SMASH: Journal of Social Management Sains and Health
E-ISSN: 3089-252X P-ISSN: 3089-2538
Vol. 3 No. 1 April 2026

Analisis Kebenaran dan Efisiensi Algoritma Dijkstra VS A-Star (A*) Dalam Penentuan Rute Tercepat ada Graf Berbobot

Sebastian Saut Marulitua Sinaga[1] Benedict Sandi Pangestu Rosa[2] Muhammad Rafli Wijaya[3] M Gali Almahdi[4] Adidtya Perdana[5]
Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Medan, Provinsi Sumatera Utara, Indonesia[1,2,3,4,5]
Email: sebastiansaut613@gmail.com[1] benedictsandi28@gmail.com[2]
rafliwijaya2005@gmail.com[3] muhammadgalialmahdi@gmail.com[4] adidtya@unimed.ac.id[5]

Abstrak

Studi ini menganalisis kebenaran dan efisiensi algoritma Dijkstra dan A* dalam menentukan jalur terpendek pada graf berbobot. Kebenaran dibuktikan secara teoritis menggunakan pendekatan invarian loop untuk Dijkstra dan sifat admissible pada heuristik A*, serta dikonfirmasi melalui eksperimen komputasi. Efisiensi dianalisis berdasarkan kompleksitas waktu dan pengujian empiris meliputi waktu eksekusi, jumlah simpul yang dikunjungi, dan ukuran antrian prioritas. Pengujian dilakukan pada tiga skenario graf berbobot dengan ukuran berbeda menggunakan metode Random Geometric Graph. Hasil menunjukkan bahwa kedua algoritma memiliki tingkat kebenaran 100% dalam seluruh pengujian. Dari segi efisiensi, algoritma A* menunjukkan performa yang lebih cepat dan mampu mengurangi jumlah eksplorasi simpul secara signifikan dibandingkan Dijkstra. Namun, A* memiliki penggunaan memori yang lebih tinggi pada graf berukuran besar. Penelitian ini menyimpulkan bahwa A* lebih unggul dalam efisiensi waktu ketika tersedia heuristik yang valid, sementara Dijkstra lebih stabil ketika heuristik tidak tersedia.

Kata Kunci: Algoritma Dijkstra, Algoritma A*, Graf Berbobot, Rute Terpendek, Efisiensi Algoritma

PENDAHULUAN

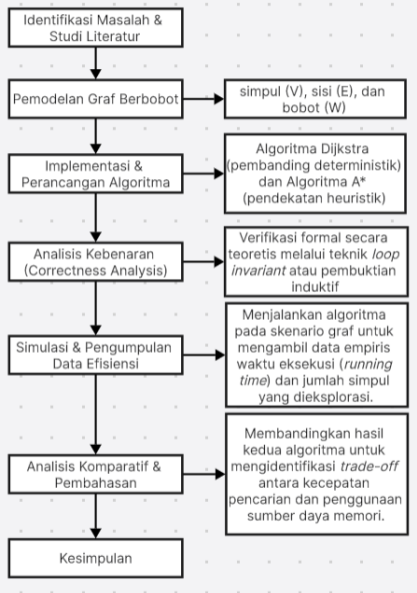
Permasalahan pencarian rute terpendek (*shortest path problem*) merupakan salah satu persoalan fundamental dalam optimasi ilmu komputer yang memiliki peran penting dalam meningkatkan efisiensi berbagai sistem modern, seperti transportasi, logistik, dan jaringan komunikasi. Seiring dengan pesatnya perkembangan teknologi informasi, kebutuhan akan pemodelan berbasis graf yang mampu merepresentasikan kondisi dunia nyata secara akurat menjadi semakin penting. Dalam struktur jaringan yang kompleks, penentuan rute paling optimal menjadi tantangan tersendiri, terutama ketika jumlah simpul besar dan kondisi sistem bersifat dinamis. Berbagai algoritma telah dikembangkan untuk menyelesaikan permasalahan tersebut, di antaranya algoritma Dijkstra dan A* yang paling banyak digunakan karena telah terbukti andal dalam berbagai penerapan. Algoritma Dijkstra merupakan metode deterministik klasik yang menjamin diperolehnya solusi optimal melalui proses relaksasi sisi secara menyeluruh pada graf. Namun demikian, meskipun memiliki jaminan kebenaran yang kuat, algoritma ini cenderung kurang efisien pada graf berskala besar karena proses pencariannya tidak memiliki arah yang spesifik. Untuk mengatasi keterbatasan tersebut, algoritma A* memanfaatkan fungsi heuristik yang mampu mengarahkan proses pencarian menuju simpul tujuan secara lebih terarah. Dengan memanfaatkan informasi tambahan tersebut, A* dapat mengurangi ruang pencarian dan meningkatkan efisiensi waktu, terutama pada graf yang kompleks.

Sejumlah penelitian sebelumnya umumnya lebih banyak berfokus pada perbandingan kinerja algoritma dari sisi waktu eksekusi. Namun demikian, masih terdapat keterbatasan penelitian yang menggabungkan analisis kebenaran secara formal dengan evaluasi kinerja

secara empiris dalam satu kajian yang terpadu. Sebagian besar penelitian hanya menitikberatkan pada hasil simulasi tanpa disertai pembuktian teoritis yang kuat terhadap kebenaran algoritma. Hal ini menimbulkan celah dalam memastikan bahwa solusi yang dihasilkan tidak hanya efisien, tetapi juga benar secara matematis dalam berbagai kondisi. Berdasarkan hal tersebut, penelitian ini bertujuan untuk menganalisis kebenaran dan efisiensi algoritma Dijkstra dan A* dalam menentukan rute terpendek pada graf berbobot. Analisis kebenaran dilakukan menggunakan pendekatan loop invariant pada algoritma Dijkstra serta sifat admissible pada heuristik A*. Sementara itu, analisis efisiensi dilakukan melalui kajian kompleksitas teoritis dan pengujian empiris pada beberapa skenario graf. Penelitian ini diharapkan dapat memberikan pemahaman yang lebih komprehensif mengenai keseimbangan antara akurasi, kinerja, dan penggunaan sumber daya dalam algoritma pencarian rute terpendek.

METODE PENELITIAN

Dalam penelitian ini, metode yang digunakan dirancang untuk memberikan gambaran yang sistematis dan terukur dalam mengevaluasi performa algoritma pencarian rute terpendek. Pendekatan yang dipilih menekankan pada keterpaduan antara analisis teoritis dan pengujian empiris, sehingga hasil yang diperoleh tidak hanya bersifat deskriptif, tetapi juga dapat dipertanggungjawabkan secara ilmiah. Penelitian ini menggunakan pendekatan eksperimental untuk menganalisis kebenaran dan efisiensi algoritma Dijkstra dan A* dalam menentukan rute terpendek pada graf berbobot. Proses penelitian dilakukan melalui tahapan perancangan graf uji, implementasi algoritma, serta evaluasi kinerja berdasarkan metrik tertentu.



Gambar 1. Alur Penelitian

1. Pemodelan Graf Berbobot. Masalah pencarian rute ditransformasikan ke dalam model matematis Graf Berbobot: *G = (V, E)* Simpul (V) merepresentasikan persimpangan atau titik lokasi, sedangkan sisi (E) merepresentasikan jalur penghubung. Setiap sisi diberikan bobot (W) berupa jarak fisik antar simpul. Model ini mengasumsikan bobot sisi bernilai non-negatif guna memenuhi syarat dasar operasional algoritma Dijkstra.
2. Implementasi dan Perancangan Algoritma. Penelitian ini mengimplementasikan dua algoritma utama dalam lingkungan komputasi yang seragam untuk menjaga validitas data:

a. Algoritma Dijkstra. Algoritma Dijkstra dirancang sebagai metode deterministik yang menjamin kebenaran solusi pada graf dengan bobot non-negatif. Perancangannya berfokus pada prosedur relaksasi sisi secara bertahap untuk memperbarui jarak minimum dari simpul sumber ke setiap simpul lainnya. Untuk mengoptimalkan efisiensi waktu komputasi, implementasi ini menggunakan struktur data *Priority Queue* berbasis *min-heap*, yang memungkinkan pengambilan simpul dengan biaya terendah dalam waktu logaritmik.
Pseudocode:
Procedure Dijkstra(Graph, Source):
Create a priority queue
PQ Distance[Source] = 0
For each vertex v in Graph:
If v != Source: Distance[v] = Infinity
Predecessor[v] = Null
PQ.Insert(v, Distance[v])
While PQ is not empty:
u = PQ.ExtractMin()
For each neighbor v of u:
alt = Distance[u] + Weight(u, v)
If alt < Distance[v]:
Distance[v] = alt
Predecessor[v] = u
PQ.DecreasePriority(v, alt)
Return Distance, Predecessor

b. Algoritma A*. Algoritma A* dirancang dengan mengintegrasikan fungsi evaluasi biaya total yang direpresentasikan melalui persamaan:
f(n) = g(n) + h(n)
Di mana g(n) adalah biaya aktual dari simpul sumber ke simpul saat ini, dan h(n) adalah fungsi heuristik yang memperkirakan biaya tersisa menuju target. Berbeda dengan Dijkstra, A* dirancang untuk memangkas ruang pencarian secara signifikan dengan hanya mengeksplorasi simpul-simpul yang secara matematis lebih mendekati tujuan.
Pseudocode:
Procedure AStar(Graph, Source, Target):
OpenList = PriorityQueue(Source)
ClosedList = EmptySet
gScore[Source] = 0
fScore[Source] = h(Source)
While OpenList is not empty:
u = OpenList.ExtractMin()
If u == Target: Return Path
ClosedList.Add(u)
For each neighbor v of u:
If v in ClosedList: Continue
temp_gScore = gScore[u] + Weight(u, v)
If temp_gScore < gScore[v] or v not in OpenList:
Predecessor[v] = u
gScore[v] = temp_gScore
fScore[v] = gScore[v] + h(v)

If v not in OpenList: OpenList.Insert(v, fScore[v])
Return Failure
Fungsi heuristik h(n) dalam penelitian ini dirancang menggunakan *Euclidean Distance* yang bersifat *admissible*, artinya nilai estimasi tidak pernah melebihi jarak asli ke tujuan sehingga menjamin optimallitas hasil akhir.

3. Analisis Kebenaran. Analisis kebenaran bertujuan untuk memberikan jaminan teoretis bahwa algoritma Dijkstra dan A* akan selalu menghasilkan jalur terpendek (solusi optimal) untuk setiap input graf berbobot non-negatif yang valid. Penekanan pada aspek ini sangat penting karena implementasi kode nyata sering kali menghadapi risiko kesalahan logika yang tidak terlihat dalam pengujian sederhana.
   a. Kebenaran Algoritma Dijkstra. Pembuktian kebenaran algoritma Dijkstra dilakukan secara formal untuk menjamin bahwa algoritma selalu menghasilkan jalur terpendek dari simpul sumber ke semua simpul lainnya pada graf berbobot non-negatif.
      1) Pre-condition (Kondisi Awal): Diberikan sebuah graf berbobot G = (V, E) di mana setiap sisi memiliki bobot non-negatif ($w(u, v) \geq 0$). Terdapat sebuah simpul sumber $s \in V$. Jarak awal $d[s] = 0$ dan $d[v] = \infty$ untuk semua $v \neq s$.
      2) Post-condition (Kondisi Akhir): Setelah algoritma selesai dieksekusi, untuk setiap simpul $v \geq V$, nilai d[v] berisi total bobot jalur terpendek yang sebenarnya dari s ke v.
      3) Loop Invariant (Invarian Perulangan): Misalkan S adalah himpunan simpul yang telah diekstraksi dari *priority queue* (simpul yang sudah dikunjungi/final). Invarian yang dipertahankan adalah: pada setiap awal iterasi *loop*, untuk setiap simpul $v \in S$, nilai d[v] adalah jarak terpendek yang optimal dari s ke v.
      4) Pembuktian induksi matematika:
         a) Basis Induksi**:** Sebelum iterasi pertama, $S = \emptyset$, sehingga invarian bernilai benar secara trivial. Saat iterasi pertama, s diekstraksi, $d[s] = 0$, yang merupakan jarak terpendek ke dirinya sendiri.
         b) Langkah Induksi: Asumsikan invarian benar untuk himpunan S. Misalkan u adalah simpul berikutnya yang diekstraksi dari *priority queue* ($u \notin S$) dengan nilai d[u] terkecil. Kita harus membuktikan bahwa d[u] adalah jarak terpendek sejati. Jika terdapat jalur lain yang lebih pendek ke u, jalur tersebut harus melalui suatu simpul $y \notin S$. Karena semua bobot sisi non-negatif ($w \geq 0$) dan y masih berada di antrean, maka $d[y] \geq d[u]$. Akibatnya, total jarak melalui y tidak mungkin lebih kecil dari d[u]. Hal ini membuktikan bahwa saat u ditambahkan ke S, d[u] sudah optimal.
         c) Analisis Penghentian (Termination): Algoritma mengambil satu simpul dari *priority queue* pada setiap iterasi dan tidak pernah memasukkannya kembali ke dalam S. Karena jumlah simpul |V| terbatas (berhingga), *loop* pasti akan berhenti setelah maksimal |V| iterasi.
   b. Kebenaran dan Optimalitas Algoritma A*. Berbeda dengan Dijkstra yang murni deterministik, kebenaran algoritma heuristik A* membutuhkan pembuktian tambahan terkait sifat fungsi evaluasi $f(n) = g(n) + h(n)$ yang memandu arah pencariannya.
      1) Pre-condition: Diberikan graf berbobot G = (V, E) dengan bobot non-negatif, simpul awal s, dan kumpulan simpul target Gt. Tersedia fungsi heuristik h(n) yang memperkirakan biaya dari simpul n ke target.
      2) Post-condition: Algoritma mengembalikan jalur terpendek dari s ke target, atau mengembalikan status gagal jika target tidak dapat dijangkau.
      3) Admissibility (Keandalan Heuristik): Agar kebenaran terjamin, h(n) harus bersifat *admissible*, artinya nilai heuristik tidak pernah melebihkan biaya sebenarnya. Secara

matematis disimbolkan dengan $0 \leq h(n) \leq h^*(n)$, dimana $h^*(n)$ adalah biaya aktual terpendek dari n ke target.

4) Pembuktian Optimalitas (Kontradiksi): Asumsikan A* mengembalikan rute yang tidak optimal dengan biaya $C > C^*$, di mana C^* adalah biaya optimal sejati. Misalkan simpul n adalah simpul yang berada pada rute optimal sejati dan saat ini masih berada di *Open List* (belum dieksplorasi). Karena heuristik bersifat *admissible*, maka $f(n) = g(n) + h(n) \leq C^*$.

   Jika algoritma memilih rute suboptimal, ia pasti mengeksekusi simpul target dengan biaya C. Namun, karena $f(n) \leq C^* < C$, *priority queue* secara logis akan selalu mengekstrak simpul n terlebih dahulu sebelum menyelesaikan rute suboptimal C. Terjadi kontradiksi. Pembuktian ini menjamin bahwa A* tidak akan pernah memilih solusi suboptimal selama heuristiknya *admissible.*

5) Analisis Penghentian (Termination): Pada graf dengan jumlah simpul berhingga dan bobot sisi minimal bernilai positif ($\varepsilon > 0$), nilai f(n) akan terus bertambah seiring bertambahnya kedalaman pencarian. Karena A* hanya mengekspansi simpul dengan $f(n) \leq C^*$, dan jumlah simpul dengan kondisi tersebut berhingga, algoritma dipastikan akan berhenti (*terminate*), baik dengan menemukan target maupun saat *Open List* kosong.

Selain analisis logika teoretis, verifikasi kebenaran juga dilakukan dengan memperhatikan aspek teknis seperti pencegahan *integer overflow* pada perhitungan biaya akumulatif dan pengelolaan status simpul dalam struktur data pendukung. Dengan menggabungkan bukti *loop invariant* pada Dijkstra dan syarat optimallitas pada A*, penelitian ini memastikan bahwa rute yang dihasilkan oleh kedua algoritma adalah valid dan dapat diandalkan untuk aplikasi navigasi.

4. Analisis Efisiensi Teoritis. Selain pembuktian kebenaran, analisis efisiensi secara teoritis diperlukan untuk memahami batas kemampuan dan perilaku kedua algoritma sebelum pengujian empiris dilakukan. Efisiensi suatu algoritma umumnya dikaji melalui dua dimensi utama, yaitu kompleksitas waktu yang mengukur pertumbuhan waktu komputasi terhadap ukuran input, dan kompleksitas ruang yang mengukur kebutuhan memori selama eksekusi berlangsung.
   a. Kompleksitas Waktu. Algoritma Dijkstra yang diimplementasikan dengan struktur data *priority queue* berbasis *min-heap* memiliki kompleksitas waktu sebesar $O((V + E) \log V)$, di mana V adalah jumlah simpul dan E adalah jumlah sisi dalam graf. Kompleksitas ini muncul karena setiap simpul diekstraksi dari *priority queue* tepat satu kali dengan biaya $O(\log V)$, dan setiap sisi direlaksasi paling banyak satu kali. Karena Dijkstra tidak memiliki mekanisme untuk membatasi arah pencariannya, algoritma ini pada dasarnya wajib memproses seluruh atau sebagian besar simpul dalam graf, menjadikan kompleksitasnya bergantung penuh pada ukuran total graf. Algoritma A* secara teoritis memiliki kompleksitas waktu $O(E \log V)$ pada kasus terbaik, yaitu ketika fungsi heuristik h(n) sangat akurat dalam memperkirakan biaya aktual ke tujuan. Namun kompleksitas A* sangat bergantung pada kualitas heuristik yang digunakan. Pada kasus terburuk, ketika $h(n) = 0$ untuk semua simpul (heuristik tidak informatif), A* tereduksi menjadi algoritma Dijkstra dengan kompleksitas yang setara. Sebaliknya, ketika heuristik bersifat *perfect*, yaitu $h(n) = h^*(n)$ untuk setiap simpul A* hanya perlu mengeksplorasi simpul-simpul yang berada tepat pada jalur optimal, sehingga kompleksitasnya mendekati $O(d \log d)$ dengan d adalah panjang jalur terpendek. Dalam penelitian ini, heuristik *Euclidean Distance* yang digunakan bersifat *admissible* namun tidak *perfect*, sehingga performa

aktualnya berada di antara kedua kasus ekstrem tersebut. Perbandingan kompleksitas waktu teoritis kedua algoritma dirangkum pada Tabel 1.

Tabel 1. Perbandingan Kompleksitas Teoritis Dijkstra dan A*

Aspek	Dijkstra	A* (kasus terbaik)	A* (kasus terburuk)
Kompleksitas Waktu	*O((V + E) log V)*	*O(E log V)*	*O((V + E) log V)*
Kompleksitas Ruang	*O(V + E)*	*O(b^d)*	*O(V + E)*
Arah Pencarian	Tidak terarah (exhaustive)	Terarah oleh h(n)	Tidak terarah (h=0)
Jaminan Optimal	Selalu (bobot ≥ 0)	Ya (h admissible)	Ya (h admissible)
Faktor Penentu Efisiensi	Densitas graf (V, E)	Kualitas heuristik h(n)	Densitas graf (V, E)

b. Kompleksitas Ruang. Dijkstra memiliki kompleksitas ruang O(V + E) yang bersifat deterministik, karena algoritma ini harus menyimpan jarak ke seluruh simpul, tabel *predecessor*, dan isi *priority queue* secara bersamaan. Ukuran memori yang dibutuhkan tumbuh secara linear dan dapat diprediksi seiring bertambahnya jumlah simpul dan sisi dalam graf. Kompleksitas ruang A* dinyatakan sebagai $O(b^d)$, di mana b adalah *branching factor* rata-rata (jumlah tetangga per simpul) dan d adalah kedalaman solusi (panjang jalur terpendek). Ini merupakan karakteristik kritis A* yang membedakannya dari Dijkstra: pada kasus terbaik A* hanya menyimpan simpul-simpul di sepanjang jalur dan cabang-cabang yang relevan, sehingga memorinya jauh lebih hemat. Namun pada kasus terburuk misalnya ketika heuristik tidak cukup informatif untuk memangkas cabang pencarian, A* dapat menyimpan hampir seluruh simpul yang pernah ditemui, menjadikan kebutuhan memorinya setara atau bahkan melebihi Dijkstra. Fenomena inilah yang akan diamati dan dikonfirmasi melalui metrik Max PQ pada eksperimen di bagian berikutnya.

5. Skenario dan Lingkungan Pengujian. Pengujian empiris pada penelitian ini dilakukan dengan menggunakan bahasa pemrograman Python. Library utama yang digunakan meliputi *heapq* untuk implementasi *priority queue* berbasis *min-heap*, NetworkX untuk pembangkitan dan pengelolaan struktur graf, serta *time.perf_counter()* untuk pengukuran waktu eksekusi dengan presisi tinggi. Untuk memperoleh data perbandingan yang representatif, penelitian ini merancang tiga skenario graf berbobot dengan karakteristik yang berbeda. Graf dibangkitkan menggunakan metode *Random Geometric Graph*, di mana setiap simpul ditempatkan secara acak pada ruang koordinat [0,1] × [0,1], dan dua simpul terhubung oleh sisi apabila jarak Euclidean antar keduanya tidak melebihi nilai radius yang ditentukan. Bobot setiap sisi ditetapkan sebagai jarak Euclidean antar simpul dikalikan dengan faktor 10, sehingga fungsi heuristik A* yang menggunakan rumus:
$h(n) = \sqrt{((x_n - x_{goal})^2 + (y_n - y_{goal})^2)} \times 10$
tetap bersifat *admissible*, karena nilai estimasi tidak pernah melebihi bobot sisi yang sebenarnya. Simpul asal dipilih sebagai simpul dengan nilai koordinat (x + y) terkecil (pojok kiri-bawah), sedangkan simpul tujuan dipilih sebagai simpul dengan nilai koordinat (x + y) terbesar (pojok kanan-atas), guna memaksimalkan jarak tempuh dan membuat perbandingan kedua algoritma lebih bermakna. Ketiga skenario yang dirancang disajikan pada Tabel 2.

Tabel 2. Spesifikasi Skenario Pengujian

Skenario	Karakteristik	Jumlah Simpul	Radius	Pengulangan
S1	Graf Kecil, Sparse	20	0.40	*30*
S2	Graf Sedang, Medium	100	0.22	*30*
S3	Graf Besar, Dense	500	0.12	30

Setiap skenario dijalankan sebanyak 30 kali pengulangan dengan nilai *seed* yang berbeda namun deterministik (seed = 42, 43, 44, ..., 71), sehingga hasil eksperimen dapat direproduksi secara konsisten. Metrik yang diukur dalam setiap pengulangan mencakup: (1) waktu eksekusi dalam satuan milidetik (ms) menggunakan *time.perf_counter()*, (2) jumlah simpul yang dikunjungi selama proses pencarian, dan (3) ukuran maksimum *priority queue* sepanjang eksekusi sebagai indikator penggunaan memori. Nilai rata-rata dari 30 pengulangan kemudian digunakan sebagai data representatif untuk perbandingan antara kedua algoritma

HASIL PENELITIAN DAN PEMBAHASAN

Verifikasi Kebenaran Algoritma

Verifikasi kebenaran merupakan aspek fundamental yang membedakan penelitian ini dari studi empiris konvensional. Hasil pengujian menunjukkan bahwa pada seluruh 90 eksekusi yang dilakukan (30 pengulangan × 3 skenario), kedua algoritma secara konsisten menghasilkan nilai jarak terpendek yang identik dengan toleransi kesalahan floating point sebesar 10^{-6}. Tingkat konsistensi sebesar 100% ini berlaku pada ketiga skenario, mulai dari graf kecil dengan 20 simpul hingga graf besar dengan 500 simpul, sebagaimana dirangkum pada Tabel 3.

Tabel 3. Hasil Verifikasi Kebenaran Kedua Algoritma

Skenario	Jumlah Simpul	Run Konsisten	Konsistensi (%)
S1 (Kecil, Sparse)	20	30 / 30	100%
S2 (Sedang, Medium)	100	30 / 30	100%
S3 (Besar, Dense)	500	30 / 30	100%

Hasil ini memiliki makna yang sangat signifikan secara ilmiah. Dari perspektif kebenaran teoritis, pembuktian melalui *loop invariant* pada Dijkstra dan sifat *admissibility* heuristik pada A* yang telah diuraikan di bagian metodologi kini mendapatkan konfirmasi empirisnya. Kedua pendekatan pembuktian teoritis dan empiris saling memperkuat dan memberikan jaminan keandalan yang komprehensif terhadap implementasi kedua algoritma. Temuan ini sejalan dengan hasil penelitian Ardiansyah et al. (2025) yang menyatakan bahwa A* dengan heuristik *admissible* secara konsisten menghasilkan jalur optimal yang setara dengan Dijkstra.

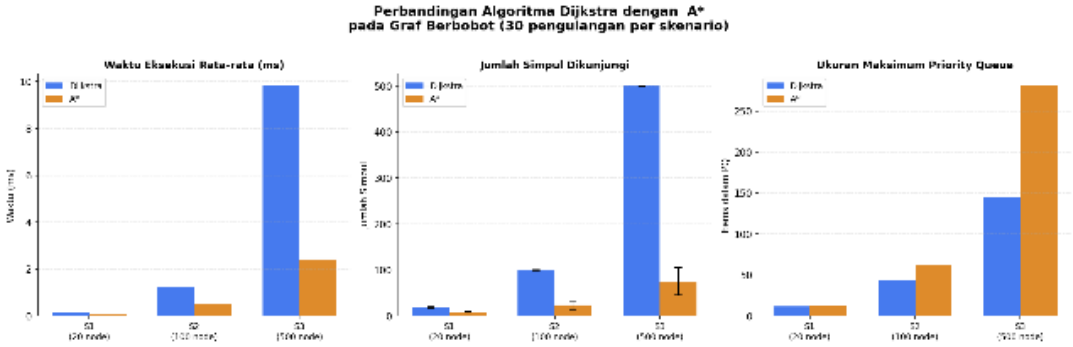
Perbandingan Efisiensi Waktu Eksekusi

Tabel 4 menyajikan perbandingan lengkap waktu eksekusi rata-rata dan jumlah simpul yang dikunjungi oleh kedua algoritma pada ketiga skenario pengujian.

Tabel 3. Perbandingan Metrik Efisiensi Dijkstra dan A*

Skenario	Algoritma	Waktu (ms)	Simpul Dikunjungi	Std Simpul	Max PQ
S1 (Kecil)	**Dijkstra**	**0.1405**	**18.8**	**1.7**	**12.1**
	A*	**0.0856**	**6.9**	**2.6**	**12.4**
S2 (Sedang)	**Dijkstra**	**1.2154**	**99.6**	**0.7**	**43.1**
	A*	**0.4890**	**23.7**	**9.2**	**62.7**
S3 (Besar)	**Dijkstra**	**9.8173**	**499.9**	**0.3**	**145.6**
	A*	**2.4141**	**75.3**	**30.1**	**280.9**

Perbandingan waktu eksekusi menunjukkan keunggulan A* yang konsisten dan semakin signifikan seiring bertambahnya skala graf. Pada skenario S1 (20 simpul), A* membutuhkan waktu rata-rata 0.0856 ms dibandingkan Dijkstra sebesar 0.1405 ms, menunjukkan percepatan sebesar 39,1%. Perbedaan ini semakin melebar pada skenario S2 (100 simpul), di mana A* selesai dalam 0.4890 ms sementara Dijkstra membutuhkan 1.2154 ms dengan percepatan 59,8%. Puncak perbedaan terjadi pada skenario S3 (500 simpul), di mana A* hanya memerlukan 2.4141 ms dibandingkan Dijkstra sebesar 9.8173 ms, menghasilkan percepatan sebesar 75,4%. Pola akselerasi yang terus meningkat ini divisualisasikan pada Gambar 2.



Gambar 2. Perbandingan Waktu Eksekusi, Simpul Dikunjungi, dan Max PQ

Percepatan yang semakin besar pada graf yang lebih besar ini dapat dijelaskan secara teoritis melalui kompleksitas asimtotik masing-masing algoritma. Dijkstra dengan kompleksitas O((V + E) log V) wajib memproses seluruh ruang simpul secara menyeluruh karena tidak memiliki arah pencarian yang terfokus. Hal ini terbukti dari data pada skenario S3 di mana Dijkstra mengunjungi rata-rata 499,9 simpul dari total 500 simpul yang tersedia nyaris seluruh graf dieksplorasi. Sebaliknya, A* dengan heuristik *Euclidean Distance* berhasil membatasi eksplorasi hanya pada simpul-simpul yang relevan secara geometris terhadap tujuan, sehingga pada S3 rata-rata hanya 75,3 simpul yang dikunjungi penghematan eksplorasi sebesar 84,9% dibandingkan Dijkstra. Temuan ini konsisten dengan hasil penelitian terdahulu yang menyatakan bahwa A* mampu memangkas ruang pencarian secara drastis pada graf berskala besar.

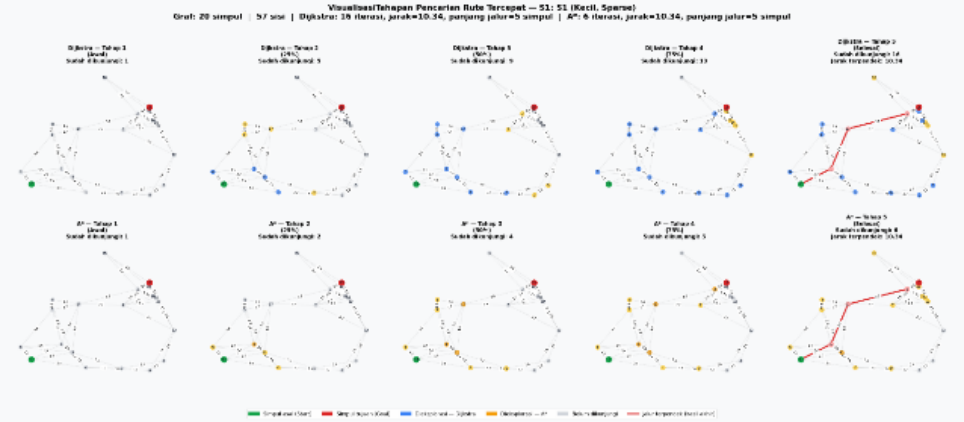
Analisis Penggunaan Memori

Ukuran maksimum *priority queue* (Max PQ) digunakan sebagai proksi untuk mengukur penggunaan memori kerja selama proses pencarian berlangsung. Hasil eksperimen mengungkapkan pola yang menarik dan berbeda dari ekspektasi awal. Pada skenario S1, ukuran Max PQ kedua algoritma hampir setara, Dijkstra sebesar 12,1 dan A* sebesar 12,4 yang dapat dipahami karena pada graf kecil heuristik belum memberikan pemangkasan yang berarti terhadap ruang pencarian. Namun pada skenario S2 dan S3, terjadi fenomena yang perlu dicermati secara mendalam: Max PQ milik A* justru lebih besar daripada Dijkstra. Pada S2, Max PQ A* mencapai 62,7 dibandingkan Dijkstra sebesar 43,1, dan pada S3 perbedaan ini semakin ekstrem dengan A* mencapai 280,9 berbanding 145,6 milik Dijkstra. Fenomena ini terjadi karena mekanisme A* yang agresif dalam memasukkan kandidat simpul ke dalam *open list* berdasarkan nilai f(n) = g(n) + h(n). Ketika heuristik mengarahkan pencarian ke satu sisi graf secara agresif, simpul-simpul di sepanjang jalur yang dieksplorasi beserta seluruh tetangganya dimasukkan ke antrean, namun kemudian sebagian besar tidak jadi diproses karena tujuan sudah ditemukan lebih cepat. Ini merupakan *trade-off* inheren pada A*: kecepatan waktu diperoleh dengan biaya penggunaan memori yang lebih tinggi. Temuan ini sejalan dengan

analisis Al Bager & Ahmed (2021) yang menegaskan bahwa efisiensi waktu A* tidak selalu diiringi efisiensi memori pada graf dengan densitas tinggi.

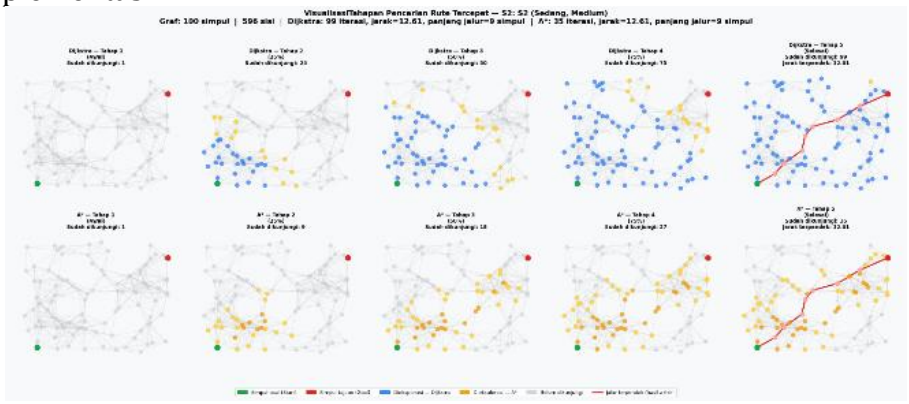
Visualisasi Tahapan Pencarian Rute

Untuk memperkuat pemahaman tentang perbedaan mekanisme kerja kedua algoritma, visualisasi tahapan pencarian ditampilkan pada Gambar 3, 4, dan 5. Setiap gambar menampilkan dua baris panel: baris atas untuk Dijkstra dan baris bawah untuk A*, masing-masing terdiri dari 5 tahap yang merepresentasikan progres eksplorasi dari awal hingga selesai (0%, 25%, 50%, 75%, dan 100%). Simpul berwarna biru merepresentasikan simpul yang sudah dikunjungi oleh Dijkstra, simpul oranye untuk A*, simpul hijau adalah titik asal, simpul merah adalah titik tujuan, dan garis merah tebal menunjukkan jalur terpendek atau rute tercepat yang ditemukan pada tahap akhir.



Gambar 3. Tahapan Pencarian pada Graf S1 (20 Simpul)

Pada skenario S1 (Gambar 3), perbedaan perilaku eksplorasi sudah terlihat meskipun grafnya kecil. Dijkstra menyebar ke seluruh penjuru graf secara merata dari awal, sementara A* langsung mengarahkan eksplorasinya ke sisi graf yang mendekati tujuan. Pada tahap akhir, Dijkstra telah mengunjungi hampir seluruh 20 simpul (rata-rata 18,8 simpul), sedangkan A* hanya mengunjungi rata-rata 6,9 simpul, kurang dari sepertiga total simpul. Meski demikian, kedua algoritma berhasil menemukan jalur dengan jarak yang identik, mengonfirmasi kebenaran implementasi.



Gambar 4. Tahapan Pencarian pada Graf S2 (100 Simpul)

Pada skenario S2 (Gambar 4), kontras antara kedua algoritma semakin jelas. Dijkstra terlihat mengisi hampir seluruh area graf secara sistematis sebelum mencapai tujuan, sedangkan A* membentuk "koridor" eksplorasi yang sempit dan terarah dari simpul asal menuju simpul tujuan. Rata-rata simpul yang dikunjungi A* hanya 23,7 dari 100 simpul,

sementara Dijkstra mengunjungi 99,6 simpul. Pola ini secara visual menegaskan bahwa heuristik *Euclidean Distance* bekerja efektif dalam mengeliminasi simpul-simpul yang tidak relevan secara geometris.

Gambar 5. Tahapan Pencarian pada Graf S3 (500 Simpul)

Skenario S3 (Gambar 5) merupakan demonstrasi paling dramatis dari perbedaan kedua algoritma. Pada tahap akhir, seluruh area graf tampak berwarna biru gelap pada panel Dijkstra, menandakan hampir semua 500 simpul telah dieksplorasi. Berbeda tajam, panel A* hanya menampilkan jalur oranye yang sempit membentang dari pojok kiri-bawah ke pojok kanan-atas, dengan sebagian besar area graf tetap berwarna abu-abu (belum dikunjungi). Simpangan baku yang tinggi pada A* di skenario ini (σ = 30,1 simpul) mencerminkan variabilitas yang wajar akibat perbedaan topologi graf pada tiap seed, juga pada beberapa konfigurasi graf, jalur geometris terpendek tidak selalu identik dengan jalur berbobot terpendek, sehingga A* perlu melakukan beberapa eksplorasi tambahan.

Diskusi: Trade-off dan Rekomendasi Pemilihan Algoritma

Berdasarkan seluruh hasil analisis di atas, dapat dirumuskan bahwa perbandingan antara Dijkstra dan A* bukan merupakan persaingan mutlak, melainkan sebuah *trade-off* yang bergantung pada karakteristik masalah dan sumber daya yang tersedia. Dari sisi kebenaran, kedua algoritma terbukti ekuivalen, keduanya selalu menghasilkan jalur terpendek yang optimal selama syarat bobot non-negatif terpenuhi dan heuristik A* bersifat *admissible*. Tidak ada kompromi terhadap kualitas solusi pada kedua algoritma. Dari sisi efisiensi, A* unggul secara substansial dalam hal waktu eksekusi dan jumlah simpul yang dikunjungi, terutama pada graf berskala besar. Namun keunggulan ini disertai konsekuensi berupa penggunaan memori kerja (*priority queue*) yang lebih besar. Oleh karena itu, rekomendasi pemilihan algoritma dapat dirumuskan sebagai berikut: (1) gunakan A* ketika tersedia informasi posisi geometris simpul (koordinat), graf berskala besar, dan kecepatan respons merupakan prioritas utama, seperti pada sistem navigasi *real-time* dan peta digital; (2) gunakan Dijkstra ketika tidak tersedia informasi posisi geometris untuk membangun heuristik yang valid, atau ketika diperlukan perhitungan jarak terpendek ke *semua* simpul sekaligus (*single-source shortest path*), seperti pada analisis jaringan telekomunikasi atau perutean paket data. Temuan ini memperkuat argumen Rachmawati & Gustin (2020) bahwa pemilihan algoritma optimal harus mempertimbangkan karakteristik spesifik dari graf dan konteks aplikasinya secara bersamaan.

KESIMPULAN

Berdasarkan hasil analisis yang telah dilakukan, dapat disimpulkan bahwa algoritma Dijkstra dan A* sama-sama mampu menentukan lintasan terpendek pada graf berbobot dengan tingkat keakuratan yang tinggi. Algoritma Dijkstra memiliki keunggulan dalam menjamin solusi

optimal secara menyeluruh melalui pendekatan deterministik, sehingga sangat cocok digunakan pada kasus yang tidak memerlukan optimasi waktu secara signifikan. Namun, pada graf dengan skala besar, kinerjanya cenderung kurang efisien karena proses eksplorasi yang tidak terarah. Di sisi lain, algoritma A* menunjukkan performa yang lebih efisien dalam hal waktu pencarian dengan memanfaatkan fungsi heuristik untuk mengarahkan proses menuju simpul tujuan. Efisiensi ini sangat bergantung pada kualitas heuristik yang digunakan, di mana heuristik yang bersifat admissible mampu tetap menjamin optimalitas solusi. Dengan demikian, terdapat trade-off antara tingkat efisiensi dan ketergantungan terhadap heuristik dalam penggunaan algoritma A*. Secara keseluruhan, penelitian ini menunjukkan bahwa pemilihan algoritma harus disesuaikan dengan karakteristik permasalahan yang dihadapi, khususnya terkait ukuran graf dan kebutuhan efisiensi. Hasil penelitian ini diharapkan dapat menjadi referensi dalam pengembangan sistem berbasis graf yang membutuhkan keseimbangan antara keakuratan dan performa.

DAFTAR PUSTAKA

A. Ardiansyah, A. M. Nasution, and M. Iqbal, "Comparative Analysis of Dijkstra and A* Algorithms for Determining the Shortest Route," *bit-Tech*, vol. 8, no. 2, pp. 2974–2983, Dec. 2025, doi: 10.32877/bt.v8i2.3474.

A. B. A. A. B. R and A. S. A. Ahmed, "Designing and Implementing Shortest and Fastest Paths; A Comparison of Bellman-Ford algorithm, A*, and Dijkstra's algorithms," *International Journal of Computer Trends and Technology*, vol. 69, no. 5, pp. 6–12, May 2021, doi: 10.14445/22312803/ijctt-v69i5p102.

A. Bernstein, D. Nanongkai, and C. Wulff-Nilsen, "Negative-Weight Single-Source Shortest Paths in Near-linear Time," *Journal of the ACM*, vol. 72, no. 4, Jul. 2025, doi: 10.1145/3742890.

A. Mohan, W. X. Leow, and A. Hobor, "Functional Correctness of C Implementations of Dijkstra's, Kruskal's, and Prim's Algorithms," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Science and Business Media Deutschland GmbH, 2021, pp. 801–826. doi: 10.1007/978-3-030-81688-9_37.

B. Şeyh, S. Kayali, U. Yuzgec, and M. Ozalp, "Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi Autonomous UAV Navigation in Simulated Environments: A Comparative Study of Dijkstra and A* Algorithms Benzetim Yapılan Ortamlarda Otonom İHA Navigasyonu: Dijkstra ve A* Algoritmalarının Karşılaştırmalı Çalışması." [Online]. Available: https://dergipark.org.tr/tr/pub/bseufbd.

B. Yang *et al.*, "A novel heuristic emergency path planning method based on vector grid map," *ISPRS Int. J. Geoinf.*, vol. 10, no. 6, Jun. 2021, doi: 10.3390/ijgi10060370.

D. Rachmawati and L. Gustin, "Analysis of Dijkstra's Algorithm and A∗ Algorithm in Shortest Path Problem," in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Jul. 2020. doi: 10.1088/1742-6596/1566/1/012061.

F. A. S. Alwafi, X. Xu, R. Saatchi, and L. Alboul, "Development and Evaluation of a Multi-Robot Path Planning Graph Algorithm," *Information (Switzerland)*, vol. 16, no. 6, Jun. 2025, doi: 10.3390/info16060431.

F. Sapundzhi, K. Danev, A. Ivanova, M. Popstoilov, and S. Georgiev, "A Performance Comparison of Shortest Path Algorithms in Directed Graphs †," *Engineering Proceedings*, vol. 100, no. 1, 2025, doi: 10.3390/engproc2025100031.

G. Pradeep Reddy, Y. V. P. Kumar, M. Kalyan Chakravarthi, and A. Flah, "Refined Network Topology for Improved Reliability and Enhanced Dijkstra Algorithm for Optimal Path Selection during Link Failures in Cluster Microgrids," *Sustainability (Switzerland)*, vol. 14, no. 16, Aug. 2022, doi: 10.3390/su141610367.

H. Mehta, P. Kanani, and P. Lande, "Google Maps," 2019.
I. H. Toroslu, "Improving The Floyd-Warshall All Pairs Shortest Paths Algorithm," Sep. 2021, [Online]. Available: http://arxiv.org/abs/2109.01872.
I. Szcześniak, B. Woźna-Szcześniak, and I. Olszewski, "Generic Dijkstra," Mar. 2026, doi: 10.1109/NOMS56928.2023.10154322.
J. Vilela, B. Fanzeres, R. Martinelli, and C. Contardo, "Efficient labeling algorithms for adjacent quadratic shortest paths," Dec. 2021, [Online]. Available: http://arxiv.org/abs/2112.04045.
K. Nosirov, E. Norov, dan S. Tashmetov, "A Review of Shortest Path Problem in Graph Theory," *Eurasian Journal of Engineering and Technology*, vol. 13, pp. 1–11, Dec. 2022.
M. R. Wayahdi, S. H. N. Ginting, and D. Syahputra, "Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest Path," *International Journal of Advances in Data and Information Systems*, vol. 2, no. 1, pp. 45–52, Feb. 2021, doi: 10.25008/ijadis.v2i1.1206.
Md. Mehedi Hassan, Md. Asadujjaman, and Md. Golam Robbani, "A Modified Approach of Dijkstra's Method for Finding Shortest Path in a Weighted Directed Graph," *American Journal of Science, Engineering and Technology*, Jul. 2023, doi: 10.11648/j.ajset.20230803.14.
N. Aldhafferi, "Time and Memory Trade-Offs in Shortest-Path Algorithms Across Graph Topologies: A*, Bellman–Ford, Dijkstra, AI-Augmented A* and a Neural Baseline," *Computers*, vol. 14, no. 12, Dec. 2025, doi: 10.3390/computers14120545.
O. Alamoudi and M. Al-Hashimi, "On the Energy Behaviors of the Bellman–Ford and Dijkstra Algorithms: A Detailed Empirical Study," *Journal of Sensor and Actuator Networks*, vol. 13, no. 5, Oct. 2024, doi: 10.3390/jsan13050067.
R. D. Saktia Purnama *et al.*, "Implementasi Penggunaan Algoritma Greedy Best First Search Untuk Menentukan Rute Terpendek Dari Cilacap Ke Yogyakarta," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 2, Apr. 2024, doi: 10.23960/jitet.v12i2.4068.
R. Duan, J. Mao, X. Mao, X. Shu, and L. Yin, "Breaking the Sorting Barrier for Directed Single-Source Shortest Paths," Jul. 2025, [Online]. Available: http://arxiv.org/abs/2504.17033
R. Lewis, "Algorithms for finding shortest paths in networks with vertex transfer penalties," *Algorithms*, vol. 13, no. 11, pp. 1–21, Nov. 2020, doi: 10.3390/a13110269.
R. Rusmi and M. Amrin Lubis, "SISTEMASI: Jurnal Sistem Informasi Determining Travel Time and Fastest Route Using Dijkstra Algorithm and Google Map." [Online]. Available: http://sistemasi.ftik.unisi.ac.id496.
S. Basu, N. Kōshima, T. Eden, O. Ben-Eliezer, and C. Seshadhri, "A Sublinear Algorithm for Approximate Shortest Paths in Large Networks," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.08624.
S. H. Barkund, A. Sharma, and H. R. Bhapkar, "Survey of Shortest Path Algorithms," *International Journal of Renewable Energy Exchange*, vol. 10, no. 11, pp. 46–57, Nov. 2022, doi: 10.58443/ijrex.10.11.2022.46-57.
S. S. Abed and S. F. Noaman, "Comparative Analysis of Shortest Path Algorithms," *South Asian Research Journal of Engineering and Technology*, vol. 7, no. 05, pp. 131–143, Dec. 2025, doi: 10.36346/sarjet.2025.v07i05.002.
V. Milin, T. Stanivuk, I. Skoko, and T. Bulić, "Dijkstra and A* Algorithms for Algorithmic Optimization of Maritime Routes and Logistics of Offshore Wind Farms," *J. Mar. Sci. Eng.*, vol. 13, no. 10, Oct. 2025, doi: 10.3390/jmse13101863.
Y. F. Riti, J. S. Iskandar, and H. Hendra, "Comparison Analysis of Graph Theory Algorithms for Shortest Path Problem," *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, vol. 12, no. 3, pp. 415–424, Nov. 2023, doi: 10.32736/sisfokom.v12i3.1756.

Y. Liu, X. Gao, B. Wang, J. Fan, Q. Li, and W. Dai, "A passage time–cost optimal A* algorithm for cross-country path planning," *International Journal of Applied Earth Observation and Geoinformation*, vol. 130, Jun. 2024, doi: 10.1016/j.jag.2024.103907.

Y. W. Shin *et al.*, "Near-optimal weather routing by using improved A* algorithm," *Applied Sciences (Switzerland)*, vol. 10, no. 17, Sep. 2020, doi: 10.3390/app10176010.