e-ISSN : 2964-5115
p-ISSN : 2964-4364

Evaluasi Algoritma Kruskal dan Prim dalam Optimalisasi Pembangunan Infrastruktur Jaringan Internet Di Daerah Terpencil

Wresti Andriani[*1], Lutfiatun Rahma Almujahidah[2], Muhammad Rio Ferdyansyah[3]
[*1,2,3]*Informatika, Universitas Muhammadiyah Tegal, Kota Tegal*
e-mail: [*1]wresty.andriani@gmail.com, [2]lutfiatunrahma135@gmail.com, [3]feryferdyansyah88@gmail.com

Abstrak

Penelitian ini membandingkan kinerja algoritma Kruskal dan Prim dalam membangun Minimum Spanning Tree (MST) pada sebuah graf berbobot. Hasil menunjukkan bahwa kedua algoritma menghasilkan MST dengan bobot total yang sama, yaitu 7.0, meskipun menggunakan pendekatan yang berbeda. Algoritma Kruskal bekerja dengan memilih edge terkecil secara global, sedangkan algoritma Prim memulai dari satu node dan menambahkan edge dengan bobot terkecil yang terhubung. Dari segi efisiensi, algoritma Kruskal lebih cocok untuk graf jarang (sparse), sementara algoritma Prim lebih optimal untuk graf padat (dense). Kompleksitas algoritma Kruskal adalah O(E log E), sedangkan algoritma Prim bergantung pada representasi graf, dengan kompleksitas O(V²) atau O(E + V log V) menggunakan heap. Penelitian ini menyimpulkan bahwa pemilihan algoritma yang tepat bergantung pada karakteristik graf, di mana Kruskal ideal untuk jaringan dengan koneksi minim, sementara Prim lebih sesuai untuk jaringan dengan konektivitas tinggi, seperti pada pengembangan infrastruktur jaringan internet di daerah terpencil.
***Kata Kunci**: Algoritma Kruskal, Algoritma Prim, Efisiensi Algoritma, Graf Berbobot, Infrastruktur Jaringan*

Abstract

This study compares the performance of Kruskal and Prim algorithms in constructing a Minimum Spanning Tree (MST) on a weighted graph. The results show that both algorithms produce an MST with the same total weight of 7.0, despite utilizing different approaches. Kruskal's algorithm operates by globally selecting the smallest edge, while Prim's algorithm starts from a single node and iteratively adds the smallest connected edge. In terms of efficiency, Kruskal is better suited for sparse graphs, while Prim is more optimal for dense graphs. The complexity of Kruskal's algorithm is O(E log E), whereas Prim's algorithm depends on the graph representation, with a complexity of O(V²) or O(E + V log V) when using a heap. This study concludes that the choice of the algorithm depends on the graph's characteristics: Kruskal is ideal for networks with minimal connections, while Prim is better suited for highly connected networks, such as in the development of internet infrastructure in remote area.
***Keywords**: Kruskal Algorithm, Prim Algorithm, Algorithm Efficiency, Weighted Graph, Network Infrastructure*

I. PENDAHULUAN

Pembangunan infrastruktur jaringan internet di daerah terpencil menjadi salah satu tantangan besar dalam penyediaan layanan telekomunikasi yang merata. Daerah-daerah ini sering kali menghadapi kendala berupa jarak yang jauh, topografi sulit, serta minimnya infrastruktur dasar, yang menyebabkan rendahnya penetrasi internet di kawasan tersebut (Dlamini, 2021). Sebagai solusi, algoritma untuk membangun *minimum spanning tree* (MST) seperti *Kruskal* dan *Prim* telah banyak digunakan dalam berbagai penelitian untuk menyusun infrastruktur jaringan yang optimal dengan biaya minimal (Ayegba et al., 2020), algoritma *Kruskal* lebih efisien pada jaringan dengan struktur graf jarang, sementara algoritma *Prim* lebih cocok untuk jaringan dengan densitas tinggi (Chen, 2023). Hal ini penting dalam perencanaan infrastruktur di daerah terpencil, di mana efisiensi dan optimasi rute komunikasi dapat menurunkan biaya dan waktu pengerjaan proyek (Melnikov & Terentyeva, 2021). Kebutuhan akan jaringan internet di daerah terpencil semakin meningkat seiring dengan berkembangnya era digitalisasi. Algoritma MST seperti *Kruskal* dan *Prim* dapat digunakan untuk optimasi jaringan internet dengan pendekatan topologi graf, yang meminimalkan panjang total jaringan sekaligus

e-ISSN : 2964-5115
p-ISSN : 2964-4364

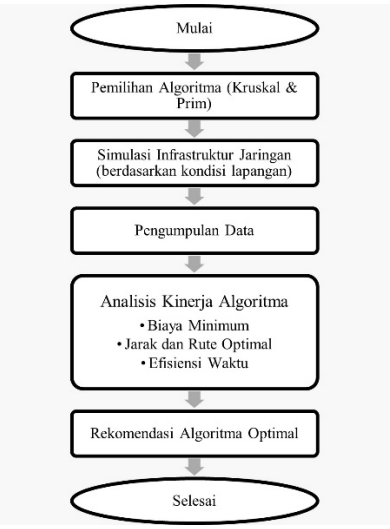
memperhatikan faktor efisiensi energi (Batta et al., 2022).

Pada penelitian sebelumnya, algoritma *Kruskal* dan *Prim* telah diterapkan dalam berbagai konteks. Penelitian (Jlassi et al., 2021) yang menerapkan Kruskal untuk memperpanjang umur jaringan sensor nirkabel, sedangkan algoritma Prim memberikan hasil optimal dalam jaringan yang lebih rapat . Selain itu, penelitian tentang penerapan algoritma-algoritma ini pada sistem monitoring kondisi turbin angin (Yang et al., 2021) menunjukkan bahwa pemilihan algoritma yang tepat dapat mengurangi interferensi dan meningkatkan efisiensi jaringan. Pada penelitian yang menunjukkan efektivitasnya dalam mengoptimalkan topologi jaringan dengan karakteristik tertentu (Rachmawati et al., 2020), yang memberikan gambaran yang jelas mengenai algoritma yang paling cocok untuk diimplementasikan pada daerah terpencil dengan topografi dan densitas jaringan tertentu. Penelitian ini menggunakan metode komparatif dengan membandingkan algoritma Kruskal dan Prim dalam membangun jaringan internet berbasis minimum spanning tree. Penggunaan kedua algoritma ini didasari oleh penelitian terdahulu yang menunjukkan efektivitasnya dalam mengoptimalkan topologi jaringan dengan karakteristik tertentu (Rachmawati et al., 2020). Hasil dari simulasi diharapkan memberikan gambaran yang jelas mengenai algoritma yang paling cocok untuk diimplementasikan pada daerah terpencil dengan topografi dan densitas jaringan tertentu.

Penelitian ini bertujuan untuk mengevaluasi kinerja algoritma *Kruskal* dan *Prim* dalam membangun jaringan internet di daerah terpencil secara optimal, dengan mempertimbangkan aspek biaya, waktu, dan efisiensi energi. Dengan adanya penelitian ini, diharapkan dapat memberikan rekomendasi algoritma terbaik sesuai dengan karakteristik jaringan dan kondisi topografi daerah terpencil.

II. METODE PENELITIAN

Penelitian ini dilakukan dengan menggunakan langkah-langkah seperti pada Gambar 1.



Gambar 1. Tahapan Metode Komparatif Perbandingkan Algoritma Kruskal dan Prim

Pada Gambar 1 menunjukkan proses dalam membangun jaringan internet berbasis *minimum spanning tree.* Tahapan dalam mencari Algoritma yang paling efisien dalam pembangunan infrastruktur jaringan internet di daerah terpencil dapat melalui tahap Pemilihan Algoritma (*Kruskal* & *Prim*), Tahapan ini adalah langkah awal di mana peneliti memilih algoritma yang akan dibandingkan, yaitu *Kruskal* dan *Prim*. Kedua algoritma ini digunakan untuk menyelesaikan masalah *minimum spanning tree* (MST) yang berfokus pada menemukan jalur terpendek untuk menghubungkan titik-titik dalam jaringan. Algoritma *Kruskal* bekerja dengan mengurutkan semua edge dalam graf berdasarkan bobotnya, kemudian menambahkan edge satu per satu ke MST, selama tidak membentuk siklus. Proses ini dilanjutkan hingga semua node terhubung dalam MST. Untuk mendeteksi siklus, algoritma ini menggunakan struktur data *Union-Find* atau *Disjoint Set*, Rumusnya seprti pada Rumus 1

$$Union - Find \leftarrow Find(u) \neq Find\ (v) \rightarrow Union(u, v) \quad (1)$$

Kompleksitas waktu algoritma *Kruskal* adalah $O(ElogE + ElogV)$, di mana E adalah jumlah edge dan V adalah jumlah node. Algoritma ini sangat efektif untuk graf yang jarang karena efisiensinya dalam mengurutkan dan memilih edge. Algoritma *Prim*, di sisi lain dimulai dengan memilih satu node dan memperluas MST dengan menambahkan edge

e-ISSN : 2964-5115
p-ISSN : 2964-4364

dengan bobot terkecil yang menghubungkan node baru ke MST. Algoritma ini terus menambahkan edge hingga semua node terhubung. Pemilihan edge dengan bobot terkecil dilakukan dengan menggunakan struktur data seperti heap untuk mempercepat prosesnya, dengan rumus:

$$min\{weight(u,v) \mid u \in MST, v \in /MST\} \quad (2)$$

Kompleksitas waktu algoritma *Prim* adalah $O(V2)$ atau $O(E + VlogV)$ jika menggunakan heap atau *priority queue*. Algoritma *Prim* lebih cocok untuk graf padat, di mana banyak node memiliki koneksi langsung. Kedua algoritma ini membantu mengoptimalkan jalur dalam pembangunan jaringan, *Kruskal* dipilih karena cocok untuk graf dengan densitas rendah, sedangkan *Prim* lebih efektif pada graf dengan densitas tinggi (Chen, 2023). Pemilihan algoritma ini didasarkan pada karakteristik topografi daerah terpencil yang umumnya memiliki struktur jaringan berbeda.

Setelah algoritma ditentukan, simulasi infrastruktur jaringan dilakukan untuk mendapatkan gambaran realistis dari jaringan yang akan dibangun di daerah terpencil. Simulasi ini menggabungkan berbagai variabel lingkungan seperti jarak antar titik, topografi, dan ketersediaan infrastruktur pendukung. Variabel-variabel ini sangat penting karena memengaruhi performa masing-masing algoritma dalam mengoptimalkan panjang jaringan dan penggunaan biaya (Rachmawati et al., 2020).

Pengumpulan Data diperoleh secara online yang dihasilkan dari data jaringan digunakan sebagai bahan dasar untuk analisis lebih lanjut. Pengumpulan data ini mencakup informasi tentang waktu eksekusi, panjang jalur terpendek yang dihasilkan, serta biaya total pembangunan jaringan untuk masing-masing algoritma. Data ini nantinya akan menunjukkan bagaimana performa algoritma *Kruskal* dan *Prim* dalam berbagai skenario jaringan yang berbeda, memungkinkan perbandingan yang akurat antara keduanya (Melnikov & Terentyeva, 2021). Data diperoleh sebanyak 10 data.

Analisis Kinerja Algoritma, pada tahap ini data yang telah dikumpulkan dianalisis untuk mengevaluasi kinerja masing-masing algoritma. Analisis ini mencakup beberapa metrik penting meliputi, Biaya Minimum, diperoleh dari perbandingkan total biaya yang diperlukan oleh masing-masing algoritma untuk menyelesaikan jaringan. Hal ini penting karena biaya adalah faktor utama dalam pengembangan infrastruktur di daerah terpencil. Jarak dan Rute Optimal, dari menghitung jarak total yang dihasilkan dari jaringan yang dibangun oleh masing-masing algoritma dan mengidentifikasi seberapa efisien rute yang dihasilkan dalam menghubungkan titik-titik jaringan, Efisiensi Waktu dari analisa waktu eksekusi dilakukan untuk melihat seberapa cepat masing-masing algoritma dapat menyelesaikan masalah jaringan pada berbagai ukuran dan kompleksitas graf (Ayegba et al., 2020).

Rekomendasi Algoritma Optimal, berdasarkan hasil analisis, peneliti memberikan rekomendasi algoritma yang paling optimal untuk implementasi di lapangan. Rekomendasi ini mempertimbangkan faktor-faktor seperti biaya terendah, rute yang efisien, dan kecepatan waktu eksekusi. Jika *Kruskal* terbukti lebih efektif pada graf dengan sedikit titik koneksi, maka akan direkomendasikan untuk daerah yang memiliki sedikit pemukiman atau titik akses. Sebaliknya, jika *Prim* lebih optimal pada graf padat, maka *Prim* akan direkomendasikan untuk daerah dengan pemukiman yang lebih terpusat (Jlassi et al., 2021).

Penelitian ini menggunakan pendekatan metode komparatif untuk membandingkan efisiensi dan efektivitas algoritma *Kruskal* dan *Prim* dalam optimasi jaringan internet di daerah terpencil. Metode komparatif dipilih untuk mengidentifikasi algoritma yang paling efisien dalam konteks penggunaan jaringan graf *minimum spanning tree* (MST), terutama dalam hal minimisasi biaya jaringan dan penyesuaian terhadap densitas topologi jaringan (Chen, 2023). Data yang digunakan dalam penelitian ini terdiri dari berbagai parameter jaringan, seperti jumlah node, jarak antar node, dan beban jaringan. Parameter ini dikumpulkan melalui simulasi komputer yang mereplikasi kondisi daerah terpencil dengan keterbatasan infrastruktur. Simulasi ini dilakukan dengan menggunakan perangkat lunak yang mampu menghitung dan mengimplementasikan algoritma graf pada struktur jaringan besar (Rachmawati et al., 2020).

Teknik pengumpulan data melibatkan simulasi yang menjalankan algoritma *Kruskal* dan *Prim* pada

e-ISSN : 2964-5115
p-ISSN : 2964-4364

berbagai skenario jaringan untuk mengamati performa masing-masing algoritma dalam konteks densitas graf yang berbeda (Yu, 2023). Simulasi ini, berbagai parameter kunci seperti waktu pemrosesan dan jumlah operasi dihitung dan dianalisis untuk menilai keunggulan relatif dari kedua algoritma dalam konteks yang relevan (Sholikhatin et al., 2020).

Analisis data dilakukan melalui pendekatan deskriptif dan kuantitatif, di mana hasil simulasi dari kedua algoritma dibandingkan dalam hal kompleksitas waktu dan efisiensi pemakaian sumber daya jaringan (Zhang, 2023). Penggunaan algoritma dengan pendekatan greedy seperti *Kruskal* dan *Prim* memungkinkan analisis mendalam terkait kebutuhan biaya minimum dan waktu yang diperlukan dalam pembentukan jaringan MST di daerah terpencil (Ayegba et al., 2020).

III. HASIL DAN PEMBAHASAN

Hasil dan pembahasan dari data yang diperoleh seperti pada Tabel 1 menunjukkan data infrastruktur internet.

Tabel 1. Data Infrastructure Internet *Kruskal Prim*

Koneksi	Jarak (km)	Biaya ($)
A-B	5	1000
A-C	7	1400
B-C	4	800
B-D	8	1600
…	…	…
E-G	9	1800

Tabel 1 menunjukkan Koneksi: Representasi hubungan antar lokasi (node) dalam jaringan. Setiap koneksi (contoh: A-B, B-C) menunjukkan dua lokasi yang terhubung. Jarak (km): Menggambarkan jarak fisik antar node, yang menjadi salah satu parameter utama untuk mengukur efisiensi algoritma dalam menentukan jalur minimum. Biaya (in $): Estimasi biaya untuk membangun koneksi antar node. Biaya ini didasarkan pada jarak, kebutuhan infrastruktur, dan faktor lain seperti medan atau kondisi geografis.

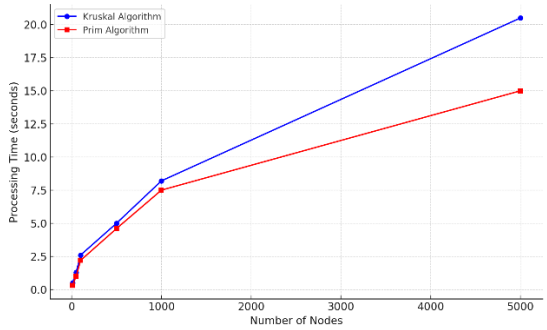
Variabel yang digunakan terdiri dari koneksi, Jarak dan biaya. Data tersebut diimplementasikan menggunakan aplikasi phyton seperti pada Gambar 2.

```
import networkx as nx
import matplotlib. pyplot as plt
# Data simulasi: Graph dengan node dan
edge (dengan bobot)
# Format: (node1, node2, weight)
edges = [
('A', 'B', 4),
('A', 'C', 3),
('B', 'C', 1),
('B', 'D', 2),
('C', 'D', 4),
('C', 'E', 2),
('D', 'E', 1)
]
# Membuat graf menggunakan NetworkX
G = nx. Graph ()
G.add_weighted_edges_from(edges)
# Visualisasi graf asli
def draw_graph (G, title):
pos = nx. spring_layout(G)
nx.draw(G, pos, with_labels=True,
node_color='lightblue', node_size=500,
font_size=10)
labels = nx.get_edge_attributes(G,
'weight')
nx.draw_networkx_edge_labels(G,
pos, edge_labels=labels)
plt.title(title)
plt. show ()
# Menghitung MST menggunakan algoritma
Kruskal
def kruskal_mst(G):
mst = nx. minimum_spanning_tree
(G, algorithm='kruskal')
return mst
# Menghitung MST menggunakan algoritma
Prim
def prim_mst(G):
mst = nx. minimum_spanning_tree
(G, algorithm='prim')
return mst

# Menampilkan hasil graf dan MST untuk
kedua algoritma
if _name_ == "_main_":
# Visualisasi graf asli
draw_graph (G, "Original Graph")
# MST menggunakan Kruskal
mst_kruskal = kruskal_mst(G)
draw_graph (mst_kruskal, "MST
using Kruskal")
print ("Total weight of MST
(Kruskal):", mst_kruskal.
size(weight='weight'))
# MST menggunakan Prim
mst_prim = prim_mst(G)
draw_graph (mst_prim, "MST using
Prim")
print ("Total weight of MST
(Prim):", mst_prim. size(weight='weight'))
```

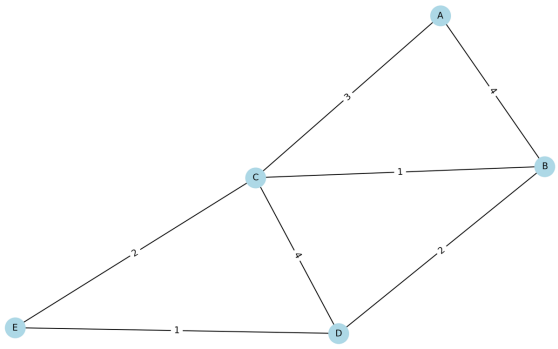
Gambar 2. Aplikasi phyton dari perbandingan algoritma kruskal dan prim

e-ISSN : 2964-5115
p-ISSN : 2964-4364



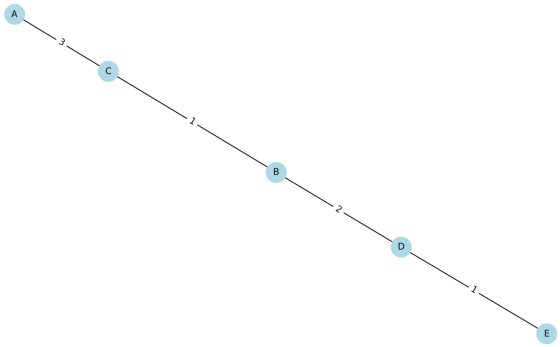
Gambar 3. Kinerja Komparatif Algoritma *Kruskal* dan *Prim*

Berdasarkan gambar 3 menunjukkan perbandingan kinerja algoritma *Kruskal* dan *Prim* dalam membangun *minimum spanning tree*, terlihat bahwa waktu permrosesan kedua algoritma meningkat seiring bertambahnya jumlah node. Namun, algoritma *Prim consistently outperform Kruskal*, terutama untuk jumlah node yang lebih besar. Pada rentang node hingga 5000, algoritma Kruskal menunjukkan peningkatan waktu pemrosesan yang lebih tajam dibandingkan *Prim*. Hal ini mengindikasikan bahwa algoritma *Prim* lebih efisien dalam menangani graf dengan jumlah node besar, menjadikannya pilihan yang lebih optimal untuk aplikasi yang membutuhkan waktu eksekusi lebih cepat pada graf skala besar. Hasil dari penelitian ini untuk algoritma *Kruskal* dan *Prim* masing-masing Total bobot dari MST (*Kruskal*): 7.0 dan Total bobot dari MST (*Prim*): 7.0. Hasilnya seperti pada gambar 4.



Gambar 4. Graph asli.

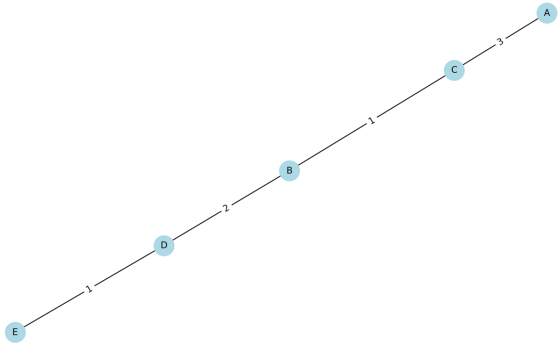
Dari Gambar 4. Menunjukkan Graph asli, Gambar ini menampilkan graf awal dengan semua node (A, B, C, D, E) dan edge (garis penghubung) yang diberikan, termasuk bobot setiap edge. Bobot pada setiap edge menunjukkan "biaya" atau "panjang" koneksi antara dua node.



Gambar 5. Graph dari algoritma Kruskal.

Gambar ini menunjukkan hasil penerapan algoritma *Kruskal* pada graf asli. *Kruskal* bekerja dengan cara memilih edge dengan bobot terkecil secara bertahap sambil memastikan tidak ada siklus yang terbentuk.

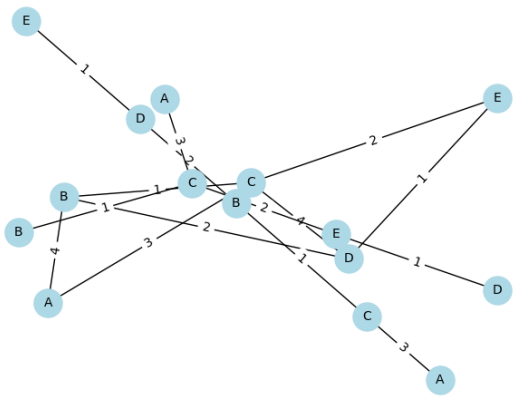
Pada hasil ini, Anda melihat bahwa MST hanya menyertakan edge dengan bobot terkecil yang menghubungkan semua node, yaitu B-C (1), D-E (1), B-D (2), dan C-E (2). Total bobot MST adalah 7.0. Sedangkan hasil algoritma *Prim* seperti pada gambar 5



Gambar 6. Graph dengan algoritma Prim.

Gambar ini menunjukkan hasil penerapan algoritma *Prim* pada graf asli. *Prim* bekerja dengan cara memulai dari satu node, kemudian menambahkan edge dengan bobot terkecil yang menghubungkan node baru ke node yang sudah terhubung. Hasilnya sama dengan *Kruskal*, karena MST yang optimal adalah unik untuk graf ini: B-C (1), D-E (1), B-D (2), dan C-E (2). Total bobot MST juga adalah 7.0.

e-ISSN : 2964-5115
p-ISSN : 2964-4364



Gambar 7. Graph hasil perbandingan Algoritma Kruskal dan Prim.

Gambar 7 menunjukkan hasil perbandingan antara algoritma Kruskal dan Prim dalam membangun *Minimum Spanning Tree* (MST) untuk sebuah graf tertentu. Berdasarkan hasil, berat total MST yang dihasilkan oleh kedua algoritma adalah sama, yaitu **7.0**. Hal ini menunjukkan bahwa kedua algoritma ini mampu menemukan MST dengan bobot minimum yang identik, meskipun pendekatan keduanya dalam membangun MST berbeda.

Tabel 2. Perbandingan kinerja Algoritma Kruskal dan Prim berdasarkan Parameter Utama

Parameter	Graf Asli	MST Menggunakan Kruskal	MST Menggunakan Prim
Visualisasi Graf	Graf dengan semua node dan edge lengkap	Graf dengan edge membentuk MST	Graf dengan edge membentuk MST
Bobot Total MST	-	7.0	7.0
Metode Pemilihan Edge	Semua edge ditampilkan	Memili edge terkecil secara global	Memilih edge terkecil dari node yang terhubung
Keunikan Hasil	Tidak ada MST, semua edge dimasukkan	MST unik dengan bobot minimum	MST unik dengan bobot minimum
Kompleksitas Algoritma	Tidak berlaku (graf asli)	Bergantung pada jumlah edge (E log E)	Bergantung pada node dan edge ($O(V^2)$ atau $O(E + V \log V)$)
Kesesuaian	-	Efisien untuk graf	Efisien untuk graf padat

Dari Tabel 2, menunjukkan (1) Visualisasi Graf: Graf Asli, Graf ini menampilkan semua node (A, B, C, D, E) dan semua edge yang menghubungkannya, lengkap dengan bobot masing-masing. Ini adalah representasi awal sebelum algoritma *Minimum Spanning Tree* (MST) diterapkan. MST Menggunakan *Kruskal*, Setelah menerapkan algoritma *Kruskal*, hanya edge yang membentuk MST (dengan bobot total minimum) yang divisualisasikan. Edge yang tidak diperlukan untuk MST dihapus. MST Menggunakan *Prim*, Hasilnya sama dengan *Kruskal* karena MST yang optimal hanya ada satu untuk graf ini. Namun, cara pemilihan edge berbeda (dijelaskan di bawah). (2) Bobot Total MST: Pada kedua algoritma (*Kruskal* dan *Prim*), total bobot MST adalah 7.0. Bobot ini dihitung dengan menjumlahkan bobot edge dalam MST: Edge B-C (1), D-E (1), B-D (2), dan C-E (2).

IV. KESIMPULAN

Kruskal dan Prim adalah algoritma yang sama-sama efektif dalam menghasilkan *Minimum Spanning Tree* (MST) dengan bobot total minimum, seperti pada graf dengan bobot total 7.0 yang digunakan dalam analisis. Meskipun hasilnya sama, cara kerja kedua algoritma berbeda: *Kruskal* memilih edge dengan bobot terkecil secara global, sedangkan Prim memulai dari satu node dan menambahkan edge terkecil yang terhubung. *Kruskal* lebih efisien untuk graf jarang (sparse), sedangkan *Prim* lebih cocok untuk graf padat (dense). Kompleksitas *Kruskal* bergantung pada sorting edge (O(E log E)), sementara *Prim* bergantung pada representasi graf ($O(V^2)$ atau $O(E + V \log V)$ dengan heap). Pemilihan algoritma tergantung pada tipe graf, di mana Kruskal ideal untuk jaringan dengan sedikit koneksi, dan Prim lebih optimal untuk jaringan yang padat, seperti jaringan komunikasi atau infrastruktur internet di daerah terpencil.

V. REFERENSI

Ayegba, P., Ayoola, J., Asani, E., & Okeyinka, A. (2020). A Comparative Study Of Minimal Spanning Tree Algorithms. *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, 1–4. https://doi.org/10.1109/ICMCECS47690.2020.240900

e-ISSN : 2964-5115
p-ISSN : 2964-4364

Batta, M. S., Aliouat, Z., Mabed, H., & Merah, M. (2022). An Improved Lifetime Optimization Clustering using Kruskal's MST and Batteries Aging for IoT Networks. *2022 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–6. https://doi.org/10.1109/ISNCC55209.2022.9851748

Chen, J. (2023). The analysis and application of Prim algorithm, Kruskal algorithm, Boruvka algorithm. *Applied and Computational Engineering*, *19*(1), 84–89. https://doi.org/10.54254/2755-2721/19/20231012

Dlamini, T. (2021). *Remote and Rural Connectivity : Infrastructure and Resource Sharing Principles*. *2021*. https://doi.org/10.1155/2021/6065119

Jlassi, W., Haddad, R., Bouallegue, R., & Shubair, R. (2021). A Combination of Kruskal and K-means Algorithms for Network Lifetime Extension in Wireless Sensor Networks. *2021 International Wireless Communications and Mobile Computing (IWCMC)*, 658–663. https://doi.org/10.1109/IWCMC51323.2021.9498594

Melnikov, B. F., & Terentyeva, Y. Y. (2021). Building communication networks: On the application of the Kruskal's algorithm in the problems of large dimensions. *IOP Conference Series: Materials Science and Engineering*, *1047*(1). https://doi.org/10.1088/1757-899X/1047/1/012089

Rachmawati, D., Herriyance, & Pakpahan, F. Y. P. (2020). Comparative Analysis of the Kruskal and Boruvka Algorithms in Solving Minimum Spanning Tree on Complete Graph. *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)*, 55–62. https://doi.org/10.1109/DATABIA50434.2020.9190504

Sholikhatin, S. A., Prasetyo, A. B., & Nurhopipah, A. (2020). IMPLEMENTASI ALGORITMA KRUSKAL DAN ALGORITMA PRIM SUATU GRAPH DENGAN APLIKASI BERBASIS DESKTOP. *Jurnal RESISTOR (Rekayasa Sistem Komputer)*, *3*(2), 89–93. https://doi.org/10.31598/jurnalresistor.v3i2.638

Yang, Y., Liu, A., Xin, H., Wang, J., Yu, X., & Zhang, W. (2021). Deployment optimization of wireless mesh networks in wind turbine condition monitoring system. *Wireless Networks*, *27*(2), 1459–1476. https://doi.org/10.1007/s11276-020-02522-w

Yu, X. (2023). Analysis and comparison of two classical greedy algorithms for minimum spanning tree. *Theoretical and Natural Science*, *5*(1), 698–702. https://doi.org/10.54254/2753-8818/5/20230464

Zhang, R. (2023). The comparison of three MST algorithms. *Applied and Computational Engineering*, *17*(1), 191–197. https://doi.org/10.54254/2755-2721/17/20230939