Pencarian Kombinasi Rute Angkot Dengan Biaya Minimal Menggunakan Algoritma A*

Makalah IF2211 Strategi Algoritma

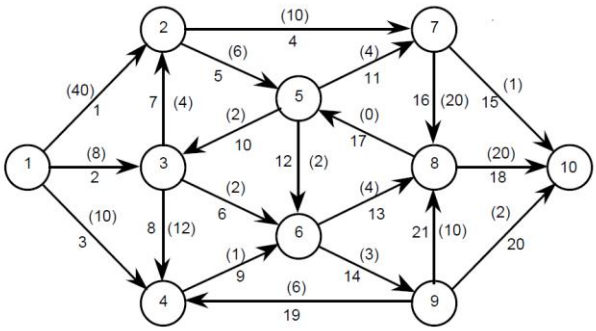
Daniel Ryan Levyson (*13516132*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
daniel93ryan@gmail.com

Abstrak*—Angkutan Kota atau yang biasa dikenal dengan angkot merupakan transportasi umum yang beroperasi hampir di setiap kota di Indonesia. Angkot beroperasi dengan rute yang sudah ditentukan dan tidak berubah-ubah. Selama masih di dalam jalur yang sesuai dengan rute, angkot bisa menaikan dan menurunkan penumpang sesuai dengan keinginan penumpang. Setelah sampai di tempat tujuan, penumpang akan membayar sesuai dengan jarak tempuh yang telah dilalui dari lokasi dimana penumpang naik. Meskipun sekarang ini sudah banyak beroperasi ojek online, namun angkot masih diandalkan oleh masyarakat untuk berpergian, oleh karena tarifnya yang murah. Namun ada permasalahan yang akan dihadapi oleh seseorang yang mengandalkan angkot dalam memenuhi kebutuhan mobilitasnya. Seseorang harus mengetahui rute-rute angkot yang ada di sekitarnya. Setelah mengetahui rute-rute yang ada, seseorang harus memilih kombinasi rute angkot yang memerlukan biaya seminimal mungkin untuk sampai ke tempat tujuannya. Selain itu, ada beberapa kombinasi yang mengharuskan orang tersebut untuk berjalan kaki untuk beralih ke angkot dengan rute yang berbeda dan biasanya seseorang ingin agar jarah tempuh yang diperlukan untuk berjalan kaki seminimal mungkin. Makalah ini akan membahas sebuah teknik algoritma yang dapat membantu pencarian kombinasi rute angkot yang meminimumkan biaya angkot serta jarak tempuh untuk berjalan kaki.

Kata Kunci—pencarian; rute; angkot; heuristik; graf

I. PENDAHULUAN

Di dalam teori graf, dikenal sebuah permasalahan untuk mencari jalur terpendek. Permasalahannya adalah menemukan jalur, yang berupa himpunan sisi, yang dapat menghubungkan dua simpul, dengan total bobot sisi yang diambil seminimal mungkin. Masalah ini sangat dekat pengaplikasiannya dalam kehidupan sehari-hari, yaitu apabila seseorang ingin pergi ke suatu tempat, jalur mana saja yang tepat untuk diambil untuk sampai ke tempat tujuan dengan jarak yang paling minimal. Setiap tempat tertentu dapat direpresentasikan oleh simpul dan akses jalan dari suatu tempat ke tempat lain dapat direpresentasikan oleh sisi yang bobotnya merupakan jarak. Beberapa pendekatan digunakan untuk menemukan cara yang menghasilkan hasil yang optimal, yaitu bobot yang paling minimal, serta memakan waktu sependek mungkin.



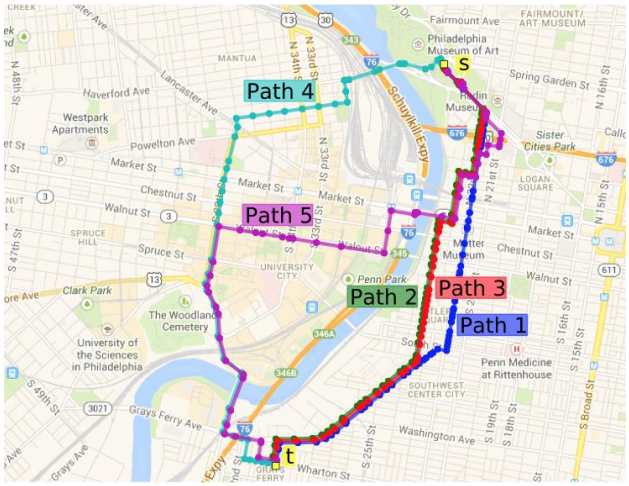
Gambar 1 Graf Berbobot
https://adhoop.files.wordpress.com/2012/03/shortest_path.png

Menyelesaikan permasalahan tersebut dengan *exhaustive search* dapat menghasilkan solusi yang optimal, namun dengan waktu yang eksponensial terhadap banyaknya simpul. Masalah ini dapat diselesaikan secara *greedy* dengan algoritma Djikstra dengan waktu yang kuadratik terhadap banyak simpul apabila graf direpresentasikan dengan matriks ketetanggaan. Algortima Djikstra mencari solusi dengan menelusuri setiap simpul dan memberi bobot kepada setiap simpul berdasarkan total bobot dari sisi yang harus dilalui dari simpul awal. Setiap penelusuran akan memperbarui bobot pada simpul apabila ditemukan jalur yang lebih kecil total bobotnya. Algoritma lain menggunakan sebuah cara untuk menebak simpul mana yang selanjutnya akan dikunjungi karena memiliki kemungkinan yang besar dibandingkan simpul lainnya untuk menghasilkan solusi yang optimal. Algoritma itu adalah A*. Tidak seperti algoritma Djikstra yang menelusuri setiap simpul secara "buta", algoritma A* menelusuri simpul-simpul yang berkemungkinan besar menghasilkan hasil yang optimal dengan cara menebak. Semakin akurat tebakan yang dibuat, algoritma A* akan jauh lebih cepat daripada algoritma Djikstra. Namun apabila tebakan yang dibuat tidak cukup informatif untuk menghasilkan hasil yang optimal, maka kompleksitas algoritma ini akan sama dengan algoritma Djikstra.

w(u,v)	a	b	c	d	d	f
a	0	4	2	INF	INF	INF
b	4	0	1	5	INF	INF
c	2	1	0	8	10	INF
d	INF	5	8	0	2	6
e	INF	INF	10	2	0	3
f	INF	INF	INF	6	3	0

Gambar 2 Contoh Tabel Djikstra
https://www.codewithc.com/wp-content/uploads/2014/07/dijkstra-algorithm-table.png

Masalah pencarian kombinasi rute angkot dengan biaya minimum merupakan modifikasi dari masalah pencarian jalur terpendek. Rute pada dasarnya merupakan sekumpulan simpul. Meskipun begitu, perpindahan dari simpul pada suatu rute ke simpul pada rute lain harus diperlakukan berbeda dengan perpindahan dari simpul ke simpul lain di rute yang sama. Dalam pencarian kombinasi rute angkot, sesungguhnya graf yang terbentuk bisa dipandang sebagai graf lengkap dengan banyak sekali simpul tergantung pada banyaknya rute angkot yang ada, karena angkot bisa berhenti dimana saja, bukan hanya di tempat tertentu. Oleh karena itu diperlukan cara tertentu untuk mengatasi banyak simpul yang ada. Fakta-fakta yang telah diuraikan sebelumnya tentu saja mempersulit masalah ini, namun sesungguhnya algoritma yang mampu memecahkan permasalahan pencarian jalur terpendek dapat pula digunakan untuk memecahkan permasalahan pencarian kombiasi rute angkot, tentunya dengan beberapa modifkasi.



Gambar 3 Rute
https://nextcity.org/daily/entry/pedestrian-open-data-safest-shortest-walk-map

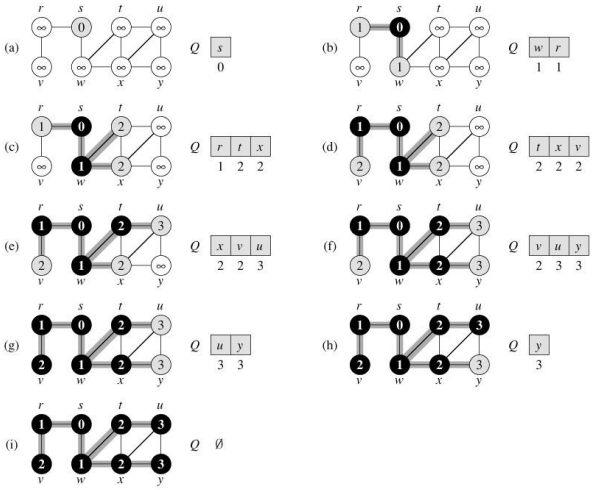
II. DASAR TEORI

A. *Algoritma Transversal Graf*

Algoritma trasnsversal graf merupakan sebuah teknik untuk menelusuri suatu graf dengan cara tertentu. Ada dua cara paling sederhana untuk menelusuri suatu graf, yaitu:

1. Breadth-First Search

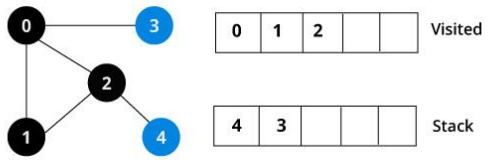
Algoritma *Breadth-First Search* atau biasa disebut BFS merupakan algoritma untuk menelusuri setiap simpul pada graf dengan cara mengunjungi semua tetangga dari simpul yang sedang dikunjungi terlebih dahulu, lalu kemudian baru mengunjungi semua tetangga dari tetangga simpul sebelumnya, dan begitu seterusnya. Untuk melakukan hal itu, struktur data *queue* digunakan untuk menyimpan urutan simpul yang akan dikunjungi selanjutnya. Setiap kali suatu simpul dikunjungi, semua tetangga dari simpul tersebut akan dimasukan ke antrian paling belakang pada *queue*. Kemudian simpul selanjutnya yang akan dikunjungi adalah simpul yang berada di antrian paling depan pada *queue*.



Gambar 4 Ilustrasi BFS
http://www.euroinformatica.ro/documentation/programming/!!!Algorithms_CORMEN!!!/images/fig555_01_0.jpg

2. Depth-First Search

Algoritma *Depth-First Searcg* atau biasa disebut DFS merupakan algoritma untuk menelusuri setiap simpul pada graf dengan cara mengunjungi dahulu salah satu tetangga pada suatu simpul, sampai tetangga tersebut tidak memiliki simpul lagi yang bisa dikunjungi. Untuk melakukan hal itu, struktur data *stack* digunakan untuk menyimpan tumpukan simpul yang akan dikunjungi selanjutnya. Setiap kali suatu simpul dikunjungi, semua tetangga dari simpul tersebut akan dimasukan ke tumpukan paling atas pada *stack*. Kemudian simpul selanjutnya yang akan dikunjungi adalah simpul yang berada di paling atas pada *stack*.



Gambar 5 Ilustrasi DFS
https://www.programiz.com/dsa/graph-dfs

B. Algoritma Branch and Bound

Algoritma *branch* and *bound* merupakan salah satu algoritma untuk menelusir graf dengan cara yang berbeda dengan baik BFS maupun DFS. Algoritma ini akan mengeksplorasi cabang demi cabang dimulai dari satu simpul akar. Setiap kali suatu cabang diekplorasi, suatu simpul akan terlihat. Setiap simpul yang belum dikunjungi akan dimasukan ke dalam himpunan kandidat. Simpul yang selanjutnya akan dikunjungi akan dipilih dari himpunan kandidat.

Setiap kandidat terasosiasi pada suatu nilai ongkos yang merupakan nilai taksiran termurah dari simpul status i ke simpul status solusi. Pada umumnya, karena letak simpul solusi tidak diketahui, nilai ongkos tersebut dihitung dengan fungsi heuristik:

c(i) = f(i) + g(i)

Dengan,

c(i) = ongkos untuk sampai ke simpul i

f(i) = ongkos mencapai simpul i dari simpul akar

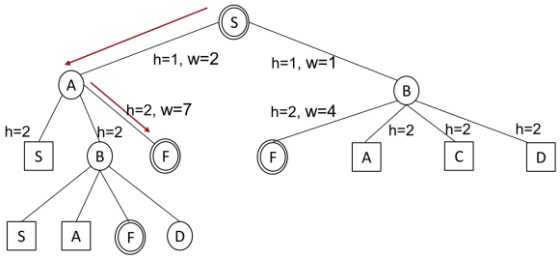
g(i) = ongkos mencapai simpul tujuan dari simpul i

Apabila algoritma transversal graf melakukan pencarian secara "buta", tanpa informasi apapun, algoritma *branch and bound* melakukan penelusuran dengan informasi taksiran ongkos, sehingga algoritma transversal graf seperti DFS dan BFS disebut juga *uninformed search*, sedangkan algoritma *branch and bound* disebut *informed search*.

Algoritma *branch and bound* memiliki pengembangan menjadi algoritma lain, seperti:

1. Greedy Best-First Search

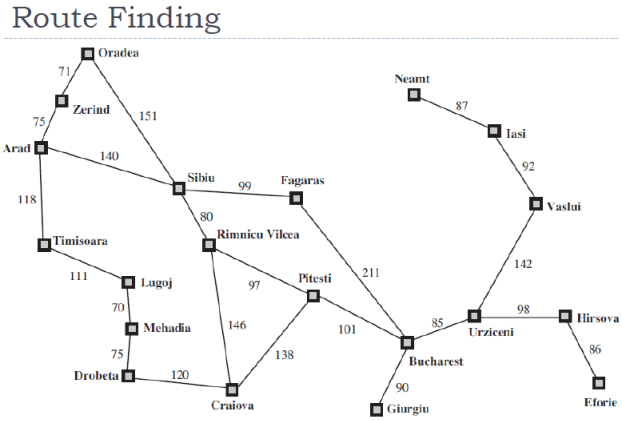
   Algoritma ini menggunakan fungsi evaluasi f(n) untuk setiap simpulnya. Fungsi f(n) akan mengestimasi ongkos yang diperlukan untuk sampai ke simpul tujuan dari simpul n. Dalam kasus pencarian jalur terpendek, f(n) merupakan panjang dari garis luruk yang ditarik dari simpul n ke simpul tujuan. Meskipun cukup cepat menemukan solusi, namun kelemahan algoritma ini adalah solusinya tidak optimal.



Gambar 6 Ilustrasi Algoritma Greed Best-First Search
https://slidewiki.org/deck/1105-1/slide/9175-2/1160-1:10;9175-2:4/view

2. A* Search

   Ide utama dari algoritma ini adalah mencoba menghindari untuk mengeksplorasi simpul yang diperkirakan berbiaya mahal. Untuk melakukan hal itu, algoritma ini menggunakan ongkos yang dihitung seperti halnya algoritma *branch and bound*. Algoritma ini akan menemukan solusi yang optimal dan dengan kompleksitas eksponensial pada kasus terburuk.



Gambar 7 Ilustrasi Algoritma A*
https://piotechno.wordpress.com/2014/01/23/uniform-cost-and-a-search-algorithm/

III. Implementasi A* Untuk Pencarian Kombinasi Rute Angkot

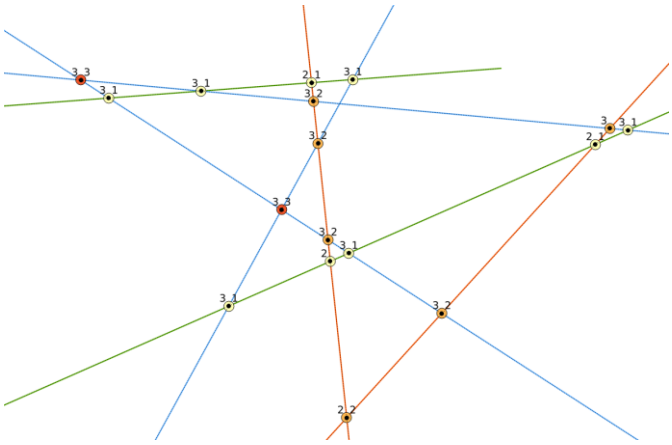
A. Rancangan Solusi

Sebuah rute angkot bisa dipandang sebagai suatu graf, sehingga permasalahan pencarian ini bisa dipandang sebagai pencarian biaya minimal yang melibatkan dua buah titik dan sekumpulan graf. Dua buah titik yang merupakan titik awal dan titik tujuan bisa saja dipandang sebagai graf yang hanya memiliki satu simpul, sehingga permasalahan ini menjadi pencarian biaya minimal yang melibatkan sekumpulan graf saja.

Untuk mencari biaya yang minimal, perlu diketahui bagaimana menghitung biaya dalam permasalahan ini. Ada dua jenis biaya yang ingin dipertimbangkan, yaitu biaya angkot dan biaya jalan kaki, kedua-duanya ingin diminimalkan. Biaya angkot dapat dihitung dari seberapa jauh perjalanan dari suatu

titik di dalam suatu graf ke titik lain di graf tersebut. Biaya jalan kaki dapat dihitung dari jarak terdekat antara dua graf.

Jarak antara graf A dengan graf B dapat didefinisikan dengan panjang dari garis terpendek yang ditarik dari titik pada graf A ke titik pada graf B. Sehingga apabila ada sisi pada graf A yang berpotongan dengan sisi pada graf B, jarak antara graf A dengan graf B adalah nol. Penghitungan jarak antara graf A dengan graf B dapat dilakukan dengan memeriksa setiap sisi pada graf A terhadap graf B dan diambil dua garis yang berjarak paling kecil.



Gambar 8 Ilustrasi Jarak Antar Graf
https://gis.stackexchange.com/questions/86308/how-to-identify-line-intersection-in-qgis-when-i-have-more-than-2-lines

Apabila setiap graf dipandang sebagai sebuah simpul pada kasus ini, maka sekumpulan graf berarti sekumpulan simpul. Sekumpulan simpul ini dapat dihubungkan dengan sebuah sisi ke masing-masing simpul lainnya, sehingga akan menjadi graf lengkap. Dalam kasus bagaimana pun, sekumpulan graf yang dipandang sebagai sekumpulan simpul akan membentuk graf lengkap, karena jarak antar simpul pada graf lengkap ini merupakan biaya berjalan kaki.

Kecenderungan tiap orang untuk memilih seberapa jauh jarak tempuh yang harus dilalui seseorang sehingga ia lebih memilih untuk berjalan kaki daripada naik angkot akan berbeda-beda. Kecenderugan ini akan mempengaruhi hasil pada pencarian kombinasi angkot. Untuk melibatkan nilai kecenderungan ini, ongkos berjalan kaki dapat dihitung dengan,

$c = d \times fw$

yang dalam hal ini,

c = ongkos berjalan kaki

d = jarak tempuh jalan kaki

fw = faktor kecenderungan untuk berjalan kaki

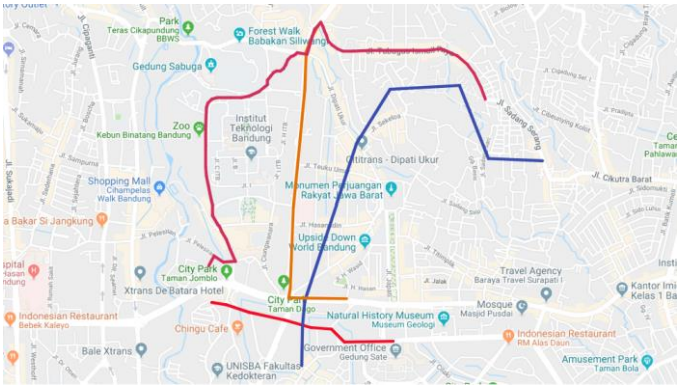
Semakin besar fw, maka berjalan kaki akan semakin cenderung untuk dihindari. Untuk menghitung ongkos naik angkot juga digunakan cara menghitung yang sama, namun *fw* digantikan dengan *ft*, yaitu faktor kecenderungan untuk menggunakan angkot.

Setiap orang tentu memiliki preferensi yang berbeda-beda dalam menentukan nilai *fw* dan *ft*. Namun dalam contoh yang akan digunakan pada makalah ini digunakan nilai *fw* yang sama dengan lima kali nilai *ft*, sehingga hasil yang diperoleh adalah hasil yang berdasar pada kecenderungan memilih naik angkot lima kali lebih mungkin untuk dipilih daripada berjalan kaki.

Setelah diketahui bagaimana menghitung biaya, fungsi heuristik dapat ditentukan. Fungsi heuristik harus bersifat optimistik, artinya nilai prediksi tidak boleh lebih besar daripada nilai aslinya. Dengan aturan tersebut, biaya naik angkot dapat digunakan untuk menentukan nilai heuristik, karena dengan adanya faktor kecenderungan, biaya berjalan kaki tidak akan menghasilkan nilai heuristik yang optimis. Dengan begitu, nilai heuristik dapat ditentukan dengan mengalikan panjang garis lurus yang ditarik dari simpul awal pada suatu graf ke titik tujuan dengan faktor kecenderungan untuk naik angkot.

B. Amalisis Hasil

Digunakan contoh empat rute angkot di sekitar Institut Teknologi Bandung, seperti digambarkan oleh garis-garis berikut.



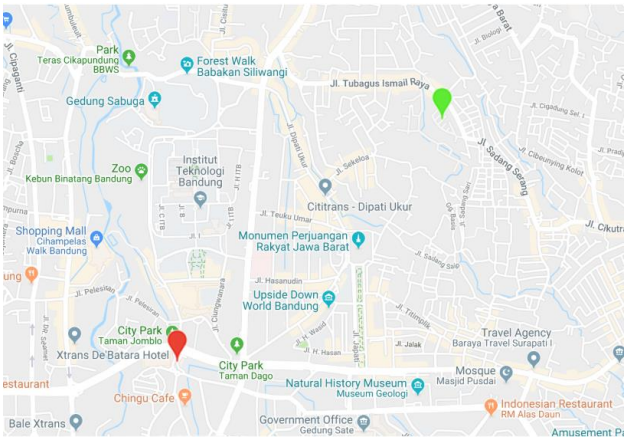
Gambar 9 RuteAngkot

1. Kasus Satu

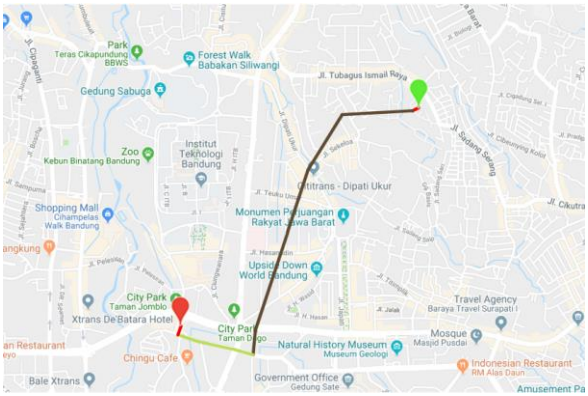
Dipilih dua titik, yang pertama terletak di daerah Tubagus Ismail, dan yang kedua di Jalan Dayang Sumbi yang akan dicari rute angkotnya.

Di atas adalah rute yang dihasilkan oleh algoritma. Jalur yang dipilih adalah jalur berwarna merah di bagian atas pada Gambar 7.

2. Kasus Dua

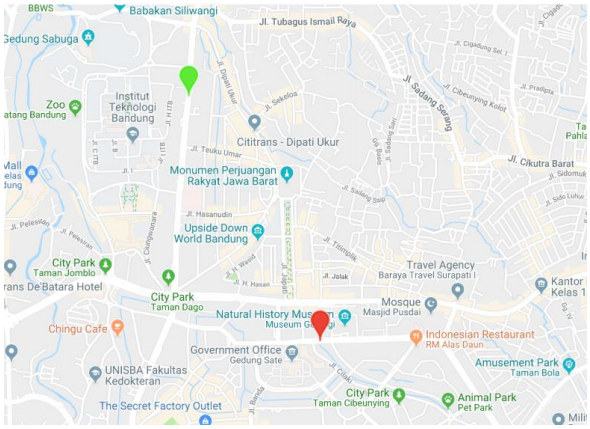


Dipilih dua titik, yang pertama terletak di daerah Tubagus Ismail, yang kedua terletak di daerah dekat Rektorat ITB.

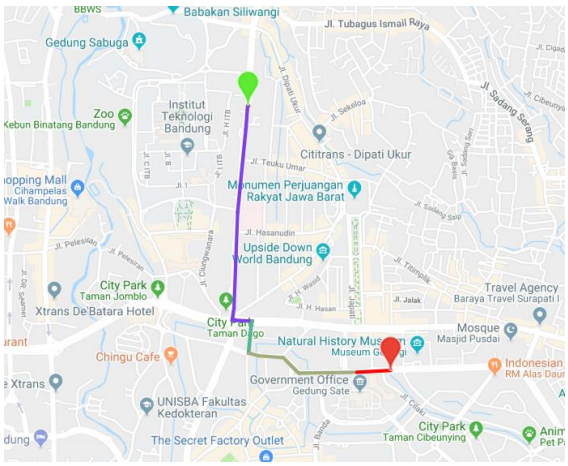


Di atas adalah rute yang dihasilkan oleh algoritma. Jalur yang dipilih adalah jalur berwarna biru dan merah pada Gambar 7.

3. Kasus Tiga



Dipilih dua titik, yang pertama terletak di dekat Rumah Sakit Santo Borromeus, yang kedua terletak di daerah dekat Taman Lansia.



Di atas adalah rute yang dihasilkan oleh algoritma. Jalur yang dipilih adalah jalur berwarna oranye, biru dan merah pada Gambar 7.

IV. Simpulan

Masalah pencarian kombinasi rute angkot memang mirip dengan masalah pencarian jalur terpendek, namun dalam beberapa aspek sangat berbeda. Algoritma A* dapat digunakan untuk memecahkan masalah pencarian rute angkot dengan biaya minimal dengan efisien. Berbeda dari masalah pencarian jalur terpendek, masalah pencarian rute angkot akan sulit diselesaikan dengan algoritma Djikstra, karena jarak tempuh antar titik pada rute, yaitu titik naik ankot dan titik turun angkot, akan berubah-ubah tergantung bagaimana kombinasi rute yang diambil.

V. Saran

Penentuan jarak antar graf bisa dioptimisasi lebih lanjut sehingga hasil yang didapatkan lebih baik. Solusi yang dirancang pada makalah ini memiliki kelemahan apabila ada dua rute yang berpotongan di titik tertentu yang cukup jauh sedemikian sehingga biaya yang dihasilkan menjadi besar, padahal apabila dikombinasikan dengan berjalan kaki dari titik

tertentu dari rute satu ke rute lain ada kemungkinan dapat mengurangi biaya.

Melihat kurangnya informasi untuk mengetahui rute-rute angkot di daerah tertentu, sehingga sulit untuk merencanakan perjalanan yang mengandalkan angkutan kota, aplikasi untuk membantu masyarakat menemukan kombinasi rute angkot pasti akan sangat berguna. Sebaiknya aplikasi seperti itu dikembangkan untuk memenuhi kebutuhan mobilitas masyarakat. Apalagi dengan adanya aplikasi seperti itu, masyarakat dapat lebih terdorong untuk menggunakan transportasi umum dariapda kendaraan pribadi.

UCAPAN TERIMA KASIH

Saya mengucapkan syukur kepada Tuhan Yang Maha Esa atas izin-Nya sama boleh menyusun makalah ini dengan baik dan tepat pada waktunya. Saya ingin mengucapkan terima kasih kepada dosen-dosen mata kuliah Strategi Algoritma Teknik Informatika ITB yang telah menyusun materi-materi kuliah dan mengajarkan saya banyak hal tentang mendesain algoritma yang efektif dan efisien. Saya ingin mengucapkan terima kasih juga kepada dosen yang mengajar kelas saya K3, yaitu Bapak Rinaldi Munir, yang telah memberi banyak ilmu, motivasi, serta inspirasi.

REFERENSI

[1] Munir, Rinaldi. 2018. Diktat Kuliah IF211 Strategi Algoritma. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[2] Slide kuliah IF2211 Strategi Algoritma, Institut Teknologi Bandung

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, 14 Mei 2018

Daniel Ryan Levyson dan 13516132