springer), pp. 1322-1325, 2016. https://doi.org/10.1007/978-1-4939-2864-4_239

[15] H. A. Taha, *Operations Research: An Introduction* (10th ed.), 2017. Pearson. https://elibrary.pearson.de/book/99.150005/9781292165561

[16] H. R. Lewis, "17 On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem (1956)," in Ideas That Created the Future: *Classic Papers of Computer Science* , MIT Press, 2021, pp.179-182. https://doi.org/10.7551/mitpress/12274.001.0001

[17] Paryati and K. Salahddine, "The Implementation of Kruskal's Algorithm for Minimum Spanning Tree in a Graph," *MATEC Web of Conferences*, vol. 348, p. 01001, Nov. 2021, https://doi.org/10.1051/matecconf/202134801001

[18] M. Alshammari, "Maintaining Minimum Spanning Trees in Fully Dynamic Graphs," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 4, pp. 1562–1567, 2024, doi: https://ijisae.org/index.php/IJISAE/article/view/6451

12 E-ISSN : 2580-5754; P-ISSN : 2580-569X

[19] L. Subelj, "Computing Well-Balanced Spanning Trees of Unweighted Networks," *Algorithms*, vol 18, no. 2, 760, 2025. https://doi.org/10.3390/a18120760
[20] Kristopher Tapp, "On the minimum spanning tree distribution in grids," *European Journal of Combinatorics*, vol. 133, 2026, 104325. https://doi.org/10.1016/j.ejc.2025.104325
[21] J. Nešetřil, E. Milková, &Helena Nešetřilová, "Otakar Borůvka on minimum spanning tree problem Translation of both the 1926 papers, comments, history," *Discrete Mathematics*, vol. 233, no. 1–3, pp 3-36, 2001. https://doi.org/10.1016/S0012-365X(00)00224-7
[22] N. R. Latha, G. Shyamala and G. R. Prasad, "Exploring the parallel implementations of the three classical MST algorithms," *International Conference on Inventive Communication and Computational Technologies (ICICCT)*, Coimbatore, India, 2017, pp. 340-346, https://doi.org/10.1109/ICICCT.2017.7975216
[23] D. Rachmawati, Herriyance, F.Y.P. Pakpahan, "Comparative Analysis of the Kruskal and Boruvka Algorithms in Solving Minimum Spanning Tree on Complete Graph," *International Conference on Data Science, Artificial Intelligence, and Business Analytics, DATABIA 2020, Proceedings, art. no. 9190504,* pp. 55 - 62, 2020. https://doi.org/10.1109/DATABIA50434.2020.9190504
[24] F. Marpaung and Arnita, "Comparative of prim's and boruvka's algorithm to solve minimum spanning tree problems," *J Phys Conf Ser*, vol. 1462, no. 1, p. 12043, Feb. 2020, https://doi.org/10.1088/1742-6596/1462/1/012043
[25] N. Robinson-O'Brien, "A formal correctness proof of Boruvka's minimum spanning tree algorithm.," Jan. 2020, https://doi.org/10.26021/10196
[26] J. Chen, "The analysis and application of Prim algorithm, Kruskal algorithm, Boruvka algorithm," *Applied and Computational Engineering*, vol. 19, no. 1, pp. 84–89, Oct. 2023, https://doi.org/10.54254/2755-2721/19/20231012
[27] R. Zhang, "The comparison of three MST algorithms," *Applied and Computational Engineering*, Oct. 2023, https://doi.org/10.54254/2755-2721/17/20230939
[28] W. Wamiliana, R. P. Sari, A. Reformasari, J. Suparman, and A. Junaidi, "Solving the Shortest Total Path Length Spanning Tree Problem Using the Modified Sollin and Modified Dijkstra Algorithms," *Science and Technology Indonesia*, vol. 8, no. 4, pp. 684–690, Oct. 2023, https://doi.org/10.26554/sti.2023.8.4.684-690
[29] S. Gupta, A. Mahajan, S. Shah, and V. Negi, "Exploring the Maze: A Comparative Study of Prims and Kruskals MST Algorithms," *Proc. IEEE ICCCNT* pp. 1–5, Jun. 2024, https://doi.org/10.1109/icccnt61001.2024.10726020
[30] W. McKinney, "*Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter,"* 3rd Edition, 2018. O'Reilly Media. https://wesmckinney.com/book/

ACKNOWLEDGEMENT

The authors would like to express their appreciation to PERUMDAM Tirta Pandalungan Jember for providing the clean water pipeline network data used in this study. The authors also gratefully acknowledge the institutional support provided by the Lembaga Penelitian dan Pengabdian Kepada Masyarakat (LP2M), University of Jember, through the Internal Grant scheme of the Research Group–Community Service (Keris-Dimas), which facilitated the provision of research resources and facilities necessary for the completion of this work. In addition, the authors acknowledge the academic contributions and constructive discussions from members of the Mathematical Optimization and Computations (MOCo) research group in the Department of Mathematics, whose insights and suggestions enhanced the quality of this research.