Analisis Efisiensi Algoritma Pencarian Graf dalam Penentuan Rute KRL Commuter Line Jabodetabek

Rhenaldy Cahyadi Putra - 13524039
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: rhenitb@gmail.com, 13524039@std.stei.itb.ac.id

***Abstract*— Sistem transportasi publik KRL Commuter Line memiliki jaringan stasiun yang kompleks sehingga memerlukan metode pencarian rute yang efisien untuk membantu penentuan perjalanan. Makalah ini menganalisis efisiensi tiga algoritma pencarian graf, yaitu Uniform Cost Search (UCS), Greedy Best-First Search (GBFS), dan A* pada jaringan KRL Commuter Line. Jaringan KRL dimodelkan sebagai graf dengan stasiun sebagai simpul dan koneksi antar stasiun sebagai sisi berbobot berdasarkan waktu tempuh perjalanan. Evaluasi dilakukan menggunakan sejumlah pasangan stasiun asal dan tujuan dengan membandingkan kualitas rute yang dihasilkan, waktu komputasi, serta jumlah simpul yang diekspansi selama proses pencarian. Untuk mendukung implementasi GBFS dan A*, digunakan heuristik berupa jarak geografis garis lurus menuju stasiun tujuan yang dihitung berdasarkan koordinat latitude dan longitude setiap stasiun. Hasil pengujian menunjukkan adanya trade-off antara optimalitas solusi dan efisiensi pencarian pada setiap algoritma. Analisis yang dilakukan memberikan gambaran mengenai karakteristik masing-masing algoritma serta kesesuaiannya untuk diterapkan pada sistem navigasi transportasi publik berbasis graf.**

Keywords*— KRL Jabodetabek, Breadth-First Search, Uniform Cost Search, Greedy Best-First Search, A*

I. Pendahuluan

Transportasi publik merupakan salah satu komponen penting dalam mendukung mobilitas masyarakat perkotaan. Di wilayah Jabodetabek, KRL Commuter Line menjadi salah satu moda transportasi utama yang melayani jutaan perjalanan penumpang setiap bulannya melalui jaringan stasiun yang tersebar di berbagai wilayah. Dengan banyaknya stasiun dan jalur yang saling terhubung, penentuan rute perjalanan yang efisien menjadi permasalahan yang penting bagi pengguna transportasi publik.

Permasalahan penentuan rute pada jaringan transportasi dapat dimodelkan sebagai permasalahan pencarian jalur pada graf. Dalam model tersebut, stasiun direpresentasikan sebagai simpul, sedangkan koneksi antar stasiun direpresentasikan sebagai sisi. Dengan representasi ini, berbagai algoritma pencarian graf dapat digunakan untuk menemukan rute dari stasiun asal menuju stasiun tujuan berdasarkan kriteria tertentu, seperti jumlah pemberhentian atau waktu perjalanan.

Berbagai algoritma pencarian graf telah dikembangkan dengan karakteristik yang berbeda. Uniform Cost Search (UCS) mempertimbangkan biaya perjalanan untuk memperoleh jalur dengan biaya total minimum. Sementara itu, Greedy Best-First Search (GBFS) dan A* memanfaatkan informasi heuristik untuk mengarahkan proses pencarian sehingga diharapkan dapat meningkatkan efisiensi eksplorasi simpul.

Meskipun algoritma-algoritma tersebut telah banyak digunakan dalam berbagai permasalahan pencarian jalur, performanya dapat berbeda bergantung pada karakteristik graf yang digunakan. Jaringan KRL Commuter Line Jabodetabek memiliki struktur jaringan yang cenderung radial, di mana sebagian besar jalur menghubungkan wilayah suburban dengan pusat kota melalui beberapa stasiun transit utama. Struktur ini menyebabkan banyak rute perjalanan melewati stasiun transit utama yang sama sehingga pola pencarian jalur yang terbentuk berbeda dibandingkan pada graf yang lebih tersebar. Karakteristik tersebut menarik untuk dianalisis guna memahami bagaimana setiap algoritma pencarian graf bekerja pada sistem transportasi publik yang nyata serta bagaimana struktur jaringan mempengaruhi efisiensi proses pencarian rute.

Gambar 1 - Peta KRL Commuter Line Jabodetabek

Oleh karena itu, penelitian ini bertujuan untuk memodelkan jaringan KRL Commuter Line Jabodetabek sebagai graf berbobot dan menganalisis efisiensi beberapa algoritma pencarian graf, yaitu UCS, GBFS, dan A*. Evaluasi dilakukan berdasarkan kualitas rute yang dihasilkan, jumlah simpul yang diekspansi, serta waktu komputasi yang dibutuhkan selama proses pencarian.

II. Landasan Teori

A. Teori Graf

Graf merupakan struktur data yang digunakan untuk merepresentasikan hubungan antar objek dalam bentuk simpul (vertex) dan sisi (edge). Secara formal, graf dapat dinyatakan sebagai pasangan himpunan

$$G = (V, E)$$

dengan V menyatakan himpunan simpul dan E menyatakan himpunan sisi yang menghubungkan pasangan simpul dalam graf. Graf banyak digunakan dalam berbagai permasalahan komputasi seperti jaringan komputer, media sosial, sistem navigasi, serta perencanaan rute transportasi.

Dalam permasalahan pencarian rute, setiap objek yang memiliki hubungan dengan objek lain dapat direpresentasikan sebagai simpul, sedangkan hubungan antar objek direpresentasikan sebagai sisi. Dengan representasi tersebut, proses pencarian jalur dapat dilakukan dengan mencari urutan simpul yang menghubungkan suatu simpul awal (source) menuju simpul tujuan (destination).

Pada penelitian ini, jaringan KRL Commuter Line Jabodetabek dimodelkan sebagai sebuah graf, dengan stasiun KRL sebagai simpul dan perjalanan antar stasiun sebagai sisi. Setiap sisi memiliki informasi tambahan berupa waktu keberangkatan dan waktu kedatangan kereta, sehingga graf yang digunakan tidak hanya merepresentasikan hubungan antar stasiun, tetapi juga karakteristik perjalanan berdasarkan jadwal operasional kereta.

B. Time Dependent Graph

Pada banyak permasalahan dunia nyata, hubungan antar simpul tidak selalu tersedia setiap saat. Kondisi tersebut dapat dimodelkan menggunakan *time-dependent graph*, yaitu graf yang ketersediaan atau bobot sisi bergantung pada waktu tertentu. Berbeda dengan graf statis yang menganggap setiap sisi selalu dapat digunakan, pada time-dependent graph suatu sisi hanya dapat dilalui pada waktu-waktu tertentu sesuai kondisi sistem yang dimodelkan.

Jaringan transportasi publik merupakan salah satu contoh penerapan time-dependent graph yang paling umum. Dalam sistem transportasi berbasis jadwal, suatu perjalanan hanya dapat dilakukan apabila kendaraan yang melayani rute tersebut tersedia pada waktu yang sesuai. Sebagai contoh, seorang penumpang hanya dapat menggunakan perjalanan kereta apabila waktu keberangkatannya lebih lambat atau sama dengan waktu kedatangan penumpang di stasiun asal.

Pada penelitian ini, jaringan KRL Commuter Line dimodelkan sebagai time-dependent graph. Setiap stasiun direpresentasikan sebagai simpul, sedangkan setiap perjalanan antar stasiun direpresentasikan sebagai sisi yang memiliki informasi waktu keberangkatan dan waktu kedatangan. Dengan demikian, suatu sisi hanya dapat digunakan apabila memenuhi batasan waktu yang berlaku.

C. Rumus Haversine

Rumus Haversine merupakan metode yang digunakan untuk menghitung jarak garis lurus (*great-circle distance*) antara dua titik pada permukaan bumi berdasarkan koordinat lintang (*latitude*) dan bujur (*longitude*). Metode ini banyak digunakan pada sistem navigasi dan pencarian rute karena mampu memperhitungkan bentuk bumi yang mendekati bola.

Secara matematis, jarak antara dua titik dapat dihitung menggunakan persamaan:

$$d = 2Rarcsin(\sqrt{sin^2(\frac{\Delta\phi}{2}) + cos(\phi 1)cos(\phi 2)sin^2(\frac{\Delta\lambda}{2})}$$

Pada penelitian ini, rumus Haversine digunakan untuk menghitung jarak geografis antar stasiun KRL berdasarkan koordinat yang diperoleh dari OpenStreetMap. Jarak yang diperoleh kemudian dikonversi menjadi estimasi waktu perjalanan dengan asumsi kecepatan rata-rata KRL sebesar 40 km/jam. Nilai estimasi waktu tersebut digunakan sebagai fungsi heuristik pada algoritma Greedy Best First Search (GBFS) dan A* Search.

D. Uniform Cost Search (UCS)

Uniform Cost Search (UCS) merupakan algoritma pencarian yang selalu memilih simpul dengan biaya kumulatif terkecil untuk diekspansi terlebih dahulu. Algoritma ini menggunakan priority queue untuk mengurutkan simpul berdasarkan biaya perjalanan yang telah ditempuh dari simpul awal. UCS bersifat optimal pada graf dengan biaya non-negatif karena solusi pertama yang mencapai tujuan dijamin memiliki biaya minimum. Dalam konteks pencarian rute KRL, biaya pada penelitian ini direpresentasikan sebagai waktu kedatangan aktual pada suatu stasiun.

E. Greedy Best First Search (GBFS)

Greedy Best First Search (GBFS) merupakan algoritma pencarian heuristik yang memilih simpul berdasarkan estimasi biaya menuju tujuan. Berbeda dengan UCS, GBFS tidak mempertimbangkan biaya yang telah ditempuh, melainkan hanya menggunakan fungsi heuristik h(n). Pendekatan ini umumnya menghasilkan proses pencarian yang lebih cepat karena ruang pencarian menjadi lebih terarah. Namun, karena mengabaikan biaya aktual, GBFS tidak menjamin bahwa solusi yang diperoleh merupakan solusi optimal.

F. A Star

A Star (A*) merupakan algoritma pencarian heuristik yang menggabungkan biaya aktual dan estimasi biaya menuju tujuan. Pemilihan simpul dilakukan berdasarkan fungsi evaluasi

$$f(n) = g(n) + h(n)$$

dengan g(n) menyatakan biaya aktual dari simpul awal ke simpul n, sedangkan h(n) menyatakan estimasi biaya dari simpul n menuju tujuan. Dengan mengkombinasikan kedua komponen tersebut, A* mampu mengarahkan pencarian secara lebih efisien dibandingkan UCS. Pada graf dengan fungsi heuristik *admissible*, A* dapat menjamin solusi optimal.

III. METODOLOGI

A. Tahapan Penelitian

Penelitian dilakukan melalui beberapa tahapan yang meliputi pengumpulan data, pemodelan jaringan KRL ke dalam bentuk graf, implementasi algoritma pencarian, serta evaluasi kinerja algoritma. Alur penelitian dapat dilihat pada Gambar 2.

Gambar 3.A.1 - Flowchart tahapan penelitian

B. Persiapan Data

B.1 Data Jadwal Perjalanan KRL

Data jadwal perjalanan KRL Commuter Line diperoleh dari dataset publik yang tersedia pada platform Kaggle. Dataset tersebut merupakan hasil scraping terhadap API publik KAI Commuter dan berisi informasi jadwal perjalanan kereta pada setiap stasiun yang dilalui. Data ini digunakan sebagai sumber utama dalam pembentukan graf perjalanan KRL. Tabel I menunjukkan data yang tersedia:

TABLE I. TABEL ATRIBUT RAW DATA YANG DIGUNAKAN

Atribut	Deskripsi
train_id	Identitas kereta
train_name	Nama kereta berdasarkan ID
route_name	Nama rute perjalanan
station_destination_name	Nama stasiun tujuan akhir
departure_time_utc7	Waktu keberangkatan dari stasiun saat ini
color	Warna identitas jalur
destination_time_utc7	Waktu tiba di stasiun tujuan akhir
station_departure_name	Nama stasiun saat ini
station_departure_id	Kode stasiun saat ini
station_destination_id	Kode stasiun tujuan akhir

Data mentah tersebut belum dapat langsung digunakan dalam proses pencarian rute karena setiap baris hanya merepresentasikan posisi kereta pada suatu stasiun pada waktu tertentu. Oleh karena itu dilakukan proses transformasi data untuk membentuk hubungan antar stasiun yang dapat direpresentasikan sebagai sisi pada graf.

B.2 Transformasi Data Perjalanan

Data jadwal perjalanan yang diperoleh dari Kaggle masih berbentuk data pemberhentian kereta pada setiap stasiun. Setiap baris hanya menunjukkan posisi suatu kereta pada waktu tertentu sehingga belum dapat langsung direpresentasikan sebagai graf.

Oleh karena itu dilakukan proses transformasi data pada file *data_preparation.py.* Proses transformasi dimulai dengan mengelompokkan data berdasarkan train_id, kemudian mengurutkan setiap kelompok berdasarkan waktu keberangkatan secara kronologis. Selanjutnya, pasangan stasiun yang berurutan pada lintasan kereta dihubungkan untuk membentuk segmen perjalanan antarstasiun.

Sebagai contoh, apabila suatu kereta tercatat berhenti di Stasiun Manggarai pada pukul 04.33 dan kemudian di Stasiun Tebet pada pukul 04.36, maka akan dibentuk satu segmen perjalanan dari Manggarai ke Tebet dengan waktu keberangkatan 04.33 dan waktu kedatangan 04.36.

Hasil dari proses tersebut adalah dataset *Commuter Line_trips.csv* yang berisi informasi perjalanan antar stasiun dan siap digunakan sebagai representasi sisi (edge) pada graf. Tabel II menunjukkan struktur data hasil transformasi.

TABLE II. TABEL DATA PERJALANAN KERETA

Atribut	Deskripsi
train_id	Identitas kereta
route_name	Nama rute perjalanan
color	Warna identitas jalur
source_station_id	Kode stasiun asal
source_station_name	Nama stasiun asal
target_station_id	Kode stasiun tujuan
target_station_name	Nama stasiun tujuan
departure_time	Waktu keberangkatan
arrival_time	Waktu kedatangan
duration_minutes	Durasi perjalanan (menit)

B.3 Data Koordinat Stasiun

Selain data jadwal perjalanan, penelitian ini juga memerlukan data koordinat geografis setiap stasiun KRL. Data koordinat

diperoleh melalui layanan Overpass API yang merupakan antarmuka kueri dari OpenStreetMap (OSM).

Proses pengambilan data dilakukan dengan mencari objek bertipe *railway = station* pada wilayah Jabodetabek. Nama stasiun yang diperoleh dari OpenStreetMap kemudian dicocokkan dengan daftar stasiun KRL yang digunakan dalam penelitian. Pada beberapa kasus dilakukan normalisasi nama stasiun dan pemetaan manual untuk mengatasi perbedaan penamaan antara dataset jadwal perjalanan dan data OpenStreetMap.

Hasil proses tersebut menghasilkan dataset *Commuter Line_station_coordinates.csv* yang berisi kode stasiun beserta koordinat lintang (latitude) dan bujur (longitude). Tabel III menunjukkan struktur data koordinat stasiun.

TABLE III. TABEL DATA KOORDINAT STASIUN

Atribut	Deskripsi
station	Nama stasiun
code	Kode stasiun
latitude	Koordinat lintang
longitude	Koordinat bujur

Data koordinat ini digunakan untuk menghitung jarak geografis antar stasiun menggunakan rumus Haversine yang selanjutnya dimanfaatkan sebagai fungsi heuristik pada algoritma Greedy Best First Search (GBFS) dan A*.

C. Pemodelan Graf

Data perjalanan yang telah ditransformasikan kemudian direpresentasikan sebagai time-dependent graph. Pada model ini, setiap stasiun direpresentasikan sebagai simpul (vertex), sedangkan perjalanan direpresentasikan sebagai sisi (edge) yang memiliki informasi waktu keberangkatan dan waktu kedatangan. Setiap sisi direpresentasikan sebagai:

$$e = (u, v, td, ta)$$

di mana u adalah stasiun asal, v adalah stasiun tujuan, td adalah waktu keberangkatan, dan ta adalah waktu kedatangan.

Dalam implementasinya, graf disimpan menggunakan struktur adjacency list. Setiap simpul menyimpan daftar perjalanan yang dapat dilakukan dari stasiun tersebut beserta informasi jadwal keretanya. Representasi ini memungkinkan algoritma pencarian mempertimbangkan aspek waktu sehingga rute yang dihasilkan sesuai dengan jadwal operasional KRL.

D. Perancangan Heuristik

Pada penelitian ini, heuristik dihitung berdasarkan jarak geografis antar stasiun menggunakan rumus Haversine yang memanfaatkan koordinat lintang (*latitude*) dan bujur (*longitude*) setiap stasiun.

Nilai jarak yang diperoleh kemudian dikonversi menjadi estimasi waktu perjalanan dengan menggunakan nilai kecepatan rata-rata KRL sebesar 40 km/jam. Estimasi waktu tersebut digunakan sebagai nilai heuristik h(n). Secara umum, fungsi heuristik dapat dinyatakan sebagai:

$$h(n) = \frac{d(n,goal)}{v} \times 60$$

dengan:

- d(n,goal) = jarak geografis dari simpul n ke tujuan (km),
- v = kecepatan rata-rata kereta (km/jam),
- h(n) = estimasi waktu tempuh dalam menit.

Heuristik ini digunakan untuk mengarahkan proses pencarian menuju stasiun tujuan secara lebih efisien.

E. Implementasi UCS, GBFS, dan A*

Penelitian ini mengimplementasikan tiga algoritma pencarian rute pada jaringan KRL yang telah dimodelkan sebagai time-dependent graph, yaitu Uniform Cost Search (UCS), Greedy Best First Search (GBFS), dan A* Search.

UCS digunakan sebagai metode dasar yang selalu memilih simpul dengan waktu kedatangan aktual paling kecil. GBFS menggunakan nilai heuristik untuk memilih simpul yang diperkirakan paling dekat dengan tujuan, sedangkan A* mengombinasikan biaya aktual dan nilai heuristik untuk memperoleh pencarian yang lebih efisien.

F. Analisis Efisiensi Algoritma

Untuk menilai performa algoritma yang diimplementasikan, dilakukan analisis efisiensi terhadap Uniform Cost Search (UCS), Greedy Best First Search (GBFS), dan A* Search pada beberapa skenario pencarian rute KRL. Analisis dilakukan dengan membandingkan hasil pencarian yang diperoleh masing-masing algoritma pada graf.

Parameter yang digunakan dalam analisis meliputi:

1. Rute yang dihasilkan, yaitu urutan stasiun yang dilalui dari stasiun asal menuju stasiun tujuan.
2. Waktu kedatangan akhir, yaitu waktu tiba pengguna di stasiun tujuan berdasarkan jadwal perjalanan KRL.
3. Jumlah simpul yang diekspansi, yaitu banyaknya simpul yang diproses selama proses pencarian berlangsung.

Hasil pengujian kemudian dianalisis untuk melihat perbedaan karakteristik masing-masing algoritma dalam menemukan rute pada jaringan KRL Jabodetabek.

IV. Hasil dan Implementasi

A. Implementasi

A.1 Implementasi Heuristik

Algoritma GBFS dan A* memerlukan fungsi heuristik untuk memperkirakan kedekatan suatu stasiun terhadap tujuan. Pada penelitian ini, heuristik dihitung berdasarkan jarak geografis antar stasiun menggunakan rumus Haversine yang memanfaatkan koordinat lintang (latitude) dan bujur (longitude) setiap stasiun.

```
distance = haversine_distance(
    station_a,
    station_b,
    stations
)
```

Fungsi tersebut menghasilkan estimasi jarak garis lurus antara dua stasiun dalam satuan kilometer. Selanjutnya, jarak tersebut dikonversi menjadi estimasi waktu perjalanan dengan menggunakan data kecepatan rata-rata KRL (40 km/jam).

```
time_minutes = (
distance / average_speed
) * 60
```

Nilai waktu yang dihasilkan digunakan sebagai fungsi heuristik h(n) pada algoritma GBFS dan A*. Semakin dekat suatu stasiun terhadap tujuan secara geografis, semakin kecil nilai heuristik yang diberikan.

A.2 Implementasi Uniform Cost Search (UCS)

UCS memilih simpul dengan biaya aktual terkecil untuk diekspansi terlebih dahulu. Pada penelitian ini biaya aktual direpresentasikan oleh waktu kedatangan pada suatu stasiun.

Untuk merealisasikan mekanisme tersebut, implementasi UCS menggunakan struktur data *priority queue*. Simpul awal dimasukkan ke dalam antrian prioritas dengan biaya awal berupa waktu keberangkatan yang diberikan pengguna.

```
heapq.heappush(
    pq,
    (
        edge["arrival"],
        ...
    )
)
```

Pada potongan kode tersebut, elemen pertama pada priority queue adalah edge["arrival"], yaitu waktu kedatangan pada stasiun berikutnya. Akibatnya, simpul dengan waktu kedatangan paling awal akan selalu diproses terlebih dahulu sesuai prinsip kerja UCS.

A.3 Implementasi Greedy Best First Search (GBFS)

Greedy Best-First Search (GBFS) memilih simpul berdasarkan nilai heuristik terkecil. Pada penelitian ini, heuristik dihitung menggunakan estimasi waktu tempuh yang diperoleh dari jarak geografis antar stasiun. Implementasi GBFS juga menggunakan *priority queue*, namun prioritas pencarian ditentukan oleh nilai heuristik.

```
heapq.heappush(
    pq,
    (
        heuristic_time(
            next_station,
            goal,
            stations
        ),
        ...
    )
```

Pada potongan kode tersebut, elemen pertama pada priority queue adalah nilai heuristik yang merepresentasikan estimasi waktu tersisa menuju tujuan. Oleh karena itu, simpul yang diperkirakan paling dekat dengan tujuan akan diproses terlebih dahulu.

A.4 Implementasi A*

A* merupakan pengembangan dari UCS dan GBFS yang menggabungkan biaya aktual perjalanan dengan informasi heuristik. Pada penelitian ini, biaya aktual g(n) direpresentasikan oleh waktu kedatangan aktual pada suatu stasiun, sedangkan heuristik h(n) merupakan estimasi waktu tersisa menuju tujuan yang dihitung menggunakan jarak Haversine. Nilai prioritas pencarian dihitung menggunakan fungsi evaluasi:

$$f(n) = g(n) + h(n)$$

Implementasi fungsi evaluasi tersebut ditunjukkan pada potongan kode berikut.

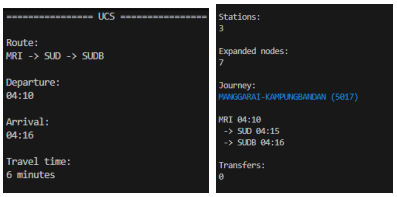
```
g = edge["arrival"]

h = heuristic_time(
    next_station,
    goal,
    stations
)

f = g + h

heapq.heappush(
    pq,
    (
        f,
        ...
    )
)
```

Pada potongan kode tersebut, nilai g(n) merepresentasikan waktu kedatangan aktual pada stasiun berikutnya, sedangkan h(n) merupakan estimasi waktu yang masih dibutuhkan untuk mencapai tujuan. Kedua komponen tersebut kemudian dijumlahkan untuk menghasilkan nilai f(n) yang digunakan sebagai prioritas pencarian.

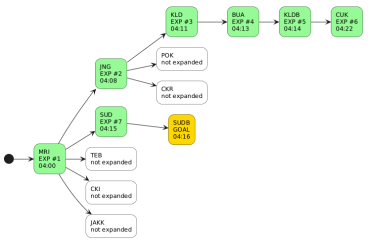
Dengan mempertimbangkan biaya aktual sekaligus informasi heuristik, A* mampu mengarahkan pencarian menuju tujuan secara lebih efisien dibandingkan UCS tanpa mengabaikan kualitas solusi yang diperoleh.

B. Hasil dan Analisis

Pengujian dilakukan pada rute Manggarai (MRI) menuju BNI City (SUDB) dengan waktu keberangkatan yang diberikan pengguna sebesar pukul 04.00 WIB. Ketiga algoritma diuji menggunakan graf dan data jadwal yang sama sehingga perbedaan hasil yang diperoleh sepenuhnya disebabkan oleh strategi pencarian masing-masing algoritma.

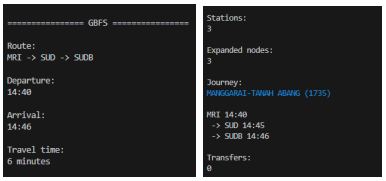


Gambar 4.B.1 dan 4.B.2 - Rute ditemukan melalui UCS

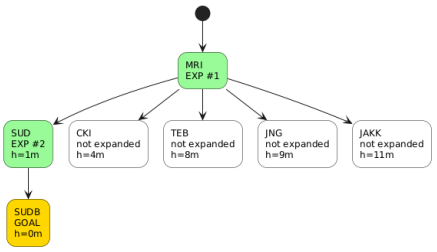


Gambar 4.B.3 - Expansion Graph UCS

UCS mengeksplorasi beberapa simpul yang memiliki waktu kedatangan paling awal sebelum mencapai tujuan. Dari Manggarai (MRI), algoritma terlebih dahulu memilih Jatinegara (JNG) karena memiliki waktu kedatangan lebih cepat dibandingkan kandidat lainnya. Proses ini berlanjut ke beberapa simpul lain yang juga memiliki biaya aktual lebih kecil sebelum akhirnya mencapai Sudirman (SUD) dan BNI City (SUDB). Hal ini terjadi karena UCS hanya mempertimbangkan biaya aktual berupa waktu kedatangan tanpa memanfaatkan informasi mengenai lokasi tujuan. Akibatnya, algoritma tidak dapat mengarahkan pencarian secara langsung ke tujuan dan harus mengevaluasi lebih banyak simpul. Meskipun demikian, pendekatan tersebut menjamin bahwa solusi yang diperoleh merupakan rute dengan waktu kedatangan paling awal yang dapat dicapai dari waktu keberangkatan yang diberikan.



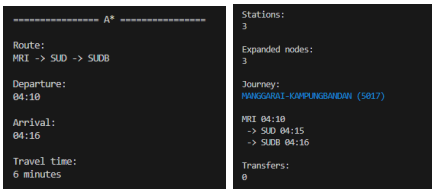
Gambar 4.B.4 dan 4.B.5 - Rute ditemukan melalui GBFS



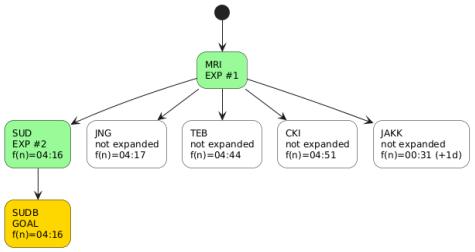
Gambar 4.B.6 - Expansion Graph GBFS

GBFS hanya mengekspansi dua simpul sebelum mencapai tujuan, yaitu Manggarai (MRI) dan Sudirman (SUD). Hal ini terjadi karena algoritma selalu memilih simpul dengan nilai heuristik terkecil, yaitu stasiun yang secara geografis paling dekat dengan tujuan. Akibatnya, ruang pencarian yang dieksplorasi menjadi sangat kecil sehingga proses pencarian dapat dilakukan dengan cepat.

Namun, karena GBFS tidak mempertimbangkan biaya aktual berupa waktu perjalanan maupun waktu tunggu kereta, algoritma tidak selalu menghasilkan rute terbaik. Pada kasus ini, GBFS memilih perjalanan Manggarai–Sudirman dengan keberangkatan pukul 14.40, meskipun tersedia perjalanan yang lebih awal. Hal tersebut terjadi karena terdapat beberapa perjalanan menuju stasiun yang sama dengan nilai heuristik identik, sehingga algoritma memilih entri yang pertama kali ditemukan pada struktur graf. Akibatnya, solusi yang diperoleh valid tetapi tidak optimal dari sisi waktu kedatangan.



Gambar 4.B.7 dan 4.B.8 - Rute ditemukan melalui A*



Gambar 4.B.9 - Expansion Graph A*

Dapat dilihat bahwa A* menghasilkan expansion graph yang identik dengan GBFS. Hal ini terjadi karena tujuan yang dicari berada sangat dekat dengan simpul awal sehingga nilai heuristik yang digunakan mampu langsung mengarahkan pencarian menuju jalur optimal. Pada ekspansi pertama, simpul Sudirman (SUD) memiliki nilai fungsi evaluasi $f(n) = g(n) + h(n)$ paling kecil dibandingkan kandidat lainnya, sehingga dipilih untuk diekspansi terlebih dahulu. Setelah mencapai Sudirman, algoritma langsung menemukan simpul tujuan BNI City (SUDB). Akibatnya, A* hanya memerlukan dua kali ekspansi simpul untuk memperoleh solusi.

```
======================================================
ALGORITHM COMPARISON
======================================================
Algorithm   Arrival     Expanded    Runtime(s)
------------------------------------------------------
UCS         04:16       7           0.013914
GBFS        14:46       3           0.005755
A*          04:16       3           0.006305
```

Gambar 4.B.10 - Perbandingan Seluruh Algoritma

UCS dan A* menghasilkan waktu kedatangan yang sama, yaitu pukul 04.16, sedangkan GBFS menghasilkan waktu kedatangan yang jauh lebih lambat, yaitu pukul 14.46. Hasil tersebut menunjukkan bahwa UCS dan A* mampu menemukan solusi optimal berdasarkan jadwal perjalanan yang tersedia, sementara GBFS tidak menjamin optimalitas karena hanya mempertimbangkan nilai heuristik.

Dari sisi jumlah simpul yang diekspansi, UCS memerlukan 7 ekspansi, sedangkan GBFS dan A* hanya memerlukan 3 ekspansi. Hal ini menunjukkan bahwa penggunaan heuristik mampu mengurangi ruang pencarian secara signifikan. Pada kasus ini, A* berhasil mempertahankan kualitas solusi yang sama dengan UCS sekaligus mengurangi jumlah simpul yang perlu dieksplorasi.

Waktu eksekusi juga menunjukkan pola yang serupa. GBFS memiliki waktu eksekusi paling rendah, diikuti oleh A*, sedangkan UCS memerlukan waktu eksekusi terbesar. Meskipun demikian, selisih waktu eksekusi antar algoritma masih relatif kecil karena ukuran graf yang digunakan tidak terlalu besar. Oleh karena itu, perbedaan utama yang terlihat pada pengujian ini terletak pada jumlah simpul yang diekspansi dan kualitas solusi yang dihasilkan.

Secara keseluruhan, A* memberikan performa terbaik pada skenario ini karena mampu menghasilkan waktu kedatangan optimal seperti UCS, tetapi dengan jumlah ekspansi simpul dan waktu komputasi yang mendekati GBFS.

V. Kesimpulan

Dari hasil pengujian yang dilakukan, algoritma A* memberikan performa paling seimbang. Dengan mengombinasikan biaya aktual dan estimasi heuristik, A* mampu menghasilkan solusi optimal yang setara dengan UCS, namun dengan jumlah ekspansi simpul yang lebih sedikit dan waktu komputasi yang lebih rendah. Oleh karena itu, A* dapat dianggap sebagai algoritma yang paling efektif untuk permasalahan pencarian rute KRL berbasis jadwal pada penelitian ini.

Lampiran

Tautan Github:

https://github.com/rhenprojects/KRLRouteFinder

Ucapan Terima Kasih

Terima kasih saya panjatkan kepada Tuhan Yang Maha Esa karena atas rahmat dan kasih karunia-Nya, makalah ini dapat diselesaikan dengan maksimal. Ucapan terima kasih yang sebesar-besarnya saya sampaikan kepada dosen mata kuliah IF2211 Strategi Algoritma K-1, Bapak Prof. Dr. Ir. Rinaldi, M.T., yang telah membagikan ilmu serta memberikan banyak perspektif baru bagi saya di dunia Informatika. Tak lupa, saya mengucapkan terima kasih kepada keluarga dan teman-teman saya yang telah memberikan dukungan moral selama proses pengerjaan makalah ini. Saya berharap makalah ini dapat menjadi referensi yang berguna, baik bagi pelajar lain yang ingin memahami materi terkait, maupun bagi penulis sendiri di masa mendatang.

References

[1] D. Dewanta, “Jabodetabek Commuter Line Schedules,” Kaggle.com, 2024. https://www.kaggle.com/datasets/dennydewanta/jabodetabek-commuter-line-schedules (diakses 13 Juni 2026).

[2] “Commuter Line,” Jakarta.go.id, 2024. https://www.jakarta.go.id/commuter-line (diakses 14 Juni 2026).

[3] “Penentuan Rute (Route/Path Planning) - Bagian 1” [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/21-Route-Planning-(2026)-Bagian1.pdf (diakses 16 Juni 2026).

[4] “Penentuan Rute (Route/Path Planning) - Bagian 2” [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/22-Route-Planning-(2026)-Bagian2.pdf (diakses 16 Juni 2026).

[5] “Graf (Bagian 1)” https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf (diakses 19 Juni 2026).

[6] “Distance between Points on the Earth’s Surface.” Available: https://www.math.ksu.edu/~dbski/writings/haversine.pdf (diakses 17 Juni 2026).

[7] Y. Wang, Y. Yuan, Y. Ma, and G. Wang, “Time-Dependent Graphs: Definitions, Applications, and Algorithms,” Data Science and Engineering, vol. 4, no. 4, pp. 352–366, Sep. 2019, doi: https://doi.org/10.1007/s41019-019-00105-0. (diakses 19 Juni 2026).

[8] “OpenStreetMap, Overpass API And Python - Pybites,” Pybites, Apr. 17, 2024. https://pybit.es/articles/openstreetmaps-overpass-api-and-python/ (diakses 17 Juni 2026).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Rhenaldy Cahyadi Putra
13524039