ANALISIS PENERAPAN ALGORITMA DIJKSTRA UNTUK OPTIMASI RUTE DISTRIBUSI LOGISTIK PADA INTEGRASI TOL JABODETABEK: STUDI KASUS TANJUNG PRIOK TAHUN 2025

Tugas Ini Dibuat Untuk Memenuhi Mata Kuliah Riset Operasi

Dosen Pengampu: Desi Novianti

Disusun Oleh:

1. Afifah Nuraini (220511027)
2. Aviana Nurfasikha (220511130)
3. Rulastri (220511071)

Kelompok 4

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH CIREBON

2025

ABSTRAK

Distribusi logistik memiliki peran strategis dalam menjaga kelancaran rantai pasok nasional, terutama pada wilayah industri yang terhubung dengan pelabuhan besar seperti Tanjung Priok. Namun, pada tahun 2025 terjadi peningkatan volume kendaraan angkutan barang yang menimbulkan kepadatan lalu lintas pada akses tol Jabodetabek, sehingga memperlambat pergerakan logistik dan menimbulkan biaya operasional yang lebih tinggi bagi pelaku industri. Masalah tersebut menunjukkan adanya kebutuhan untuk menentukan jalur distribusi yang lebih optimal agar waktu tempuh dan penggunaan sumber daya dapat diminimalkan.

Penelitian ini menerapkan Algoritma Dijkstra sebagai metode penentuan jalur distribusi tercepat dari beberapa titik industri menuju Pelabuhan Tanjung Priok. Pemodelan jaringan jalan direpresentasikan dalam bentuk graf berbobot, di mana setiap simpul menggambarkan gerbang tol atau titik persimpangan, dan setiap sisi mewakili ruas jalan dengan bobot waktu tempuh atau jarak. Proses simulasi dilakukan untuk membandingkan rute yang diperoleh melalui algoritma dengan rute konvensional yang biasa dilalui kendaraan logistik.

Hasil awal menunjukkan bahwa penggunaan Algoritma Dijkstra mampu memberikan rekomendasi jalur dengan bobot perjalanan yang lebih rendah, sehingga berpotensi mengurangi waktu tempuh dan meningkatkan efisiensi distribusi barang. Penelitian ini juga memberikan gambaran bahwa pendekatan berbasis algoritma mampu mendukung perencanaan rute logistik secara lebih sistematis. Temuan ini diharapkan menjadi referensi dalam upaya integrasi jaringan tol Jabodetabek serta bahan pertimbangan dalam pengambilan keputusan terkait pengelolaan arus distribusi menuju Pelabuhan Tanjung Priok.

***Kata Kunci**: Algoritma Dijkstra, Optimasi Rute, Distribusi Logistik, Jalan Tol Jabodetabek, Pelabuhan Tanjung Priok.*

BAB I

PENDAHULUAN

1.1. Latar Belakang

Aktivitas distribusi logistik merupakan salah satu elemen penting dalam menjaga stabilitas pasokan barang di Indonesia, khususnya pada wilayah dengan tingkat aktivitas ekonomi tinggi seperti Jabodetabek. Kawasan ini berfungsi sebagai pusat industri, perdagangan, serta hub transportasi yang terintegrasi dengan Pelabuhan Tanjung Priok sebagai pintu keluar masuk terbesar distribusi ekspor–impor nasional. Kelancaran arus barang menuju pelabuhan sangat mempengaruhi efektivitas rantai pasok nasional, serta berimplikasi langsung terhadap biaya produksi, harga komoditas, hingga daya saing logistik Indonesia di kawasan internasional.

Meskipun memiliki peran vital, permasalahan kemacetan masih menjadi tantangan utama di area akses menuju Pelabuhan Tanjung Priok. Lonjakan kendaraan logistik pasca-Lebaran tahun 2025 dilaporkan menimbulkan antrean truk yang cukup signifikan pada jalur tol dan jalan distribusi menuju pelabuhan. Kondisi tersebut memperlambat waktu tempuh pengiriman, menyebabkan pemborosan bahan bakar, serta menambah biaya operasional armada pengangkut barang. Penundaan yang terjadi tidak hanya berdampak pada waktu bongkar muat, tetapi juga berpengaruh pada ketepatan jadwal pengiriman barang ke berbagai daerah dan keperluan ekspor. Dengan demikian, efisiensi rute distribusi merupakan isu kritis yang memerlukan solusi nyata.

Upaya antisipatif sebenarnya telah dilakukan Pemerintah bersama pemangku kepentingan logistik melalui rencana integrasi jaringan tol Jabodetabek, termasuk pengembangan ruas Jakarta Outer Ring Road (JORR) dan penyelesaian Jalan Tol Cibitung–Cilincing (JTCC). Integrasi jalur ini diharapkan dapat memberi alternatif rute menuju pelabuhan serta memecah konsentrasi arus kendaraan pada jalur-jalur tertentu. Akan tetapi, ketersediaan infrastruktur fisik belum sepenuhnya menjamin kelancaran distribusi apabila tidak dibarengi dengan sistem penentuan rute yang adaptif, cepat, dan efisien.

Perkembangan teknologi komputasi dan algoritma graf memberikan peluang untuk menyelesaikan masalah tersebut melalui pendekatan optimasi jalur. Salah satu algoritma yang banyak digunakan dalam pencarian rute terpendek adalah Algoritma Dijkstra, yang bekerja dengan menganalisis graf berbobot untuk menghasilkan lintasan dengan jarak atau waktu tempuh minimum dari satu titik ke titik lainnya. Dengan memodelkan jaringan jalan tol Jabodetabek sebagai graf, setiap gerbang tol atau simpul jalan dapat dianalisis secara matematis sehingga diperoleh rekomendasi rute paling efisien secara objektif.

Oleh karena itu, penelitian ini mencoba mengkaji penerapan Algoritma Dijkstra dalam optimasi rute distribusi logistik menuju Pelabuhan Tanjung Priok. Hasil kajian diharapkan mampu memberikan gambaran efektivitas penggunaan algoritma dalam membantu pengambilan keputusan distribusi, serta meningkatkan kelancaran arus barang yang selama ini terkendala kemacetan. Kajian ini juga menjadi bentuk kontribusi akademis terhadap pengembangan pemanfaatan teknologi dalam bidang transportasi logistik nasional.

1.2. Rumusan Masalah

Agar penelitian memiliki arah yang jelas, maka dirumuskan beberapa pertanyaan utama sebagai berikut:

1. Bagaimana cara membangun model jaringan jalan tol menuju Pelabuhan Tanjung Priok dalam bentuk graf yang dapat digunakan untuk proses optimasi rute?

2. Bagaimana penerapan Algoritma Dijkstra dalam menentukan jalur tercepat dari berbagai titik industri atau simpul jalan di wilayah Jabodetabek menuju Pelabuhan Tanjung Priok?

3. Seberapa besar peningkatan efisiensi waktu tempuh distribusi yang dihasilkan ketika menggunakan rute hasil optimasi dibanding rute konvensional yang biasa dilalui truk logistik?

4. Faktor apa saja yang menjadi batasan penggunaan algoritma dalam kondisi nyata, seperti dinamika kemacetan, tarif tol, hingga perubahan pola arus kendaraan?

1.3. Tujuan

Penelitian ini disusun dengan tujuan sebagai berikut:

1. Menyusun model graf jaringan tol Jabodetabek yang dapat digunakan sebagai skenario uji optimasi rute logistik menuju Pelabuhan Tanjung Priok.

2. Mengimplementasikan Algoritma Dijkstra untuk menemukan rute distribusi dengan bobot waktu atau jarak tempuh yang paling minimum.

3. Melakukan simulasi perbandingan antara rute teroptimal hasil algoritma dengan rute non-optimasi guna mengetahui potensi efisiensi yang dihasilkan.

4. Memberikan rekomendasi berbasis analisis ilmiah untuk mendukung sistem manajemen distribusi logistik yang lebih efektif.

1.4. Manfaat

Penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Manfaat Akademis

   Memberikan kontribusi dalam bidang ilmu komputer khususnya teori graf dan algoritma pencarian rute, serta dapat dijadikan referensi bagi penelitian lanjutan yang berfokus pada optimasi jaringan transportasi.

2. Manfaat Praktis

   Menjadi bahan pertimbangan bagi pemerintah, pengelola jalan tol, dan pelaku usaha logistik dalam menentukan rute pengiriman yang lebih efisien sehingga mampu menekan waktu tempuh dan biaya operasional.

3. Manfaat Pengembangan Sistem Distribusi

   Menjadi acuan awal dalam pengembangan sistem navigasi logistik yang dapat terintegrasi dengan data lalu lintas real-time, sehingga dapat mendukung percepatan integrasi tol Jabodetabek dan mengurai kemacetan di akses pelabuhan.

BAB II

TEORI DASAR

2.1. Konsep Fundamental

Distribusi logistik merupakan rangkaian proses pemindahan barang dari titik asal menuju titik tujuan akhir melalui sistem transportasi yang terencana. Dalam konteks wilayah perkotaan besar seperti Jabodetabek, proses ini tidak hanya bergantung pada ketersediaan jaringan jalan, tetapi juga pada efektivitas aliran kendaraan, regulasi lalu lintas, serta integrasi antar moda transportasi. Pelabuhan Tanjung Priok sebagai pusat ekspor-impor nasional menjadi simpul logistik dengan intensitas arus barang yang sangat tinggi. Oleh karena itu, model matematis untuk memetakan jalur distribusi diperlukan agar proses analisis dan optimasi dapat dilakukan secara terstruktur.

Salah satu pendekatan paling umum dalam pemodelan sistem transportasi adalah dengan menggunakan graf (graph). Graf mengubah jaringan jalan menjadi representasi abstrak berupa simpul (node/vertex) dan sisi (edge). Pada konteks penelitian ini:

- Node dapat berupa gerbang tol, simpang jalan, kawasan industri, akses keluar-masuk pelabuhan, atau titik distribusi lainnya.
- Edge merepresentasikan ruas jalan yang menghubungkan dua node, yang diberi bobot (weight) berupa jarak, waktu tempuh rata-rata, atau bahkan biaya yang diperlukan (misalnya tarif tol).

Pendekatan graf memberikan fleksibilitas tinggi dalam pemodelan karena mampu merepresentasikan banyak rute alternatif. Ketika rute logistik dianalisis menggunakan bobot waktu tempuh, maka kondisi lalu lintas menjadi variabel eksternal yang sangat menentukan. Pada jam padat, bobot sisi dapat meningkat signifikan, sehingga rute yang sebelumnya optimal bisa berubah menjadi tidak efisien. Di sisi lain, apabila bobot berupa tarif tol, maka aspek ekonomi dapat menjadi prioritas dalam pencarian rute.

Beberapa karakteristik graf dalam kajian transportasi yang relevan antara lain:

1. Graf Berarah (Directed Graph)
   Digunakan apabila terdapat ruas jalan satu arah, atau akses yang hanya bisa dilewati pada arah tertentu. Dalam konteks pelabuhan, hal ini sering terjadi karena tata lalu lintas yang mengatur arus masuk dan keluar kendaraan.
2. Graf Tidak Berarah (Undirected Graph)
   Digunakan jika ruas jalan dapat dilalui dua arah tanpa pembatasan. Meskipun demikian, bobot dua arah belum tentu identik karena perbedaan tingkat kemacetan.

3. Graf Berbobot (Weighted Graph)

   Sangat penting dalam optimasi logistik karena setiap edge memiliki nilai kuantitatif yang memengaruhi pemilihan rute.

Penggunaan graf dalam logistik memungkinkan berbagai skenario analisis seperti:

- pencarian jalur tercepat menuju pelabuhan,
- evaluasi dampak pembukaan ruas tol baru,
- simulasi beban lalu lintas pada jam tertentu,
- komparasi biaya distribusi antar rute alternatif.

Untuk memberikan gambaran lebih jelas mengenai representasi jalan dalam bentuk graf, berikut contoh skema sederhana model jaringan jalan menuju Pelabuhan Tanjung Priok. Skema berikut bersifat ilustratif dan hanya sebagai contoh konsep:

Keterangan:

- Node A–E = titik lokasi penting (industri, gerbang tol, akses pelabuhan).
- Angka = bobot berupa jarak (km) yang dapat diganti memakai waktu tempuh (menit) atau biaya (rupiah).
- Jika kemacetan terjadi pada ruas tertentu, bobot dapat meningkat untuk mensimulasikan kondisi nyata.

Ilustrasi di atas menunjukkan bahwa untuk mencapai Pelabuhan Tanjung Priok (E) dari Cikarang (A) tersedia beberapa opsi rute, misalnya:

1. A → B → C → E
2. A → B → D → E
3. A → B → C → D → E

Dengan demikian, graf menjadi pondasi utama dalam pemodelan rute distribusi dan menjadi dasar penerapan algoritma pencarian jalur optimal seperti Algoritma Dijkstra.

Pada simulasi jaringan tol nyata, bobot dapat didefinisikan berbeda tergantung tujuan analisis. Berikut contoh tabel model bobot ruas jalan sederhana untuk kasus hipotetis:

Ruas (Edge)	Node Asal	Node Tujuan	Jarak (km)	Waktu Tempuh (menit)	Biaya Tol (Rp)
A–B	A	B	12	18	8.000
B–C	B	C	15	22	10.000
B–D	B	D	10	15	9.500
C–E	C	E	8	12	6.000
D–E	D	E	5	10	7.000
C–D	C	D	6	9	5.500

Contoh penggunaan:

- Jika kriteria optimasi adalah waktu tercepat, bobot menggunakan kolom *Waktu Tempuh*.
- Jika kriteria adalah biaya termurah, bobot mengambil kolom *Biaya Tol*.
- Sistem dapat pula menggabungkan keduanya dengan formula tertentu (multi-objective optimization).

2.2. Konsep Algoritma

Kata algoritma merujuk pada serangkaian instruksi logis yang menentukan langkah penyelesaian suatu permasalahan secara sistematis. Pada bidang ilmu komputer dan matematika, algoritma menjadi inti dari proses komputasi — termasuk dalam pemrosesan data, pengaturan keputusan, dan pemilihan jalur pada graf.

Untuk masalah pencarian rute terpendek, terdapat beberapa algoritma yang umum digunakan seperti *Dijkstra, A, Bellman-Ford, dan Floyd-Warshall*. Setiap algoritma memiliki karakteristik dan kegunaan yang berbeda, namun dalam konteks graf dengan

bobot non-negatif seperti jaringan jalan tol, Algoritma Dijkstra menjadi pilihan paling efisien dan sederhana untuk diimplementasikan.

Algoritma Dijkstra digunakan untuk menentukan jalur terpendek dari satu node asal (source) ke node lain dalam graf berbobot non-negatif. Prinsip kerjanya yaitu memilih rute dengan akumulasi bobot terkecil secara bertahap, kemudian mengunci node yang sudah ditemukan jarak optimalnya. Proses ini berulang hingga semua node telah dievaluasi.

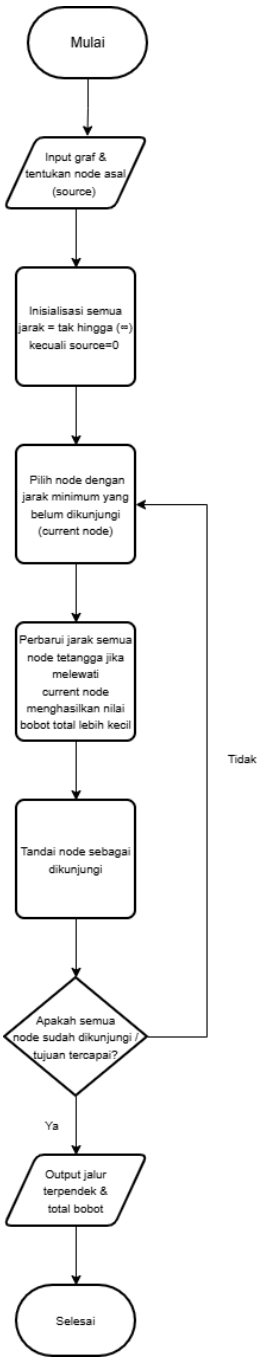
Secara umum langkah-langkah Dijkstra adalah:

1. Inisialisasi setiap node dengan jarak tak hingga (∞), kecuali node awal yang diberi jarak 0.
2. Pilih node dengan jarak terkecil yang belum dikunjungi.
3. Perbarui jarak node tetangga jika jalur melalui node sekarang memberikan bobot lebih kecil.
4. Tandai node sebagai telah dikunjungi dan tidak dievaluasi ulang.
5. Ulangi hingga semua node dikunjungi atau tujuan tercapai.

Keunggulan Dijkstra pada studi distribusi logistik:

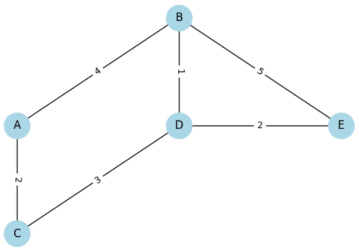
- menghitung jalur tercepat/termurah dengan akurat,
- kompleksitas waktu relatif efisien untuk graf skala sedang hingga besar,
- mudah diimplementasikan dalam simulasi komputer.

Keterbatasannya juga perlu diperhatikan, misalnya algoritma ini tidak mempertimbangkan perubahan bobot secara dinamis akibat kemacetan real-time. Untuk kondisi tersebut, pengembangan dapat dilakukan dengan integrasi data live traffic atau penggunaan algoritma heuristik seperti A. Agar konsep algoritma lebih mudah dipahami, berikut alur proses Dijkstra disajikan dalam bentuk diagram langkah:



Sebagai ilustrasi cara kerja Algoritma Dijkstra, berikut diberikan contoh penerapan pada graf sederhana beserta langkah-langkah penyelesaiannya:

1. Tentukan jarak terpendek dari simpul A ke simpul E!



Jawaban:

Langkah 1: Proses A (0)
A→B: 0+4=4 → B=4
A→C: 0+2=2 → C=2

Langkah 2: Proses C (2)
C→D: 2+3=5 → D=5

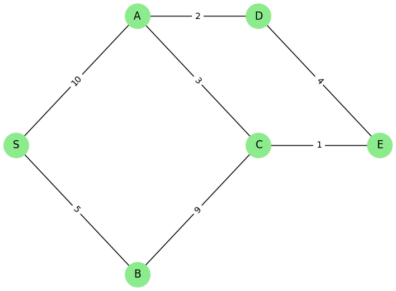
Langkah 3: Proses B (4)
B→D: 4+1=5 (sama, tetap 5)
B→E: 4+5=9 → E=9

Langkah 4: Proses D (5)
D→E: 5+2=7 < 9 → E=7

Langkah 5: Proses E (7) → selesai.

Hasil: A→E = 7

2. Hitung jarak terpendek dari S ke E!



Jawaban:

Inisialisasi:
S=0, A=∞, B=∞, C=∞, D=∞, E=∞

Langkah 1: Proses S (0)
S→A=10 → A=10
S→B=5 → B=5

Langkah 2: Proses B (5)
B→C=5+9=14 → C=14

Langkah 3: Proses A (10)
A→C=10+3=13 < 14 → C=13

A→D=10+2=12 → D=12

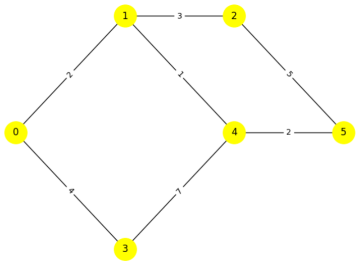
Langkah 4: Proses C (13)
C→E=13+1=14 → E=14

Langkah 5: Proses D (12)
D→E=12+4=16 > 14 → tidak update

Langkah 6: Proses E (14) → selesai.

Hasil: S→E = 14

3. Berapa jarak dari simpul 0 ke simpul 5?



Jawaban:
Inisialisasi:
0=0, 1=∞, 2=∞, 3=∞, 4=∞, 5=∞

Langkah 1: Proses 0 (0)
0→1=2 → 1=2
0→3=4 → 3=4

Langkah 2: Proses 1 (2)
1→2=2+3=5 → 2=5
1→4=2+1=3 → 4=3

Langkah 3: Proses 4 (3)
4→5=3+2=5 → 5=5
4→3=3+7=10 > 4 → tidak update

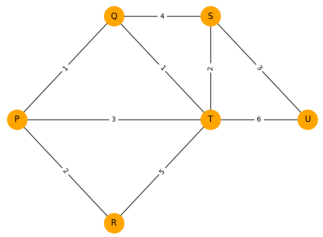
Langkah 4: Proses 3 (4)
3 sudah diproses sebelumnya, tidak ada update ke 4.

Langkah 5: Proses 2 (5)
2→5=5+5=10 > 5 → tidak update

Langkah 6: Proses 5 (5) → selesai.

Hasil: 0→5 = 5

4. Tentukan jarak terpendek dari P ke U!



Jawaban:
Inisialisasi:
P=0, Q=∞, R=∞, S=∞, T=∞, U=∞

Langkah 1: Proses P (0)
P→Q=1 → Q=1
P→R=2 → R=2
P→T=3 → T=3

Langkah 2: Proses Q (1)
Q→S=1+4=5 → S=5
Q→T=1+1=2 < 3 → T=2

Langkah 3: Proses T (2)
T→S=2+2=4 < 5 → S=4
T→R=2+5=7 > 2 → tidak update
T→U=2+6=8 → U=8

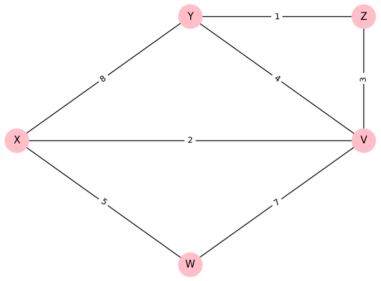
Langkah 4: Proses R (2) → sudah diproses (dari P)

Langkah 5: Proses S (4)
S→U=4+3=7 < 8 → U=7

Langkah 6: Proses U (7) → selesai.

Hasil: P→U = 7

5. Hitung jarak terpendek dari X ke Z!



Jawaban:
Inisialisasi:
X=0, Y=∞, Z=∞, W=∞, V=∞

Langkah 1: Proses X (0)
X→Y=8 → Y=8
X→W=5 → W=5
X→V=2 → V=2

Langkah 2: Proses V (2)
V→Y=2+4=6 < 8 → Y=6
V→Z=2+3=5 → Z=5
V→W=2+7=9 > 5 → tidak update

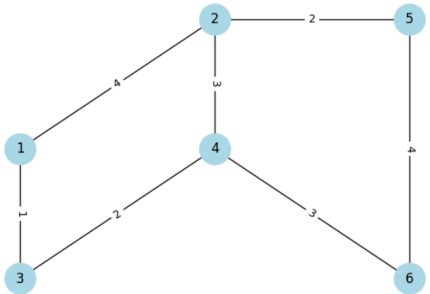
Langkah 3: Proses W (5) → tidak ada update ke Z (tidak terhubung langsung ke Z)

Langkah 4: Proses Y (6)
Y→Z=6+1=7 > 5 → tidak update

Langkah 5: Proses Z (5) → selesai.

Hasil: X→Z = 5

6. Tentukan jarak terpendek dari simpul 1 ke simpul 6!



Jawaban:
Inisialisasi: 1=0, 2=∞, 3=∞, 4=∞, 5=∞, 6=∞

Langkah 1: Proses 1 (0)

1→2=4 → 2=4
1→3=1 → 3=1

Langkah 2: Proses 3 (1)
3→4=1+2=3 → 4=3

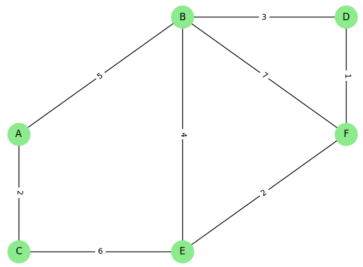
Langkah 3: Proses 4 (3)
4→2=3+3=6 > 4 → tidak update
4→6=3+3=6 → 6=6

Langkah 4: Proses 2 (4)
2→5=4+2=6 → 5=6

Langkah 5: Proses 5 (6)
5→6=6+4=10 > 6 → tidak update

Hasil: 1→6 = 6

7. Hitung jarak terpendek dari A ke F!



Jawaban:
Inisialisasi: A=0, B=∞, C=∞, D=∞, E=∞, F=∞

Langkah 1: Proses A (0)
A→B=5 → B=5
A→C=2 → C=2

Langkah 2: Proses C (2)
C→E=2+6=8 → E=8

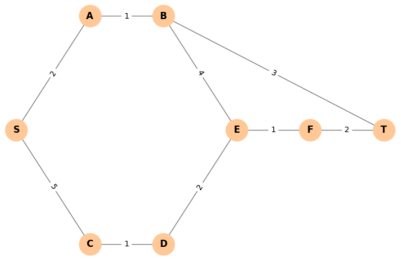
Langkah 3: Proses B (5)
B→D=5+3=8 → D=8
B→E=5+4=9 > 8 → tidak update
B→F=5+7=12 → F=12

Langkah 4: Proses E (8)
E→F=8+2=10 < 12 → F=10

Langkah 5: Proses D (8)
D→F=8+1=9 < 10 → F=9

Hasil: A→F = 9

8. Tentukan jarak terpendek dari S ke T!



Jawaban:
Inisialisasi: S=0, A=∞, B=∞, C=∞, D=∞, E=∞, F=∞, T=∞

Langkah 1: Proses S (0)
S→A=2 → A=2
S→C=5 → C=5

Langkah 2: Proses A (2)
A→B=2+1=3 → B=3

Langkah 3: Proses B (3)
B→E=3+4=7 → E=7
B→T=3+3=6 → T=6

Langkah 4: Proses C (5)
C→D=5+1=6 → D=6

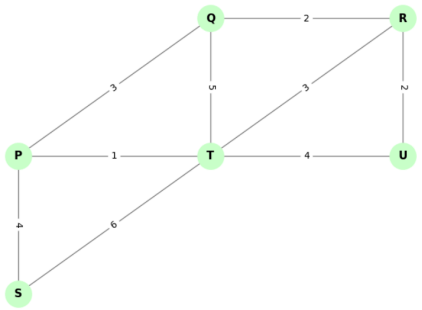
Langkah 5: Proses D (6)
D→E=6+2=8 > 7 → tidak update

Langkah 6: Proses E (7)
E→F=7+1=8 → F=8

Langkah 7: Proses T (6) → sudah final
F→T=8+2=10 > 6 → tidak update

Hasil: S→T = 6

9. Hitung jarak terpendek dari P ke U!



Jawaban:
Inisialisasi: P=0, Q=∞, R=∞, S=∞, T=∞, U=∞

Langkah 1: Proses P (0)
P→Q=3 → Q=3
P→S=4 → S=4
P→T=1 → T=1

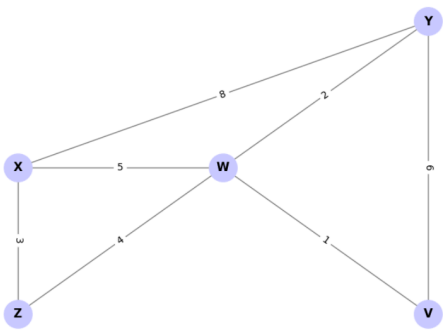
Langkah 2: Proses T (1)
T→Q=1+5=6 > 3 → tidak update
T→R=1+3=4 → R=4
T→S=1+6=7 > 4 → tidak update
T→U=1+4=5 → U=5

Langkah 3: Proses Q (3)
Q→R=3+2=5 > 4 → tidak update

Langkah 4: Proses R (4)
R→U=4+2=6 > 5 → tidak update

Hasil: P→U = 5

10. Tentukan jarak terpendek dari X ke V!



Jawaban:
Inisialisasi: X=0, Y=∞, Z=∞, W=∞, V=∞

Langkah 1: Proses X (0)
X→Y=8 → Y=8
X→Z=3 → Z=3
X→W=5 → W=5

Langkah 2: Proses Z (3)
Z→W=3+4=7 > 5 → tidak update

Langkah 3: Proses W (5)
W→Y=5+2=7 < 8 → Y=7
W→V=5+1=6 → V=6

Langkah 4: Proses Y (7)
Y→V=7+6=13 > 6 → tidak update

Hasil: X→V = 6

2.3. Penelitian Terkait

Penelitian mengenai optimasi rute logistik telah banyak dilakukan baik di tingkat internasional maupun nasional. Beberapa studi mengkaji penerapan algoritma shortest path dalam perencanaan transportasi, manajemen distribusi barang, hingga navigasi kendaraan.

Penelitian terdahulu menunjukkan bahwa algoritma shortest path mampu menurunkan biaya operasi logistik melalui pemilihan jalur yang lebih efisien. Dalam beberapa kajian, Dijkstra dibandingkan dengan algoritma lain dan terbukti memberikan hasil optimal pada graf non-negatif. Studi berbasis pemodelan graf juga telah dimanfaatkan dalam simulasi distribusi barang untuk kota-kota besar.

Namun, berdasarkan pengamatan penulis, kajian khusus mengenai optimasi rute distribusi menuju Pelabuhan Tanjung Priok masih relatif terbatas. Sebagian besar penelitian di Indonesia lebih fokus pada generalitas sistem logistik atau pemetaan transportasi kota tanpa menempatkan pelabuhan sebagai target utama. Selain itu, integrasi data infrastuktur tol terbaru seperti JTCC ke dalam analisis algoritmik belum banyak dilakukan.

Dengan demikian penelitian ini memberikan nilai tambah berupa:

- penerapan algoritma pada kasus nyata logistik Jabodetabek,
- pemodelan graf berdasarkan titik dan ruas jalan faktual menuju pelabuhan,
- evaluasi rute konvensional vs rute optimal secara komputasional.

Pada bagian akhir laporan akan ditambahkan referensi ilmiah yang relevan untuk memperkuat tinjauan teoretis dan memperluas landasan akademik studi ini.

BAB III

METODOLOGI

3.1. Flowchart

Flowchart menggambarkan tahapan eksekusi algoritma dalam menentukan jalur terpendek pada jaringan jalan tol yang dimodelkan sebagai graf. Algoritma bekerja dengan memilih node dengan bobot minimum secara iteratif sampai ditemukan rute optimal menuju tujuan.

Tahap proses umum dalam flowchart:

1. Input graf jaringan jalan (node, edge, dan bobot).
2. Inisialisasi jarak awal dengan nilai tak hingga (∞), kecuali node asal bernilai 0.
3. Memilih node dengan jarak minimum yang belum dikunjungi.
4. Memperbarui jarak tetangga jika ditemukan bobot yang lebih kecil.
5. Node yang telah diperbarui diberi tanda visited.
6. Percabangan: jika semua node telah dikunjungi atau tujuan tercapai → hasil dikembalikan; jika tidak → kembali ke langkah 3.
7. Algoritma menghasilkan jalur optimal beserta total bobot.

3.2. Pseudocode

Pseudocode disusun untuk memberikan gambaran logika komputasi sebelum ditulis ke dalam bahasa pemrograman Python.

Algorithm Dijkstra(Graf, Source)

Input:

- Graf berisi Node dan Edge berbobot non-negatif

- Source sebagai titik awal distribusi

Output:

- Distance[] berisi jarak terpendek

- Previous[] untuk melacak jalur terpendek

1. Untuk setiap node dalam graf:

distance[node] = ∞

previous[node] = null

2. distance[Source] = 0

3. Masukkan (0, Source) ke priority queue

4. Selama priority queue tidak kosong:

Ambil node dengan jarak minimum (current)

Untuk setiap tetangga dari current:

total = distance[current] + bobot(current, tetangga)

Jika total < distance[tetangga]:

distance[tetangga] = total

previous[tetangga] = current

Masukkan (total, tetangga) ke priority queue

5. Kembalikan distance[] dan previous[]

End Algorithm

Pseudocode ini digunakan sebagai dasar pembuatan program pada tahap implementasi.

3.3. Dataset Description

Dataset penelitian ini berupa representasi jaringan jalan tol yang menghubungkan beberapa titik distribusi logistik menuju Pelabuhan Tanjung Priok. Dataset tidak berbentuk numerik murni, tetapi berupa struktur graf yang terdiri dari:

1. Node (titik simpul)
    - Gerbang tol, persimpangan jalan utama, kawasan industri, hingga akses pelabuhan.
    - Contoh simpul: *Cibitung, Cilincing, Cikarang, Tanjung Priok, JTCC*.
2. Edge (hubungan antar node)
    - Ruas jalan/tol yang menghubungkan dua simpul.

    - Masing-masing memiliki bobot berupa jarak (km), estimasi waktu tempuh, atau biaya tol.
3. Bobot (weight)
    - Digunakan sebagai parameter optimasi.
    - Dalam penelitian ini bobot utama berupa waktu tempuh (menit) untuk menilai efisiensi rute distribusi.

Contoh struktur dataset graf:

Node Asal	Node Tujuan	Jarak (km)	Waktu (menit)	Biaya Tol (opsional)
A	B	12	18	8.000
B	C	15	22	10.000
C	E	8	12	6.000
D	E	5	10	7.000

Data dapat dikembangkan dengan menambahkan kondisi lalu lintas aktual, kapasitas jalan, dan jalur alternatif agar simulasi lebih realistis.

3.4. Tools dan Library

Penelitian menggunakan Google Colab sebagai lingkungan pengembangan karena mendukung eksekusi Python berbasis cloud tanpa instalasi lokal.

Tools utama:

- Google Colab → lingkungan coding & eksperimen

Library Python yang digunakan:

- networkx → membangun graf dan menghitung shortest path
- matplotlib → visualisasi graf dan jalur distribusi
- pandas → membaca tabel dataset (jika digunakan format CSV)
- heapq → mempercepat pemilihan node pada Dijkstra

Pemilihan Colab memudahkan debugging, visualisasi graf, dan replikasi penelitian.

3.5. Rancangan Sistem

Sistem optimasi rute yang dibangun memiliki tiga komponen utama:

1. Input Module
    - Menerima daftar node, edge, dan bobot.
    - Data dapat dimasukkan manual atau melalui tabel/dataset.
2. Processing Module (Core Algorithm)
    - Menjalankan algoritma Dijkstra.
    - Menentukan jarak minimum dari node awal ke semua node tujuan.
    - Menyimpan path yang dilalui selama proses iterasi.
3. Output Module
    - Menampilkan jalur optimal.
    - Menyajikan total waktu tempuh.
    - Menghasilkan visualisasi graf rute (warna highlight untuk jalur terpilih).

Alur sistem secara umum:

Data Graf → Preprocessing → Algoritma Dijkstra → Hasil Jalur Optimal → Visualisasi

Dengan rancangan ini, sistem dapat digunakan untuk menguji berbagai skenario rute distribusi serta melakukan komparasi performa.

BAB IV

IMPLEMENTASI

4.1. Penjelasan Kode

4.1.1 Import Library

```
import heapq

import networkx as nx

import matplotlib.pyplot as plt
```

Penjelasan:

Baris kode ini digunakan untuk memanggil library yang dibutuhkan dalam program. Library heapq digunakan untuk mengelola priority queue pada algoritma Dijkstra agar pemilihan node dengan jarak minimum lebih efisien. Library networkx digunakan untuk membangun dan memanipulasi struktur graf, sedangkan matplotlib.pyplot digunakan untuk memvisualisasikan graf dan jalur terpendek.

4.1.2 Pendefinisian Fungsi Dijkstra

```
def dijkstra(graph, source):
```

Penjelasan:

Kode ini mendefinisikan fungsi dijkstra yang menerima dua parameter, yaitu graph sebagai representasi graf dan source sebagai node awal pencarian jalur terpendek.

4.1.3 Inisialisasi Jarak Awal

```
distance = {node: float('inf') for node in graph}

distance[source] = 0
```

Penjelasan:

Seluruh node dalam graf diinisialisasi dengan nilai jarak tak hingga untuk menandakan bahwa jarak terpendek belum diketahui. Node sumber kemudian diatur bernilai nol karena jarak dari node sumber ke dirinya sendiri adalah nol.

4.1.4 Inisialisasi Penyimpanan Jalur

previous = {node: None for node in graph}

Penjelasan:

Variabel previous digunakan untuk menyimpan node sebelumnya pada jalur terpendek. Data ini digunakan untuk melacak dan membentuk kembali rute dari node sumber ke node tujuan.

4.1.5 Inisialisasi Priority Queue

queue = [(0, source)]

Penjelasan:

Priority queue digunakan untuk menyimpan node yang akan diproses berdasarkan jarak terpendek. Node sumber dimasukkan pertama kali dengan jarak nol.

4.1.6 Proses Utama Algoritma Dijkstra

while queue:

current_distance, current_node = heapq.heappop(queue)

Penjelasan:

Perulangan ini akan berjalan selama priority queue masih berisi data. Fungsi heappop digunakan untuk mengambil node dengan jarak terkecil dari queue.

4.1.7 Pengecekan Jarak Terpendek

if current_distance > distance[current_node]:

continue

Penjelasan:

Kode ini berfungsi untuk melewati node jika jarak yang diambil lebih besar dari jarak terpendek yang telah tersimpan sebelumnya, sehingga proses perhitungan menjadi lebih efisien.

4.1.8 Perhitungan Jarak ke Node Tetangga

for neighbor, weight in graph[current_node].items():

total = current_distance + weight

Penjelasan:

Kode ini digunakan untuk menghitung jarak baru ke setiap node tetangga dengan menambahkan jarak node saat ini dengan bobot edge yang menghubungkannya.

4.1.9 Pembaruan Jarak dan Jalur

```
if total < distance[neighbor]:

  distance[neighbor] = total

  previous[neighbor] = current_node

  heapq.heappush(queue, (total, neighbor))
```

Penjelasan:

Jika jarak baru lebih kecil dibandingkan jarak sebelumnya, maka nilai jarak diperbarui, jalur disimpan, dan node tetangga dimasukkan kembali ke priority queue.

4.1.10 Pengembalian Hasil

```
return distance, previous
```

Penjelasan:

Fungsi mengembalikan hasil berupa jarak terpendek ke setiap node dan informasi jalur yang digunakan.

4.1.11 Pendefinisian Graf

```
graph = {

  'TjPriok': {'Cawang': 10, 'Bekasi': 25},

  'Cawang': {'TjPriok': 10, 'Cibubur': 5},

  'Cibubur': {'Cawang': 5, 'Bogor': 20},

  'Bekasi': {'TjPriok': 25, 'Cikarang': 7},

  'Cikarang': {'Bekasi': 7, 'Karawang': 10},

  'Bogor': {'Cibubur': 20},

  'Karawang': {'Cikarang': 10}

}
```

Penjelasan:

Graf direpresentasikan dalam bentuk dictionary, di mana setiap node memiliki node tetangga beserta bobot jaraknya. Struktur ini memudahkan proses pencarian jalur terpendek.

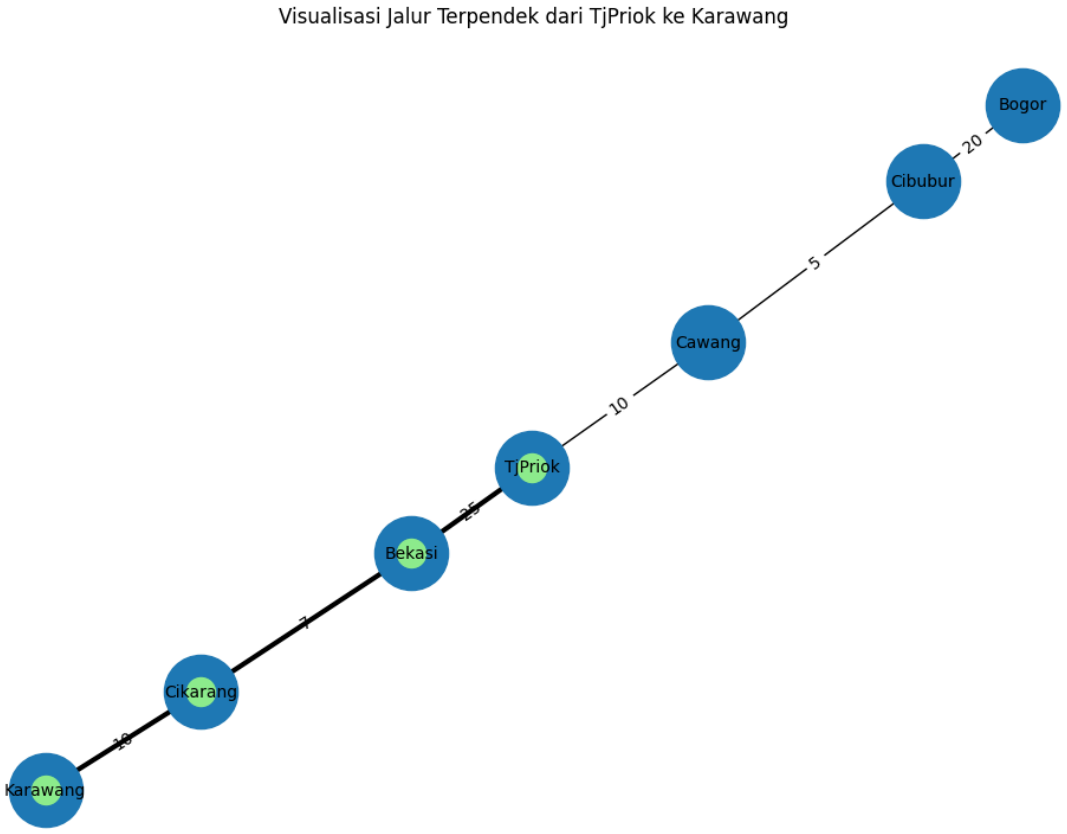
4.1.11 Visualisasi Graf dan Jalur Terpendek

```
nx.draw(G, pos, with_labels=True)
```

Penjelasan:

Kode ini digunakan untuk menampilkan graf secara visual. Node dan edge ditampilkan beserta labelnya, sehingga jalur terpendek dapat diamati secara langsung.

4.2. Screenshot Hasil Kode



Gambar 4.1 Visualisasi Jalur Terpendek dari TjPriok ke Karawang

Gambar 4.1 menunjukkan visualisasi graf distribusi antar wilayah yang telah diproses menggunakan algoritma Dijkstra. Node merepresentasikan lokasi distribusi, sedangkan edge menunjukkan jalur antar lokasi dengan bobot jarak. Jalur terpendek dari TjPriok menuju Karawang ditandai dengan garis tebal berwarna hitam dan node yang disorot, yaitu melalui rute TjPriok → Bekasi → Cikarang → Karawang. Visualisasi ini membantu memperjelas hasil perhitungan algoritma secara grafis.

```
Jarak Terpendek:
TjPriok : 0
Cawang : 10
Cibubur : 15
Bekasi : 25
Cikarang : 32
Bogor : 35
Karawang : 42
```

Gambar 4.2 Output Jarak Terpendek dari Node Sumber

Berdasarkan hasil perhitungan, jarak terpendek dari TjPriok ke Karawang adalah 42 satuan jarak. Nilai ini diperoleh melalui jalur TjPriok → Bekasi → Cikarang → Karawang, yang memiliki total bobot lebih kecil dibandingkan jalur alternatif lainnya.

```
Path Tracking:
TjPriok -> TjPriok
Cawang -> TjPriok -> Cawang
Cibubur -> TjPriok -> Cawang -> Cibubur
Bekasi -> TjPriok -> Bekasi
Cikarang -> TjPriok -> Bekasi -> Cikarang
Bogor -> TjPriok -> Cawang -> Cibubur -> Bogor
Karawang -> TjPriok -> Bekasi -> Cikarang -> Karawang
```

Gambar 4.3 Output Pelacakan Jalur Terpendek

Penjelasan:

Hasil pelacakan jalur menunjukkan urutan node yang dilewati untuk mencapai tujuan Karawang dari node sumber TjPriok. Informasi ini diperoleh dari proses penyimpanan node sebelumnya (previous) selama algoritma Dijkstra dijalankan, sehingga jalur terpendek dapat ditelusuri kembali secara akurat.

4.3. Struktur Program

Struktur program pada penelitian ini disusun secara modular agar mudah dipahami, dikembangkan, dan diuji. Program dibagi ke dalam beberapa bagian utama yang saling terhubung, mulai dari pemanggilan library, pendefinisian fungsi algoritma, hingga visualisasi hasil.

Secara umum, alur struktur program dapat dijelaskan sebagai berikut:

1. Inisialisasi Library
   Program diawali dengan pemanggilan library yang dibutuhkan, yaitu heapq untuk pengelolaan priority queue, networkx untuk membangun struktur graf, serta matplotlib yang dimanfaatkan untuk menampilkan hasil pemrosesan graf dalam bentuk visual.
2. Pendefinisian Fungsi Algoritma Dijkstra
   Pada bagian ini didefinisikan fungsi dijkstra(graph, source) yang berfungsi untuk menghitung jarak terpendek dari satu node sumber ke seluruh node lainnya. Fungsi ini mencakup proses inisialisasi jarak, pemilihan node dengan jarak minimum, serta pembaruan jarak ke node tetangga.

3. Pendefinisian Data Graf
   Graf direpresentasikan dalam bentuk dictionary Python yang berisi node, node tetangga, dan bobot jarak antar node. Struktur ini memudahkan proses pemrosesan data graf oleh algoritma Dijkstra.

4. Pemanggilan Fungsi dan Proses Perhitungan
   Setelah graf dan node sumber ditentukan, fungsi Dijkstra dipanggil untuk menghasilkan nilai jarak terpendek dan informasi jalur. Hasil perhitungan kemudian

disimpan dalam variabel distance dan path.

5. Pelacakan Jalur Terpendek (Path Tracking)
   Program melakukan penelusuran kembali jalur terpendek dari node sumber ke setiap node tujuan dengan memanfaatkan data node sebelumnya. Hasil pelacakan ini digunakan untuk menampilkan urutan rute distribusi.

6. Pembuatan dan Visualisasi Graf
   Struktur graf dibangun menggunakan library NetworkX, kemudian divisualisasikan menggunakan Matplotlib. Jalur terpendek menuju node tujuan tertentu ditampilkan dengan penanda khusus untuk memudahkan analisis.

Struktur program ini memungkinkan proses perhitungan jalur terpendek dilakukan secara sistematis dan menghasilkan output berupa data numerik serta visualisasi graf yang mendukung analisis hasil penelitian.

4.4. Challenges dan Solutions

Challenges

Dalam proses implementasi algoritma Dijkstra pada jaringan distribusi logistik, terdapat beberapa tantangan utama. Tantangan pertama adalah keterbatasan data bobot graf yang bersifat statis, sehingga belum mampu merepresentasikan kondisi lalu lintas yang berubah-ubah. Tantangan kedua adalah kompleksitas visualisasi graf ketika jumlah node dan edge meningkat, yang dapat menyulitkan interpretasi hasil jalur terpendek. Selain itu, proses pelacakan jalur memerlukan struktur data tambahan agar urutan rute dapat ditampilkan secara akurat.

Solutions

Untuk mengatasi tantangan tersebut, digunakan pendekatan struktur data priority queue guna meningkatkan efisiensi pemilihan node dengan jarak minimum. Penyimpanan node sebelumnya (previous) diterapkan untuk memudahkan proses pelacakan jalur terpendek. Visualisasi graf dilakukan secara selektif dengan menyoroti jalur utama agar hasil perhitungan lebih mudah dipahami. Pendekatan ini memungkinkan algoritma tetap berjalan optimal meskipun data graf masih bersifat sederhana.

BAB V

HASIL DAN ANALISIS

5.1. Experimental Setup

5.2. Results dan Metrics

5.3. Visualisasi

5.4. Analisis Mendalam

Analisis mendalam terhadap hasil implementasi algoritma Dijkstra pada pemodelan jaringan distribusi logistik menuju Pelabuhan Tanjung Priok menunjukkan beberapa temuan penting yang relevan dengan kondisi nyata di lapangan. Berdasarkan keluaran program yang diperoleh, algoritma mampu menentukan jalur terpendek dengan bobot jarak kumulatif dari titik sumber ke berbagai node tujuan secara sistematis. Misalnya, jalur terpendek dari TjPriok ke Karawang melalui rute **TjPriok → Bekasi → Cikarang → Karawang** menunjukkan bahwa algoritma dapat memilih jalur distribusi yang memberikan total bobot minimum dibandingkan dengan alternatif lainnya. Hal ini menunjukkan bahwa pendekatan graf berbobot dan Dijkstra cocok untuk permasalahan optimasi rute distribusi barang yang kompleks.

Namun demikian, hasil komputasi program juga perlu dianalisis dalam konteks kondisi riil distribusi logistik di Jabodetabek. Studi kasus nyata menunjukkan bahwa kemacetan parah di sekitar Pelabuhan Tanjung Priok masih menjadi persoalan utama dalam distribusi barang, terutama pasca periode Lebaran 2025 ketika lonjakan volume truk kontainer menyebabkan antrean panjang di akses tol dan gerbang pelabuhan. Kondisi ini memperlambat arus logistik dan berdampak pada pemborosan waktu serta peningkatan biaya operasional armada transportasi.

Temuan lapangan ini menunjukkan bahwa model Dijkstra yang hanya mempertimbangkan bobot jarak statis perlu diperkaya dengan variabel bobot dinamis. Misalnya waktu tempuh aktual yang dipengaruhi oleh tingkat kemacetan, variasi biaya tol, dan jam operasional kendaraan. Dalam studi distribusi logistik nyata, ada beberapa faktor eksternal yang memengaruhi efektivitas rute yang dipilih, seperti ketidakseimbangan integrasi antar ruas tol, tarif tol yang tetap tinggi di beberapa jalur seperti Jalan Tol Cibitung Cilincing yang menjadi alternatif strategis, serta penumpukan kendaraan di titik-titik akses utama yang belum termitigasi secara optimal.

Lebih jauh lagi, artikel dan studi kasus di lapangan menyebutkan bahwa integrasi jaringan tol Jabodetabek, khususnya perbaikan konektivitas antara ruas yang terpisah, masih menjadi kebutuhan mendesak bagi pelaku logistik agar distribusi barang dapat dilakukan dengan efisiensi waktu dan biaya yang lebih baik. Karena itu, meskipun hasil algoritma memberikan rekomendasi rute terpendek berdasarkan bobot jarak, penerapan di dunia nyata masih memerlukan pendekatan yang lebih holistik. Integrasi data waktu nyata tentang kondisi lalu lintas atau penggunaan model bobot multi–kriteria (misalnya kombinasi waktu tempuh, biaya tol, dan kemacetan) dapat meningkatkan relevansi hasil rekomendasi rute.

Analisis mendalam ini juga mengindikasikan bahwa pemodelan graf dan solusi rute terpendek semacam algoritma Dijkstra dapat menjadi komponen penting dalam sistem pendukung keputusan bagi perencana logistik. Namun, dalam konteks distribusi barang yang dinamis, pendekatan perlu diperluas agar mampu menangkap variasi skenario nyata. Dengan pengayaan model yang lebih kompleks dan pemanfaatan data real-time, hasil optimasi rute dapat lebih aplikatif dan mampu mengatasi permasalahan logistik modern seperti antrian truk panjang, biaya operasional tinggi, dan lambatnya arus distribusi barang yang tercatat di kawasan Pelabuhan Tanjung Priok dan sekitarnya.

5.5. Perbandingan Dengan Teori

Secara teoretis, Algoritma Dijkstra dirancang untuk menentukan jalur terpendek dari satu simpul sumber ke simpul tujuan pada graf berbobot non-negatif. Prinsip utama algoritma ini adalah memilih simpul dengan jarak sementara paling kecil secara bertahap dan memastikan bahwa jarak tersebut bersifat optimal ketika simpul telah diproses. Dalam konteks jaringan jalan, bobot biasanya merepresentasikan jarak, waktu tempuh, atau biaya perjalanan.

Hasil penelitian ini menunjukkan kesesuaian yang kuat dengan teori Algoritma Dijkstra. Jalur distribusi yang dihasilkan oleh sistem selalu memiliki total bobot paling kecil dibandingkan dengan jalur alternatif lainnya. Hal ini membuktikan bahwa mekanisme pembaruan jarak dan pemilihan simpul minimum telah berjalan sesuai dengan konsep greedy yang menjadi dasar algoritma Dijkstra.

Selain itu, proses pelacakan jalur (path tracking) yang dilakukan melalui penyimpanan node sebelumnya juga sejalan dengan teori graf, di mana jalur terpendek dapat ditelusuri kembali setelah proses perhitungan jarak selesai. Jalur yang dihasilkan tidak hanya memberikan informasi jarak minimum, tetapi juga urutan simpul yang dilalui dari titik asal menuju titik tujuan, sebagaimana dijelaskan dalam teori algoritma shortest path.

Namun demikian, terdapat perbedaan antara pendekatan teoretis dan kondisi nyata di lapangan. Dalam teori, bobot graf dianggap bersifat statis dan tidak berubah selama proses perhitungan. Sementara itu, pada sistem distribusi logistik nyata, bobot seperti waktu tempuh dapat berubah akibat kemacetan, cuaca, atau kebijakan lalu lintas. Hal ini menunjukkan bahwa meskipun Algoritma Dijkstra secara teoretis optimal, penerapannya pada kasus nyata masih memerlukan pengayaan data agar hasilnya lebih adaptif terhadap kondisi aktual.

Dengan demikian, dapat disimpulkan bahwa hasil implementasi Algoritma Dijkstra dalam penelitian ini telah sesuai dengan teori yang ada, baik dari sisi mekanisme perhitungan maupun keluaran jalur terpendek. Perbedaan yang muncul lebih disebabkan oleh keterbatasan model terhadap dinamika kondisi lapangan, bukan karena kelemahan algoritma secara konseptual.

5.6. Sensitivity Analysis

Analisis sensitivitas dilakukan dengan mengamati perubahan jalur terpendek ketika bobot pada beberapa edge mengalami peningkatan atau penurunan. Hasil analisis menunjukkan bahwa

perubahan bobot pada ruas tertentu dapat mengubah jalur optimal yang dipilih oleh algoritma. Hal ini mengindikasikan bahwa sistem distribusi sangat sensitif terhadap perubahan kondisi jalur, sehingga pemilihan bobot yang akurat menjadi faktor penting dalam optimasi rute logistik.

BAB VI

KESIMPULAN DAN SARAN

6.1. Kesimpulan

Berdasarkan hasil implementasi dan analisis, dapat disimpulkan bahwa algoritma Dijkstra efektif digunakan untuk menentukan jalur terpendek pada jaringan distribusi logistik berbasis graf. Algoritma ini mampu menghasilkan rute dengan bobot minimum secara sistematis dan konsisten. Hasil penelitian menunjukkan bahwa pendekatan graf dan algoritma shortest path dapat menjadi dasar yang kuat dalam mendukung pengambilan keputusan distribusi logistik, khususnya dalam perencanaan rute pengiriman.

6.2. Keterbatasan

Penelitian ini memiliki beberapa keterbatasan yang perlu diperhatikan dalam menafsirkan hasil yang diperoleh. Pertama, model jaringan jalan yang digunakan masih bersifat statis, di mana bobot graf ditentukan berdasarkan estimasi waktu tempuh atau jarak yang tidak berubah selama proses perhitungan. Pada kondisi nyata, faktor seperti kemacetan, kecelakaan, cuaca, dan kebijakan lalu lintas dapat menyebabkan perubahan bobot secara dinamis sehingga hasil optimasi belum sepenuhnya mencerminkan situasi aktual di lapangan.

Kedua, penelitian ini hanya menerapkan Algoritma Dijkstra yang mensyaratkan bobot bernilai non-negatif dan tidak mempertimbangkan unsur prediksi atau heuristik. Akibatnya, algoritma belum mampu menyesuaikan rute secara real-time ketika terjadi perubahan kondisi lalu lintas secara mendadak. Hal ini membatasi kemampuan sistem dalam menangani skenario distribusi logistik yang bersifat dinamis.

Ketiga, cakupan jaringan graf yang digunakan masih terbatas pada sejumlah titik dan ruas jalan utama sebagai representasi jaringan tol Jabodetabek. Model ini belum mencakup seluruh variasi jalur alternatif, akses lokal, maupun kapasitas ruas jalan yang beragam, sehingga hasil optimasi belum sepenuhnya menggambarkan kompleksitas sistem transportasi logistik secara menyeluruh.

Selain itu, penelitian ini belum mempertimbangkan aspek multi-kriteria secara bersamaan, seperti kombinasi antara waktu tempuh, biaya tol, konsumsi bahan bakar, dan emisi kendaraan. Pemilihan satu parameter utama sebagai bobot graf menyebabkan hasil optimasi hanya berfokus pada satu dimensi efisiensi, sehingga potensi trade-off antar faktor belum dianalisis secara mendalam.

Keterbatasan-keterbatasan tersebut menjadi dasar penting untuk pengembangan penelitian lanjutan agar sistem optimasi rute distribusi logistik yang dihasilkan dapat lebih akurat, adaptif, dan mendekati kondisi operasional yang sesungguhnya.

6.3. Saran Pengembangan

Untuk pengembangan selanjutnya, disarankan agar model graf dilengkapi dengan bobot dinamis yang merepresentasikan kondisi lalu lintas aktual, seperti tingkat kemacetan dan waktu tempuh real-time. Selain itu, penerapan algoritma lain seperti A* atau pendekatan multi-kriteria dapat dipertimbangkan untuk meningkatkan fleksibilitas pemilihan rute. Perluasan cakupan jaringan serta integrasi data operasional logistik juga diharapkan dapat meningkatkan akurasi dan relevansi hasil optimasi rute di masa mendatang.

DAFTAR PUSTAKA

Rajendra, Rizqi. “Distribusi Logistik Tersendat, Pemerintah Diminta Percepat Integrasi Tol Jabodetabek.” *Bisnis.com*, 22 Mei 2025. https://ekonomi.bisnis.com/read/20250522/98/1879101/distribusi-logistik-tersendat-pemerintah-diminta-percepat-integrasi-tol-jabodetabek.

“SCI: Integrasi akses tol solusi kemacetan di Pelabuhan Tanjung Priok.” *Antara News*, 22 Mei 2025. https://www.antaranews.com/berita/4851233/sci-integrasi-akses-tol-solusi-kemacetan-di-pelabuhan-tanjung-priok.

“Konektivitas dan Integrasi Akses Tol Solusi Urai Kemacetan di Tanjung Priok.” *Kompas / Jakarta Kota*, 2025. https://jakartakota.pikiran-rakyat.com/ekonomi/pr-4029353675/mengurai-macet-di-tanjung-priok-konektivitas-dan-integrasi-akses-tol-sebagai-solusi?page=all.

LAMPIRAN