JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Visualisasi Dan Perbandingan Efisiensi Algoritma A*, Greedy, Dan Dijkstra Pada Data Openstreetmap Kota Medan

Paris Alvito*, Muhammad Ikhsan

Fakultas Sains dan Teknologi, Ilmu Komputer, Universitas Islam Negeri Sumatera Utara, Medan, Indonesia
Email: 1* alvinparis65@gmail.com, mhdikhsan@uinsu.ac.id
Email Penulis Korespondensi: alvinparis65@gmail.com
Submitted **10-07-2025;** Accepted **01-08-2025;** Published **14-08-2025**

Abstrak

Penelitian ini membandingkan efisiensi algoritma A*, Greedy, dan Dijkstra dalam pencarian rute terpendek pada jaringan jalan Kota Medan menggunakan data geospasial dari OpenStreetMap. Sistem visualisasi interaktif dikembangkan berbasis web menggunakan React.js, MapLibre GL, dan Deck.gl untuk menampilkan proses pencarian secara real-time. Pengujian dilakukan pada dua skala graf dan enam parameter: waktu eksekusi, jumlah *node* yang dieksplorasi, jarak rute, penggunaan memori, jumlah *node* dalam rute, dan skalabilitas. Hasil menunjukkan bahwa algoritma A* paling efisien secara keseluruhan dengan waktu 0.13 detik dan eksplorasi 17 *node* pada graf kecil, serta 0.29 detik dan 52 *node* pada graf besar. Dijkstra menghasilkan rute optimal namun dengan eksplorasi dan memori lebih besar, sedangkan Greedy tercepat (0.11 detik) tetapi kurang akurat. Penelitian ini memberikan kontribusi terhadap pemahaman algoritma pencarian jalur dan penerapannya dalam sistem berbasis peta.

Kata Kunci: Algoritma A*; Algoritma Dijkstra; Algoritma Greedy; OpenStreetMap; Visualisasi; Pencarian Rute Terpendek

Abstract

This study compares the efficiency of A*, Greedy, and Dijkstra algorithms in finding the shortest path on the road network of Medan City using geospatial data from OpenStreetMap. An interactive visualization system was developed using web-based technologies such as React.js, MapLibre GL, and Deck.gl to display the pathfinding process in real-time. The evaluation was conducted on two graph scales using six parameters: execution time, number of explored nodes, path length, memory usage, number of nodes in the path, and scalability. The results show that the A* algorithm is the most efficient overall, achieving 0.13 seconds with 17 nodes explored on the small graph, and 0.29 seconds with 52 nodes on the large graph. Dijkstra yields the most accurate paths but with significantly more *node* exploration and memory consumption, while Greedy is the fastest (0.11 seconds) but less accurate. This research contributes to the understanding of pathfinding algorithms and their implementation in map-based systems.

Keywords: A* Algorithm; Dijkstra Algorithm; Greedy Algorithm; OpenStreetMap; Visualization; Shortest Path Search.

1. PENDAHULUAN

Pencarian rute terpendek merupakan salah satu permasalahan fundamental dalam ilmu komputer yang memiliki penerapan luas di berbagai bidang, seperti transportasi, logistik, sistem navigasi, dan perencanaan kota [1]. Kemampuan untuk menemukan jalur tercepat atau terpendek antara dua titik sangat penting dalam pengambilan keputusan yang efisien, baik dalam sistem navigasi real-time maupun dalam perancangan jaringan transportasi yang optimal. Di tengah kemajuan teknologi informasi, pengembangan metode pencarian jalur yang efisien dan tepat sasaran menjadi aspek yang sangat krusial [2].

Kota Medan, sebagai kota metropolitan di Pulau Sumatera, memiliki struktur jalan yang kompleks dan terus berkembang. Kompleksitas ini menciptakan tantangan tersendiri dalam proses pencarian rute optimal, khususnya dalam konteks implementasi algoritma pada data geografis dunia nyata [3]. Oleh karena itu, Kota Medan dipilih sebagai studi kasus dalam penelitian ini untuk merepresentasikan permasalahan nyata dalam pencarian rute terpendek yang umum dihadapi di wilayah perkotaan padat [4].

Sejumlah penelitian sebelumnya telah mengkaji efektivitas berbagai algoritma pencarian jalur seperti Dijkstra, Greedy, dan A*. Misalnya, Lakutu et al. (2023) membandingkan algoritma Dijkstra dan Greedy untuk pengiriman barang di Gorontalo, dan menemukan bahwa Dijkstra menghasilkan rute lebih optimal namun memerlukan eksplorasi lebih banyak [5]. Penelitian Sugianti et al. (2020) menunjukkan bahwa A* lebih efisien pada grid kompleks dibandingkan Greedy, sedangkan BFS paling lambat [6]. Beberapa studi lainnya menitikberatkan pada akurasi hasil dan waktu tempuh, namun jarang yang membahas proses eksplorasi algoritma secara visual atau integrasi dengan data geografis nyata seperti OpenStreetMap.

Penelitian ini hadir untuk mengisi celah (GAP) tersebut dengan tidak hanya membandingkan efisiensi ketiga algoritma (Dijkstra, A*, dan Greedy), tetapi juga menyajikan visualisasi interaktif berbasis web yang menggambarkan langkah-langkah eksplorasi pada peta digital Kota Medan secara *real-time*. Visualisasi ini memberikan pendekatan baru dalam memahami cara kerja internal algoritma yang sebelumnya hanya dijelaskan secara tekstual atau numerik .

Data geospasial yang digunakan diperoleh dari OpenStreetMap (OSM) melalui Overpass API dalam format GeoJSON. Peneliti mengambil dua graf yang berbeda dari area Kota Medan, masing-masing mewakili area dengan tingkat kerumitan jalur yang berbeda. Hal ini memungkinkan evaluasi algoritma dilakukan pada kondisi graf berskala kecil dan besar untuk mengukur performa dalam berbagai skenario [7].

Setiap graf kemudian diolah menjadi struktur data simpul dan sisi (*node* and edge) sebagai input dalam proses pencarian jalur [8]. Evaluasi dilakukan berdasarkan enam parameter penting, yaitu: (1) waktu eksekusi, (2) jumlah *node*

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

yang dieksplorasi, (3) jarak rute terpendek, (4) penggunaan memori, (5) jumlah *node* pada rute terpendek, dan (6) skalabilitas [9]. Sistem pengujian dan visualisasi dikembangkan dengan teknologi modern: React.js sebagai antarmuka pengguna, MapLibre GL untuk peta *real-time* berbasis WebGL, dan Deck.gl untuk visualisasi layer eksplorasi algoritma dalam bentuk 2D/3D [13]. Sistem ini memungkinkan pengguna mengamati secara langsung bagaimana algoritma menjelajahi *node* untuk menemukan jalur terpendek.

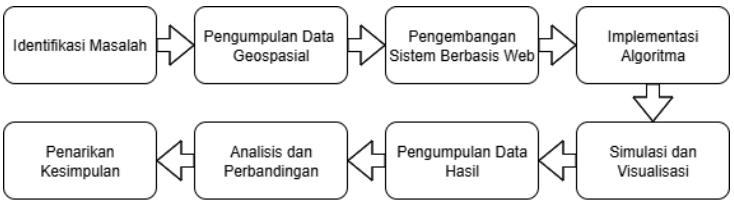
Berbeda dari penelitian terdahulu yang umumnya hanya menyajikan hasil dalam bentuk tabel atau grafik, sistem ini menekankan pada pengamatan langsung proses pencarian, sehingga memberikan nilai edukatif tinggi dalam konteks pembelajaran algoritma praktis. Dosen, mahasiswa, maupun pengembang aplikasi dapat memanfaatkan sistem ini untuk memahami kelebihan dan kekurangan setiap algoritma dalam konteks graf dunia nyata.

Kontribusi utama dari penelitian ini adalah menghadirkan pendekatan baru dalam analisis dan visualisasi algoritma pencarian jalur secara interaktif berbasis data geospasial nyata, yang sekaligus membuka peluang penelitian lebih lanjut di bidang sistem navigasi dan visualisasi algoritmik berbasis web.Sistem ini dirancang tidak hanya sebagai alat penelitian tetapi juga sebagai media pembelajaran dan eksplorasi akademik. Dosen dan mahasiswa dapat memanfaatkan sistem untuk memahami perilaku algoritma secara visual dan interaktif, sementara peneliti dan pengembang aplikasi dapat melihat potensi penerapannya dalam membangun solusi navigasi atau sistem logistik yang efisien. Dengan pendekatan berbasis data nyata, pengujian menyeluruh terhadap enam parameter efisiensi, dan visualisasi berbasis *web* yang interaktif, penelitian ini diharapkan memberikan kontribusi signifikan terhadap pemahaman algoritma pencarian rute serta membuka jalan untuk penelitian lanjutan dalam pengembangan sistem geospasial cerdas yang efisien dan edukatif [10].

2. METODOLOGI PENELITIAN

2.1 Kerangka Kerja Penelitian

Penelitian ini menggunakan pendekatan kuantitatif eksperimental untuk menganalisis dan membandingkan efisiensi tiga algoritma pencarian jalur terpendek, yaitu Dijkstra, A*, dan Greedy Best First Search. Ketiga algoritma dipilih karena masing-masing mewakili pendekatan berbeda dalam pencarian rute: Dijkstra dengan eksplorasi menyeluruh (*exhaustive search*), A* dengan pendekatan heuristik, dan Greedy dengan strategi lokal terbaik [11]. Lingkungan pengujian dibangun dalam bentuk simulasi berbasis web dengan visualisasi real-time untuk meningkatkan pemahaman terhadap proses pencarian jalur. Data jalan diperoleh dari OpenStreetMap (OSM) melalui Overpass API dalam format GeoJSON, yang mencakup simpul (*node*) dan sisi (*edge*) jaringan jalan Kota Medan. Dua wilayah berbeda digunakan: satu dengan graf kecil dan satu dengan graf besar, untuk menguji performa algoritma dalam dua skala yang berbeda [12].



Gambar 1. Kerangka Penelitian

2.1 Kajian Teori Algoritma

a. Algoritma Dijkstra

Merupakan algoritma pencarian jalur terpendek yang menjamin solusi optimal dari simpul awal ke simpul tujuan dalam graf berbobot non-negatif. Algoritma ini melakukan eksplorasi semua jalur potensial dengan strategi greedy minimum *cost* . Meskipun akurat, Dijkstra cenderung lambat pada graf besar karena eksplorasi menyeluruh terhadap *node* yang belum tentu relevan [13].

b. Algoritma A*

(A Star) merupakan pengembangan dari Dijkstra dengan penambahan fungsi heuristik f(n) = g(n) + h(n) untuk memperkirakan jarak total dari *node* awal ke tujuan. g(n) adalah biaya dari awal ke *node* saat ini, sedangkan h(n) adalah estimasi jarak ke tujuan. Fungsi heuristik ini mempercepat pencarian dengan menghindari eksplorasi *node* yang tidak relevan, dan menghasilkan jalur optimal jika heuristik yang digunakan bersifat *admissible* [14].

c. Algoritma Greedy Best First Search

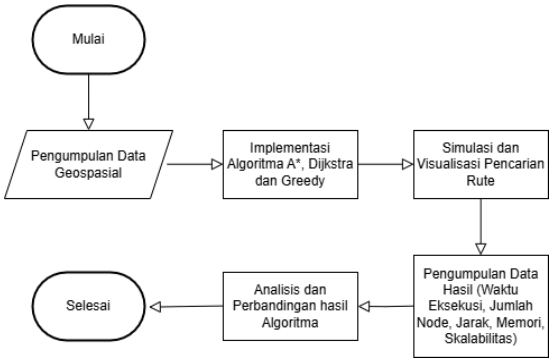
Greedy BFS menggunakan pendekatan lokal dengan hanya mempertimbangkan nilai heuristik h(n) tanpa memperhatikan g(n). Algoritma ini sangat cepat namun tidak menjamin hasil optimal karena cenderung memilih jalur yang tampak paling dekat tanpa mempertimbangkan total biaya. Dalam konteks graf dunia nyata, pendekatan ini dapat menghasilkan rute lebih panjang dibanding algoritma lain [12].

2.2 Tahapan Metode

Penelitian diawali dengan identifikasi kebutuhan sistem yang tidak hanya menampilkan hasil pencarian jalur, tetapi juga

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

memvisualisasikan prosesnya secara interaktif. Data graf jalan diperoleh dari OpenStreetMap (OSM) melalui Overpass API dalam format GeoJSON. Dua area graf di Kota Medan dipilih untuk pengujian algoritma pada skala kecil dan besar, mencakup simpul (*nodes*) dan sisi (*edges*) antar titik jalan. Pengujian dan pencatatan data dilakukan otomatis oleh sistem, lalu dianalisis secara kuantitatif guna mengevaluasi performa dan efisiensi tiap algoritma. Hasil perbandingan dijadikan dasar dalam menyimpulkan serta merekomendasikan algoritma sesuai kondisi graf dan kebutuhan pengguna, baik untuk pendidikan maupun aplikasi pencarian rute.



Gambar 2. *Flowchart* Penelitian

Tahapan penelitian ini dijelaskan sebagai berikut:

a. Pengumpulan Data Geospasial: Data graf jaringan jalan Kota Medan diperoleh dari OpenStreetMap melalui Overpass API, lalu diformat dalam bentuk GeoJSON untuk kemudahan pemrosesan spasial dan kompatibilitas dengan sistem visualisasi.
b. Implementasi Algoritma: Algoritma Dijkstra, A*, dan Greedy diimplementasikan dalam bahasa pemrograman JavaScript untuk dijalankan dalam sistem berbasis web.
c. Simulasi dan Visualisasi Pencarian Rute: Sistem visualisasi dibangun menggunakan React.js (UI), MapLibre GL (rendering peta WebGL), dan Deck.gl (visualisasi eksplorasi *node* dan jalur akhir). Pengguna dapat memilih titik awal dan tujuan untuk mengamati proses pencarian setiap algoritma.
d. Pengumpulan Data Hasil: Sistem secara otomatis mencatat enam parameter evaluasi dari setiap algoritma, keenam parameter tersebut adalah (1) Waktu Eksekusi, (2) Jumlah *Node* yang Dieksplorasi, (3) Jarak Rute Terpendek, (4) Penggunaan Memori, (5) Jumlah *Node* dalam Rute, (6) Skalabilitas terhadap ukuran graf.
e. Analisis dan Perbandingan Hasil Algoritma: Setiap algoritma diuji sebanyak 20 kali pada masing-masing graf. Rata-rata dari hasil uji dicatat, kemudian dilakukan perbandingan performa secara kuantitatif untuk mengevaluasi efisiensi dari masing-masing algoritma.
f. Penyusunan Kesimpulan: Hasil analisis digunakan untuk menarik kesimpulan mengenai kekuatan dan kelemahan algoritma pada berbagai kondisi graf, serta rekomendasi pemilihan algoritma untuk implementasi di dunia nyata.

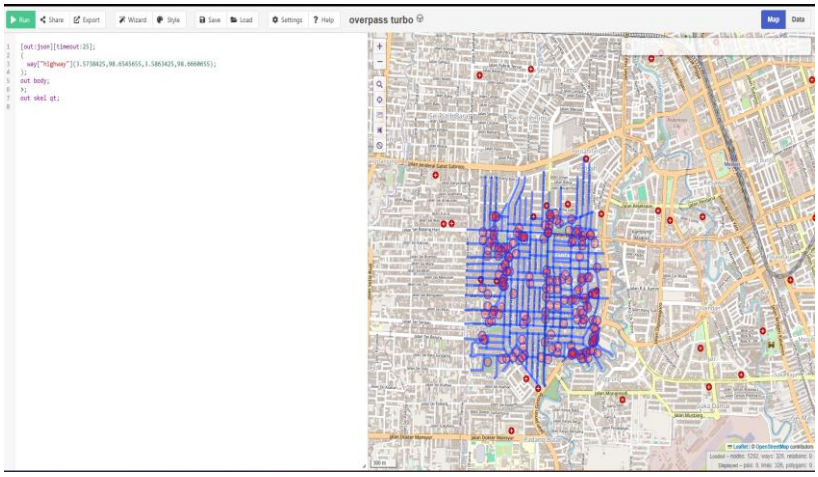
3. HASIL DAN PEMBAHASAN

3.1 Pengumpulan Data dan Struktur Graf

Pengumpulan data dilakukan dengan memanfaatkan layanan *Overpass Turbo* (https://overpass-turbo.eu/), sebuah antarmuka interaktif untuk *Overpass API* yang digunakan dalam mengekstrak data geospasial dari OpenStreetMap (OSM). Dalam penelitian ini, dua wilayah di Kota Medan dipilih sebagai graf uji, masing-masing mewakili graf skala kecil dan graf skala besar. Kriteria pemilihan wilayah didasarkan pada kompleksitas jaringan jalan, kepadatan simpul, dan variasi struktur jalan untuk memastikan pengujian algoritma dapat mencerminkan kondisi nyata [15].

Query Overpass ditulis untuk mengekstrak data jalan dalam radius tertentu pada dua lokasi berbeda di Kota Medan. Hasil dari *query* berupa data dalam format GeoJSON yang mencakup *node* (simpul) dan *way* (sisi) yang menggambarkan struktur jaringan jalan [15]. Data yang diperoleh masih mentah, sehingga dilakukan proses pembersihan dan optimalisasi menggunakan Geojson.io. Proses ini mencakup penghapusan entitas redundan, perbaikan struktur *node*, dan penyederhanaan geometri agar sesuai dengan kebutuhan pemodelan graf. Graf dengan skala kecil memiliki area radius kurang lebih 800 meter persegi, sementara graf dengan skala besar memiliki area radius kurang lebih 1.6 Km.

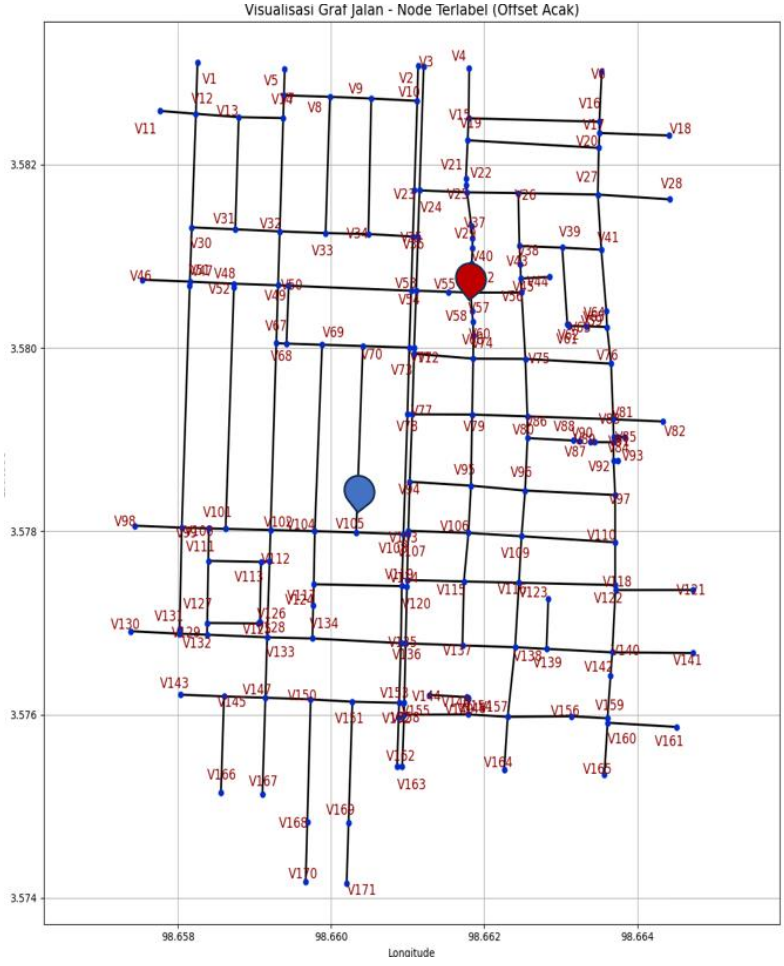
JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 3. Hasil *Query* pada *Web Overpass Turbo* untuk menentukan graf yang akan digunakan diarea kota medan

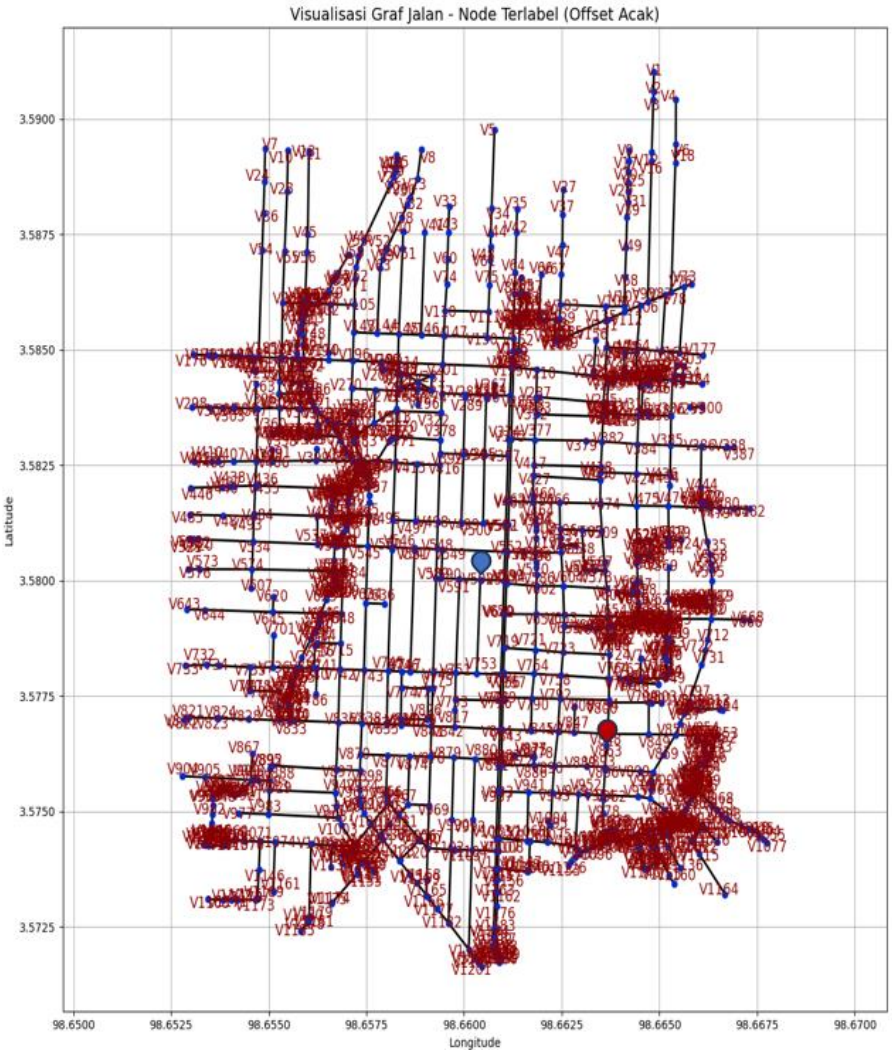
Setelah optimasi data GeoJSON, titik awal dan tujuan ditentukan secara manual berdasarkan posisi strategis di peta. Graf yang telah dioptimalkan digunakan sebagai input untuk implementasi dan pengujian algoritma, baik secara manual maupun melalui sistem berbasis web. Untuk mendukung pengujian, graf dimodifikasi agar menampilkan seluruh simpul beserta koordinat dan penomorannya, sehingga setiap node dapat diidentifikasi dengan jelas. [16]. Proses visualisasi dilakukan secara otomatis menggunakan program Python yang dijalankan di Google Colab, dengan dukungan pustaka matplotlib.pyplot untuk plotting dan shapely.geometry untuk manipulasi objek geospasial. Data map.geojson yang memuat graf sebelumnya menjadi dasar input, dan diproses dalam beberapa tahapan sebagai berikut:

1. Ambil semua koordinat unik
2. Urutkan dari kiri atas ke kanan bawah
3. Labelkan dengan V1, V2, ..., Vn
4. Gambar garis jalur, Gambar titik dan label



Gambar 4. Hasil program python, graf skala besar yang sudah dioptimasi dan diberi label

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 5. Hasil program python, graf skala besar yang sudah dioptimasi dan diberi label

Visualisasi graf menunjukkan bahwa graf skala kecil memiliki 171 node (V1–V171) dengan titik awal di V105 dan tujuan di V58, sedangkan graf skala besar memuat 1.202 node, dari V592 ke V863. Beberapa node terlihat berdekatan atau bertumpang tindih, khususnya di area persimpangan tak bersudut siku-siku. Node tersebut tidak dapat dihapus karena akan mengganggu akurasi dan menyebabkan ketidaksesuaian dengan representasi asli dari OpenStreetMap (OSM) [15]. Penyesuaian yang diperbolehkan hanyalah dalam bentuk pemangkasan skala graf atau penambahan nama jalan kosong saat proses optimasi. Selanjutnya, diperlukan perhitungan jarak antar-*node* ke *node* tetangga terdekat dalam satuan meter. Proses ini melibatkan konversi koordinat dari data GeoJSON, di mana setiap baris array dalam atribut *coordinates* mewakili satu *node* dengan format [*Longitude, Latitude*]. Untuk memperoleh jarak aktual antar dua *node*, koordinat tersebut terlebih dahulu dikonversi ke satuan radian, lalu dihitung menggunakan rumus Haversine guna memperoleh jarak geodesik yang akurat.
Rumus Konversi ke Radian:

$$\text{radian} = \text{derajat} \times \left(\frac{\pi}{180}\right)$$

Rumus *Harvesine*:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \arctan 2(\sqrt{a}, \sqrt{1-a})$$

$$d = R \cdot c$$

Keterangan:
1. φ1,φ2: Lintang (*latitude*) titik 1 dan 2 (dalam radian)
2. λ1,λ2: Bujur (longitude) titik 1 dan 2 (dalam radian)
3. Δφ: Selisih lintang (latitude) antar dua titik
4. Δ λ: Selisih lintang (longtitude) antar dua titik

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

5. *R*: Jari-jari bumi = 6371 km atau 6371000 meter
6. *d*: Jarak antara dua titik dalam meter

Tabel 1. Daftar jarak dari setiap *node* yang terhubung dengan *node* tetangga pada graf skala kecil

No	Titik Awal	Koordinat Titik Awal	Titik Tujuan	Koordinat Titik Tujuan	Jarak (Meter)
1	V1	98.6612097, 3.5830717	V12	98.6611553, 3.5825246	61
2	V2	98.6618034, 3.5830496	V10	98.661796, 3.5826708	42
3	V3	98.6612097, 3.5830717	V24	98.6611553, 3.5817185	149.76
..	..	..	..	..	..
225	V171	98.6602038, 3.574162	V169	98.6602286, 3.5748159	72.36

Langkah selanjutnya adalah menghitung nilai heuristik berdasarkan titik tujuan yang sudah kita tentukan. Nilai heuristik (h(n)) adalah prediksi jarak atau biaya tersisa dari *node* sekarang ke tujuan. Setiap *node* akan memiliki nilai heuristik yang berbeda-beda, tergantung seberapa jauh *node* itu dari tujuan. Tujuan dari h(n) adalah memandu pencarian menuju tujuan seefisien mungkin, dengan memperkirakan "arah terbaik" untuk melangkah [17]. Nilai heuristik diperlukan untuk algoritma A* dan Greedy. Perhitungan manual dilakukan dengan menetapkan V58 sebagai tujuan, lalu menghitung heuristik seluruh node menggunakan pendekatan rumus geodesic sphere untuk hasil yang akurat.
Rumus *geodesic sphere:*

$$d = R \cdot \arccos\left(\sin(\phi_1) \cdot \sin(\phi_2) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \cos(\Delta\lambda)\right)$$

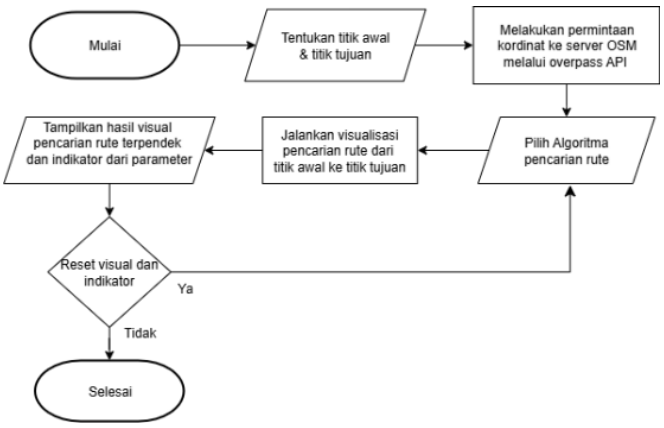
Keterangan:
1. ϕ1 = latitude V1 dalam radian
2. ϕ2 = latitude V58 dalam radian
3. Δλ = selisih longitude dalam radian
4. *R* = jari-jari bumi (6,371,000 meter)

Tabel 1. Daftar nilai heuristik pada seluruh *node* yang ada digraf skala kecil

Node	Longitude	Latitude	Nilai Heuristik ke V58
V1	98.65826	3.583109	497.92
V2	98.66114	3.583074	305.19
V3	98.66121	3.583072	302.97
..	..	..	..
V171	98.6602	3.574162	713.98

3.2 Visualisasi dan Implementasi Algoritma dalam Sistem berbasis Web

Penelitian ini tidak hanya menguji algoritma secara numerik, tetapi juga memvisualisasikan proses pencarian jalur secara interaktif melalui sistem berbasis web. Sistem dikembangkan menggunakan React.js, MapLibre GL untuk peta WebGL, dan Deck.gl untuk animasi node dan jalur. Visualisasi memungkinkan pengguna mengamati langsung eksplorasi graf oleh algoritma Dijkstra, A*, dan Greedy pada peta Kota Medan. Sistem bersifat modular, terdiri dari Interface.jsx, Map.jsx, dan direktori models/algorithms. Seluruh proses berjalan di sisi klien, termasuk animasi bertahap dan pencatatan metrik performa seperti waktu eksekusi, jumlah node, panjang rute, dan konsumsi memori. Data graf diperoleh melalui Overpass API dan dikonversi menggunakan Graph.js serta Node.js.



Gambar 6. Flowchart sistem berbasis web

Data jaringan jalan Kota Medan diperoleh dari OpenStreetMap (OSM) dalam format GeoJSON melalui Overpass API. Sistem memfilter data hanya pada ruas jalan bertag *highway* yang relevan bagi kendaraan, lalu mengubahnya menjadi graf berbasis simpul (*node*) dan sisi (edge) berdasarkan koordinat geografis. Jarak antar simpul dihitung

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

menggunakan rumus geodesik. Pengguna dapat menentukan titik awal dan tujuan, yang kemudian memicu pengambilan data OSM secara dinamis untuk membentuk graf lokal [15].

Visualisasi menggunakan MapLibre GL sebagai peta dasar dan Deck.gl untuk menampilkan node serta edge secara interaktif. Warna, ketebalan garis, dan animasi merepresentasikan status eksplorasi dan progres algoritma. Parameter seperti radius pencarian, kecepatan animasi, jenis algoritma, dan warna elemen dapat dikonfigurasi real-time. Rute terpendek ditandai garis merah tebal, sementara node dan jalur eksplorasi diberi warna sesuai status. Antarmuka menyediakan kontrol penuh seperti tombol mulai/reset, pemilihan titik melalui peta, dan pengaturan tampilan. Sistem merespons interaksi dengan animasi real-time dan menampilkan metrik performa secara dinamis untuk memperjelas efisiensi algoritma. [6].

3.3 Hasi dan Pengujian pada Algoritma

Dataset dan graf yang telah disiapkan sebelumnya kemudian akan digunakan oleh algoritma Dijkstra, A* dan greedy untuk diuji dengan ke enam parameter tersebut. Proses pengujian dilakukan 2 cara, dengan perhitungan yang dilakukan secara manual, lalu dilakukan oleh sistem berbasis web. Setiap algoritma diuji sebanyak 20 kali pada masing-masing ukuran graf, lalu dihitung nilai rata-rata sebagai representasi objektif dari performa waktu eksekusi

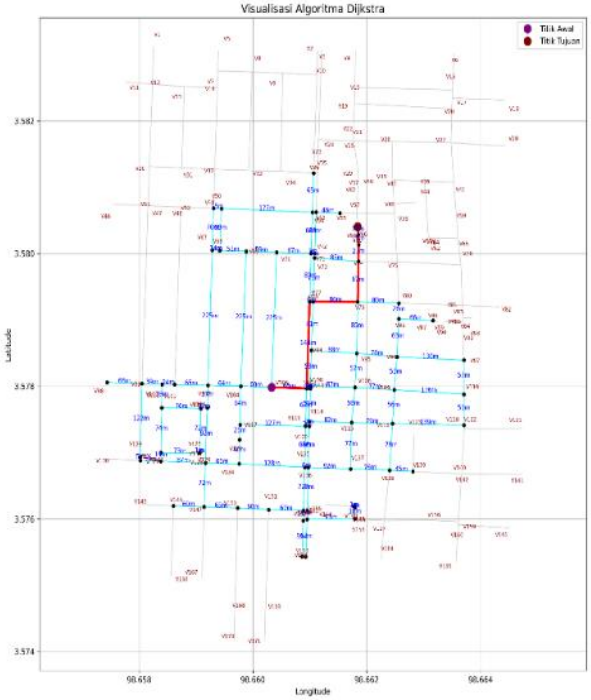
a. Pengujian pada Algoritma Dijkstra

Algoritma Dijkstra digunakan untuk mencari jalur terpendek dari satu titik ke titik lain dalam graf berbobot positif. Rumus utamanya adalah: d(u) = min(d(u), d(v) + w(v,u)), di mana:

1. d(u) = jarak minimum dari titik awal ke *node* u
2. d(v) = jarak minimum ke tetangga v
3. w(v,u) = bobot dari v ke u

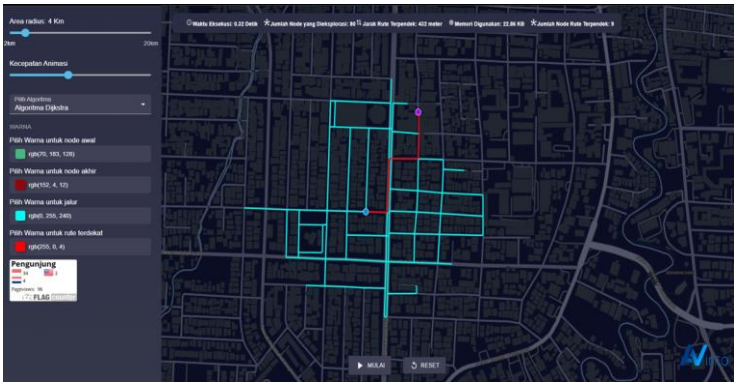
Tabel 3. proses perhitungan Dijkstra pada graf kecil

No	*Node* Saat Ini	Jarak dari Awal (Meter)	*Node* Tetangga yang akan Dicek	*Node* Sebelumnya	Jumlah *Node* Sebelumnya
1	V105	0	V70 (225.1 m), V104 (60.0 m), V107 (68.6 m)	V105	1
2	V104	60.04	V117 (64.1 m), V69 (225.1 m), V102 (64.2 m), V105 (60.0 m)	V105 -> V104	2
3	V107	68.65	V105 (68.6 m), V119 (62.8 m), V77 (144.7 m), V108 (6.8 m)	V105 -> V107	2
..	..	..	..	..	..
80	V58	432.59	V57 (21.9 m), V60 (13.8 m)	V105 -> V107 -> V77 -> V78 -> V79 -> V74 -> V66 -> V60 -> V58	9



Gambar 7. Graf hasil pencarian rute terpendek algoritma dijkstra skala kecil

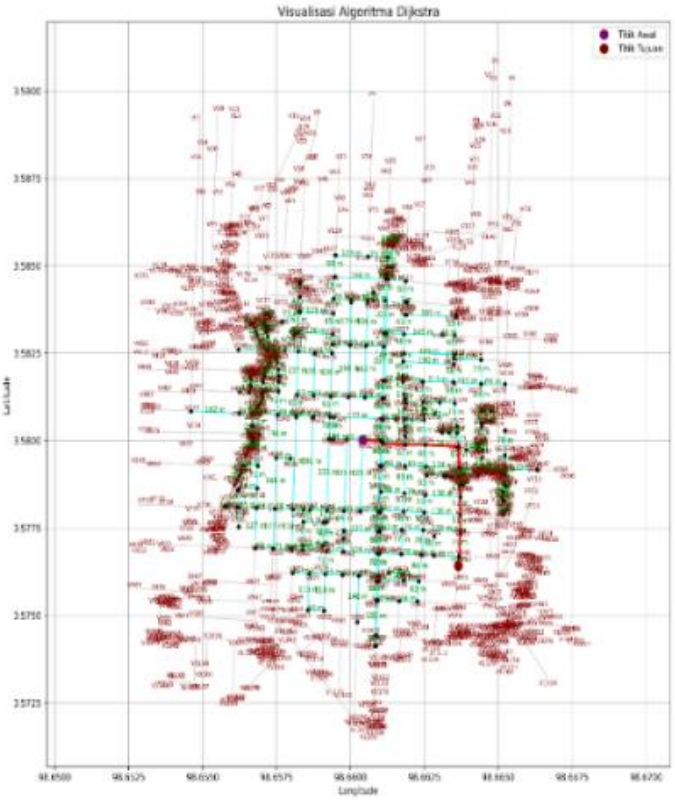
JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



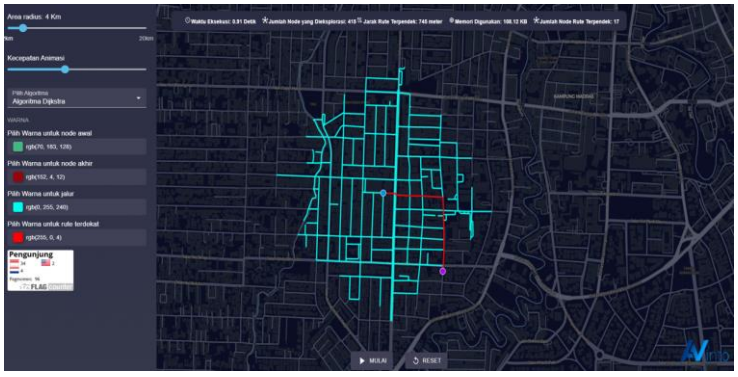
Gambar 8. Hasil pencarian rute terpendek algoritma dijkstra skala kecil dengan sistem

Tabel 5. Hasil pengujian algoritma Dijkstra skala kecil

Waktu Eksekusi	Jumlah *Node* yang Dieksplorasi	Jarak Rute Terpendek	Memori Digunakan	Jumlah *Node* Rute Terpendek
0.32 Detik	80	432 Meter	22.86 KB	9



Gambar 9. Graf hasil pencarian rute terpendek algoritma dijkstra skala besar



Gambar 10. Hasil pencarian rute terpendek algoritma dijkstra skala besar dengan sistem

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Tabel 5. Hasil pengujian algoritma Dijkstra skala besar

Waktu Eksekusi	Jumlah *Node* yang Dieksplorasi	Jarak Rute Terpendek	Memori Digunakan	Jumlah *Node* Rute Terpendek
0.91 Detik	415	745 Meter	108.12 KB	17

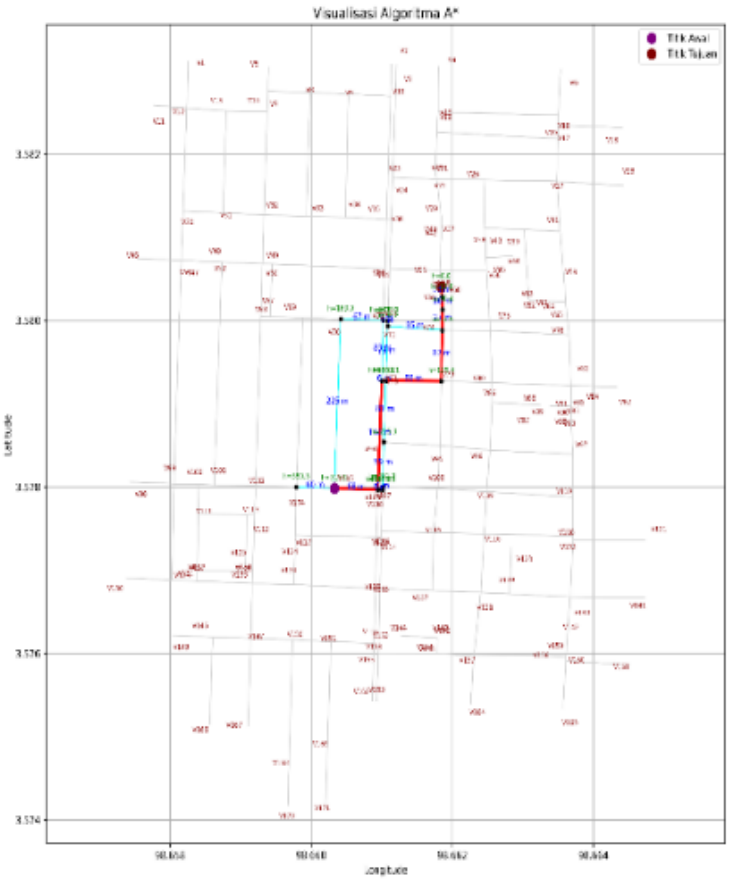
b. Pengujian pada Algoritma A*

Algoritma A* merupakan metode pencarian jalur yang menggabungkan pendekatan algoritma Dijkstra dengan fungsi heuristik untuk mempercepat pencarian ke tujuan. A* bekerja dengan mengevaluasi setiap simpul berdasarkan fungsi biaya total:

1. g(n): Jarak dari titik awal ke *node* saat ini.
2. h(n): Estimasi jarak dari *node* saat ini ke titik tujuan (menggunakan jarak garis lurus).
3. f(n) = g(n) + h(n): Total estimasi biaya dari awal sampai tujuan melalui *node* ini.

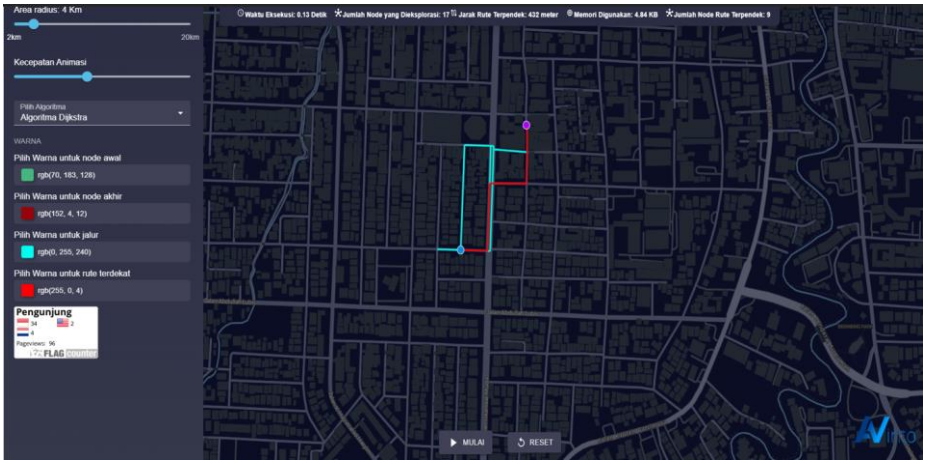
Tabel 6. Proses lengkap dan hasil perhitungan algoritma A*

No	*Node* Saat Ini	Jarak dari Awal (Meter)	Heuristik ke Tujuan (Meter)	*Node* Tetangga yang akan Dicek	*Node* Sebelumnya	Jumlah *Node* Sebelumnya
1	V105	0	315.95	V70 (225.1 m), V104 (60.0 m), V107 (68.6 m)	V105	1
2	V107	68.65	287.3	V105 (68.6 m), V119 (62.8 m), V77 (144.7 m), V108 (6.8 m)	V105 -> V107	2
3	V108	75.5	285.26	V107 (6.8 m), V103 (4.2 m), V114 (55.2 m)	V105 -> V107 -> V108	3
..	..	…	..	..	...	..
17	V58	432.59	0	V57 (21.9 m), V60 (13.8 m)	V105 -> V107 -> V77 -> V78 -> .. -> V58	9



Gambar 11. Graf hasil pencarian rute terpendek algoritma A* skala kecil

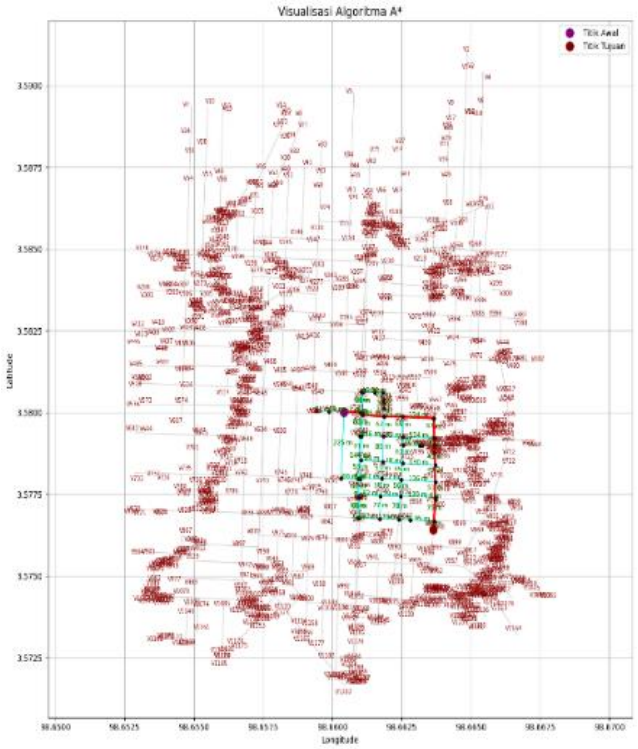
JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



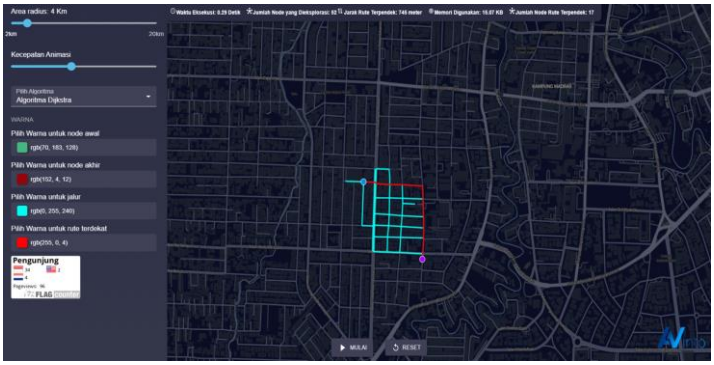
Gambar 12. Hasil pencarian rute terpendek algoritma A* skala kecil dengan sistem

Tabel 9. Hasil pengujian algoritma A* skala kecil

Waktu Eksekusi	Jumlah *Node* yang Dieksplorasi	Jarak Rute Terpendek	Memori Digunakan	Jumlah *Node* Rute Terpendek
0.13 Detik	17	432 Meter	4.84 KB	9



Gambar 13. Graf hasil pencarian rute terpendek algoritma A* skala besar



Gambar 14. Hasil pencarian rute terpendek algoritma A* skala besar dengan sistem

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Tabel 7. Hasil pengujian algoritma A* skala besar

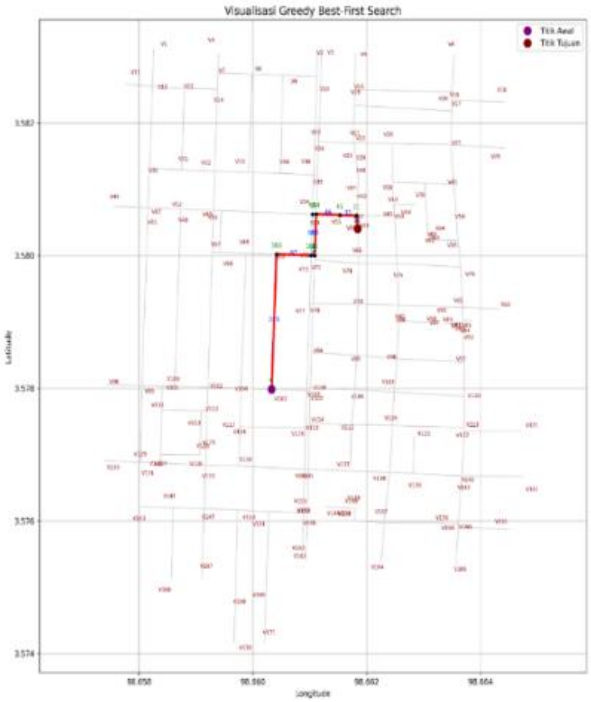
Waktu Eksekusi	Jumlah *Node* yang Dieksplorasi	Jarak Rute Terpendek	Memori Digunakan	Jumlah *Node* Rute Terpendek
0.29 Detik	52	745 Meter	15.07 KB	17

c. Pengujian Algoritma Greedy

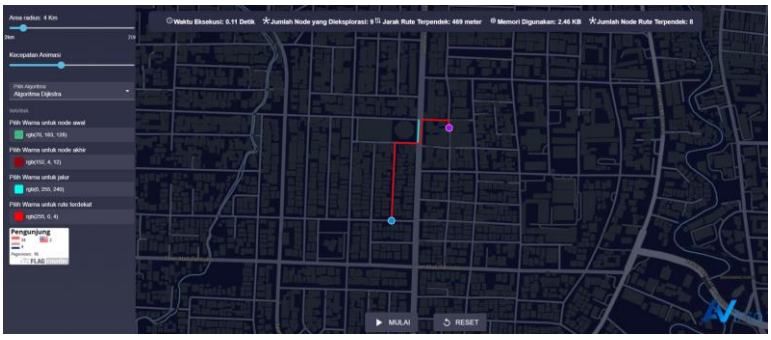
Algoritma Greedy BFS adalah metode pencarian jalur yang mengandalkan pendekatan heuristik untuk memilih jalur tercepat menuju tujuan berdasarkan estimasi jarak terdekat dari simpul saat ini ke simpul tujuan [18]. Pada setiap langkah, algoritma hanya mempertimbangkan nilai heuristik $h(n)$, tanpa memperhitungkan biaya dari titik awal $g(n)$, sehingga fungsi evaluasinya menjadi f(n) = h(n). Berbeda dengan A* yang menggunakan f(n) = g(n) + h(n) (jarak dari start + heuristik), Greedy hanya fokus pada seberapa dekat *node* ke tujuan secara langsung, tanpa mempertimbangkan jarak yang telah ditempuh sebelumnya.

Tabel 8. Proses lengkap dan hasil perhitungan algoritma Greedy

No	*Node* Saat Ini	Estimasi Heuristik (h)	*Node* Sebelumnya	Jumlah *Node* Sebelumnya	Jalur yang Ditempuh
1	V105	0		0	V105
2	V70	163.89	V105	1	V105 -> V70
3	V71	101.29	V105 -> V70	2	V105 -> V70 -> V71
..	..	..	..	..	..
9	V58	0	V105 -> V70 -> V71 -> V72 -> V53 -> V55 -> V57	7	V105 -> V70 -> V71 -> V72 -> V53 -> V55 -> V57 -> V58



Gambar 15. Graf hasil pencarian rute terpendek algoritma Greedy skala kecil

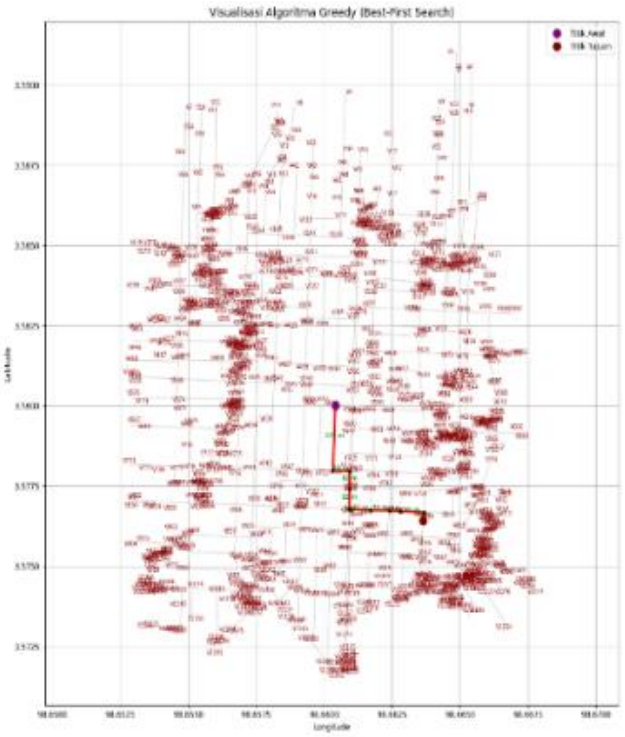


Gambar 16. Hasil pencarian rute terpendek algoritma Greedy skala kecil dengan sistem

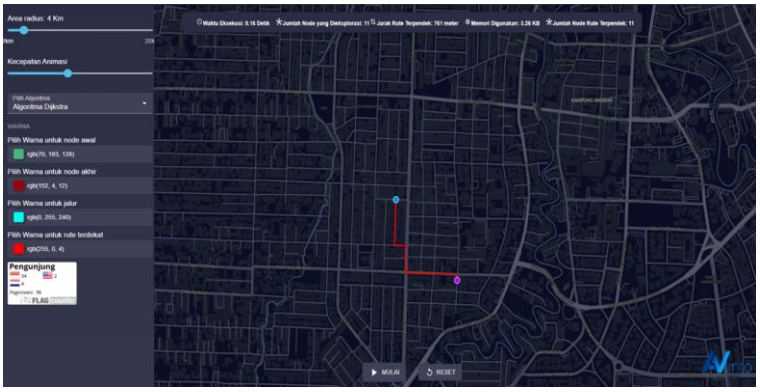
JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Tabel 9. Hasil pengujian algoritma Greedy skala kecil

Waktu Eksekusi	**Jumlah *Node* yang Dieksplorasi**	**Jarak Rute Terpendek**	**Memori Digunakan**	**Jumlah *Node* Rute Terpendek**
0.11 Detik	9	469 Meter	2.46 KB	8



Gambar 17. Graf hasil pencarian rute terpendek algoritma Greedy skala besar



Gambar 18. Hasil pencarian rute terpendek algoritma Greedy skala besar dengan sistem

Tabel 10. Hasil pengujian algoritma Greedy skala kecil

Waktu Eksekusi	**Jumlah *Node* yang Dieksplorasi**	**Jarak Rute Terpendek**	**Memori Digunakan**	**Jumlah *Node* Rute Terpendek**
0.16 Detik	11	761 Meter	3.26 KB	11

3.4 Perbandingan Efisiensi Ketiga Algoritma Berdasarkan Parameter

a. Waktu Eksekusi

Waktu eksekusi algoritma diukur berdasarkan selisih antara saat algoritma mulai dijalankan hingga rute terpendek berhasil ditemukan. Pengukuran dilakukan menggunakan fungsi javascript *performance.now()* dengan satuan milidetik dan presisi tinggi. Proses dimulai setelah pemanggilan fungsi *start*(), dan dihentikan ketika simpul tujuan tercapai serta jalur optimal berhasil dilacak. Nilai ini disimpan sebagai salah satu metrik visualisasi sistem [19]. Karena sistem dijalankan di lingkungan web berbasis JavaScript dengan animasi melalui *requestAnimationFrame*, waktu eksekusi dipengaruhi oleh beban prosesor, rendering frame, aktivitas browser, dan *garbage collection*. Untuk menjaga konsistensi, kecepatan animasi diatur sedang agar seimbang antara akurasi visual dan ketepatan waktu. Hasil dapat dilihat pada tabel dibawah ini.

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Tabel 11. Hasil perbandingan waktu eksekusi

Algoritma	Waktu Eksekusi (Graf Kecil)	Waktu Eksekusi (Graf Besar)
Dijkstra	0.32 detik	0.91 detik
A*	0.13 detik	0.29 detik
Greedy	0.17 detik	0.16 detik

b. Jumlah *Node* yang Dieksplorasi

Jumlah *node* yang dieksplorasi menunjukkan efisiensi pencarian algoritma. Semakin sedikit *node* yang dieksplorasi untuk mencapai tujuan, semakin efisien algoritma tersebut. A* secara konsisten mengeksplorasi lebih sedikit *node* dibanding Dijkstra karena memanfaatkan heuristik dalam pencarian [20]. Hasil dapat dilihat pada tabel dibawah ini.

Tabel 12. Hasil perbandingan jumlah *node* yang dieksplorasi

Algoritma	*Node* Dieksplorasi (Graf Kecil)	*Node* Dieksplorasi (Graf Besar)
Dijkstra	80	415
A*	17	52
Greedy	9	11

c. Jarak Rute Terpendek

Jarak rute terpendek menjadi tolak ukur keoptimalan jalur yang ditemukan oleh algoritma. Baik Dijkstra maupun A* mampu menemukan rute dengan panjang yang sama, karena keduanya bersifat optimal[12]. Sebaliknya, Greedy sering kali menghasilkan jalur yang lebih panjang karena hanya mempertimbangkan estimasi jarak ke tujuan tanpa menghitung total jarak secara akumulatif. Hasil dapat dilihat pada tabel dibawah ini.

Tabel 13. Hasil perbandingan jarak rute terpendek

Algoritma	Jarak Terpendek (Graf Kecil)	Jarak Terpendek (Graf Besar)
Dijkstra	432 meter	745 meter
A*	432 meter	745 meter
Greedy	469 meter	761 meter

d. Memori yang Digunakan

Parameter ini ditentukan berdasarkan estimasi total penggunaan memori (RAM) oleh seluruh simpul yang dieksplorasi selama proses pencarian rute. Perhitungan difokuskan pada struktur data *processed Nodes*, yakni kumpulan *node* yang telah diproses oleh algoritma pencarian jalur seperti Dijkstra, A*, dan Greedy [20]. Estimasi dilakukan melalui *estimateMemory(nodes)* yang menghitung ukuran byte tiap komponen *node*. Setiap *node* diasumsikan menggunakan 64 byte dasar, ditambah 16 byte per elemen *neighbors*, 32 byte per edge, dan 8–32 byte untuk atribut seperti *referer*, *parent*, *distanceFromStart, f, g,* dan *h*. Total byte dikonversi ke satuan terbaca (B/KB/MB) oleh *formatMemory(bytes)* dan ditampilkan sebagai metrik efisiensi. Hasil menunjukkan algoritma dengan eksplorasi simpul lebih sedikit atau struktur data ringan lebih hemat memori. Hasil dapat dilihat pada tabel dibawah ini.

Tabel 14. Hasil perbandingan jumlah memori yang digunakan

Algoritma	Memori (Graf Kecil)	Memori (Graf Besar)
Dijkstra	22.86 KB	108.12 KB
A*	4.84 KB	15.07 KB
Greedy	2.46 KB	3.26 KB

e. Jumlah *Node* pada Rute Terpendek

Jumlah simpul dalam rute terpendek menggambarkan seberapa kompleks jalur yang ditemukan. Jalur dengan lebih sedikit simpul biasanya lebih langsung dan efisien, meskipun tetap tergantung pada kondisi topologi jaringan jalan. Dijkstra dan A* memiliki jumlah simpul rute yang sama karena jalur yang ditemukan identik, sedangkan Greedy menghasilkan simpul rute lebih sedikit namun tidak selalu optimal [19]. Hasil dapat dilihat pada tabel dibawah ini.

Tabel 15. Hasil perbandingan jumlah *node* pada rute terpendek

Algoritma	Jumlah *Node* Rute (Graf Kecil)	Jumlah *Node* Rute (Graf Besar)
Dijkstra	9	17
A*	9	17
Greedy	8	11

F. Ukuran Skala (Skalabilitas)

Skalabilitas mengacu pada bagaimana algoritma berperilaku saat ukuran graf meningkat. A* menunjukkan performa paling stabil, tetap efisien meskipun skala graf meningkat drastis. Dijkstra mengalami penurunan performa karena eksplorasi yang meningkat secara signifikan. Greedy tetap cepat, tetapi akurasi jalurnya semakin tidak dapat

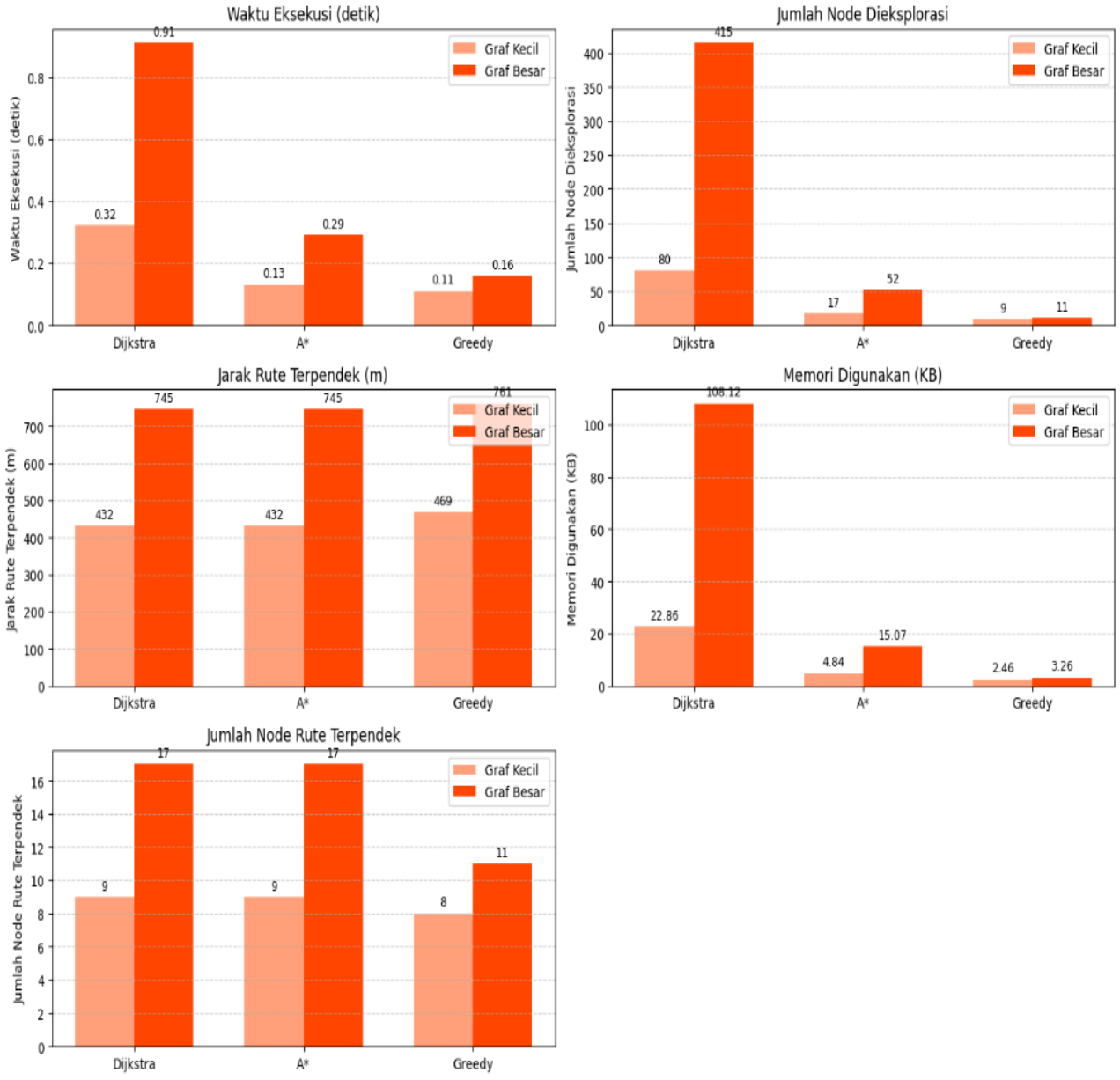
JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

diandalkan pada skala besar [20]. Perbandingan graf kecil dan besar menunjukkan bahwa A* memiliki peningkatan waktu dan memori yang proporsional, sementara Dijkstra meningkat lebih tajam. Ini menegaskan bahwa A* paling skalabel di antara ketiganya. Ketiga algoritma mampu menyelesaikan perhitungan di graf kecil maupun besar. Namun dari segi skalabilitas:

1. A* memiliki performa paling stabil, dengan peningkatan waktu dan memori yang masih efisien ketika skala diperbesar.
2. Dijkstra mengalami lonjakan eksplorasi dan penggunaan memori yang signifikan pada graf besar.
3. Greedy tetap cepat, namun kualitas hasil rute terpendek menurun saat graf semakin kompleks.

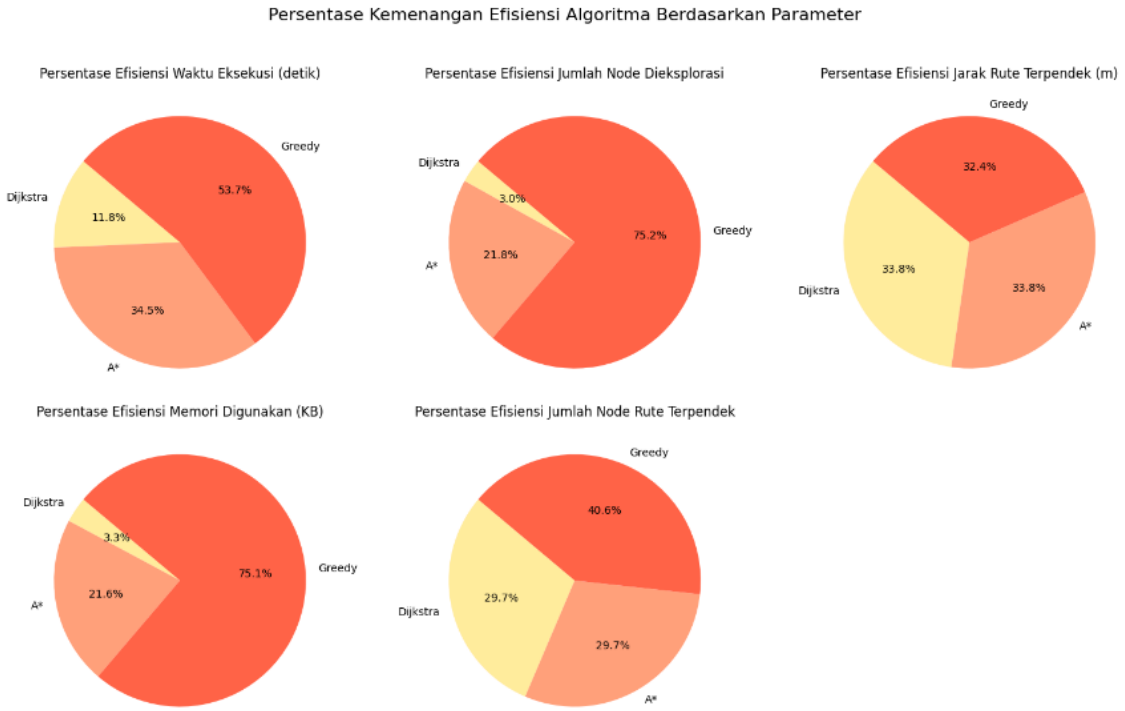
Tabel 16. Hasil perbandingan efisiensi pada setiap algortima berdasarkan parameter

No	Algoritma	Waktu Eksekusi	Jumlah *Node* yang Dieksplorasi	Jarak Rute Terpendek	Memori Di Gunakan	Jumlah *Node* Rute Terpendek	Ukuran Skala graf
1	Dijkstra	0.32 Detik	80	432 Meter	22.86 KB	9	Kecil
2	A*	0.13 Detik	17	432 Meter	4.84 KB	9	Kecil
3	Greedy	0.11 Detik	9	469 Meter	2.46 KB	8	Kecil
4	Dijkstra	0.91 Detik	415	745 Meter	108.12 KB	17	Besar
5	A*	0.29 Detik	52	745 Meter	15.07 KB	17	Besar
6	Greedy	0.16 Detik	11	761 Meter	3.26 KB	11	Besar



Gambar 19. Perbandingan keseluruhan parameter pada pengujian algoritma

JURIKOM (Jurnal Riset Komputer), Vol. 12 No. 4, Agustus 2025
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v12i4.8869
Hal 539-554
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 20. Presentase keunggulan algoritma berdasarkan parameternya

Tabel 17. Kelebihan dan kekurangan masing masing algoritma

Algoritma	Kelebihan	Kekurangan	Cocok Untuk
Dijkstra	Akurat, selalu temukan rute optimal; tidak butuh heuristik.	Lambat di graf besar; eksplorasi dan memori tinggi.	Graf kecil atau kasus yang butuh akurasi mutlak.
A*	Cepat, efisien, optimal; gabungkan Dijkstra + heuristik.	Bergantung pada kualitas heuristik; sedikit lebih berat.	Graf besar/kompleks; peta nyata seperti OpenStreetMap.
Greedy	Sangat cepat, ringan, eksplorasi dan memori minim.	Tidak selalu optimal; rawan jebakan lokal.	Real-time, graf besar, perangkat terbatas (IoT, mobile).

3.5 Analisis Perbandingan dengan Penelitian Sebelumnya

Penelitian ini tidak hanya menyajikan hasil numerik, tetapi juga menghadirkan visualisasi interaktif proses eksplorasi rute oleh algoritma, sesuatu yang belum dijelaskan secara rinci pada penelitian sebelumnya. Contohnya, Lakutu et al. hanya membandingkan hasil numerik Dijkstra dan A* berdasarkan waktu dan jarak rute pada dataset offline tanpa menampilkan eksplorasi spasial tiap node. [5]. Melalui simulasi visual berbasis web, pengguna dapat mengamati langsung animasi eksplorasi simpul dan jalur, sehingga memahami perilaku algoritma secara lebih mendalam. Penelitian ini juga menguji tiga algoritma sekaligus (A*, Dijkstra, dan Greedy) menggunakan enam parameter evaluasi pada dua skala graf. [18] hanya membandingkan dua algoritma dan menggunakan satu parameter. Penelitian [12] membahas Greedy BFS secara teoritis, namun belum diuji dalam konteks visualisasi graf spasial. Dengan demikian, penelitian ini memberikan kontribusi dalam hal kelengkapan pengujian parameter, pendekatan visualisasi, serta evaluasi skalabilitas secara langsung.

4. KESIMPULAN

Penelitian ini berhasil mengembangkan sistem visualisasi berbasis web untuk membandingkan efisiensi algoritma A*, Greedy, dan Dijkstra dalam pencarian rute terpendek menggunakan data OpenStreetMap. Sistem menampilkan proses pencarian secara interaktif dan dievaluasi berdasarkan enam parameter efisiensi. Hasil menunjukkan bahwa algoritma A* memberikan performa terbaik secara keseluruhan, unggul dalam waktu eksekusi, eksplorasi *node*, dan penggunaan memori, terutama pada graf besar. Dijkstra memberikan hasil akurat namun kurang efisien dalam eksplorasi dan memori. Sementara Greedy paling ringan dan cepat, tetapi rutenya kurang optimal. Secara keseluruhan, A* menempati peringkat tertinggi berdasarkan kombinasi seluruh parameter.

REFERENCES

[1] J. Y. Pratama, "Analisis Perbandingan Algoritma Dijkstra dan A-Star dalam Menentukan Rute Terpendek," *JIMUJurnal Ilm. Multidisipliner*, vol. 2, no. 03, pp. 668–682, 2024, doi: 10.70294/jimu.v2i03.423.

[2] N. R. I, M. Z. Hilmi, E. Proborini, and F. A. Husain, "Pencarian Rute Terpendek dengan Algoritma Dijkstra Shortest Route Search with Dijkstra ' s algorithm Shortest Route Search with Dijkstra ' s Algorithm Shortest Route Search with Dijkstra ' s algorithm," vol. 6, no. 1, pp. 50–57, 2025.
[3] M. Ikhsan and Haris, "Ekowisata Rammang-Rammang Sebagai Laboratorium Pembelajaran Kontekstual Geografi Di Kabupaten Maros," *Jambura Geo Educ. J.*, vol. 3, no. 2, pp. 43–51, 2022, doi: 10.34312/jgej.v3i2.15366.
[4] T. Dealva Arsyad, I. Agi Berutu, E. Keisha Silalahi, and D. Yandra Niska, "Implementasi Algoritma a* (a - Star) Untuk Optimasi Jalur Bus Listrik Di Kota Medan Dengan Visualisasi 2D," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 9, no. 4, pp. 6138–6143, 2025, doi: 10.36040/jati.v9i4.14001.
[5] N. F. Lakutu, S. L. Mahmud, M. R. Katili, and N. I. Yahya, "Algoritma Dijkstra dan Algoritma Greedy Untuk Optimasi Rute Pengiriman Barang Pada Kantor Pos Gorontalo," *Euler J. Ilm. Mat. Sains dan Teknol.*, vol. 11, no. 1, pp. 55–65, 2023, doi: 10.34312/euler.v11i1.18244.
[6] N. Sugianti, A. Mardhiyah, and N. R. Fadilah, "Komparasi Kinerja Algoritma BFS, Dijkstra, Greedy BFS, dan A* dalam Melakukan Pathfinding," *JISKA (Jurnal Inform. Sunan Kalijaga)*, vol. 5, no. 3, pp. 194–204, 2020, doi: 10.14421/jiska.2020.53-07.
[7] F. L. Tobing, F. A. T. Tobing, and Prayogo, "Analisis Perbandingan Algoritma Dfs, Bfs Dan Dijkstra Untuk Menentukan Rute Terpendek Pada Peta Geografis," *J. Widya*, vol. 3, no. 1, pp. 59–67, 2022, [Online]. Available: https://jurnal.amikwidyaloka.ac.id/index.php/awl
[8] A. Bonifasius Simbolon, D. Aulia Artika, Y. Christian Sitanggang, and P. Harliana, "Implementasi Algoritma Dijkstra Dalam Menganalisis Rute Terpendek Dan Efisiensi Jarak Tempuh Dari Stasiun Kereta Api Medan Menuju 6 Kampus Di Area Medan Estate," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 9, no. 1, pp. 162–168, 2025, doi: 10.36040/jati.v9i1.12155.
[9] E. Rahmawati and C. Agustina, "Implementasi Algoritma Greedy dan Djikstra untuk Efektifitas Rute Pariwisata Populer di Borobudur," *J. Teknol. Inf. dan Terap.*, vol. 8, no. 2, pp. 72–75, 2021, doi: 10.25047/jtit.v8i2.216.
[10] R. A. M. P. N. Hikmah, "Perbandingan Algoritma a Star Dan Algoritma Dijkstra Untuk Menentukan Rute Terpendek Kantor Pos Di Provinsi Lampung," *NBER Work. Pap.*, p. 39, 2023, [Online]. Available: http://www.nber.org/papers/w16019
[11] J. Dima, M. S. Hamzah, C. G. Tallo, and D. Y. A. Fallo, "Tinjauan Literatur tentang Pemanfaatan Algoritma Greedy untuk Pencarian Jalur Terpendek," vol. 7, no. 01, pp. 519–528, 2025.
[12] H. Sulaiman, Y. Yuliani, E. Fitri, N. Herlinawati, and S. Watmah, "Algoritma Dijkstra untuk Pendistribusian Carica Nida Food Wonosobo," *J. Sist. dan Teknol. Inf.*, vol. 8, no. 2, p. 203, 2020, doi: 10.26418/justin.v8i2.38223.
[13] A. Maghfirotul Inayah, N. Cintya Resti, and N. Fadilatul Ilmiyah, "Analisa Perbandingan Algoritma Floyd-Warshalldan Algoritma Dijkstrauntuk Penentuanrute Terdekat," *J. Ilm. Mat. Realis. (JI-MR*, vol. 4, no. 2, pp. 146–155, 2023.
[14] A. Dwi Rifka Setyawan, J. Dedy Irawan, and A. Faisol, "Rancang Bangun Aplikasi Penentuan Rute Terpendek Distribusi Jamu Tradisional Menggunakan Metode a*," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 7, no. 4, pp. 2218–2225, 2023, doi: 10.36040/jati.v7i4.7450.
[15] Alia, M. A. Irwansyah, and H. Novriando, "Aplikasi WebGis Fasilitas Umum Menggunakan Library Leaflet dan OpenStreetMap," *J. Sist. dan Teknol. Inf.*, vol. 9, no. 3, p. 334, 2021, doi: 10.26418/justin.v9i3.44442.
[16] H. Hendra and Y. F. Riti, "Perbandingan Algoritma Dijkstra Dan Floyd-Warshall Dalam Menentukan Rute Terpendek Stasiun Gubeng Menuju Wisata Surabaya," *JIKA (Jurnal Inform.*, vol. 6, no. 3, p. 297, 2022, doi: 10.31000/jika.v6i3.6528.
[17] R. Perayoga, P. Hendradi, and A. Setiawan, "Implementasi Algoritma Dijkstra Pada Pencarian Rute Terpendek Objek Wisata," *KLIK Kaji. Ilm. Inform. dan Komput.* , vol. 4, no. 3, pp. 1471–1482, 2023, doi: 10.30865/klik.v4i3.1495.
[18] J. S. Iskandar and Y. F. Riti, "Perbandingan Algoritma Greedy dan Algoritma Dijkstra dalam Pencarian Rute Terpendek dari Kabupaten Tuban ke Kota Surabaya," *J. PETIK*, vol. 8, no. 2, pp. 96–106, 2022.
[19] C. T. S. Garingging, S. L. Gaol, M. A. Lubis, F. T. Sianturi, B. M. Sembiring, and S. P. Sipayung, "Implementasi Algoritma Dijkstra dalam Menentukan Rute Terpendek dari Unika St. Thomas Medan ke Lapangan Merdeka," *J. Minfo Polgan*, vol. 14, no. 1, pp. 789–800, 2025, doi: 10.33395/jmp.v14i1.14892.
[20] S. Skiena, "The Algorithm Design Manual 3rd edition," in *Algorithms and applications*, vol. 42, C. Springer, Ed., Stony Brook University, 2020, p. 823. [Online]. Available: http://www.springer.com/series/3191%0Ahttp://www.ncbi.nlm.nih.gov/pubmed/20549881