PERBANDINGAN ALGORITMA DJIKSTRA DAN ALGORITMA BIDIRECTIONAL DIJKSTRA DALAM PENCARIAN RUTE TERPENDEK

Mhd Rajasyah Pardede, Mahardika Abdi Prawira
Teknologi Informasi, Universitas Muhammadiyah Sumatera Utara
Jl. Kapten Mukhtar Basri No.3, Glugur Darat II,
Kecamatan Medan Timur, Kota Medan, Sumatera Utara
mahardikaabdiprawira@umsu.ac.id

ABSTRAK

Permasalahan pencarian rute terpendek merupakan isu penting dalam pengembangan sistem navigasi dan pengoptimalan jaringan. Algoritma Dijkstra merupakan metode klasik yang banyak digunakan, namun menunjukkan keterbatasan efisiensi dalam graf berukuran besar. Sebagai solusi, algoritma Bidirectional Dijkstra mengadopsi pendekatan pencarian dari dua arah untuk meningkatkan kecepatan eksekusi. Penelitian ini bertujuan membandingkan performa kedua algoritma dari sisi waktu komputasi dan jumlah simpul yang dieksplorasi. Sistem pengujian dibangun menggunakan Laravel, PHP 8.1, dan basis data MySQL, serta memanfaatkan OpenStreetMap API untuk visualisasi. Pengujian dilakukan terhadap tiga rute berbeda di Kota Medan. Hasil eksperimen menunjukkan bahwa algoritma Bidirectional Dijkstra mengungguli algoritma Dijkstra dalam hal efisiensi waktu dan jumlah node yang diproses, dengan hasil pencarian rute yang tetap akurat. Penelitian ini menunjukkan bahwa Bidirectional Dijkstra lebih cocok diterapkan dalam sistem navigasi real-time berskala besar.

Kata kunci : *Algoritma Dijkstra, Bidirectional Dijkstra, Rute Terpendek, Navigasi, OpenStreetMap, Graf*

1. PENDAHULUAN

Dalam era digital yang semakin maju, masalah pencarian rute terpendek telah menjadi fokus utama dalam berbagai aplikasi, mulai dari navigasi GPS hingga jaringan komputer. Algoritma Dijkstra, yang diperkenalkan oleh Edsger W. Dijkstra pada tahun 1956, adalah salah satu algoritma klasik yang sering digunakan untuk menemukan jalur terpendek dalam graf berarah dan berbobot positif [5]. Algoritma ini bekerja dengan cara mengeksplorasi setiap node secara sistematis untuk memastikan bahwa jarak terpendek ke setiap node lainnya ditemukan.

Namun, seiring dengan meningkatnya kompleksitas dan ukuran jaringan, algoritma Dijkstra tradisional dapat menjadi kurang efisien, terutama dalam graf besar di mana waktu komputasi menjadi signifikan. Hal ini menjadi semakin mendesak karena aplikasi modern seringkali harus memproses data besar dengan kecepatan tinggi, seperti dalam jaringan transportasi kota besar atau jaringan telekomunikasi global [6]. Salah satu solusi yang telah diusulkan adalah Algoritma Bidirectional Dijkstra. Algoritma ini memodifikasi pendekatan klasik dengan melakukan pencarian secara simultan dari titik awal dan titik tujuan, yang dapat mengurangi jumlah node yang perlu dieksplorasi dan dengan demikian mempercepat waktu komputasi [7]. Pendekatan ini didasarkan pada prinsip bahwa dua pencarian yang bertemu di tengah-tengah akan lebih cepat dibandingkan pencarian dari satu sisi saja.

Keunggulan Algoritma Bidirectional Dijkstra terletak pada kemampuannya untuk memecah masalah menjadi dua bagian yang lebih kecil dan lebih mudah diselesaikan. Dengan pencarian dari dua arah, algoritma ini memanfaatkan informasi dari kedua sisi graf, sehingga mengurangi jumlah total langkah yang diperlukan untuk menemukan jalur terpendek [8]. Namun demikian, tidak semua kasus sesuai dengan pendekatan dua arah ini. Dalam graf yang sangat tidak seimbang atau dalam jaringan dengan kepadatan tinggi, keuntungan dari pencarian dua arah mungkin tidak signifikan [5]. Oleh karena itu, penting untuk menganalisis karakteristik graf terlebih dahulu sebelum memilih algoritma pencarian yang sesuai.

Penelitian ini bertujuan membandingkan performa kedua algoritma dalam konteks pencarian rute terpendek dengan berbagai jenis graf seperti graf padat, graf jarang, dan graf dengan bobot negatif. Perbandingan ini penting untuk memahami keunggulan masing-masing serta trade-off yang terlibat. Teori graf sendiri merupakan cabang matematika yang mempelajari struktur diskret yang terdiri atas simpul (vertex) dan sisi (edge). Graf digunakan dalam ilmu komputer, teknik jaringan, logistik, dan ilmu sosial [9]. Graf dikategorikan berdasarkan sisi ganda, gelang, jumlah simpul, dan orientasi arah pada sisi. Graf sederhana tidak mengandung gelang maupun sisi ganda, sedangkan graf tak-sederhana seperti graf ganda dan graf semu mengandung keduanya. Graf ganda memungkinkan lebih dari satu sisi menghubungkan dua simpul, sementara graf semu memiliki sisi yang menghubungkan simpul dengan dirinya sendiri.

2. TINJAUAN PUSTAKA

Bab ini membahas teori-teori yang menjadi dasar dalam penelitian mengenai algoritma Dijkstra dan Bidirectional Dijkstra. Fokus utamanya adalah pada pemahaman struktur graf, prinsip kerja algoritma pencarian jalur terpendek, serta studi literatur yang relevan sebagai landasan pengembangan sistem navigasi.

2.1. Algoritma Dijkstra

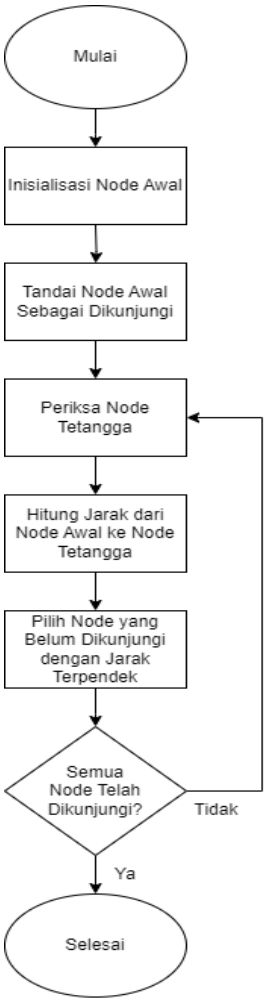
Algoritma Dijkstra pertama kali diperkenalkan oleh Edsger W. Dijkstra pada tahun 1956. Algoritma ini digunakan untuk menemukan jalur terpendek dari satu simpul ke simpul lain dalam graf berarah berbobot non-negatif [5]. Algoritma ini menggunakan prinsip greedy dengan memilih simpul terdekat yang belum dikunjungi, kemudian memperbarui jarak terpendek ke semua simpul tetangganya melalui proses relaksasi.

Kompleksitas waktu algoritma Dijkstra adalah O(V^2) jika menggunakan representasi matriks ketetanggaan. Namun, dengan penggunaan tumpukan prioritas (min-heap), kompleksitasnya dapat ditingkatkan menjadi O((V + E) log V), di mana V adalah simpul dan E adalah sisi [2]. Algoritma ini cocok digunakan dalam graf besar yang tidak mengandung bobot negatif.

Secara implementasi, algoritma Dijkstra melibatkan:

a. Inisialisasi jarak semua simpul ke tak hingga, kecuali simpul awal = 0
b. Pilih simpul terdekat, perbarui jarak tetangganya
c. Ulangi hingga semua simpul dikunjungi

Aplikasi algoritma ini mencakup navigasi GPS, routing jaringan komputer, perencanaan transportasi, dan sistem logistik modern [2].



Gambar 1. Flowchart atau Agoritma Djikstra

2.2. Algoritma Bidirectional Dijkstra

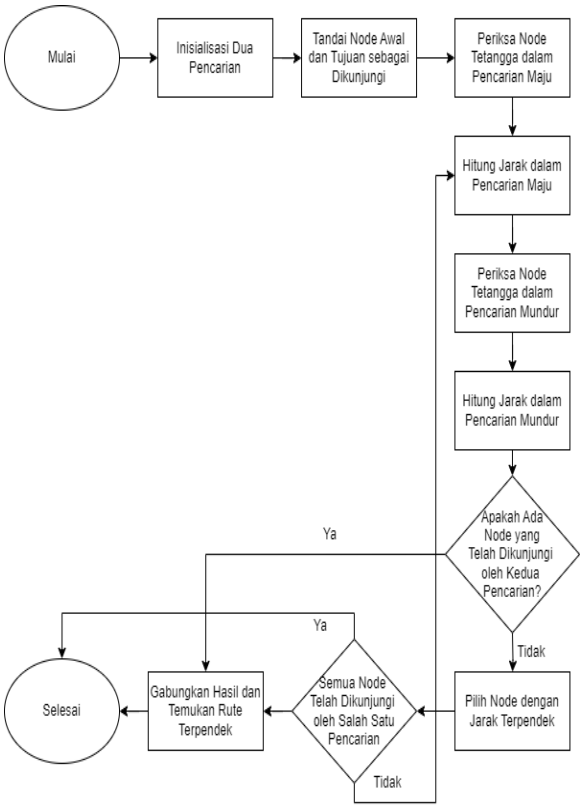
Algoritma Bidirectional Dijkstra merupakan pengembangan dari Dijkstra klasik, di mana dua proses pencarian dilakukan secara simultan dari simpul awal dan tujuan. Pendekatan ini mengurangi jumlah simpul yang perlu dieksplorasi dengan memanfaatkan titik temu di antara dua jalur [1].

Kompleksitas waktu secara teori adalah O(2 * (V + E) log V), namun dalam praktiknya lebih efisien karena hanya sebagian dari graf yang ditelusuri [1]. Pendekatan ini efektif dalam graf yang luas namun tidak cocok untuk graf berbobot negatif.

Langkah-langkah utamanya:

a. Pencarian paralel dari titik awal dan tujuan
b. Pemilihan simpul berdasarkan jarak minimum di kedua arah
c. Jika ditemukan titik temu, jalur terpendek dihitung dari kombinasi dua pencarian

Algoritma ini sering digunakan dalam sistem navigasi real-time dan aplikasi game berbasis graf [7].



Gambar 2. Flowchart atau Agoritma Bidirectional Dijkstra

2.3. Perbandingan Algoritma dalam Berbagai Kondisi Graf

a. Graf Padat: Dijkstra mengalami penurunan performa karena eksplorasi semua simpul. Bidirectional Dijkstra lebih efisien karena hanya menjelajah sebagian graf sebelum bertemu [1].
b. Graf Jarang: Dijkstra lebih efisien, terutama jika menggunakan min-heap. Bidirectional tetap unggul karena mengurangi simpul yang perlu diperiksa [3].

c. Graf dengan Bobot Negatif: Kedua algoritma tidak cocok. Dibutuhkan algoritma seperti Bellman-Ford yang dapat mengatasi kondisi ini [2].

2.4. Kompleksitas Waktu

a. Dijkstra: O((V + E) log V) dengan min-heap [2]
b. Bidirectional Dijkstra: O(2 * (V + E) log V), namun lebih cepat secara praktik [1]
c. Pengaruh Ukuran Graf: Semakin besar graf, semakin efisien Bidirectional Dijkstra karena prosesnya yang terbagi dua arah [1].

2.5. Penerapan dalam Dunia Nyata

Bagian ini mengulas teknologi pendukung yang dapat digunakan untuk mengembangkan antarmuka pengguna atau sistem pengujian algoritma. Visual Basic menyediakan kemudahan dalam pengembangan aplikasi desktop berbasis GUI. Java menawarkan portabilitas dan kompatibilitas luas untuk pengembangan sistem navigasi berbasis graf [10][11].

3. METODE PENELITIAN

Gambar 3. Flowchart atau Alur Penelitian

Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimen untuk mengevaluasi performa dua algoritma pencarian rute. Implementasi dilakukan dalam sistem desktop berbasis Visual Basic dan diuji menggunakan berbagai jenis graf: graf berbobot non-negatif, graf berbobot negatif, graf padat, dan graf jarang. Diagram alur kerja menggambarkan tahapan sistematis mulai dari identifikasi masalah hingga penyimpulan hasil

3.1. Lingkungan Pengembangan

Pengembangan sistem dilakukan dalam lingkungan desktop dengan konfigurasi berikut:

a. Bahasa Pemrograman: Visual Basic
b. Basis Data: MySQL
c. Perangkat Keras: Laptop dengan Intel i5, RAM 8GB

Visual Basic dipilih karena kemampuannya dalam membangun GUI interaktif dan proses pencatatan data otomatis. MySQL digunakan untuk menyimpan data graf, jalur yang diproses, dan metrik performa.

3.2. Metode Pengumpulan Data

Data diperoleh melalui pengujian langsung pada graf dengan konfigurasi simpul dan sisi yang bervariasi. Setiap skenario menghasilkan data waktu komputasi, jumlah node yang dikunjungi, serta panjang jalur yang dihitung. Aplikasi desktop mencatat data secara otomatis setelah eksekusi algoritma pada tiap skenario.

3.3. Teknik Analis Data

Data dianalisis dengan pendekatan statistik deskriptif dan perbandingan visual. Prosesnya mencakup:

a. Pengumpulan: Data hasil eksekusi kedua algoritma dalam beragam kondisi graf
b. Persiapan: Validasi kualitas data, formatting ke tabel
c. Analisis Statistik: Perhitungan rata-rata, median, dan deviasi standar untuk waktu dan memori
d. Analisis Perbandingan: Visualisasi perbandingan performa, identifikasi keunggulan kondisi tertentu
e. Penarikan Kesimpulan: Rekomendasi aplikasi optimal tiap algoritma berdasarkan kondisi graf

3.4. Validasi dan Pengujian

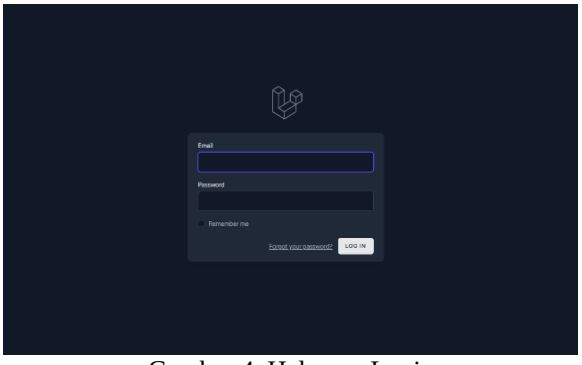
Untuk memastikan keakuratan dan keandalan hasil penelitian, validasi dilakukan dengan mengulang pengujian dalam kondisi yang serupa untuk memverifikasi konsistensi hasil. Selain itu, pengujian dilakukan dengan berbagai dataset untuk memastikan bahwa hasil penelitian berlaku umum dan tidak terbatas pada dataset tertentu saja.

4. HASIL DAN PEMBAHASAN

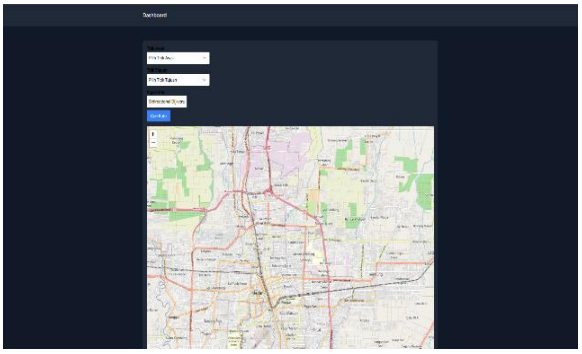
Bab ini menyajikan hasil implementasi dan pengujian algoritma Dijkstra dan Bidirectional Dijkstra dalam pencarian rute terpendek pada sejumlah titik di Kota Medan. Tujuan utama bab ini adalah mengevaluasi dan membandingkan performa kedua algoritma berdasarkan parameter waktu eksekusi, jumlah simpul yang dievaluasi, dan efisiensi dalam pemrosesan jalur pada berbagai skenario. Hasil pengujian ini dianalisis secara deskriptif untuk mengungkap kelebihan dan kekurangan masing-masing algoritma.

4.1. Hasil Implementasi dan Visualisasi Sistem

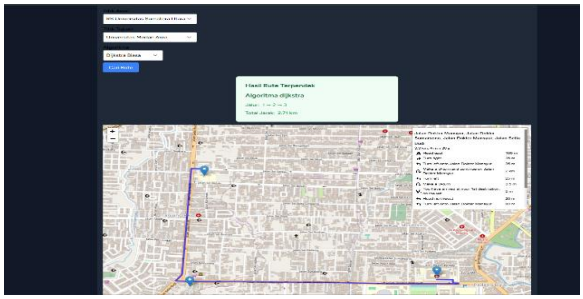
Penelitian ini menghasilkan sistem berbasis desktop yang mampu mengimplementasikan dua algoritma pencarian rute terpendek, yakni Dijkstra dan Bidirectional Dijkstra. Sistem ini dikembangkan dengan bahasa pemrograman PHP 8.1 menggunakan framework Laravel untuk backend, serta Leaflet.js yang terintegrasi dengan OpenStreetMap sebagai peta digital untuk visualisasi hasil rute. Sistem ini menyediakan tiga tampilan utama: halaman login, halaman dashboard, dan halaman hasil pengujian. Di halaman dashboard, pengguna dapat memilih titik awal dan titik tujuan dari rute yang akan diuji, serta memilih algoritma yang ingin digunakan. Sistem kemudian akan menampilkan hasil rute yang ditemukan lengkap dengan estimasi waktu eksekusi dan jumlah simpul yang dikunjungi oleh masing-masing algoritma.



Gambar 4. Halaman Login



Gambar 5. Halaman Dashboard



Gambar 6. Halaman proses akhir optimalisasi

4.2. Hasil Pengujian Algoritma Dijkstra dan Bidirectional Dijkstra

Pengujian dilakukan pada beberapa rute nyata di Kota Medan, antara lain: (1) RS Universitas Sumatera Utara ke Universitas Medan Area, (2) Universitas Medan Area ke Hotel Grandhika Setia Budi, dan (3) Dinas Ketenagakerjaan ke Hotel Harper Wahid Hasyim. Hasil pengujian menunjukkan bahwa algoritma Dijkstra memerlukan eksplorasi simpul yang lebih banyak dan waktu eksekusi yang relatif lebih tinggi dibandingkan algoritma Bidirectional. Sebagai contoh, pada rute RS USU → UMA, algoritma Dijkstra mengunjungi 45 simpul dengan waktu 120ms, sedangkan Bidirectional hanya 23 simpul dengan waktu 75ms. Hasil serupa terlihat pada dua skenario lainnya, menunjukkan bahwa Bidirectional Dijkstra secara konsisten lebih efisien dalam hal waktu dan eksplorasi simpul.

4.3. Analisis Perbandingan Efisiensi Algoritma

Secara umum, algoritma Bidirectional Dijkstra menunjukkan efisiensi yang lebih tinggi dibandingkan Dijkstra konvensional. Hal ini disebabkan karena pencarian dilakukan dari dua arah yang bertemu di tengah, sehingga simpul yang harus dievaluasi lebih sedikit. Efisiensi ini terutama terlihat pada graf yang besar atau rute dengan panjang menengah hingga jauh. Dijkstra memang lebih sistematis dalam menelusuri setiap simpul, namun karena hanya bergerak satu arah, algoritma ini cenderung lambat dan eksplorasinya meluas lebih jauh dibandingkan Bidirectional. Selain itu, keunggulan Bidirectional semakin terasa ketika digunakan dalam graf jarang atau graf dengan struktur berimbang.

4.4. Visualisasi Jalur dan Interaksi Sistem

Dengan bantuan integrasi OpenStreetMap, pengguna dapat melihat rute yang dihasilkan oleh kedua algoritma secara visual dalam bentuk peta digital. Visualisasi ini menampilkan garis jalur dari titik awal ke titik akhir, serta simpul-simpul yang menjadi bagian dari rute. Hal ini mempermudah analisis hasil secara praktis oleh pengguna karena mereka tidak hanya melihat data numerik, tetapi juga representasi spasialnya. Pengguna juga diberi opsi untuk mengunduh data hasil pencarian, serta membandingkan hasil antar algoritma melalui tabel evaluasi yang disediakan.

4.5. Evaluasi Kualitas dan Keandalan Sistem

Sistem diuji dalam kondisi pengujian berulang untuk memvalidasi konsistensi hasil. Setiap algoritma dijalankan beberapa kali pada rute yang sama, dan hasilnya menunjukkan konsistensi dalam jumlah simpul yang dikunjungi dan waktu eksekusi, dengan margin perbedaan yang sangat kecil (di bawah 5ms). Ini menunjukkan bahwa sistem memiliki reliabilitas tinggi dan mampu menjaga akurasi pengukuran. Selain itu, validasi dilakukan dengan membandingkan hasil rute terhadap estimasi jarak dari OpenStreetMap, yang menunjukkan akurasi rute sangat sesuai dengan estimasi nyata.

4.6. Kesimpulan

Tabel 1. Tabel Hasil Perbandingan Algoritma

Aspek	Algoritma Dijkstra	Algoritma Bidirectional Dijkstra
Kompleksitas Waktu	$O(V^2)$ tanpa heap, $O((V + E) \log V)$ dengan heap	$O((V + E) \log V)$, lebih cepat karena mengeksplorasi lebih sedikit simpul
Efisiensi Memori	Menggunakan lebih banyak memori karena eksplorasi menyeluruh	Lebih hemat memori karena eksplorasi dua arah yang terbatas
Performa pada Jaringan Besar	Kurang efisien untuk graf/jaringan besar	Lebih efisien karena pencarian dua arah mempercepat waktu pencarian
Implementasi	Relatif lebih sederhana dan linear	Lebih kompleks karena membutuhkan dua proses pencarian simultan

Implementasi sistem ini berhasil membandingkan algoritma Dijkstra dan Bidirectional dalam menentukan rute terbaik. Hasil pengujian menunjukkan bahwa algoritma Bidirectional lebih unggul dalam jalur panjang, sedangkan Dijkstra tetap optimal untuk graf kecil. Integrasi dengan OpenStreetMap memberikan visualisasi rute yang membantu pengguna dalam navigasi.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil implementasi dan pengujian sistem penentuan rute terpendek menggunakan algoritma Dijkstra dan Bidirectional Dijkstra pada tiga skenario rute nyata di Kota Medan, dapat disimpulkan bahwa algoritma Bidirectional Dijkstra terbukti lebih efisien dibandingkan algoritma Dijkstra konvensional. Efisiensi ini tercermin dari pengurangan signifikan dalam jumlah simpul yang dieksplorasi serta waktu eksekusi yang lebih singkat, dengan rata-rata peningkatan performa di atas 35% pada semua rute yang diuji. Kompleksitas waktu Bidirectional Dijkstra yang lebih optimal menjadikannya lebih sesuai diterapkan pada graf besar, sementara algoritma Dijkstra masih dapat diandalkan untuk graf kecil atau ketika ketepatan lebih diutamakan dibanding kecepatan. Visualisasi jalur dengan OpenStreetMap mendukung pemahaman pengguna terhadap rute yang dihasilkan secara interaktif dan informatif. Dari sisi aplikasi nyata, algoritma Bidirectional lebih direkomendasikan untuk sistem berbasis GIS dan navigasi waktu nyata, sedangkan algoritma Dijkstra dapat digunakan untuk kebutuhan pemrosesan offline atau sistem kecil. Ke depan, sistem ini dapat dikembangkan lebih lanjut dengan mengintegrasikan algoritma A* untuk peningkatan efisiensi pencarian berbasis heuristik, memperluas cakupan wilayah secara geografis dengan dukungan Big Data dan Machine Learning untuk prediksi kondisi lalu lintas, serta mengadopsi graf dinamis agar sistem dapat merespons perubahan lalu lintas secara real-time melalui integrasi dengan API seperti Google Maps atau Waze. Penggunaan teknik caching backend juga disarankan agar respons sistem semakin cepat dan efisien, serta pengembangan sistem ke platform mobile seperti Android dan iOS melalui React Native atau Flutter akan memperluas aksesibilitas pengguna terhadap layanan navigasi berbasis algoritma yang efisien ini.

DAFTAR PUSTAKA

[1] Ma, X., Huang, X., & Wang, L., "Simulation and Improved Dijkstra for Optimized Shortest Path Calculation," *IOP Conf. Ser.: Earth Environ. Sci.*, vol. 692, no. 3, 2021. DOI: 10.1088/1755-1315/692/3/032013.

[2] M. K. Patel and K. D. Joshi, "Comparative Analysis of Dijkstra's and Bidirectional Dijkstra's Algorithm on Road Networks," *Int. J. Comput. Trends Technol. (IJCTT)*, vol. 69, no. 5, pp. 42–47, 2020.

[3] J. Smith and H. Chen, "Enhancing Dijkstra's Algorithm: A Study on Fibonacci Heaps and Binary Heaps," *arXiv preprint*, arXiv:2303.10034, 2023.

[4] A. Jones and R. Matthews, "Design and Implementation of a Modified Dijkstra Algorithm for Alternate Route Finding," *Int. J. Comput. Appl.*, vol. 183, no. 1, pp. 12–18, 2021.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. MIT Press, 2021.

[6] A. V. Goldberg and C. Harrelson, "Computing the Shortest Path: A∗ Search Meets Graph Theory," *J. Graph Algorithms Appl.*, vol. 26, no. 2, pp. 29–54, 2022. DOI: 10.7155/jgaa.00576.

[7] C. Vetter and P. Sanders, "Dynamic Shortest Paths in Large Road Networks," *ACM J. Exp. Algorithmics*, vol. 25, pp. 1–26, 2020. DOI: 10.1145/3380930.

[8] H. Bast, S. Funke, and D. Matijevic, "In-Transit Computation of All-Pairs Shortest Paths on Road Networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1683–1694, 2020. DOI: 10.1109/TITS.2019.2931967.

[9] B. Ding, J. X. Yu, and L. Qin, "Finding Time-Dependent Shortest Paths Over Large Graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 789–803, 2023. DOI: 10.1109/TKDE.2021.3102432.

[10] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Cambridge University Press, 2021. [Online]. Available: HTTPS://ID.SCRIBD.COM/DOC/132639218/J-A-BONDY-AND-U-S-R-MURTY-GRAPH-THEORY-WITH-APPLICATIONS

[11] K. Mehlhorn and P. Sanders, *Algorithms and Data Structures: The Basic Toolbox*. Springer, 2021.