PERBANDINGAN EFISIENSI ALGORITMA DJIKSTRA DAN ALGORITMA A* (A STAR) DALAM MENEMUKAN RUTE OPTIMAL ANTARA SUN PLAZA DAN PODOMORO MENGGUNAKAN PYTHON

Dean Siregar, Evelyn Keisha Silalahi, Clara Kresensia Panjaitan, Putri Harliana
Ilmu Komputer, Universitas Negeri Medan
Jl. William Iskandar Ps. V Medan, Indonesia
deansiregar1609.4233250006@mhs.unimed.ac.id

ABSTRAK

Menentukan rute terpendek merupakan tantangan utama dalam sistem navigasi, terutama di wilayah perkotaan seperti Medan. Penelitian ini memodelkan jaringan rute antara Sun Plaza dan Podomoro City Deli Medan dalam bentuk graf berbobot, kemudian menganalisisnya menggunakan algoritma Dijkstra dan A*. Implementasi kedua algoritma dilakukan menggunakan bahasa pemrograman Python. Hasil penelitian menunjukkan bahwa algoritma Dijkstra menghasilkan rute A → E → F → G → H → D dengan total jarak 2,87 km dan kecepatan rata-rata 48 km/jam, sementara algoritma A* menghasilkan rute A → B → C → D dengan total jarak 2,05 km dan kecepatan rata-rata 26 km/jam. Waktu eksekusi kedua algoritma hampir identik. Kesimpulan penelitian menunjukkan bahwa untuk graf sederhana, kedua algoritma memberikan hasil serupa, tetapi Dijkstra lebih efisien dalam implementasi. Penelitian ini memberikan wawasan praktis untuk pengembangan aplikasi navigasi dan menyarankan studi lanjutan dengan membandingkan algoritma lain pada graf yang lebih kompleks.

Kata kunci : *Algoritma Djikstra , Algoritma A*l , Graf, Phython*

1. PENDAHULUAN

Di era modern yang semakin berkembang, aplikasi seperti Google Maps telah menjadi bagian penting dalam kehidupan sehari-hari, memungkinkan perjalanan cepat dari satu tempat ke tempat lain. Penentuan rute terpendek menjadi masalah penting yang membutuhkan algoritma efisien. Algoritma Dijkstra dan A-Star dipilih karena keduanya merupakan solusi yang paling umum digunakan dalam menyelesaikan masalah pencarian jalur optimal. Kemampuan algoritma ini untuk mencari rute terpendek atau tercepat membuatnya relevan dalam berbagai bidang, terutama dalam sistem navigasi dan perencanaan rute [1].

Dengan efisiensi dan keakuratannya, kedua algoritma ini menjadi landasan penting dalam pengembangan sistem modern yang mendukung mobilitas Masyarakat [2].

Identifikasi masalah dalam penelitian ini adalah perlunya evaluasi komprehensif untuk menentukan algoritma yang lebih efisien dan efektif dalam menemukan rute optimal antara dua lokasi tertentu, Dalam konteks navigasi otomatis, bobot ini dapat mencerminkan jarak fisik antara lokasi, waktu yang diperlukan untuk mencapai suatu titik, atau biaya lainnya yang terkait dengan perjalanan dari satu lokasi ke lokasi lainnya [3].

Khususnya di wilayah Kota Medan, seperti Sun Plaza dan Podomoro City Deli Medan. Dengan perbedaan lingkungan dan struktur jalan,penting untuk mengukur kinerja algoritma dalam situasi nyata. Dalam pengembangan algoritma pencarian rute terdekat, bebeberap penelitian telah dilakukan untuk menghasilkan efisiensi dan akurasi pada algoritma yang digunakan.

Tujuan penelitian ini adalah membandingkan efisiensi algoritma Dijkstra dan A* dalam menemukan rute optimal menggunakan implementasi Python. Penelitian ini akan menilai kedua algoritma berdasarkan waktu eksekusi, jumlah simpul yang dieksplorasi, kecepatan rata-rata dan efektivitas rute yang dihasilkan.

Penelitian ini penting karena dapat memberikan wawasan praktis tentang algoritma mana yang lebih cocok digunakan dalam perencanaan rute di lingkungan perkotaan. Selain itu, hasil penelitian ini juga dapat menjadi referensi bagi pengembang aplikasi navigasi yang membutuhkan algoritma pencarian jalur optimal dengan efisiensi tinggi. Dengan demikian, penelitian ini diharapkan dapat mendukung pengembangan teknologi berbasis graf yang lebih efektif dan relevan dengan kebutuhan pengguna di era digital.

2. TINJAUAN PUSTAKA

Penelitian sebelumnya oleh Jordi Yoga Pratama, Pada penelitian ini menganilisis waktu komputasi dan akurasi rute. Hasil menunjukkan bahwa algoritma djikstra dan A* memiliki kemampuan yang sama dalam menemukan rute terpendek, tetapi pada algoritma A*(A-Star) menunjukakan lebih efisien dalam mengukur waktu komputasi karena memanfaatkan heuristik yang mempercepat proses dalam pencarian rute terdekat[1].

2.1. Navigasi

Sistem Pemosisian Global (GPS) adalah sistem navigasi yang menggunakan satelit untuk menyediakan informasi mengenai lokasi, kecepatan, arah, dan waktu. GPS ini dapat dimanfaatkan untuk

menentukan lokasi toko dengan cepat dan efisien, yang berguna dalam proses distribusi. [4].

Dengan perkembangan smartphone, juga pada saat ini terdapat sebuah alat GPS yang dalam pemetaan atau menavigasi untuk menunjukan posisi suatu tempat atau suatu kendaraan yaitu GPS GARMIN MONTANA [5].

Google maps merupakan sebuah fitur yang ada pada Smartphoe android dan proyeksi geometris. Pencitraan timbul dari sebuah variasi dari sumber-sumber yang melibatkan banyak orang. Sehingga ketidakakuratan pada data terkait dengan hal tersebut. Google maps secara kontinyu mengambil input dan meningkatkan kualitas dari data yang ada[6].

2.2. Algoritma Djikstra

Algoritma Dijkstra digunakan untuk membuktikan jarak paling pendek dari pilihan rute yang tersedia. Algoritma Dijkstra mengevaluasi semua potensi bobot minimum untuk setiap titik. Algoritma Dijkstra merupakan tipe algoritma rakus (greedy algorithm) berprinsip menggunakan layanan minimum pada sejumlah jarak yang ada guna memberikan berbagai solusi untuk mengetahui jarak terpendek[7].

Prinsip dari algoritma greedy adalah tidak mempedulikan node atau node lain yang belum tersentuh, selama algoritma ini telah menemukan titik akhir tujuan, algoritma ini akan berhenti dan tidak akan mencari lagi [8].

Selain itu, sebuah penelitian membuktikan bahwa integrasi Algoritma Dijkstra dengan sistem informasi geografis (GIS) dapat meningkatkan akurasi dan efisiensi dalam menentukan rute terpendek dari pusat kota surabaya ke tempat bersejarah[9].

2.3. Algoritma A* (A-Star)

Dalam pencarian suatu jalur tertentu atau path finding dan graph traversal atau sering disebut dengan grafik melintang tidak lepas dari proses plotting sebuah jalur yag melintang hal tersebut merupakan pengertiandari A Star atau dalam sains komputer sering di sebut A* yang tergolong ke dalam algoritma sedangkan proses plotting tersebut dinamakan node yang terjadi pada jalur melintang secara efisien antara berbagai titik yang ada. Algoritma ini diperluas guna berbagai bidang karena penampilan dan juga akurasinya yang terkenal. Dengan memakai heuristic A* akan mencapai penampilan yang sangat bagus atau baik[10]

2.4. Algoritma

Algoritma adalah serangkaian langkah-langkah logis yang disusun secara terstruktur untuk menyelesaikan suatu masalah. Algoritma sangat erat kaitannya dengan permasalahan rute terpendek, karena dalam menentukan rute terpendek terdapat tahapan-tahapan yang disusun secara teratur. Proses penentuan rute terpendek melibatkan analisis semua jalan yang saling terhubung satu sama lain, yang kemudian membentuk sebuah graf.[11].

2.5. Teori Graf

Teori graf adalah cabang ilmu yang telah ada sejak lama, namun masih memiliki berbagai aplikasi hingga saat ini. Graf digunakan untuk menggambarkan objek-objek diskrit beserta hubungan antar objek tersebut. Secara visual, graf dapat digambarkan dengan objek-objek yang diwakili oleh titik, bulatan, atau noktah, sedangkan hubungan antar objek tersebut digambarkan dengan garis penghubung. Sebagai contoh, peta dapat dianggap sebagai graf, di mana kota-kota digambarkan sebagai bulatan dan jalan-jalan sebagai garis penghubung antar kota.[12]

2.6. Jenis-jenis Graf

a. Graf sederhana adalah graf yang tidak memiliki gelang atau sisi ganda.
b. Graf tak-sederhana adalah graf yang mengandung sisi ganda atau gelang.
c. Graf tak-berarah adalah graf yang sisi-sisinya tidak memiliki orientasi arah.
d. Graf berarah adalah graf di mana setiap sisi diberikan orientasi arah.[12]

2.7. Flowchart

Flowchart adalah representasi grafis yang menggambarkan langkah-langkah dan urutan prosedur dalam sebuah program. Dalam flowchart program, simbol-simbol khusus digunakan untuk menggambarkan alur proses dan keterkaitan antara instruksi-instruksi dalam program tersebut. Sementara itu, flowchart sistem menggambarkan urutan proses dalam sebuah sistem, termasuk alat input, output, dan jenis media yang digunakan dalam pengolahan data. Beberapa pedoman yang perlu diperhatikan oleh analis dan pengembang dalam membuat flowchart adalah sebagai berikut:

a. Flowchart harus disusun dengan arah dari kiri ke kanan dan dari atas ke bawah.
b. Setiap aktivitas yang digambarkan harus jelas dan mudah dimengerti oleh pembaca.
c. Setiap aktivitas harus memiliki titik awal dan akhir yang jelas.
d. Deskripsi dari setiap langkah aktivitas harus menggunakan kata kerja yang jelas.
e. Langkah-langkah dalam aktivitas harus diatur sesuai dengan urutan yang benar.
f. Lingkup dan jangkauan aktivitas yang digambarkan harus dipertimbangkan dengan cermat. Jika ada percabangan yang tidak relevan dengan sistem, sebaiknya gunakan simbol konektor dan tempatkan percabangan tersebut di halaman terpisah atau hilangkan.
g. Simbol yang digunakan dalam flowchart harus sesuai dengan standar konvensional. [13].

3. METODE PENELITIAN

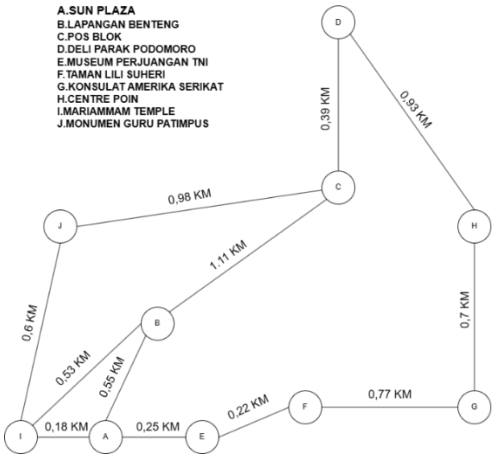
Metode penelitan ini dibagi jadi 5 tahapan diantaranya pemodelan Graf, Implementasi Algortima, Pengujian, Analisis perbandingan dan kesimpulan.

3.1. Pemodelan Graf

Penelitian ini memodelkan jaringan rute antara Sun Plaza dan Podomoro dalam bentuk graf berbobot.Graf tersebut terdiri dari simpul-simpul yang menunjukkan lokasi-lokasi utama,seperti A, B, C dan sebagainya,serta sisi-sisi yang menghubungkan simpul-simpul tersebut dengan bobot yang menunjukkan jarak antar lokasi yang diambil dari google earth.Gambar di bawah ini menunjukkan rute optimal Sunplaza dan Podomoro.

Gambar 1 . Map rute optimal antara sun plaza dan podomoro

Gambar di atas menunjukkan rute optimal antara Sun Plaza dan Podomoro dalam bentuk peta melalui beberapa titik utama yang ditandai sebagai A hingga J. Rute ini menghubungkan lokasi-lokasi strategis di area perkotaan dengan jalur yang dirancang untuk menghindari kemacetan dan meminimalkan jarak tempuh. Jalur optimalnya menggabungkan ruas jalan utama dan sekunder, dengan titik persimpangan yang strategis seperti D, C, dan H untuk mempermudah navigasi di wilayah padat penduduk. Rute ini dirancang menggunakan algoritma seperti Dijkstra atau A*, yang mempertimbangkan jarak, waktu, dan kondisi jalan secara efisien.



Gambar 2 . Graf rute optimal antara sun plaza dan podomoro

Gambar di atas menunjukkan graf yang merepresentasikan rute optimal antara Sun Plaza dan Podomoro dengan simpul-simpul (nodes) yang diberi label dari A hingga J. Graf ini menampilkan hubungan antar titik melalui garis penghubung (edges) yang mewakili jalan atau jalur antar lokasi. Setiap simpul mencerminkan lokasi spesifik, sementara edge dapat melambangkan jarak, waktu tempuh, atau bobot lainnya. Struktur graf ini mempermudah analisis rute menggunakan algoritma seperti Dijkstra atau A* untuk menentukan jalur tercepat atau terpendek berdasarkan bobot yang diberikan, sehingga mendukung perencanaan perjalanan yang efisien.

Tabel 1 . Jarak antara rute

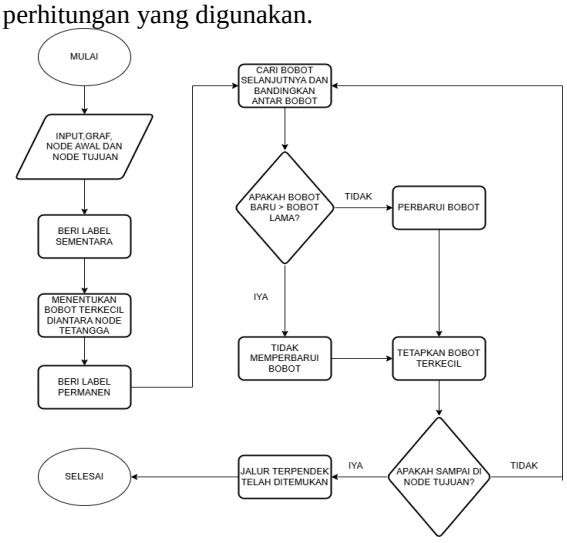
RUTE	JARAK (KM)	KECEPATAN(KM/H)
A-B	0,55	60
B-C	1,11	20
C-D	0,39	20
A-E	0,25	60
E-F	0,22	60
F-G	0,77	60
G-H	0,7	60
H-D	0,93	40
A-I	0,18	60
I-J	0,6	60
J-C	0,98	40
I-B	0,53	60

3.2. Implementasi Algoritma

Penelitian ini mengimplementasikan dua algoritma untuk menemukan rute optimal antara Sun Plaza dan Podomoro dengan menggunakan algoritma Djikstra dan A*.implementasi dilakukan menggunakan bahasa pemrograman Python.

3.2.1. Implementasi Djikstra

Algoritma Djikstra digunakan untuk menemukan rute terpendek dari satu sumber ke satu tujuan.gambar dibawah ini adalah Flowchart serta perhitungan yang digunakan.



Gambar 3 . Flowchart algoritma djikstra

Flowchart algoritma Dijkstra yang digunakan untuk menemukan jalur terpendek pada graf. Proses dimulai dengan memasukkan graf, node awal, dan node tujuan, lalu memberikan label sementara pada node. Langkah berikutnya adalah menentukan bobot terkecil di antara node tetangga dan memberi label permanen pada node dengan bobot terkecil. Bobot selanjutnya dicari dan dibandingkan, di mana bobot akan diperbarui jika nilai baru lebih kecil dari nilai sebelumnya. Proses ini berlanjut hingga jalur terpendek ke node tujuan ditemukan, dan algoritma selesai. Flowchart ini menggambarkan langkah sistematis dari algoritma untuk memastikan efisiensi dalam perhitungan jalur terpendek.

4.2.1. Perhitungan Manual Dijkstra

a. Inisialisasi
   Simpul awal ditetapkan sebagai A. Jarak ke semua simpul lainnya diatur menjadi ∞, kecuali untuk simpul awal (A = 0). Jalur optimal ke simpul lainnya belum ditentukan.
   Berikut adalah jarak awal:
   - A: 0
   - B: ∞
   - C: ∞
   - D: ∞
   - E: ∞
   - F: ∞
   - G: ∞
   - H: ∞
   - I: ∞
   - J: ∞

b. Iterasi 1
   Mulai dari simpul A
   Pertama, dari simpul A, kita akan memeriksa jarak ke tetangga-tetangganya:
   - Jarak A ke B = 0.55
   - Jarak A ke E = 0.25
   - Jarak A ke I = 0.18

   Selanjutnya, perbarui jarak berdasarkan jalur terpendek yang telah ditemukan sejauh ini:
   A: 0, B: 0.55, C: ∞, D: ∞, E: 0.25, F: ∞, G: ∞, H: ∞, I: 0.18, J: ∞
   Akhirnya, tandai simpul A sebagai selesai.

c. Iterasi 2
   Dari simpul I (dengan jarak terkecil: 0.18)
   Periksa tetangga dari I:
   - Jarak I ke B: 0.18 + 0.53 = 0.71
   - Jarak I ke J: 0.18 + 0.6 = 0.78

   Perbarui jarak sebagai berikut:
   A: 0, B: 0.55, C: ∞, D: ∞, E: 0.25, F: ∞, G: ∞, H: ∞, I: 0.18, J: 0.78
   Tandai simpul I sebagai selesai.

d. Iterasi 3
   Dimulai dari simpul E, yang memiliki jarak terkecil sebesar 0.25. Di tahap ini, kita memeriksa tetangga dari E:
   Dari E ke F, jarak yang diperoleh adalah 0.25 + 0.22, sehingga total jaraknya menjadi 0.47. Setelah perhitungan, jarak terupdate menjadi:
   A: 0, B: 0.55, C: ∞, D: ∞, E: 0.25, F: 0.47, G: ∞, H: ∞, I: 0.18, J: 0.78.
   Simpul E kemudian ditandai sebagai selesai.

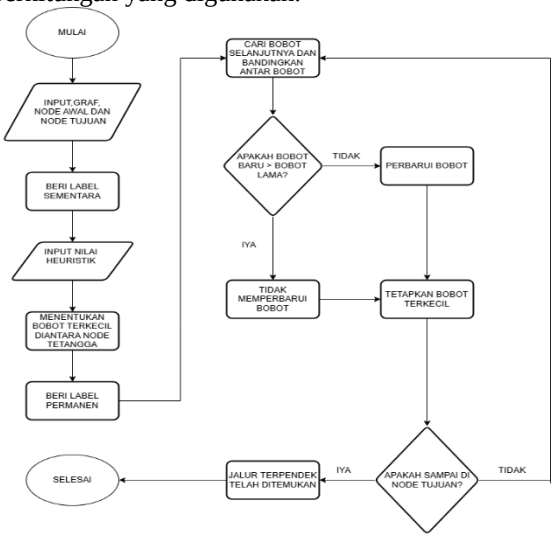
e. iterasi 4
   Kita melanjutkan dari simpul F yang memiliki jarak terkecil 0.47. Selanjutnya, kita memeriksa tetangga dari F:
   Dari F ke G, jarak yang diperoleh adalah 0.47 + 0.77, yang menghasilkan total jarak 1.24. Setelah pembaruan, jarak menjadi:
   A: 0, B: 0.55, C: ∞, D: ∞, E: 0.25, F: 0.47, G: 1.24, H: ∞, I: 0.18, J: 0.78.
   Simpul F juga ditandai sebagai selesai.

f. Iterasi 5
   Dari simpul G (jarak terpendek: 1.24), kita memeriksa tetangga G.
   Jaalip nama family nya likmirak dari G ke H dihitung sebagai 1.24 + 0.7 = 1.94.
   Jarak yang diperbarui adalah:
   A: 0, B: 0.55, C: ∞, D: ∞, E: 0.25, F: 0.47, G: 1.24, H: 1.94, I: 0.18, J: 0.78
   Simpul G ditandai sebagai selesai.

g. Iterasi 6
   Dari simpul H (jarak terpendek: 1.94), kita memeriksa tetangga H.
   Jarak dari H ke D dihitung sebagai 1.94 + 0.93 = 2.87.
   Jarak yang diperbarui menjadi:
   A: 0, B: 0.55, C: ∞, D: 2.87, E: 0.25, F: 0.47, G: 1.24, H: 1.94, I: 0.18, J: 0.78
   Simpul H kini ditandai sebagai selesai.
   Rute terpendek: A → E → F → G → H → D, dengan total jarak 2.87 km.

4.2.2. Implementasi A*

Algoritma A* digunakan untuk menghitung jarak terpendek antara semua apsangan simpul dalam graf.gambar dibawah ini adalah Flowchart serta perhitungan yang digunakan.



Gambar 4 . Flowchart algoritma a*

Flowchart algoritma A* yang digunakan untuk menemukan jalur terpendek pada graf. Proses dimulai dengan memasukkan graf, node awal, dan node tujuan, lalu memberikan label sementara pada node,menginputkan nilai heuristik. Langkah berikutnya adalah menentukan bobot terkecil di antara node tetangga dan memberi label permanen pada node dengan bobot terkecil. Bobot selanjutnya dicari dan dibandingkan, di mana bobot akan diperbarui jika nilai baru lebih kecil dari nilai sebelumnya. Proses ini berlanjut hingga jalur terpendek ke node tujuan ditemukan, dan algoritma selesai. Flowchart ini menggambarkan langkah sistematis dari algoritma untuk memastikan efisiensi dalam perhitungan jalur terpendek.

a. Perhitungan Manual A*
   Rumus A* Untuk setiap simpul:
   $$f(n) = g(n) + h(n)$$
   Keterangan:
   - g(n): Jarak dari simpul awal ke simpul saat ini.
   - h(n): Perkiraan jarak dari simpul saat ini ke tujuan (heuristik).

b. Inisialisasi
   - Simpul awal: A

   Heuristik h(n) (jarak garis lurus ke D):
   - A: 2.05
   - B: 1.5
   - C: 0.39
   - D: 0
   - E: 1.8
   - F: 2.4
   - G: 1.6
   - H: 0.93
   - I: 1.9
   - J: 1.37

c. Iterasi 1
   Dari simpul A, hitung nilai f(n) untuk tetangga A:
   - A → B:
     - g(B) = 0.55
     - h(B) = 1.5
     - f(B) = 0.55 + 1.5 = 2.05
   - A → E:
     - g(E) = 0.25
     - h(E) = 1.8
     - f(E) = 0.25 + 1.8 = 2.05
   - A → I:
     - g(I) = 0.18
     - h(I) = 1.9
     - f(I) = 0.18 + 1.9 = 2.08

   Pilih simpul dengan nilai f(n) terkecil: B atau E (f = 2.05). Misalkan kita pilih B

d. Iterasi 2
   Dimulai dari simpul B, kita perlu menghitung nilai f(n) untuk tetangga B:
   Untuk B menuju C:
   - Hitung g(C): 0.55 + 1.11 = 1.66
   - Hitung h(C): 0.39
   - Jadi, f(C) = 1.66 + 0.39 = 2.05

   Setelah melakukan perhitungan, simpul dengan nilai f(n) terkecil adalah C, dengan f = 2.05.

e. Iterasi 3
   Dari simpul C, lakukan perhitungan nilai f(n) untuk tetangga C:
   C → D:
   - g(D) = 1.66 + 0.39 = 2.05
   - h(D) = 0
   - f(D) = 2.05 + 0 = 2.05

   Dengan demikian, simpul yang dipilih berdasarkan nilai f(n) terkecil adalah D dengan f = 2.05.
   Rute terpendek: A → B → C → D , dengan total jarak 2.05 km.

4. HASIL DAN PEMBAHASAN

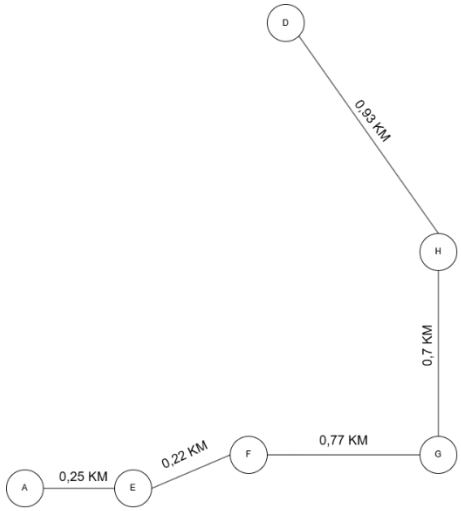
Pada bagian ini, akan dijelaskan hasil implementasi algoritma yang digunakan dalam penelitian serta pembahasan mengenai performa algoritma tersebut. Pembahasan difokuskan pada evaluasi rute terpendek yang ditemukan oleh algoritma Dijkstra dan A*, termasuk pengujian kinerjanya dalam hal kecepatan dan akurasi, dengan mempertimbangkan perangkat keras yang digunakan.

4.1. Pengujian

Pengujian dilakukan untuk mengevaluasi kinerja Algoritma Djikstra dan A* dalam menemukan rute terpendek antara dua simpul pada graf yang telah di modelkan.pengujian ini dilakukan menggunakan prosessor Ryzen 9 5000 series.Tabel dibawah ini adalah hasil dari pengujian yang telah dilakukan.

Tabel 2 . Hasil pengujian algoritma djikstra

From/To	A	B	C	D	E	F	G	H	I	J
A	0	0.55	1.66	2.05	0.25	0.47	1.24	1.94	0.18	0.78
B	0.55	0	1.11	1.5	0.8	1.02	1.79	2.43	0.53	1.13
C	1.66	1.11	0	0.39	1.91	2.13	2.02	1.32	1.58	0.98
D	2.05	1.5	0.39	0	2.3	2.4	1.63	0.93	1.97	1.37
E	0.25	0.8	1.91	2.3	0	0.22	0.99	1.69	0.43	1.03
F	0.47	1.02	2.13	2.4	0.22	0	0.77	1.47	0.65	1.25
G	1.24	1.79	2.02	1.63	0.99	0.77	0	0.7	1.42	2.02
H	1.94	2.43	1.32	0.93	1.69	1.47	0.7	0	2.12	2.3
I	0.18	0.53	1.58	1.97	0.43	0.65	1.42	2.12	0	0.6
J	0.78	1.13	0.98	1.37	1.03	1.25	2.02	2.3	0.6	0
Jarak total: 2.87 KM										
Waktu eksekusi DJIKSTRA* untuk rute A ke D: 0.000000 detik										

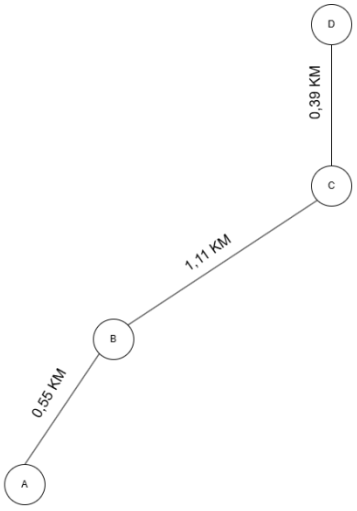


Gambar 5. Rute yang paling efisien berdasarkan algoritma djikstra

Gambar graf di atas menunjukkan bahwa rute paling efisien berdasarkan perhitungan algoritma Dijkstra adalah rute A ->E -> F -> G -> H -> D. Rute ini dipilih karena memiliki total jarak terpendek, yaitu 2,87 km, setelah mempertimbangkan bobot setiap simpul yang terhubung dari titik awal (A) hingga titik tujuan (D). Proses perhitungan ini melibatkan penetapan bobot awal pada setiap simpul yang kemudian diperbarui berdasarkan jalur dengan bobot terkecil. Dalam hal ini, algoritma Dijkstra bekerja dengan cara mengeliminasi jalur-jalur lain yang memiliki bobot lebih besar untuk memastikan efisiensi rute yang dipilih. Dengan demikian, jalur ini tidak hanya memenuhi kriteria jarak terpendek, tetapi juga memastikan perjalanan yang optimal sesuai data graf.

Tabel 3 . Hasil pengujian algoritma a*

From/To	A	B	C	D	E	F	G	H	I	J
A	0	0.55	1.66	2.05	0.25	0.47	1.24	1.94	0.18	0.78
B	0.55	0	1.11	1.5	0.8	1.02	1.79	2.43	0.53	1.13
C	1.66	1.11	0	0.39	1.91	2.13	2.02	1.32	1.58	0.98
D	2.05	1.5	0.39	0	2.3	2.4	1.63	0.93	1.97	1.37
E	0.25	0.8	1.91	2.3	0	0.22	0.99	1.69	0.43	1.03
F	0.47	1.02	2.13	2.4	0.22	0	0.77	1.47	0.65	1.25
G	1.24	1.79	2.02	1.63	0.99	0.77	0	0.7	1.42	2.02
H	1.94	2.43	1.32	0.93	1.69	1.47	0.7	0	2.12	2.3
I	0.18	0.53	1.58	1.97	0.43	0.65	1.42	2.12	0	0.6
J	0.78	1.13	0.98	1.37	1.03	1.25	2.02	2.3	0.6	0
Jarak total: 2.05 KM										
Waktu eksekusi A* untuk rute A ke D: 0.000000 detik										



Gambar 6 . Rute yang paling efisien berdasarkan algoritma a*

Gambar graf di atas menunjukkan bahwa rute paling efisien berdasarkan perhitungan algoritma A* adalah rute A -> B -> C -> D. Rute ini dipilih karena memiliki total jarak terpendek, yaitu 2,05 km, setelah mempertimbangkan bobot setiap simpul yang terhubung dari titik awal (A) hingga titik tujuan (D). Proses perhitungan algoritma A* melibatkan kombinasi antara jarak yang sudah ditempuh (g-cost) dan estimasi jarak ke tujuan (h-cost), sehingga menghasilkan jalur dengan bobot total terkecil. Berbeda dengan Dijkstra, algoritma A* menggunakan heuristik untuk mempercepat pencarian jalur optimal dengan mengeliminasi jalur-jalur yang kurang efisien. Dengan demikian, rute ini memenuhi kriteria jarak terpendek sekaligus memastikan perjalanan yang optimal berdasarkan data graf.

4.2. Analisis Perbandingan

dilakukan analisis terhadap kinerja algoritma Dijkstra dan A* dalam menemukan rute terpendek pada graf yang telah dimodelkan. Analisis mencakup aspek kecepatan eksekusi, efisiensi rute yang dihasilkan, total jarak tempuh,serta kecepatan rata-rata.Pengujian dilakukan dengan menggunakan dataset graf berbobot yang sama untuk memastikan perbandingan dilakukan secara adil. Perbandingan ini bertujuan untuk memahami keunggulan dan kekurangan masing-masing algoritma dalam konteks perhitungan rute optimal antara dua simpul utama, yaitu Sun Plaza dan Deli Park Podomoro. Hasil perbandingan disajikan dalam Gambar dibawah ini.

Tabel 4 . Hasil perbandingan algoritma

ALGORITMA	WAKTU EKSEKUSI	RUTE YANG OPTIMAL	TOTAL JARAK	KECEPATAN RATA-RATA
DJIKSTRA	0,00000 SECOND	A->E->F->G->H->D	2.87 KM	48 KM/H
A*(STAR)	0,00000 SECOND	A->B->C->D	2.05 KM	26 KM/H

Kecepatan eksekusi dan rute yang berbeda pada algoritma Dijkstra dan A* kemungkinan besar disebabkan oleh struktur graf yang tidak terlalu kompleks. Meskipun kedua algoritma menghasilkan rute yang berbeda—Dijkstra menghasilkan rute A -> E -> F -> G -> H -> D dengan total jarak 2.87 km, sedangkan A* menghasilkan rute A -> B -> C -> D dengan total jarak 2.05 km—kedua algoritma memiliki waktu eksekusi yang hampir identik. Kecepatan rata-rata untuk rute Dijkstra adalah 48 km/jam, sedangkan A* memiliki kecepatan rata-rata 26 km/jam. Perbedaan ini lebih dipengaruhi oleh perbedaan dalam struktur graf dan jalur yang ditempuh.

5. KESIMPULAN DAN SARAN

Kesimpulan dari penelitian ini adalah berhasil memodelkan jaringan rute antara Sun Plaza dan Deli Park Podomoro menggunakan graf berbobot serta mengimplementasikan algoritma Dijkstra dan A* untuk menemukan rute terpendek. Hasil pengujian menunjukkan bahwa kedua algoritma menghasilkan rute yang berbeda. Algoritma Dijkstra menghasilkan rute A -> E -> F -> G -> H -> D dengan total jarak 2.87 km, sedangkan algoritma A* menghasilkan rute A -> B -> C -> D dengan total jarak 2.05 km. Berdasarkan hasil pengujian, Dijkstra memberikan kecepatan rata-rata 48 km/jam, sementara A* memiliki kecepatan rata-rata 26 km/jam.Meskipun kedua algoritma memberikan hasil yang berbeda, Dijkstra memiliki total jarak yang lebih panjang, tetapi kecepatan rata-rata yang lebih tinggi. Hal ini menunjukkan bahwa meskipun A* memberikan rute yang lebih pendek, Dijkstra dapat lebih efisien dari segi kecepatan rata-rata.Untuk penelitian selanjutnya, dapat diperluas dengan membandingkan algoritma lainnya, seperti Bellman-Ford atau algoritma berbasis pencarian lainnya, untuk menemukan metode yang paling efisien dalam berbagai jenis jaringan rute.

DAFTAR PUSTAKA

[1] J. Yoga Pratama, “Analisis Perbandingan Algoritma Dijkstra dan A-Star dalam Menentukan Rute Terpendek,” Online, 2024.

[2] A. P. Denio, "Penerapan Teori Graf dalam Pengembangan Algoritma Pencarian Optimal pada Sistem Navigasi Otomatis," *Duniailmu.org*, vol. 3, no. 7, pp. 1–6, 2023.

[3] A. Putra and D. Pendidikan Matematika, “Penerapan Teori Graf dalam Pengembangan Algoritma Pencarian Optimal pada Sistem Navigasi Otomatis.”

[4] R. Rumini and D. Lesmana, “Algoritma Dijkstra untuk Menentukan Jalur Tercepat pada Pendistribusian Barang Berbasis Mobile,” *Jurnal Sistem dan Teknologi Informasi (Justin)*, vol. 8, no. 4, p. 362, Oct. 2020, doi: 10.26418/justin.v8i4.42250.

[5] “Luciana Singal-Comparative Analysis of Google Maps Coordinates Points and Professional GPS Tools in Manado City.” [Online]. Available: https://www.google.com/maps

[6] H. Hendri, “PRAYER ROOM QIBLA DIRECTION AT SCHOOL IN BUKITTINGGI : (Qibla Study in Junior High School and Senior High Schools Prayer Room),” *Al-Hilal: Journal of Islamic Astronomy*, vol. 1, no. 1, Oct. 2019, doi: 10.21580/al-hilal.2019.1.1.5189.

[7] I. Journal ----of ----, G. Nurendrawan Ramadhan, R. Khaira Alma Bachrun, A. Syaifulloh, F. Sains dan Teknologi, and U. Sunan Ampel Surabaya, “PENERAPAN ALGORITMA DIJKSTRA UNTUK MENENTUKAN RUTE TERPENDEK TEMPAT TINGGAL KE KAMPUS 2 UIN SUNAN AMPEL SURABAYA,” vol. 1, 2024, doi: 10.21927/ijubi.v7i1.3964.

[8] F. Lourence Tobing and F. Adline Twince Tobing, “ANALISIS PERBANDINGAN ALGORITMA DFS, BFS DAN DIJKSTRA UNTUK MENENTUKAN RUTE TERPENDEK PADA PETA GEOGRAFIS,” *Jurnal Widya*, vol. 3, no. 1, pp. 59–67, 2022, doi: 10.54593/awl.v3i1.83.

[9] F. Kuncoro, I. A. Zulkarnain, and G. A. Buntoro, "Penerapan Algoritma Dijkstra dalam Menentukan Rute Terpendek pada TPA Mrican," *Antivirus: Jurnal Ilmiah Teknik Informatika*, vol. 18, no. 2, pp. 200–211, Nov. 2024. doi: 10.35347/antivirus.v18i2.3785.

[10] F. L. Tobing, F. A. T. Tobing, and Prayogo, "Analisis Perbandingan Algoritma DFS, BFS, dan Dijkstra untuk Menentukan Rute Terpendek pada Peta Geografis," *Jurnal Widya*, vol. 3, no. 1, pp. 59–67, Apr. 2022. doi: 10.54593/awl.v3i1.83.

[11] J. Adriano Pane, I. Fitriani, and M. Lestari, “IMPLEMENTASI ALGORITMA DIJKSTRA DALAM MENENTUKAN RUTE TERPENDEK MENUJU MUSEUM DI JAKARTA,” 2024.

[12] P. Jalur, T. Dengan, M. Graf, D. Greedy, and K. Sehari-Hari, “Makalah IF3051 Strategi Algoritma-Sem. I Tahun,” 2011.

[13] A. Zalukhu *et al.*, “PERANGKAT LUNAK APLIKASI PEMBELAJARAN FLOWCHART,” *Jurnal Teknologi Informasi dan Industri*, vol. 4, no. 1, 2023