p-ISSN : 2406-7733
e-ISSN : 2597-9922

PERBANDINGAN ALGORITMA GREEDY DAN ALGORITMA DJIKSTRA DALAM MENCARI RUTE TERPENDEK PADA PENDISTRIBUSIAN LOGISTIK

Hafizh Pebrian[1], Mohammad Dzaky[2], Muhammad Ezar Al Rivan[3]
Program Studi Informatika, Fakultas Ilmu Komputer dan Rekayasa, Universitas Multi Data Palembang
Jl. Rajawali No. 14, Palembang
E-mail: hafizhpebrian_2226250077@mhs.mdp.ac.id[1], *mohammaddzaky_2226250083@mhs.mdp.ac.id[2] , meedzhar@mdp.ac.id[3]

Abstrak - Kelancaran logistik sangat bergantung pada efisiensi rute pengiriman. Penentuan rute secara manual sering menimbulkan inefisiensi dan pembengkakan biaya operasional. Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja algoritma Dijkstra dan *Greedy* dalam mencari lintasan terpendek pada jaringan distribusi logistik. Pengujian dilakukan menggunakan metode kuantitatif eksperimental dengan merepresentasikan data ke dalam graf berbobot dari jaringan distribusi 35 kota di Indonesia, menggunakan studi kasus rute Banda Aceh menuju Bandung. Parameter yang dievaluasi meliputi total jarak tempuh, waktu pemrosesan (*execution time*), dan penggunaan memori (*memory usage*). Hasil pengujian menunjukkan bahwa algoritma Dijkstra jauh lebih unggul dengan akurasi optimalitas 100%, menghasilkan rute terpendek sejauh 2019,20 km yang melewati 10 *node*. Sebaliknya, algoritma *Greedy* terjebak pada optimum lokal sehingga rute yang dihasilkan memutar sejauh 2863,75 km dengan akurasi hanya 70,51%. Dari segi komputasi, Dijkstra terbukti lebih efisien dengan waktu eksekusi 0,4889 ms dan konsumsi memori 16,33 KB, dibandingkan *Greedy* yang memakan waktu 8,2643 ms dan memori 22,28 KB. Kesimpulannya, algoritma Dijkstra sangat direkomendasikan karena terbukti lebih presisi, stabil, dan efisien secara operasional untuk diimplementasikan pada sistem navigasi logistik berskala luas.

Kata Kunci: Algoritma Dijkstra, Algoritma *Greedy*, Distribusi Logistik, Rute Terpendek, Teori Graf

I. PENDAHULUAN

Kelancaran sistem logistik dan rantai pasok memiliki dampak besar terhadap pertumbuhan ekonomi nasional. Dalam operasionalnya, efisiensi pengiriman menjadi kunci utama untuk menekan biaya dan mempertahankan daya saing perusahaan. Sayangnya, kegiatan distribusi sering kali terhambat oleh rute yang kurang efektif dan kondisi jalan yang bervariasi, terutama di kawasan perkotaan yang sibuk seperti Palembang. Pengalaman atau insting pengemudi yang selama ini diandalkan sering memunculkan bias dalam memperkirakan jarak tempuh yang sebenarnya. Merujuk pada kajian Widodo et al., (2023), kesalahan penentuan rute bisa menyebabkan lonjakan biaya operasional hingga 30%. Keterlambatan ini juga berisiko melanggar *Service Level Agreement* (SLA) dan menurunkan tingkat kepercayaan pelanggan (Fayaqun, 2024). Untuk mengatasi kendala tersebut, diperlukan solusi digital yang mampu merencanakan rute secara matematis dan presisi.

Salah satu pendekatan yang terbukti tangguh untuk memetakan jalur distribusi secara otomatis adalah penerapan Teori Graf. Dua metode yang paling umum dievaluasi dalam pencarian lintasan terpendek adalah algoritma *Greedy* dan Dijkstra, yang masing-masing menawarkan pola komputasi yang kontras. Algoritma *Greedy* memproses data berdasarkan keputusan terbaik di setiap titik simpul (*local optimum*), sehingga waktu eksekusinya tergolong sangat cepat. Meski begitu, rute akhir yang dihasilkan algoritma ini tidak selalu menjamin jarak yang paling efisien secara keseluruhan (Iskandar & Riti, 2022). Sebaliknya, algoritma Dijkstra mengevaluasi semua kemungkinan rute untuk memastikan hasil yang paling presisi, namun karakteristik penelusuran ini menuntut kapasitas komputasi yang lebih besar (Lakutu et al., 2023). Beban komputasi ini sering kali menjadi kendala tersendiri, mengingat kurir logistik umumnya hanya mengandalkan ponsel pintar (*mobile devices*) sebagai alat navigasi utama.

Kinerja kedua algoritma tersebut juga sangat bergantung pada struktur topologi wilayah distribusinya. Riset terdahulu, seperti yang dilakukan Sulistyowati et al., (2025), mencatat bahwa penggunaan algoritma *Greedy* sangat menguntungkan pada distribusi skala lokal karena proses responsnya yang instan. Di sisi lain, Harahap & Triase, (2024) menekankan bahwa navigasi di kawasan padat lalu lintas jauh lebih membutuhkan kestabilan jarak yang ditawarkan oleh algoritma Dijkstra. Perlu digarisbawahi bahwa temuan dari berbagai studi tersebut tidak bisa langsung digeneralisasi pada sistem logistik yang kompleks, khususnya pada rute yang melibatkan banyak titik pengantaran (*drop-point*) dalam satu kali perjalanan. Hingga saat ini, masih terdapat celah penelitian terkait bagaimana performa aktual kedua algoritma ini saat dihadapkan pada jaringan logistik berskala luas.

p-ISSN : 2406-7733
e-ISSN : 2597-9922

Pemilihan algoritma secara sembarangan tanpa metrik yang terukur berisiko memunculkan masalah baru, seperti aplikasi yang lambat atau rekomendasi rute yang meleset. Untuk itu, penelitian ini bertujuan menganalisis membandingkan kinerja algoritma Dijkstra dan *Greedy*. Fokus pengujian akan diarahkan pada aspek kecepatan waktu pemrosesan (*execution time*), efisiensi alokasi memori, serta kalkulasi total jarak tempuh (*total distance*). Hasil analisis ini diharapkan dapat memberikan rekomendasi teknis berbasis data mengenai algoritma navigasi logistik yang paling tangguh, stabil, dan siap diimplementasikan.

II. TINJAUAN PUSTAKA

Sistem Logistik

Sistem logistik merupakan proses pengelolaan aliran barang, informasi, dan sumber daya dari titik asal hingga sampai ke konsumen akhir. Dalam kegiatan logistik, distribusi menjadi salah satu aspek yang sangat penting karena berhubungan langsung dengan ketepatan waktu pengiriman dan efisiensi biaya operasional. Perencanaan distribusi yang baik dapat membantu perusahaan meningkatkan kualitas layanan sekaligus mempertahankan daya saing di tengah persaingan bisnis yang semakin ketat. Oleh karena itu, optimasi rute pengiriman menjadi salah satu faktor utama dalam mendukung efektivitas sistem logistik (Setyadi, 2024).

Teori Graf

Teori graf merupakan cabang ilmu matematika yang digunakan untuk memodelkan hubungan antar objek dalam bentuk simpul (*vertex*) dan sisi (*edge*). Dalam sistem transportasi dan distribusi, setiap lokasi dapat direpresentasikan sebagai simpul, sedangkan jalan yang menghubungkan lokasi tersebut direpresentasikan sebagai sisi yang memiliki bobot tertentu, seperti jarak atau waktu tempuh. Melalui representasi graf, berbagai permasalahan pencarian rute dapat diselesaikan secara sistematis menggunakan algoritma pencarian lintasan terpendek. Oleh karena itu, teori graf menjadi dasar penting dalam pengembangan sistem navigasi dan optimasi distribusi logistik (Gajjar et al., 2024).

Algoritma *Greedy*

Algoritma *Greedy* merupakan metode penyelesaian masalah yang bekerja dengan memilih alternatif terbaik pada setiap langkah berdasarkan informasi yang tersedia saat itu. Pendekatan ini dikenal sebagai konsep *local optimum*, yaitu memilih solusi yang dianggap paling menguntungkan pada kondisi tertentu tanpa mengevaluasi seluruh kemungkinan solusi secara keseluruhan. Keunggulan utama algoritma *Greedy* terletak pada proses komputasinya yang relatif sederhana dan cepat sehingga cocok digunakan pada aplikasi yang membutuhkan respons dalam waktu singkat. Namun, karena hanya berfokus pada keputusan lokal, algoritma ini tidak selalu menghasilkan solusi terbaik secara global (*global optimum*) (Szeszlér, 2022).

Algoritma Dijkstra

Algoritma Dijkstra merupakan salah satu algoritma yang paling banyak digunakan untuk menyelesaikan permasalahan lintasan terpendek pada graf berbobot positif. Algoritma ini bekerja dengan menghitung jarak minimum dari simpul awal menuju simpul lainnya melalui proses evaluasi bertahap terhadap seluruh jalur yang memungkinkan. Dibandingkan dengan algoritma *Greedy*, Dijkstra mampu menghasilkan solusi yang lebih optimal karena mempertimbangkan akumulasi bobot pada setiap jalur. Meskipun demikian, proses perhitungan yang lebih kompleks menyebabkan algoritma ini membutuhkan sumber daya komputasi yang lebih besar (Awalloedin & Gata, 2022).

Shortest Path

Shortest path atau lintasan terpendek merupakan permasalahan yang bertujuan untuk menemukan jalur dengan total bobot minimum antara dua simpul pada suatu graf. Bobot yang digunakan dapat berupa jarak, waktu tempuh, biaya perjalanan, maupun parameter lainnya sesuai kebutuhan sistem. Konsep *shortest path* banyak diterapkan pada berbagai bidang, seperti sistem navigasi, transportasi, dan distribusi logistik. Dalam konteks logistik, pencarian lintasan terpendek dapat membantu perusahaan menentukan rute pengiriman yang lebih efisien sehingga waktu distribusi dan biaya operasional dapat diminimalkan (Taneo et al., 2025).

Pengukuran Kinerja Algoritma

Pengukuran kinerja algoritma dilakukan untuk mengetahui tingkat efisiensi dan efektivitas suatu algoritma dalam menyelesaikan permasalahan tertentu. Evaluasi kinerja diperlukan untuk membandingkan kemampuan algoritma dalam menghasilkan solusi yang optimal dengan penggunaan sumber daya yang tersedia. Pada penelitian ini, pengukuran kinerja dilakukan berdasarkan tiga parameter utama, yaitu waktu pemrosesan (*execution time*), penggunaan memori (*memory usage*), dan total jarak tempuh (*total distance*) yang dihasilkan oleh algoritma.

1. Execution Time

Execution time merupakan waktu yang dibutuhkan oleh algoritma untuk menyelesaikan proses komputasi sejak data mulai diproses hingga menghasilkan keluaran. Parameter ini digunakan untuk mengukur tingkat kecepatan algoritma dalam

p-ISSN : 2406-7733
e-ISSN : 2597-9922

menyelesaikan suatu permasalahan. Semakin kecil nilai *execution time* yang diperoleh, maka semakin cepat algoritma tersebut dalam menghasilkan solusi. Pada sistem logistik yang memerlukan respons secara cepat, waktu pemrosesan menjadi salah satu indikator penting dalam menentukan kualitas kinerja algoritma (Ula et al., 2026).

2. *Memory Usage*

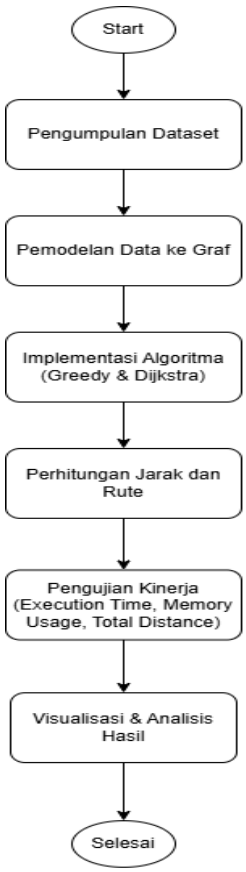
Memory usage adalah jumlah memori yang digunakan oleh algoritma selama proses eksekusi berlangsung. Pengukuran penggunaan memori bertujuan untuk mengetahui tingkat efisiensi algoritma dalam memanfaatkan sumber daya perangkat. Algoritma yang mampu menghasilkan solusi dengan penggunaan memori yang lebih kecil cenderung lebih sesuai untuk diterapkan pada perangkat dengan kapasitas terbatas, seperti telepon pintar yang digunakan dalam sistem navigasi logistic (Hadiewijaya & Wasito, 2024).

3. *Total Distance*

Total distance merupakan total jarak tempuh yang dihasilkan oleh algoritma dalam menentukan rute perjalanan dari titik awal menuju titik tujuan. Parameter ini digunakan untuk mengevaluasi tingkat efisiensi rute yang dihasilkan oleh suatu algoritma. Semakin kecil total jarak yang diperoleh, maka semakin efisien rute yang dihasilkan karena dapat mengurangi konsumsi bahan bakar, mempercepat waktu pengiriman, dan menekan biaya operasional distribusi (Hadiewijaya & Wasito, 2024).

III. METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah metode kuantitatif dengan pendekatan eksperimen. Metode ini bertujuan untuk menganalisis dan membandingkan kinerja algoritma *Greedy* dan Dijkstra dalam menentukan rute terpendek pada sistem distribusi logistik. Pendekatan ini penting untuk memperoleh hasil pengujian yang objektif berdasarkan parameter yang telah ditentukan, sehingga dapat diketahui algoritma yang lebih efektif untuk diterapkan pada proses distribusi. Tahapan penelitian meliputi pengumpulan data, pemodelan jaringan distribusi dalam bentuk graf berbobot, implementasi algoritma *Greedy* dan Dijkstra, pengujian kinerja algoritma, serta evaluasi hasil berdasarkan parameter *execution time*, *memory usage*, dan *total distance*. Tahapan-tahapan penelitian yang dilakukan ditunjukkan pada gambar 1.



Gambar 1. Tahapan Penelitian

Pengumpulan Dataset

Dataset yang digunakan dalam penelitian ini merupakan *dataset* yang diambil dari platform Kaggle yang bersifat *open* publik berisi data-data kota yang ada di Indonesia. *Dataset* dibangun secara manual berdasarkan koordinat geografis nyata yang diperoleh melalui *Google Maps API*, kemudian disusun dalam format adjacency list untuk memudahkan proses pemodelan ke dalam struktur graf. Proses pengumpulan data dilakukan dengan mempertimbangkan representativitas jaringan jalan yang mencakup berbagai kondisi.

Pemodelan Data ke Graf

Data distribusi yang telah dikumpulkan kemudian dikonversi ke dalam representasi graf berbobot tak berarah (*undirected weighted graph*). Setiap titik lokasi pengiriman direpresentasikan sebagai simpul (*vertex*), sedangkan jalur penghubung antar lokasi direpresentasikan sebagai sisi (*edge*) dengan bobot berupa jarak dalam kilometer. Pemodelan dilakukan menggunakan struktur data *adjacency list* dalam bahasa pemrograman Python, yang dipilih karena efisiensinya dalam merepresentasikan graf jarang (*sparse graph*) yang umum dijumpai pada jaringan distribusi skala kota. Visualisasi awal graf juga dilakukan menggunakan pustaka *NetworkX* untuk

p-ISSN : 2406-7733
e-ISSN : 2597-9922

memverifikasi kebenaran struktur graf sebelum proses implementasi algoritma dilakukan.

Implementasi Algoritma

Kedua algoritma diimplementasikan secara mandiri menggunakan bahasa pemrograman Python tanpa memanfaatkan fungsi bawaan pustaka graf, guna memastikan konsistensi pengukuran kinerja. Algoritma *Greedy* diimplementasikan dengan pendekatan *nearest neighbor*, yaitu pada setiap simpul yang dikunjungi, algoritma selalu memilih sisi dengan bobot terkecil yang menuju simpul yang belum dikunjungi. Sementara itu, algoritma Dijkstra diimplementasikan menggunakan struktur data *priority queue* berbasis *min-heap* menggunakan modul *heapq* untuk memastikan efisiensi pemilihan simpul dengan jarak minimum pada setiap iterasi. Kedua implementasi mencatat jalur yang dilalui dan total bobot kumulatif sebagai luaran utama, yang selanjutnya digunakan dalam tahap pengujian kinerja.

Algoritma *Greedy* memilih simpul berikutnya berdasarkan bobot sisi terkecil dari simpul yang sedang dikunjungi. Secara matematis, pemilihan simpul berikutnya dapat diformulasikan pada persamaan 1.

$$v^* = argmin\,\{w(u,v)|v \in V \backslash S\} \qquad (1)$$

di mana *v** adalah simpul tujuan yang dipilih, *w(u, v)* adalah bobot sisi dari simpul *u* ke simpul *v*, *V* adalah himpunan seluruh simpul dalam graf, dan *S* adalah himpunan simpul yang telah dikunjungi. Total jarak yang dihasilkan *Greedy* dihitung dengan menjumlahkan seluruh bobot sisi pada rute yang terbentuk seperti yang ditunjukkan pada Persamaan 2.

$$D_{Greedy} = \sum w\,(v_i, v_i + 1), i = 0{,}1, \dots, n1 \qquad (2)$$

Sedangkan algoritma Dijkstra menghitung jarak kumulatif terpendek dari simpul sumber ke setiap simpul lainnya secara iteratif. Pembaruan jarak dilakukan dengan relaksasi sisi seperti pada Persamaan 3.

$$d[v] = min(d[v], d[u] + w(u,v) \qquad (3)$$

di mana *d[v]* adalah estimasi jarak terpendek dari simpul sumber ke simpul *v*, *d[u]* adalah jarak terpendek yang sudah dikonfirmasi ke simpul *u*, dan *w(u, v)* adalah bobot sisi dari simpul *u* ke simpul *v*. Proses relaksasi ini diulang hingga seluruh simpul telah diproses, sehingga nilai *d[v]* pada akhir eksekusi merupakan jarak terpendek yang dijamin optimal dari simpul sumber ke simpul tujuan.

Perhitungan Jarak dan Rute

Setelah kedua algoritma berhasil menentukan rute, dilakukan perhitungan total jarak tempuh (*total distance*) dengan menjumlahkan seluruh bobot sisi yang membentuk rute yang dihasilkan. Rute yang diperoleh dari masing-masing algoritma kemudian dibandingkan untuk mengetahui selisih efisiensi jarak yang dihasilkan. Perbandingan ini mencakup analisis konsistensi rute pada berbagai kombinasi titik asal dan tujuan yang diujikan, sehingga penilaian tidak hanya didasarkan pada satu skenario pengiriman saja, melainkan mencerminkan performa rata-rata pada berbagai kondisi jaringan distribusi yang berbeda.

Pengujian Kinerja

Pengujian kinerja dilakukan berdasarkan tiga parameter utama, yaitu *execution time*, *memory usage*, dan *total distance*. Pengukuran *execution time* dilakukan menggunakan fungsi *time.perf_counter()* yang dicatat sebelum dan setelah eksekusi fungsi algoritma. Pengukuran *memory usage* dilakukan menggunakan modul *tracemalloc* bawaan Python untuk memantau alokasi memori selama proses eksekusi berlangsung. Setiap skenario pengujian diulang sebanyak sepuluh kali untuk mendapatkan nilai rata-rata yang stabil dan meminimalkan pengaruh fluktuasi kinerja sistem. Pengujian dilakukan pada perangkat dengan spesifikasi yang sama untuk seluruh percobaan guna memastikan validitas perbandingan.

Execution time dihitung sebagai selisih waktu sebelum dan sesudah eksekusi algoritma, kemudian dirata-rata dari seluruh pengulangan berdasarkan Persamaan 4 dan Persamaan 5.

$$T_i = t_{end_i} - t_{start_i} \qquad (4)$$

$$T = \left(\frac{1}{n}\right) \sum T_i \; i = 1, 2, \dots, n \qquad (5)$$

di mana T_i adalah waktu eksekusi pada percobaan ke-i, t_{end_i} dan t_{start_i} adalah waktu akhir dan awal eksekusi, serta $\underline{T}$ adalah rata-rata *execution time* dari n percobaan.

Memory usage dihitung sebagai selisih antara penggunaan memori puncak saat algoritma berjalan dengan kondisi memori sebelum eksekusi dimulai, kemudian dirata-rata seperti pada Persamaan 6 dan Persamaan 7.

$$M_i = m_{peak_i} - m_{before_i} \qquad (6)$$

$$\underline{M} = \left(\frac{1}{n}\right) \sum M_i \; i = 1, 2, \dots, n \qquad (7)$$

di mana M_i adalah penggunaan memori pada percobaan ke-i, m_{peak_i} adalah puncak alokasi

p-ISSN : 2406-7733
e-ISSN : 2597-9922

memori, m_{before_i} adalah memori sebelum eksekusi, dan $\underline{M}$ adalah rata-rata *memory usage* dari n percobaan.

Adapun *total distance* diperoleh langsung dari akumulasi bobot sisi pada rute yang dihasilkan masing-masing algoritma sebagaimana pada Persamaan 2. Untuk mengukur selisih efisiensi jarak antara kedua algoritma, digunakan rumus persentase efisiensi yang ditunjukkan pada Persamaan 8.

$$E = \left(D_{Greedy} - \frac{D_{Dijkstra}}{D_{Dijkstra}} \right) \times 100\% \qquad (8)$$

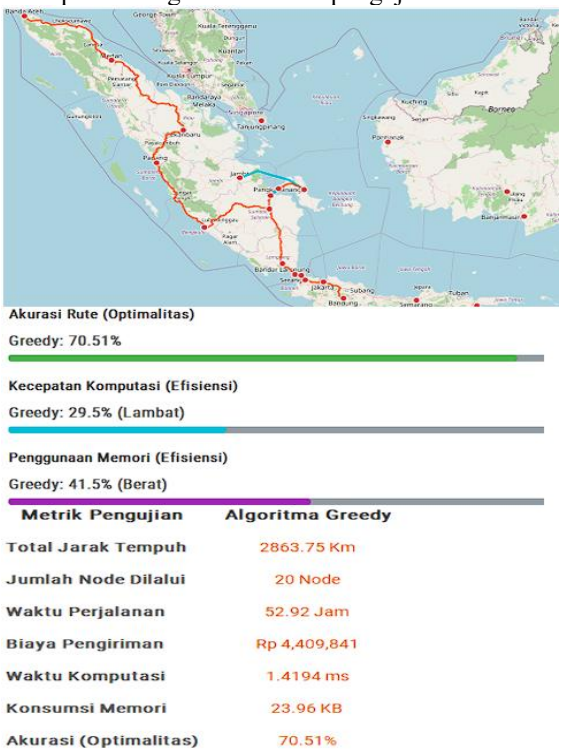
di mana E adalah persentase selisih efisiensi jarak, D_{Greedy} adalah total jarak yang dihasilkan algoritma *Greedy*, dan $D_{Dijkstra}$ adalah total jarak yang dihasilkan algoritma Dijkstra. Nilai E yang positif menunjukkan bahwa Dijkstra menghasilkan rute yang lebih efisien dibandingkan *Greedy*.

Visualisasi dan Analisis Hasil

Hasil pengujian divisualisasikan melalui antarmuka sistem berbasis web yang dikembangkan secara khusus untuk penelitian ini. Sistem menampilkan tiga komponen visualisasi utama: (1) peta interaktif jaringan distribusi 35 kota yang memperlihatkan jalur rute *Greedy* (warna oranye) dan Dijkstra (warna biru) secara bersamaan; (2) tabel perbandingan metrik pengujian yang merangkum nilai total jarak tempuh, jumlah *node* dilalui, waktu komputasi, konsumsi memori, dan akurasi optimalitas dari kedua algoritma; serta (3) panel analisis performa berupa indikator batang yang menampilkan perbandingan relatif kecepatan komputasi dan penggunaan memori secara visual. Analisis dilakukan secara deskriptif-komparatif dengan membandingkan nilai ketiga parameter kinerja utama antara algoritma *Greedy* dan Dijkstra. Dari hasil analisis tersebut, ditarik kesimpulan mengenai kelebihan dan kekurangan masing-masing algoritma dalam konteks penerapannya pada sistem distribusi logistik, serta rekomendasi pemilihan algoritma yang paling sesuai berdasarkan kebutuhan operasional di lapangan.

IV. HASIL DAN PEMBAHASAN

Pada penelitian ini, diimplementasikan sebuah sistem pencarian rute berbasis antarmuka grafis (GUI) menggunakan Python, yang difungsikan khusus untuk membandingkan algoritma *Greedy* dan Dijkstra secara komprehensif. Melalui antarmuka yang interaktif, sistem ini secara langsung menyajikan visualisasi peta jaringan distribusi, membandingkan hasil pencarian rute terbaik, dan menampilkan metrik kinerja dari masing-masing algoritma. Sebagai bahan pembahasan utama terkait efisiensi kedua algoritma tersebut, sistem diuji menggunakan data pemodelan jaringan logistik pada 35 kota di Indonesia, di mana pencarian rute dari Banda Aceh menuju Bandung ditetapkan sebagai studi kasus pengujian.



Gambar 2. Tampilan GUI Sistem Pencarian Rute Algoritma *Greedy*



Gambar 3. Tampilan GUI Sistem Pencarian Rute Algoritma Dijkstra

p-ISSN : 2406-7733
e-ISSN : 2597-9922

Berdasarkan simulasi pencarian rute yang divisualisasikan pada antarmuka sistem, kinerja kedua algoritma dievaluasi secara komprehensif. Hasil rekapitulasi pengujian metrik kinerja algoritma pada skenario rute Banda Aceh menuju Bandung dirangkum pada Tabel 1.

Tabel 1. Perbandingan Metrik Kinerja Algoritma Dijkstra dan *Greedy*

Metrik Pengujian	Algoritma Dijkstra	Algoritma *Greedy*
Total Jarak Tempuh	2019,20 Km	2863,75 Km
Jumlah Node Dilalui	10 Node	20 Node
Waktu Perjalanan	34,60 Jam	52,92 Jam
Biaya Pengiriman	Rp 3.143.025	Rp 4.409.841
Waktu Komputasi	0,4889 ms	8,2643 ms
Konsumsi Memori	16,33 KB	22,28 KB
Akurasi (Optimalitas)	100,00% (Optimal)	70,51%

Berdasarkan Tabel 1 dan visualisasi pada sistem, terdapat perbedaan kinerja yang sangat signifikan antara kedua algoritma. Dalam hal penentuan rute, Algoritma Dijkstra terbukti 100% optimal dengan menghasilkan total jarak tempuh terpendek sejauh 2019,20 Km yang hanya melewati 10 *node* (simpul). Visualisasi rute Dijkstra menunjukkan pola jalur yang lebih terarah dengan menyusuri pesisir timur Pulau Sumatera. Sebaliknya, Algoritma *Greedy* menghasilkan jarak tempuh yang jauh lebih panjang, yakni 2863,75 Km melewati 20 *node*, dengan tingkat akurasi rute hanya mencapai 70,51%. Rute yang dihasilkan oleh *Greedy* terlihat lebih memutar (*detour*) hingga melewati sisi barat Sumatera dan Kepulauan Bangka Belitung sebelum akhirnya mencapai titik tujuan di Pulau Jawa. Hal ini terjadi karena sifat dasar algoritma *Greedy* yang hanya mencari nilai optimum lokal (*local optimum*) pada setiap persimpangan simpul tanpa mengevaluasi total bobot lintasan secara keseluruhan.

Ketidakmampuan algoritma *Greedy* dalam menemukan rute global yang optimal berdampak langsung pada tingginya inefisiensi operasional. Rute dari algoritma *Greedy* membutuhkan estimasi waktu perjalanan yang jauh lebih lama (52,92 jam dibandingkan 34,60 jam pada Dijkstra) serta menyebabkan lonjakan biaya pengiriman menjadi Rp 4.409.841. Angka ini berselisih cukup jauh dibandingkan rute Dijkstra yang hanya memakan biaya Rp 3.143.025.

Selain unggul dalam penentuan lintasan (efisiensi jarak dan biaya), Algoritma Dijkstra juga menunjukkan performa komputasi yang lebih superior pada metrik waktu pemrosesan (*execution time*) dan penggunaan memori (*memory usage*). Waktu komputasi Dijkstra tercatat sangat cepat, yakni hanya 0,4889 ms, berbanding jauh dengan *Greedy* yang memakan waktu 8,2643 ms. Dari segi penggunaan sumber daya, konsumsi memori Dijkstra tergolong lebih ringan (16,33 KB) dibandingkan *Greedy* (22,28 KB). Tingginya beban waktu komputasi dan konsumsi memori pada algoritma *Greedy* dalam studi kasus ini berbanding lurus dengan banyaknya jumlah *node* yang harus dievaluasi (20 *node*) akibat pemilihan jalur optimum lokal yang keliru di awal iterasi. Secara keseluruhan, Algoritma Dijkstra terbukti jauh lebih presisi, stabil, dan efisien untuk diimplementasikan pada sistem navigasi logistik berskala luas.

V. KESIMPULAN DAN SARAN

Kesimpulan

Berdasarkan hasil pengujian dan analisis terhadap kinerja algoritma Dijkstra dan *Greedy* pada studi kasus pencarian rute distribusi logistik, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Optimalitas dan Efisiensi Jarak: Algoritma Dijkstra terbukti 100% optimal dalam menentukan rute terpendek, menghasilkan total jarak 2019,20 km (melewati 10 *node*). Algoritma *Greedy* gagal mencapai hasil global yang optimal karena hanya berfokus pada titik optimum lokal pada setiap iterasi, sehingga menghasilkan jarak yang lebih jauh, yakni 2863,75 km (melewati 20 *node*) dengan akurasi hanya 70,51%.
2. Kinerja Komputasi: Berlawanan dengan asumsi teoretis yang sering menyebut *Greedy* lebih ringan, pada skala jaringan spasial ini, algoritma Dijkstra justru lebih efisien. Waktu pemrosesan Dijkstra tercatat sangat cepat (0,4889 ms) dengan konsumsi memori yang ringan (16,33 KB), jauh mengungguli algoritma *Greedy* yang membutuhkan waktu 8,2643 ms dan memori 22,28 KB. Tingginya beban komputasi *Greedy* diakibatkan oleh banyaknya jumlah simpul tidak perlu yang dikunjungi.
3. Dampak Operasional: Penerapan algoritma Dijkstra mampu memangkas estimasi waktu perjalanan secara signifikan (34,60 jam) dan menekan biaya pengiriman hingga menjadi Rp 3.143.025. Di sisi lain, rute dari algoritma *Greedy* menyebabkan inefisiensi waktu (52,92 jam) dan pembengkakan biaya hingga Rp 4.409.841. Secara keseluruhan, algoritma Dijkstra merupakan pilihan terbaik yang stabil dan akurat untuk sistem navigasi logistik berskala luas.

p-ISSN : 2406-7733
e-ISSN : 2597-9922

Saran

Untuk penelitian dan pengembangan selanjutnya, terdapat beberapa saran yang dapat dilakukan guna menyempurnakan sistem penentuan rute distribusi ini:

1. Penerapan Bobot Dinamis (*Dynamic Weighting*): Penelitian ini menggunakan bobot jarak statis antarkota. Ke depannya, disarankan untuk mengintegrasikan data lalu lintas waktu nyata (*real-time traffic*), kondisi jalan, atau hambatan cuaca sebagai bobot tambahan agar simulasi rute lebih mencerminkan kondisi riil di lapangan.
2. Skenario *Multi-Drop Point*: Mengingat kurir logistik sering kali harus mengantarkan banyak paket sekaligus dalam satu rute perjalanan, pengujian selanjutnya dapat dikembangkan ke dalam penyelesaian kasus *Vehicle Routing Problem* (VRP) atau *Traveling Salesperson Problem* (TSP) dengan banyak titik henti (*multiple destinations*).
3. Komparasi dengan Algoritma Heuristik Lain: Algoritma Dijkstra menuntut evaluasi kumulatif yang presisi. Untuk menangani graf berukuran lebih raksasa (skala nasional hingga ke tingkat jalan terkecil), disarankan untuk membandingkan kinerjanya dengan algoritma heuristik kecerdasan buatan, seperti Algoritma A* (A-Star) atau *Ant Colony Optimization*, guna melihat keseimbangan optimal antara kecepatan dan ketepatan rute.

DAFTAR PUSTAKA

Awalloedin, N., & Gata, W. (2022). Algoritma Dijkstra dalam penentuan rute terpendek pada jalan raya antar kota Jakarta - Tangerang. *Jurnal Sistem Informasi, Teknologi Informasi dan Komputer*, *13*(1), 8–13.

Fayaqun, R. (2024). Penerapan algoritma Dijkstra dalam pendistribusian paket permakanan bagi lansia tunggal dan penyandang disabilitas tunggal di Kelurahan Klampok Kota Blitar. *Jurnal Teknik Industri Terintegrasi*, *7*(3), 1867–1873.

Gajjar, K., Vibhuti, A., Manish, J., & Abhiruk, K. (2024). Reconfiguring shortest paths in graphs. *Algorithmica*, *86*(10), 3309–3338. https://doi.org/10.1007/s00453-024-01263-y.

Hadiewijaya, A., & Wasito, B. (2024). Perbandingan efisiensi waktu dan memori pada C# dan Java dengan metode benchmarking. *Journal Binary Digital – Technology, 7*(2). https://doi.org/10.32877/bt.v7i2.1906

Harahap, A. S., & Triase. (2024). Algoritma Greedy untuk mendukung keputusan pemilihan rute distribusi bantuan tercepat pasca banjir. *Jurnal Sistem Informasi, 13*, 1689–1704.

Iskandar, J. S., & Riti, Y. F. (2022). Perbandingan algoritma Greedy dan algoritma Dijkstra dalam pencarian rute terpendek dari Kabupaten Tuban ke Kota Surabaya. *Jurnal Pendidikan Teknologi Informasi dan Komunikasi*, *8*(2), 96–106.

Lakutu, N. F., Katili, M. R., Mahmud, S. L., & Imansyah, N. (2023). Algoritma Dijkstra dan algoritma Greedy untuk optimasi rute pengiriman barang pada Kantor Pos Gorontalo. *Jurnal Ilmiah Matematika, Sains dan Teknologi, 11*(1), 55–65.

Setyadi, W. (2024). Sistem informasi logistik untuk optimalisasi pengelolaan stok barang pada toko bangunan. *Journal Binary Digital - Technology, 7*(2). https://doi.org/10.32877/bt.v7i2.1776**.**

Sulistyowati, HS., Nur, M., & Fajri, D. (2025). Implementasi algoritma Dijkstra untuk menemukan rute terdekat pengambilan barang bekas berbasis web pada lapak rongsok Brosot. *Jurnal Teknologi Informatika & Komputer, 2*(2), 30–45.

Szeszlér, D. (2022). Sufficient conditions for the optimality of the greedy algorithm in greedoids. *Journal of Combinatorial Optimization*, *44*(1), 287–302. https://doi.org/10.1007/s10878-021-00833-y.

Taneo, R. E., Ndun, R., Fallo, D. Y. A., & Do, F. (2025). Optimasi jalur terpendek menggunakan algoritma Dijkstra dan Greedy pada sistem informasi geografis. *Jurnal Kridatama Sains dan Teknologi, 7*(01), 572–582.

Ula, N. M., Aulia, G. S., Suraya, R. G., Firmansyah, A., Prayogo, I., & Pujiono. (2026). Optimasi pengurutan data : Perbandingan waktu eksekusi Bubble vs Insertion vs Heap sort. *Jurnal Pengembangan dan Observasi Sistem Komputer, 13*(1).

Widodo, E. M., Rifa, A., & Fuadyani, R. (2023). Sistem distribusi logistik dengan pendekatan sistem dinamik untuk mitigasi bencana Gunung Merapi. *Jurnal Manajemen dan Teknik Industri-Produksi, 23*(2). https://doi.org/10.350587/Matrik.