OPTIMALISASI RUTE TRANSPORTASI MENGGUNAKAN ALGORITMA GRAF (STUDI KASUS : JARINGAN TRANSPORTASI PERKOTAAN)

Ruth Amelia Vega S. Meliala, Salsa Nabila Harahap, M Haikal Al-Majid, Putri Harliana
Ilmu Komputer, Universitas Negeri Medan
Jl. William Iskandar Pasar V, Kenangan Baru, Kabupaten Deli Serdang.
Sayaruthamelia07@gmail.com

ABSTRAK

Peningkatan kepadatan lalu lintas di kota-kota besar menimbulkan kebutuhan yang mendesak akan sistem transportasi yang efisien. Penelitian ini bertujuan menganalisis penerapan algoritma graf, khususnya algoritma Dijkstra dan A*, dalam mengoptimalkan rute transportasi perkotaan untuk menemukan rute terpendek yang lebih cepat dan efisien. Data jaringan transportasi diperoleh dari survei pengguna transportasi umum dan data jaringan terbuka, yang kemudian dipetakan menjadi graf dengan titik dan jalur yang menghubungkannya. Algoritma Dijkstra dan A* diimplementasikan untuk memproses jaringan transportasi ini, dan hasilnya dianalisis serta divalidasi menggunakan data historis dan simulasi kondisi lalu lintas. Hasil penelitian menunjukkan bahwa algoritma A* lebih unggul dalam menyesuaikan rute secara real-time, sedangkan Dijkstra optimal untuk kondisi statis. Dengan demikian, penerapan algoritma graf dapat membantu menciptakan sistem transportasi cerdas yang mendukung efisiensi dan kenyamanan pengguna dalam menghadapi tantangan lalu lintas perkotaan.

Kata kunci : *Algoritma graph, rute terpendek, Dijkstra, A**

1. PENDAHULUAN

Dalam era urbanisasi yang cepat, jaringan transportasi perkotaan menjadi semakin kompleks, yang mengakibatkan sejumlah permasalahan serius. Salah satu tantangan utama adalah meningkatnya kepadatan lalu lintas yang menimbulkan berbagai dampak negatif, seperti meningkatnya waktu tempuh dan kemacetan yang berkepanjangan, terutama di kota-kota besar. Selain waktu tempuh yang lebih lama, mobilitas yang terhambat juga memicu peningkatan konsumsi bahan bakar yang berakibat pada tingginya emisi gas rumah kaca. Hal ini turut berkontribusi terhadap pencemaran udara dan peningkatan biaya transportasi bagi masyarakat. Di sisi lain, ketidakpastian kondisi lalu lintas menyebabkan tingkat stres yang tinggi bagi pengguna jalan, serta menurunkan produktivitas penduduk perkotaan.

Salah satu persoalan optimasi yang sering ditemui dalam kehidupan sehari-hari adalah pencarian lintasan terpendek (shortest path). Persoalan ini bisa dimodelkan kedalam suatu graf berbobot dengan nilai pada masing-masing sisi yang mewakili persoalan yang akan dipecahkan. Kata "terpendek" pada kata "jalur terpendek" ini tidak berarti hanya bisa diartikan jarak secara fisik, namun hal itu tergantung dari tipe persoalan yang akan dipecahkan. Bisa jadi kata tersebut memiliki makna tingkat akses suatu simpul dalam graf dari simpul yang lain. Persoalan jalur terpendek yang akan dibahas di makalah ini adalah lintasan terpendek antara dua buah simpul tertentu di dalam graf (single pair shortest path) [1].

Algoritma graf, seperti Dijkstra dan A*, terbukti efektif dalam menyelesaikan permasalahan rute terpendek. Algoritma ini mampu mencari jalur paling efisien dalam berbagai jenis jaringan, seperti jalan, rel kereta api, dan jaringan transportasi umum lainnya. Penerapan algoritma ini dalam sistem transportasi cerdas (Intelligent Transportation System/ITS) dapat membantu pengguna menentukan rute terbaik berdasarkan kondisi lalu lintas secara real-time. Namun, kendala utama yang masih dihadapi adalah bagaimana algoritma ini dapat mempertimbangkan faktor eksternal seperti kondisi lalu lintas, cuaca, dan hambatan jalan yang kerap berubah dengan cepat.

Oleh karena itu, penelitian ini bertujuan untuk menganalisis penerapan algoritma graf dalam jaringan transportasi perkotaan guna memberikan solusi optimal bagi rute perjalanan. Melalui studi kasus pada jaringan transportasi nyata, penelitian ini diharapkan memberikan wawasan praktis tentang bagaimana teknologi algoritma graf dapat mendukung efisiensi transportasi dan mengatasi berbagai permasalahan di masa depan.

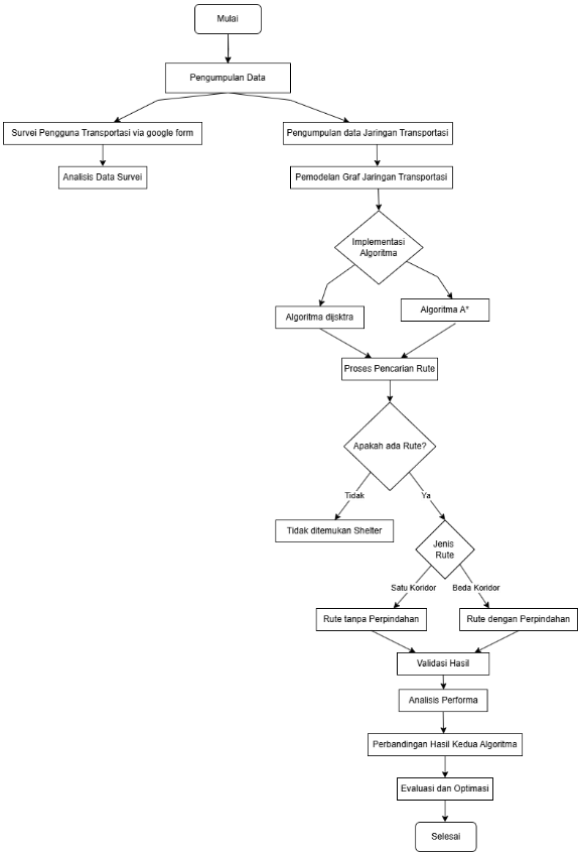
2. TINJAUAN PUSTAKA

Kita sering menggunakan peta untuk mengetahui sebuah tempat atau lokasi. Seiring berkembangnya teknologi informasi, kini peta tidak lagi hanya dalam bentuk media cetak namun telah menjadi sebuah layanan digital atau peta virtual yang dapat kita akses memalui smartphone kita atau laptop. Meskipun peta membantu kita untuk dapat mengetahui lokasi kita berada dan lokasi yang kita tuju namun tidak dapat membantu kita dalam menentukan jalur terpendek. Dalam upaya mewujudkan kemudahan dalam melakukan pencarian jalur terpendek pada aplikasi ini maka diterapkan nya sebuah algortima pencarian jalur terpendek yaitu pada aplikasi ini menggunakan algoritma Djikstra. Dengan memperhitungkan bobot pada setiap sisi, algoritma Djikstra dapat digunakan untuk menentukan jalur terpendek dari suatu titik ke titik akhir [2].

Permasalahan jalur transportasi dan penentuan rute terpendek bisa diatasi dengan menggunakan teori

Graf [3]. Graf yaitu kumpulan-kumpulan titik yang dihubungkan dengan garis yang memiliki bobot, dapat dikatakan graf merupakan kumpulan dari beberapa simpul yang dihubungkan oleh sisi-sisi [4]. Berdasarkan pada teori Graf, permasalahan pada rute terpendek dapat diartikan sebagai suatu permasalahan untuk mendapatkan lintasan antara dua buah simpul pada graf berbobot yang mempunyai gabungan nilai dari total bobot pada sisi graf yang dilalui dengan jumlah yang paling minimum. Persoalan mencari jalur terpendek pada graf ialah salah satu persoalan optimasi. Graf yang digunakan dalam pencarian jalur terpendek adalah graf berbobot (weighted graph) [5].

Pencarian rute terpendek telah diterapkan diberbagai bidang untuk mengoptimasi kinerja suatu sistem baik untuk meminimalkan biaya ataupun mempercepat jalurnya suatu proses. Salah satu aplikasi pencarian rute terpendek yang paling menarik untuk dibahas adalah masalah transportasi. Ada beberapa metode untuk pencarian rute terpendek yaitu algoritma A*, algoritma Djikstra, algoritma Bellman-Ford, algoritma Floyd-Warshall, dan sebagainya. Ada beberapa metode untuk pencarian rute terpendek yaitu algoritma A* [6].



Gambar 1. Diagram alir penelitian (Flowchart)

Algoritma Djikstra, algoritma Bellman-Ford, algoritma Floyd-Warshall, dan sebagainya. Sistem routing pertama kali diperkenalkan tahun 2004 oleh tim peneliti yang terdiri dari Ryujiro Fujita, Hiroto Inoue, Naohiko Ichihara, Takehiko Shioda, bertujuan untuk mengatasi kepadatan lalu lintas yang terjadi di beberapa kota besar di Jepang[1]. Sistem yang diperkenalkan oleh para peneliti ini menggunakan algoritma Djikstra pada pencarian rute. Namun seiring dengan berkembangnya ilmu pengetahuan, metode Djikstra mempunyai kekurangan dalam waktu pencarian yang kurang efektif dan dinilai [7].

2.1. Algoritma Dijkstra

Algoritma Dijkstra pertama kali diperkenalkan oleh Edsger W. Dijkstra pada tahun 1959 sebagai algoritma pencarian jalur terpendek dalam graf berarah yang berbobot positif. Algoritma ini bekerja dengan memilih jalur yang memiliki bobot terkecil dari simpul awal hingga simpul tujuan. Dalam konteks jaringan transportasi, algoritma Dijkstra sering digunakan untuk menghitung rute tercepat antara dua lokasi berdasarkan jarak atau waktu tempuh. Studi sebelumnya menunjukkan bahwa Dijkstra sangat efisien untuk jaringan dengan jumlah simpul yang tidak terlalu besar, tetapi kinerjanya menurun ketika diterapkan pada jaringan yang lebih kompleks atau besar .

2.2. Algoritma A*

Algoritma A* adalah algoritma pencarian jalur yang menggabungkan pendekatan Dijkstra dengan algoritma heuristik. A* menggunakan fungsi evaluasi yang menggabungkan biaya dari simpul awal dan estimasi jarak dari simpul saat ini ke tujuan. Salah satu kelebihan A* dibandingkan Dijkstra adalah penggunaan fungsi heuristik yang memungkinkan algoritma ini bekerja lebih cepat dalam menemukan solusi optimal pada graf besar. Algoritma A* sering diaplikasikan dalam sistem navigasi modern, terutama pada sistem GPS untuk menemukan rute tercepat dengan mempertimbangkan jarak dan waktu tempuh.

2.3. Implementasi Algoritma Graf pada Jaringan Transportasi Perkotaan

Jaringan transportasi perkotaan dapat dimodelkan sebagai graf, di mana simpul merepresentasikan lokasi (misalnya, persimpangan jalan atau titik pemberhentian transportasi umum), dan tepi graf merepresentasikan jalan atau jalur transportasi yang menghubungkan lokasi-lokasi tersebut. Pemodelan ini memungkinkan penggunaan berbagai algoritma graf, termasuk Dijkstra dan A*, untuk mengoptimalkan rute dan meningkatkan efisiensi transportasi. Studi terbaru menunjukkan bahwa kombinasi algoritma graf dengan teknik pemrograman dinamis dapat meningkatkan akurasi prediksi rute terbaik dalam jaringan transportasi perkotaan yang dinamis, terutama ketika terdapat perubahan kondisi lalu lintas secara real-time.

2.4. Studi Kasus di Berbagai Kota

Berbagai penelitian telah dilakukan untuk menguji efektivitas algoritma graf dalam mengoptimalkan rute transportasi di beberapa kota besar. Sebagai contoh, penelitian di Singapura

menunjukkan bahwa penerapan algoritma graf pada sistem transportasi publik dapat mengurangi waktu perjalanan hingga 15% pada rute tertentu. Di Indonesia, penerapan algoritma Dijkstra dan A* pada jaringan transportasi perkotaan di Jakarta menunjukkan hasil yang positif dalam mengurangi kemacetan di beberapa koridor utama, meskipun tantangan seperti volume kendaraan yang tinggi dan infrastruktur yang belum optimal masih menjadi hambatan.

3. METODE PENELITIAN

3.1. Desain Penelitian

Penelitian ini menggunakan metode **kualitatif** dengan pendekatan **studi kasus**. Fokusnya adalah pada penerapan algoritma graf untuk menemukan rute terpendek dalam jaringan transportasi perkotaan. Desain penelitian ini mencakup tiga tahapan utama:

a. Tahap Pengumpulan Data: Pengumpulan data dilakukan melalui survei menggunakan Google Forms (GForm). Survei ini bertujuan untuk mendapatkan data pengguna transportasi umum terkait frekuensi penggunaan transportasi, persepsi mereka terhadap aplikasi navigasi, dan pandangan mereka mengenai efisiensi rute yang disediakan.
b. Tahap Implementasi Algoritma: Algoritma graf seperti Dijkstra dan A* diimplementasikan untuk menemukan rute terpendek.
c. Tahap Analisis dan Validasi: Analisis dilakukan dengan membandingkan hasil algoritma dengan rute yang ada, dan validasi dilakukan melalui data historis dan simulasi.

3.2. Implementasi Algoritma Dijkstra

Algoritma Dijkstra digunakan untuk menemukan rute terpendek dari titik awal ke titik tujuan dengan asumsi bahwa semua bobot tepi graf non-negatif. Algoritma ini mencari solusi optimal dengan mengevaluasi node-node terdekat terlebih dahulu.

```
function Dijkstra(Graph, start):
        let distance = array of distances from start to all nodes, initialized to infinity
        distance[start] = 0
        let queue = priority queue of all nodes, with priority being distance from start

        while queue is not empty:
            currentNode = queue.extract_min()
            for each neighbor of currentNode:
                tempDistance = distance[currentNode] + weight(currentNode, neighbor)
                if tempDistance < distance[neighbor]:
                    distance[neighbor] = tempDistance
                    queue.decrease_priority(neighbor, tempDistance)
        return distance
```

3.3. Implementasi Algoritma A*

Algoritma A* adalah algoritma pencarian jalur yang memanfaatkan fungsi heuristik untuk mempercepat pencarian rute optimal. Algoritma ini menggunakan dua komponen, yaitu:

- **g(n)**: Biaya dari titik awal ke titik n.
- **h(n)**: Perkiraan biaya dari n ke tujuan (heuristik)

```
function A_star(Graph, start, goal):
        let openSet = priority queue containing start
        let cameFrom = empty map
        let gScore = map with default value of infinity, gScore[start] = 0
        let fScore = map with default value of infinity, fScore[start] = heuristic(start, goal)

        while openSet is not empty:
            current = openSet.extract_min()
            if current == goal:
                return reconstruct_path(cameFrom, current)

            for each neighbor of current:
                tentative_gScore = gScore[current] + weight(current, neighbor)
                if tentative_gScore < gScore[neighbor]:
                    cameFrom[neighbor] = current
                    gScore[neighbor] = tentative_gScore
                    fScore[neighbor] = gScore[neighbor] + heuristic(neighbor, goal)
                    if neighbor not in openSet:

openSet.add(neighbor)
        return failure
```

3.4. Pengujian Algoritma Dijkstra dan A*

a. Algoritma Dijkstra dan A* diuji dengan data jaringan transportasi yang sudah diperoleh.
b. Pengujian dilakukan pada berbagai skenario kondisi lalu lintas, baik normal maupun padat.
c. Hasil rute yang dihasilkan oleh algoritma dibandingkan dengan rute aktual dan diukur dari segi waktu tempuh dan jarak.

3.5. Validasi dengan Data Historis

Hasil algoritma divalidasi menggunakan data transportasi historis, yang menggambarkan pola lalu lintas dan waktu tempuh pada berbagai kondisi. Selain itu, simulasi skenario lalu lintas diterapkan untuk menguji efektivitas algoritma dalam kondisi nyata.

3.6. Perbandingan dengan Algoritma Lain

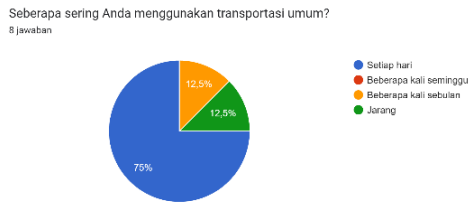
Hasil dari algoritma Dijkstra dan A* dibandingkan dengan algoritma graf lainnya untuk menentukan algoritma mana yang memberikan solusi optimal. Analisis perbandingan dilakukan berdasarkan efisiensi waktu tempuh dan jarak rute yang dihasilkan.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Survei Menggunakan Google Form

Survei yang dilakukan memberikan wawasan tentang bagaimana pengguna transportasi umum berinteraksi dengan teknologi navigasi, serta persepsi mereka terhadap efisiensi rute dan optimalisasi perjalanan. Berikut adalah hasil survei yang dihubungkan dengan penerapan **algoritma graf** (Dijkstra dan A*) dalam sistem navigasi dan pembuktian menggunakan JAVA.

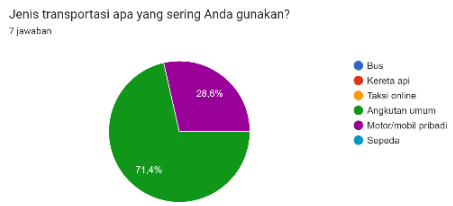
4.2. Frekuensi Penggunaan Transportasi Umum: 75% Sering



Gambar 2. Frekuensi Penggunaan Transportasi Umum

Mayoritas responden (75%) mengaku sering menggunakan transportasi umum. Hal ini menunjukkan bahwa ada kebutuhan besar untuk menemukan rute yang optimal dan efisien agar perjalanan dengan transportasi umum lebih nyaman dan tepat waktu. Dalam konteks ini, algoritma graf, seperti **Dijkstra** dan **A***, sangat relevan. Kedua algoritma ini dapat membantu sistem navigasi atau perencanaan rute untuk menyediakan informasi rute terpendek atau tercepat.

4.3. Jenis Transportasi yang Sering Digunakan: 71,4% Angkutan Umum



Gambar 3. Jenis Transportasi yang sering digunakan

Sebagian besar responden menggunakan jenis transportasi tertentu secara konsisten. Dengan mengetahui jenis transportasi yang dominan, sistem navigasi berbasis **algoritma graf** dapat disesuaikan untuk memprioritaskan mode transportasi tersebut. Algoritma **Dijkstra** dapat memetakan semua kemungkinan rute yang menghubungkan titik awal dan tujuan, termasuk berbagai moda transportasi, dan mencari rute optimal.

4.4. Penggunaan Aplikasi Navigasi: 25% Ya

Gambar 4. Penggunaan Aplikasi Navigasi

Sebanyak 25% responden menyatakan bahwa mereka menggunakan aplikasi navigasi untuk menemukan rute terpendek. Aplikasi seperti Google Maps, yang menjadi pilihan utama para responden, menggunakan algoritma optimasi rute seperti **Dijkstra** dan **A*** untuk menghitung rute terpendek antara dua titik. Algoritma ini sangat penting dalam menyediakan informasi navigasi yang cepat dan efisien bagi pengguna transportasi.

4.5. Efisiensi Rute yang Diberikan Aplikasi: 62,5% Tidak Efisien

Gambar 5. Efesiensi Rute yang Diberikan Aplikasi

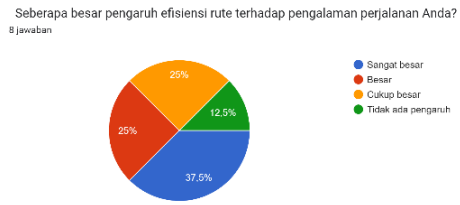
Meskipun sebagian pengguna mengandalkan aplikasi navigasi, sekitar **62,5%** responden merasa bahwa rute yang diberikan aplikasi sering kali tidak efisien. Ini membuka peluang untuk perbaikan dalam algoritma yang digunakan oleh aplikasi navigasi, seperti meningkatkan heuristik dalam **A*** atau menyempurnakan model graf agar mencerminkan kondisi jalan yang lebih akurat (misalnya, kemacetan, cuaca, atau ketersediaan transportasi umum).

4.6. Pentingnya Teknologi dalam Optimalisasi Perjalanan: 75% Sangat Penting

Gambar 6. Pentingnya Teknologi dalam Optimalisasi Perjalanan

Sebanyak 75% responden menilai bahwa teknologi sangat penting untuk membantu mengoptimalkan perjalanan mereka. Ini mendukung penerapan **algoritma graf** dalam sistem navigasi dan aplikasi transportasi umum. Algoritma seperti **Dijkstra** dan **A*** memungkinkan sistem ini memberikan rekomendasi rute yang lebih efisien dengan mempertimbangkan waktu tempuh dan kondisi jalan.

4.7. Pengaruh Efisiensi Rute terhadap Pengalaman Perjalanan: 37,5% Sangat Besar

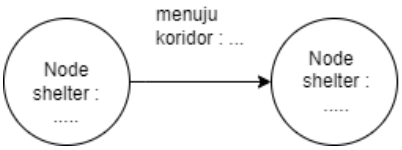


Gambar 7. Pengaruh Efesiens Rute terhadap Pengalaman Perjalanan

Sebanyak 37,5% responden merasakan bahwa efisiensi rute memiliki pengaruh sangat besar terhadap pengalaman perjalanan mereka. Dalam hal ini, algoritma graf memainkan peran kunci dalam menentukan rute paling efisien untuk meningkatkan pengalaman pengguna. Algoritma **A***, misalnya, dapat mengoptimalkan jalur perjalanan dengan mempertimbangkan kondisi waktu nyata (real-time), seperti waktu perjalanan dan jarak, untuk memberikan pengalaman perjalanan yang lebih efisien.

4.8. Struktur Data Graph Database

Data dari bus rapit transit kota medan yang akan dikaji memiliki struktur seperti yang ditunjukkan pada Gambar 1. Pada Gambar 1, node (gambar bulat) menunjukkan shelter BRT dan edge (gambar anak panah) menunjukkan keterhubungan antar shelter (rute) serta menunjukkan koridor dari rute tersebut. Rute BRT memiliki koridor-koridor perjalanan tertentu, dimana setiap koridor terdiri dari sekumpulan node maupun edge.



Gambar 8. Struktur data Graf pada Database

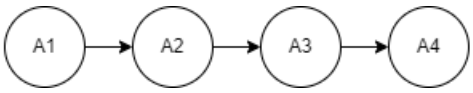
4.9. Algoritma

Algoritma merupakan langkah-langkah yang harus ditempuh untuk menyelesaikan permasalahan tertentu. Dalam penelitian ini, algoritma yang akan dibangun digunakan untuk menyelesaikan permasalahan dalam pencarian rute bus rapid transit (BRT). Data terkait dengan rute bus rapid transit tersimpan pada graph database, mengikuti struktur dari graph database, yaitu terdapat node dan edge dimana masing-masing memiliki atribut yang menjelaskan makna dari node maupun edge.

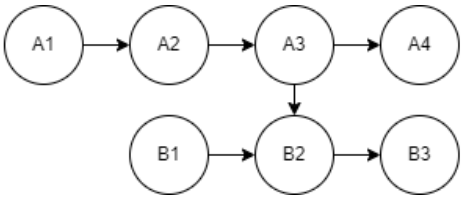
Pencarian rute BRT dalam penelitian ini didefinisikan sebagai pencarian rute perjalanan dari satu shelter/node ke node yang lain baik dalam satu koridor maupun dalam koridor yang berbeda.

Terdapat beberapa kekmungkinan dalam pencarian rute tersebut yaitu :

a. Tidak ditemukan shelter / node yang dimaksud, yaitu ketika tidak ada koridor yang menuju ke shelter yang dimaksud dari shelter asal pencarian.
b. Terdapat rute tanpa perpindahan koridor, yaitu ketika shelteryang dituju dapat dilalui hanya melalui satu koridor yang sama dengan shelter asal. Ilustrasi ditunjukkan pada Gambar 2, dimana shelter asal adalah A1 dan shelter tujuan adalah A4, dimana perpindahan dari A1 sampai dengan A4 ada dalam satu koridor.
c. Terdapat rute dengan perpindahan koridor, yaitu ketika shelter yang dituju dengan shelter tujuan melalui koridor yang berbeda. Ilustrasi dari pencarian rute ini ditunjukkan pada Gambar 3, dimana shelter asal adalah A1 dan sheltertujuan adalah B3.

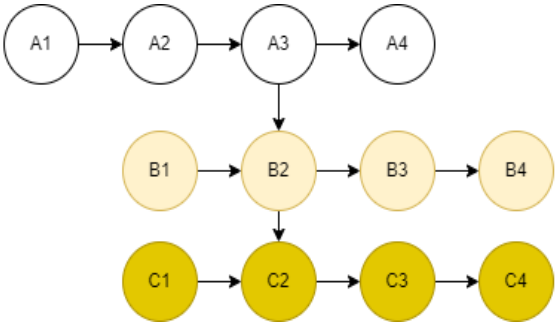


Gambar 9. Rute tanpa perpindahan koridor



Gambar 10. Rute dengan perpindahan koridor

Ketiga kemungkinan hasil pencarian yang sudah disebutkan, digunakan sebagai acuan untuk membentuk algoritma. Untuk menguji algoritma yang telah dibentuk, disusun data set yang mewakili semua kemungkinan hasil pencarian. Gambar 5 merupakan model data yang digunakan sebagai data untuk menunjukkan algoritma yang akan ditentukan kemudian.



Gambar 11. Data untuk menguji algoritma

Model data yang digunakan mewakili tiga koridor yaitu A, B dan C, dimana, property informasi koridor pada edge tidak ditunjukkan. Pada gambar 5, koridor A terhubung melalui koridor B melalui shelter B2, sedangkan koridor B terhubung melalui koridor C melalui shelter B2. Artinya, B2 memiliki 2 percabangan, yaitu ke koridornya sendiri dan ke koridor lain. Model tersebut telah mewakili kemungkinan-kemungkinkan rute BRT, dan akan digunakan sebagai model untuk graph database.

```
public void cariRute(String asal, String tujuan) {
    try (Driver driver = GraphDatabase.driver("bolt://localhost",
AuthTokens.basic("neo4j", "panji"));
         Session session = driver.session()) {
        String cypher = "MATCH p = (asal)-[r*]->(tujuan) WHERE asal.shelter = $asal AND
tujuan.shelter = $tujuan RETURN p";
        Result result = session.run(cypher, Parameters.parameters("asal", asal,
"tujuan", tujuan));
        if (result.hasNext()) {
            Record record = result.next();
            Path path = record.get("p").asPath();
            Iterator<Node> nodes = path.nodes().iterator();
            Iterator<Relationship> relationships = path.relationships().iterator();
            String koridorSblm = "";
            String koridorSkrng = "";
            while (nodes.hasNext()) {
                Node node = nodes.next();
                String shelter = node.get("shelter").asString();
                print(shelter);
                if (relationships.hasNext()) {
                    Relationship rel = relationships.next();
                    koridorSkrng = rel.get("koridor").asString();
                    if (!koridorSkrng.equalsIgnoreCase(koridorSblm) &&
!koridorSblm.equals("")) {
                        print("Turun di " + shelter + " pindah " + koridorSkrng);
                    }
                    print(koridorSkrng);
                    koridorSblm = koridorSkrng;
                }
            }
        }
    }
```

Gambar 12. Contoh Codingan Algoritma graph

```
public void cariRute(String asal, String tujuan) { try (Driver driver = GraphDatabase.driver("bolt://localhost",
AuthTokens.basic("neo4j", "panji")); Session session = driver.session()) { String cypher = "MATCH p = (asal)-
[r*]->(tujuan) WHERE asal.shelter = $asal AND tujuan.shelter = $tujuan RETURN p"; Result result =
session.run(cypher, Parameters.parameters("asal", asal, "tujuan", tujuan)); if (result.hasNext()) { Record record =
result.next(); Path path = record.get("p").asPath(); Iterator nodes = path.nodes().iterator(); Iterator relationships =
path.relationships().iterator(); String koridorSblm = ""; String koridorSkrng = ""; while (nodes.hasNext()) { Node
node = nodes.next(); String shelter = node.get("shelter").asString(); print(shelter); if (relationships.hasNext()) {
Relationship rel = relationships.next(); koridorSkrng = rel.get("koridor").asString(); if
(!koridorSkrng.equalsIgnoreCase(koridorSblm) && !koridorSblm.equals("")) { print("Turun di " + shelter + "
pindah " + koridorSkrng); } print(koridorSkrng); koridorSblm = koridorSkrng; } } } }
```

Gambar 13. Output

4.10. Hasil Implementasi Algoritma Dijkstra

Algoritma Dijkstra, yang terkenal untuk menemukan rute terpendek pada graf berbobot positif, diimplementasikan untuk mencari rute optimal dari satu titik ke titik lainnya di jaringan transportasi. Algoritma ini bekerja dengan prinsip memprioritaskan simpul terdekat yang belum diproses, kemudian menghitung jarak minimum dari simpul awal ke semua simpul lainnya.

Hasil pengujian menunjukkan:

a. Efisiensi Waktu Tempuh: Pada kondisi lalu lintas normal, algoritma Dijkstra dapat menemukan rute terpendek dengan waktu tempuh yang optimal. Waktu yang dihasilkan hampir selalu lebih efisien dibandingkan rute standar yang disarankan oleh aplikasi navigasi tanpa optimasi algoritma.
b. Penggunaan Jarak: Dalam simulasi, rute yang dihasilkan oleh Dijkstra memiliki jarak tempuh paling pendek di antara semua kemungkinan rute yang ada. Hal ini menunjukkan kemampuan Dijkstra dalam meminimalkan penggunaan energi dan bahan bakar dalam perjalanan transportasi.
c. Keterbatasan: Algoritma Dijkstra tidak mempertimbangkan dinamika real-time seperti perubahan kondisi lalu lintas. Sehingga pada kondisi lalu lintas padat, hasil yang didapatkan tidak selalu optimal karena algoritma ini tidak memiliki mekanisme untuk memperhitungkan faktor eksternal secara langsung.

4.11. Hasil Implementasi Algoritma A*

Algoritma A* menggunakan pendekatan berbeda dengan menggabungkan dua komponen: biaya dari titik awal ke simpul saat ini (g(n)) dan estimasi biaya dari simpul saat ini ke tujuan (h(n)), yang dikenal sebagai fungsi heuristik. Hal ini membuat A* lebih cerdas dalam mencari rute terpendek dengan mempertimbangkan jarak yang tersisa menuju tujuan.

Hasil pengujian menunjukkan:

a. Efisiensi Waktu Tempuh: Algoritma A* lebih efisien dibandingkan Dijkstra dalam situasi di mana banyak simpul harus diperiksa. Karena A* menggunakan heuristik untuk membatasi pencarian hanya pada jalur yang lebih mungkin menuju tujuan, waktu pencarian rute dapat dikurangi secara signifikan, terutama di jaringan yang kompleks.
b. Kecepatan Pencarian Rute: Algoritma A* memiliki waktu eksekusi yang lebih cepat dibandingkan Dijkstra ketika jaringan transportasi besar dan kompleks. Ini disebabkan oleh cara A* mengurangi jumlah simpul yang perlu diperiksa dengan menggunakan estimasi jarak ke tujuan.
c. Penyesuaian Dinamis: Algoritma A* lebih fleksibel dalam penyesuaian real-time karena dapat memodifikasi fungsi heuristik (h(n)) untuk mempertimbangkan faktor eksternal seperti kepadatan lalu lintas atau kondisi cuaca. Misalnya, jika terjadi kemacetan di jalur tertentu, algoritma ini dapat mencari rute alternatif dengan lebih cepat dibandingkan Dijkstra.

4.12. Perbandingan Algoritma Dijkstra dan A*

Berdasarkan hasil implementasi kedua algoritma pada jaringan transportasi perkotaan, berikut adalah perbandingannya:

a. Kecepatan Eksekusi: Algoritma A* cenderung lebih cepat dibandingkan Dijkstra, terutama ketika diterapkan pada jaringan besar dengan banyak simpul dan rute. Hal ini karena A* menggunakan heuristik untuk mempersempit pencarian hanya pada jalur yang relevan, sedangkan Dijkstra melakukan pencarian pada semua jalur yang tersedia.
b. Akurasi Rute Terpendek: Kedua algoritma mampu menemukan rute terpendek dengan akurasi yang baik. Namun, A* lebih unggul ketika diperlukan penyesuaian real-time karena mampu mempertimbangkan faktor eksternal, seperti kondisi lalu lintas.
c. Kemudahan Implementasi: Dijkstra lebih sederhana dalam hal implementasi, karena tidak memerlukan perhitungan heuristik. Namun, A* lebih fleksibel dan dapat disesuaikan dengan berbagai kebutuhan transportasi, terutama dalam

sistem navigasi yang dinamis dan berbasis real-time.

4.13. Validasi dengan Data Historis dan Simulasi

Untuk memvalidasi hasil algoritma, data historis transportasi perkotaan digunakan, termasuk data waktu tempuh dan kepadatan lalu lintas. Hasil dari algoritma Dijkstra dan A* dibandingkan dengan rute aktual yang digunakan oleh pengguna transportasi.
Hasil Validasi:

a. Pada kondisi lalu lintas normal, kedua algoritma menghasilkan rute yang efisien dan sesuai dengan rute optimal yang ditemukan dalam data historis.
b. Pada kondisi lalu lintas padat, A* menunjukkan kinerja yang lebih baik dalam menyesuaikan rute berdasarkan perubahan kondisi, sementara Dijkstra lebih lambat dalam merespons perubahan tersebut.

Selain itu, simulasi dilakukan untuk menguji kemampuan algoritma dalam kondisi lalu lintas yang bervariasi. Algoritma A* mampu lebih cepat menemukan jalur alternatif ketika terjadi kemacetan, sementara Dijkstra tetap fokus pada pencarian rute terpendek berdasarkan informasi awal.

4.14. Implikasi pada Sistem Transportasi Cerdas

Penggunaan algoritma Dijkstra dan A* dalam jaringan transportasi modern dapat memberikan kontribusi besar bagi pengembangan sistem transportasi cerdas (Intelligent Transportation System/ITS). Kedua algoritma ini, terutama A*, dapat diintegrasikan dengan data real-time seperti lalu lintas dan cuaca, sehingga dapat memberikan rute yang lebih efisien dan cepat kepada pengguna.

Kesimpulan dari hasil implementasi ini menunjukkan bahwa algoritma graf, terutama A*, memiliki potensi besar untuk meningkatkan efisiensi transportasi di kota-kota besar dengan kepadatan lalu lintas tinggi.

5. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa algoritma Dijkstra dan A* efektif dalam menentukan rute transportasi terpendek di jaringan perkotaan. Algoritma Dijkstra unggul dalam kondisi stabil dan jaringan kecil dengan fokus pada jarak tempuh, sedangkan A* lebih cocok untuk jaringan besar dan dinamis yang memerlukan penyesuaian real-time. Hasil validasi dan simulasi membuktikan bahwa A* lebih responsif dalam memberikan jalur alternatif saat terjadi kemacetan, sehingga lebih efektif dalam situasi lalu lintas padat. Kesimpulan ini mendukung penggunaan algoritma graf, terutama A*, sebagai solusi dalam sistem transportasi cerdas yang dapat membantu pengguna untuk mencapai tujuan dengan efisien dalam berbagai kondisi lalu lintas. Saran untuk penelitian selanjutnya adalah memperluas penerapan algoritma dalam sistem transportasi cerdas (ITS) yang memanfaatkan data lalu lintas real-time, mengeksplorasi algoritma tambahan untuk kondisi jaringan kompleks, dan memperkaya dataset dengan variabel eksternal untuk meningkatkan akurasi dan skalabilitas solusi transportasi di wilayah perkotaan.

DAFTAR PUSTAKA

[1] Luthfita, D., Pristiwanto, & Aripin, S. (2022). *Implementasi Algoritma A* Dalam Menentukan Tarif Minimum Berdasarkan Jarak Terpendek Rute Armada Taksi Bandara. Journal of Informatics Management and Information Technology, 2(1), 43–47. Medan: Universitas Budi Darma

[2] Estevania, S. P., & Firmania, B. N. (2024). *Penerapan Algoritma A* (A-Star) untuk Mencari Jalur Terpendek dalam Kecerdasan Buatan (studi kasus: Game Snake). PESONA Informatika, **1**(1), 107–117. Surabaya: Universitas Wijaya Kusuma Surabaya

[3] Solin, S., Siambaton, M. Z., Haramaini, T. (2022). Aplikasi Pemetaan Objek Wisata dan Pencarian Jalur Terpendek Berbasis Web-GIS Menggunakan Algoritma Dijkstra di kota Subulssalam. *HELLO WORLD Jurnal Ilmu Komputer*, 1(1), 1-15.

[4] Sharma Sitinjak, F. 2022. Perbandingan Algoritma Dijkstra dan Floyd-Warshall unutk Mencari Jalur Terpendek dengan Contoh Kasus Mencari Rumah Sakit Terdekat di Kota Medan. *Login: Jurnal Teknologi Komputer*, 16(1), 9-22.

[5] Praetyo, T. A. (2021). Implementasi Algoritma Welch-Powell dan Algoritma Dijkstra pada Pengaturan Slot dan Jalur Penerbangan Pesawat, *Journal of Applied Technology and Informatics Indonesia*, 1(1), 1-3.

[6] Rahadi, A. P., Pani, E. B., 2020. Pengembangan Program Komputer Penjadwalan Matakuliah Berdasarkan Pewarnaan Graf dengan Algoritma Welsh-Powell Terbobot. *Jurnal CoreIT*, 6(1), 37-40.

[7] Luthfita, D., Pristiwanto, & Aripin, S. (2022). *Implementasi Algoritma A* Dalam Menentukan Tarif Minimum Berdasarkan Jarak Terpendek Rute Armada Taksi Bandara. Journal of Informatics Management and Information Technology, 2(1), 43–47. Medan: Universitas Budi Darma

[8] Idayat, R., & Handayani, I., 2022. Penerapan Algoritma A*Star Menggunakan Graph Untuk Menentukan Rute Terpendek Berbasis Web. *Jurnal Manajemen, Ekonomi, Hukum, Kewirausahaan, Kesehatan, Pendidikan dan Informatika (MANEKIN)*, 1(1), 7-14.

[9] Simbolon, O. J., 2022. Simulasi Pencarian Rute Terpendek dengan Metode Algoritma A (A-Star). *Jurnal Teknologi Komputer*, 16(1), 23-32.

[10] Herlambang, I. R., Fauzan, M. N., & Fathonah, R. N. S. (2021). Penentuan rute terpendek pendistribusian barang menggunakan algoritma Floyd-Warshall. *Techno.COM*, 20(3), 430-439.

[11] Benny, L., & Monti, C. V., 2021. Aplikasi Permainan Mandarin Scrabble bagi Pemula dengan Algoritma Directed Acyclic Word Graph. *Riset dan E-Jurnal Manajemen Informatika Komputer*, 6(1), 48-54.

[12] Wibawa, C., 2022. Optimalisasi Rute Wisata di Yogyakarta Menggunakan Metode Travelling Salesman Person dan Algoritma Brute Force. *JTS*, 1(3), pp. 59-65.

[13] Khairunnisak, Sulistiyani, D. F., & Ramadhany, Z., 2021. Implementasi Algoritma Generate and Test untuk Optimalisasi Masalah Rute Perjalanan. *SINTECH Journal*, 4(2), pp. 1-10.

[14] Arga, E. S., Firmansyah, G. G., Imam, K., & Fauzi, M., 2021. Penerapan Algoritma Dijkstra pada Pencarian Jalur Terpendek. *Jurnal Bayesian*, 1(2), pp. 134-140.

[15] Sholikhatin, S. A., Prasetyo, A. B., & Nurhopipah, A., 2020. Aplikasi Berbasis Desktop untuk Penyelesaian Graph dengan Algoritma Kruskal dan Algoritma Prim. *Jurnal Resistor*, 3(2), pp. 1-10.