Pembuatan Aplikasi Pencarian Jalur Angkutan Umum Memanfaatkan Algoritma Dijkstra

Vincent Sebastian The/13513057
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
vincent.s.the@gmail.com

Abstrak-Angkutan umum bisa dikatakan adalah moda transportasi utama bagi mayoritas masyarakat Indonesia. Banyaknya moda angkutan umum seperti mikrolet, kereta, transjakarta (pada kota Jakarta) membuat banyaknya pilihan transportasi dari satu tempat ke tempat lain. Banyaknya rute angkutan umum di sisi lain juga menimbulkan kebingungan, yaitu kebingungan mencari angkutan umum dengan rute yang tepat untuk mencapai tujuan masing-masing. Hal ini terutama dirasakan oleh orang-orang yang bukan berada di kota-nya, misalnya wisatawan dari Jakarta yang berkunjung ke Bandung akan kesulitan berpergian dengan angkutan umum karena tidak mengetahui rute-rute angkutan umum pada kota tersebut. Hal ini juga seringkali dirasakan oleh orang-orang yang sudah tinggal di kota-nya namun tiap hari-nya hanya melakukan perjalanan dari tempat tinggal ke tempat kerja. Selain masalah tidak tahu rute-rute angkutan umum, tidak dapat diprediksi-nya waktu tempuh menggunakan angkutan umum juga membuat banyak orang menggunakan kendaraan pribadi. Permasalahan ini yang penulis akan coba untuk pecahkan. Penulis akan menuliskan ide dan konsep aplikasi yang mampu memberikan rute terbaik dan tercepat untuk berpindah dari satu tempat ke tempat lainnya dalam waktu paling efisien beserta dengan prediksi waktu perjalanan dalam tulisan ini. Dalam konsep aplikasi ini, permasalahan akan dipecahkan dengan algoritma Greedy Dijkstra dengan memperhatikan segala kondisi seperti kecepatan setiap moda transportasi, kemacetan jalan, dll. Pada pemecahan masalah ini, tempat dan rute perjalanan akan dimodelkan menjadi graf berbobot dengan bobot sisi disesuaikan dengan aspek-aspek yang dijelaskan di atas.

***Keywords*—graf, *upagraf*, Dijkstra, Path, Greedy**

I. PENDAHULUAN

Setiap angkutan umum beroperasi pada rute-rute sesuai trayeknya. Pada tulisan ini, akan ditinjau dua moda transportasi umum, yaitu mikrolet (angkot) dan bus Transjakarta (berasumsi pemodelan yang dibuat mengambil tempat di Jakarta), pada pemodelan ini, berjalan kaki juga merupakan suatu cara transportasi. Angkot dapat berhenti dimana saja sepanjang jalan trayeknya. Sedangkan bus transjakarta hanya dapat berhenti pada halte-halte transjakarta.

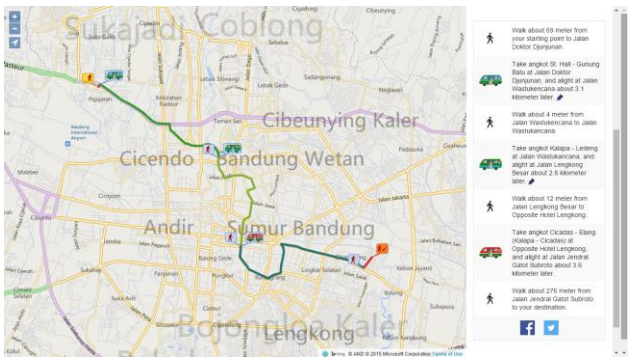
Pada aplikasi ini, akan dilakukan pemodelan graf. Tempat-tempat akan dimodelkan menjadi simpul, sedangkan jalan-jalan akan dimodelkan menjadi sisi. Pemodelan terhadap jalur mikrolet dan busway akan dilakukan dengan mendaftarkan simpul dan sisi mana saja yang dilewati oleh mikrolet dan busway tersebut. Tahap selanjutnya dari proses ini adalah memperkirakan kecepatan dari setiap moda transportasi tersebut, misalkan karena mikrolet seringkali *ngetem*, maka kecepatan mikrolet bisa dihitung lebih rendah daripada kecepatan transjakarta. Sedangkan berpindah dengan jalan kaki adalah moda yang paling lambat namun tidak terpengaruhi oleh kemacetan. Akhirnya, proses menghitung tingkat kemacetan pada periode-periode tertentu juga dilakukan untuk membuat hasil rute dan perkiraan waktu tempuh menjadi lebih akurat.

Dalam menghitung rute optimal menggunakan angkutan umum, aplikasi ini menggunakan algoritma greedy Dijkstra (algoritma akan dijelaskan pada bab landasan teori). Pemodelan seperti yang sudah dijelaskan sebelumnya, simpul adalah lokasi-lokasi pada peta, sedangkan simpul-simpul adalah jalan. Bobot setiap simpul akan ditentukan oleh beberapa aspek. Aspek yang menentukan bobot setiap sisi adalah jarak antar simpul, jenis moda transportasi umum, dan tingkat kemacetan jalan. Algoritma akan memulai dari titik awal dan akan terus mengunjungi simpul-simpul lainnya mulai dengan simpul yang paling cepat bisa dikunjungi sampai tiba di sumpul tujuan.

Kecepatan moda transportasi dihitung dari rata-rata kecepatan setiap moda. Pada kasus ini, yang paling cepat adalah bus Transjakarta, lalu mikrolet, dan yang paling lambat adalah berjalan kaki. Tingkat kemacetan dapat diukur secara periodik bergantung kepada waktu jam dan hari. Tingkat kemacetan tidak akan mempengaruhi kecepatan saat berjalan kaki.

Ada website yang sudah mengadaptasi sistem semacam ini, website tersebut adalah kiri.travel. Website yang sudah cukup lama didirikan oleh VIROES INC. tersebut menyediakan info serupa dengan aplikasi yang penulis ingin rancang. Website ini juga bertujuan untuk membuat lebih banyak orang beralih menggunakan transportasi umum sehingga kemacetan dapat berkurang dan polusi juga dapat berkurang. Namun website tersebut hanya menyediakan layanan untuk beberapa kota saja, dan data informasi rute angkutan umum juga tidak lengkap, bahkan untuk kota Jakarta. Website tersebut juga tidak memperhitungkan tingkat kemacetan sehingga

seringkali prediksi waktu berbeda dari waktu tempuh yang sebenarnya.



Gambar 1: Tampilan website kiri.travel saat menunjukkan rute angkutan umum (http://kiri.travel)

Pada implementasi aplikasi ini, nantinya peta yang akan digunakan akan menggunakan *Google Maps API*, penggunaan kakas ini, selain mudah digunakan dan banyak fitur yang bisa digunakan, juga agar pembuat aplikasi dapat lebih fokus mengembangkan algoritma pencarian rute yang lebih efisien, tidak menghabiskan waktu memikirkan implementasi map.

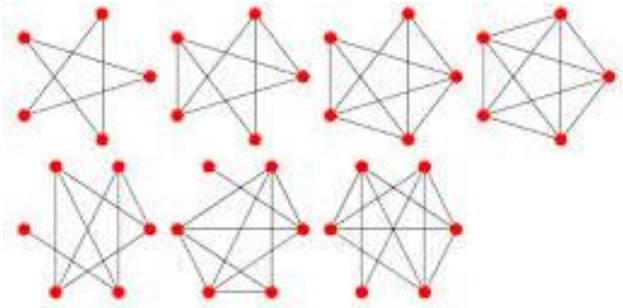
II. LANDASAN TEORI

A. Teori Graf

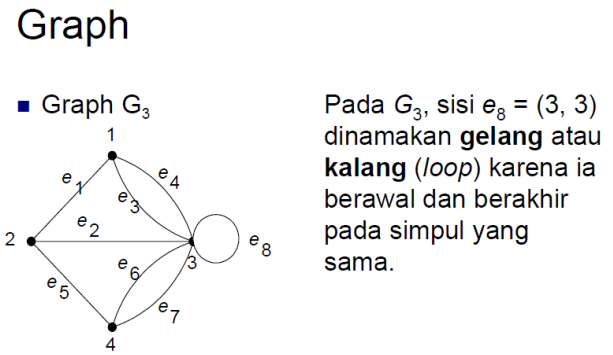
Graf adalah struktur data yang merepresentasikan data yang saling berhubungan. Benda atau objek dalam suatu graf digambarkan sebagai simpul (*vertex*). Sedangkan hubungan antara simpul disebut busur atau sisi (*edge*).

Dalam matematika diskrit, graf dinotasikan sebagai berikut G = (V,E), dengan V adalah himpunan tidak kosong dari simpul (vertices) = { V1, V2, … , Vn) dalam graf dan E adalah himpunan sisi (edges) yang menghubungkan dua simpul dalam graf tersebut = { e1, e2, … , en }.

Terdapat beberapa jenis graf berdasarkan ada tidaknya sisi ganda dan gelang, yaitu graf sederhana (tidak mengandung sisi ganda dan sisi gelang) dan graf tak sederhana(memiliki sisi ganda atau sisi gelang). Sisi ganda adalah adanya lebih dari satu sisi yang menghubungkan dua simpul yang sama, dan sisi gelang adalah sebuah sisi yang menghubungkan satu simpul dengan dirinya sendiri (disebut cincin karena biasanya berbentuk cincin). Pada tulisan ini, semua graf yang terbentuk akan diasumsikan sebagai graf sederhana. Tidak ada sisi gelang karena tidak ada bus yang pergi dari satu halt eke halte tersebut lagi. Sedangkan bila ada sisi ganda, sisi tersebut akan dibuat menjadi satu sisi yang memiliki berat yang sama (dibahas berikutnya).



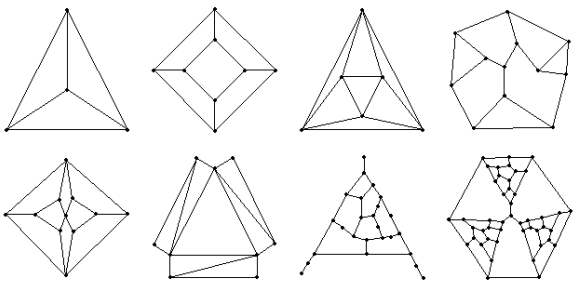
Gambar 2: Graf sederhana (http://mathworld.wolfram.com/)



Gambar 3 : Graf tak sederhana (darkrabbitblog.blogspot.com)

Sedangkan berdasarkan orientasinya graf terdiri atas 2 yaitu graf berarah dan graf dan graf tidak berarah. Graf berarah berarti sisi-sisi memiliki orientasi (dari simpul satu ke simpul lain dan tidak sebaliknya). Sedangkan graf tidak berarah tidak memiliki orientasi (dari satu simpul melewati satu sisi dapat kembali melewati sisi yang sama lagi). Pada pemodelan tulisan ini, graf berarah diperlukan untuk memodelkan jalan-jalan yang hanya dapat dilalui satu arah.

Karena pada permasalahan ini peta akan dimodelkan menjadi graf, maka graf yang terbentuk adalah graf planar (atau sering juga disebut *graph map*). Graf planar adalah graf yang bila semua sisi dan simpulnya digambar, akan ada konfigurasi sedemikian rupa sehingga tidak ada dua sisi yang tumpang tindih pada gambar tersebut. Graf planar ini memiliki beberapa ciri khusus contohnya memiliki derajat warna 4.



Gambar 4. Contoh graf planar. (http://www.math.rutgers.edu/)

Beberapa terminology graf antara lain :

Berikut adalah karakteristik-karakteristik graf yang akan digunakan dalam menyelesaikan permasalahan pada tulisan ini

1. Ketetanggaan (adjacent)

Dua buah simpul pada graf dikatakan bertetangga apabila keduanya terhubung langsung oleh sebuah sisi. Dengan kata lain ada sisi dimana sisi tersebut menghubungkan simpul S1 dan siimpul S2. Prinsip ketetanggan pada pemodelan tulisan ini terlihat jelas saat pemodelan lintasan bus transjakarta karena transjakarta hanya dapat berhenti pada halte, sehingga sisi-sisi hanya menghubungkan simpul-simpul yang mewakili halte.

2. Bersisian (Incidency)

Untuk sembarang sisi e = (S1, S2), e dikatakan bersisian dengan S1 dan S2 karena e menghubungkan simpul S1 dan S2.

3. Lintasan (Path)

Lintasan (path) adalah jalur dari satu simpul ke simpul lainnya, pada gambar 3, lintasan dari simpul 2 ke simpul 3 dapat berupa melewti sisi e1 lalu ke sisi e4, dengan begitu akan melewati simpul 2 – 1 – 3.

4. Graf Berbobot (Weighted Graph)

Graf berbobot adalah graf yang sisinya mempunyai nilai, bobot tersebut bisa mewakili panjang sisi atau dapat juga mewakili maksud lain dari graf tersebut. Pada pemodelan pada tulisan ini bobot sisi adalah waktu tempuh yang diperlukan untuk mencapai satu sisi ke sisi lainnya. Bobot ditentukan oleh beberapa faktor yaitu jenis moda transportasi, jarak, dan tingkat kemacetan.

5..Upagraf

Misalkan $G = (V, E)$ adalah sebuah graf. $G_1 = (V_1, E_1)$ adalah upagraf (*subgraph*) dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$. Komplemen dari upagraf G_1 terhadap graf G adalah graf $G_2 = (V_2, E_2)$ sedemikian sehingga $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota-anggota E_2 bersisian dengannya.

B. Teori Greedy

Greedy adalah sebuah algoritma yang mengikuti aturan pemecahan masalah heuristik. Algoritma greedy pada setiap langkanya membuat solusi optimal lokal dengan harapan dapat menemukan solusi optimal untuk masalah keseluruhan atau masalah global. Dalam banyak permasalahan, algoritma greedy tidak akan menemukan solusi optimal, namun banyak algoritma greedy yang dipakai untuk mencari solusi masalah-masalah kompleks dengan cara mencarai langkah lokal optimal. Solusi yang dihasilkan bukanlah solusi optimal global, namun dengan algoritma ini, solusi yang cukup optimal bisa didapatkan dengan kompleksitas waktu yang kecil. Strategi pemecahan solusi semcam ini banyak dipakai untuk memecahkan masalah heuristik.

Secara keseluruhan, algoritma greedy memiliki lima bagian

- himpunan kandidat: himpunan dimana solusi berada.
- fungsi seleksi: fungsi yang memilih kandidat terbaik untuk ditambahkan ke solusi.
- fungsi kemungkinan: fungsi yang memperkirakan apakah kandidat dapat berkontribusi ke solusi.
- fungsi objektif: fungsi yang memberikan nolai ke suatu solusi.
- fungsi solusi: fungsi yang mengindikasikan algoritma telah menemukan solusi lengkap.

1. Algoritma Dijkstra

Algoritma Dijkstra merupakan algoritma yang digunakan untuk mencari lintasan terpendek (*shortest path*) pada sebuah graf berbobot yang ditemukan oleh ilmuwan. komputer Edsger W. Dijkstra. Algoritma ini mirip dengan algoritma BFS (Breadth First Search).

Penulis memilih memakai algoritma dijkstra untuk tulisan ini karena algoritma dijkstra merupakan algoritma yang paling efisien dalam mencari lintasan terpendek suatu graf. Kompleksitas algoritma Dijkstra adalah $O(V^2)$ bila menggunakan larik biasa, dimana V adalah jumlah simpul dalam graf, atau $O(E \log V)$ bila menggunakan struktur data *heap* (*priority queue)* dimana V adalah jumlah simpul dalam graf dan E adalah jumlah sisi dalam graf.

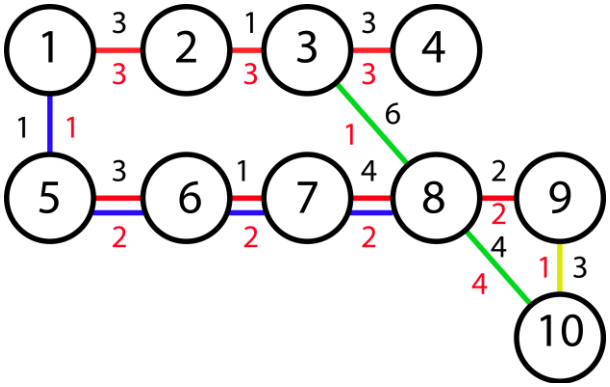
Algoritma Dijkstra ini bekerja dengan mengunjungi simpul-simpul berurutan dari simpul yang paling dekat secara bobot dari simpul awal. Algoritma Dijkstra dapat disimpulkan sebagai berikut

1. Berikan jarak tak terhingga untuk semua simpul pada graf. Kecuali untuk simpul awal, berikan jarak 0.
2. Tandakan simpul awal sebagai simpul yang sedang dikunjungi, tandai simpul lain sebagai belum dikunjungi.
3. Untuk semua simpul yang ber-ketetanggaan dengan simpul yang sedang dikunjungi, hitung jarak tentatif-nya yaitu jarak simpul yang sedang dikunjungi ditambah dengan bobot sisi yang menghubungi kedua simpul. Bandingkan dengan jarak tentatif sebelumnya. Bila lebih kecil, ganti jarak tentatif dengan nilai yang baru.
4. Saat sudah selesai mengecek semua tetangga simpul yang sedang dikunjungi, tandai simpul sekarang menjadi simpul yang sudah dikunjungi. Simpul yang sudah dikunjungi tidak akan pernah dicek kembali.
5. Jika simpul tujuan sudah dikunjungi, maka berhenti. Ini menandakan algoritma sudah menemukan jarak terpendek menuju simpul tujuan.
6. Kembali ke langkah 3 dengan mengambil simpul dengan jarak tentatif terkecil sebagai simpul yang sedang dikunjungi.

III. APLIKASI ALGORITMA DIJKSTRA UNTUK MENCARI RUTE OPTIMAL ANGKUTAN UMUM

A. PEMODELAN PETA MENJADI GRAF BERBOBOT

Supaya tulisan ini lebih berfokus pada pembahasan algoritma penentuan rute optimal, maka jalur angkutan umum yang berada pada kota beserta angkutan umum yang tersedia akan dimodelkan menjadi sebuah graf. Pada tulisan ini pememcahan masalah akan menggunakan model graf seperti di bawah.



Gambar 5. Pemodelan graf buatan untuk tujuan tulisan ini.

Untuk gambar di atas diberikan 10 buah simpul yang mewakilkan tempat-tempat. Angka hitam pada sisi melambangkan jarak simpul yang bersisian dengan sisi dalam satuan kilometer. Angka merah pada sisi melambangkan faktor kemacetan, angka 2 berarti sisi tersebut harus dilalui dengan waktu dua kali lebih lama dibanding waktu normal.

Warna-warna pada sisi melabangkan transportasi yang menghubungkan tempat-tempat yang bersisian dengan sisi tersebut. Misalkan warna merah mengartikan busway sedangkan warna lain melambangkan jalur-jalur mikrolet sesuai dengan warnanya. Berikut daftar rute angkutan umum pada graf di atas

- Busway koridor 1: 1 – 2 – 3 – 4
- Busway koridor 2: 5 – 6 – 7 – 8 – 9
- Mikrolet jurusan M1: 1 – 5 – 6 – 7 – 8 – 9
- Mikrolet jurusan M2: 3 – 8 – 10
- Mikrolet jurusan M3: 9 – 10

Sedangkan untuk kecepatan setiap moda transportasi, digunakan acuan sebagai berikut:

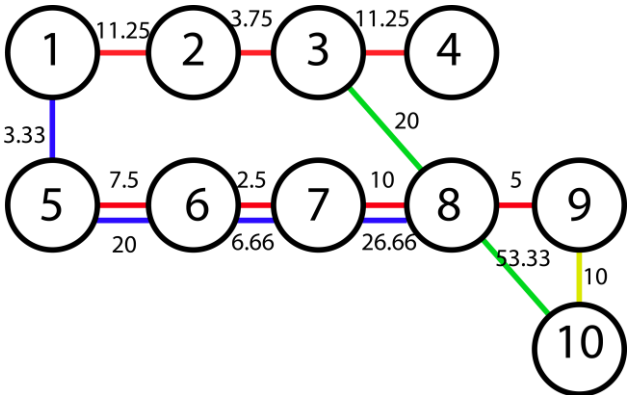
- Busway: kecepatan 0.8 km per menit
- Mikrolet: kecepatan 0,3 km per menit

Berdasarkan informasi-informasi tersebut, kita dapat menentukan bobot-bobot sebenarnya atau waktu tempuh dalam menit dari setiap sisi pada graf. Bobot sebenarnya suatu sisi pada graf dapat diturunkan dari persamaan

$$W = J / V \times K$$

Dengan W adalah bobot sebenarnya dari suatu sisi yaitu waktu tempuh untuk berpindah dari sisi yang bersisian, J adalah jarak dua simpul yang bersisian, V adalah kecepatan moda transportasi, dan K adalah faktor kemacetan sisi tersebut.

Dengan menerapkan persamaan tersebut pada graf buatan tulisan ini, maka akan didapat graf sebagai berikut



Gambar 6. Graf yang sudah diberi bobot waktu perjalanan antar simpul bersisian

Pada gambar graf di atas sisi yang memiliki dua bobot menandakan ada dua jenis moda transportasi untuk berpindah antar simpul tersebut, dalam hal ini busway koridor 2 dan mikrolet jurusan M1.

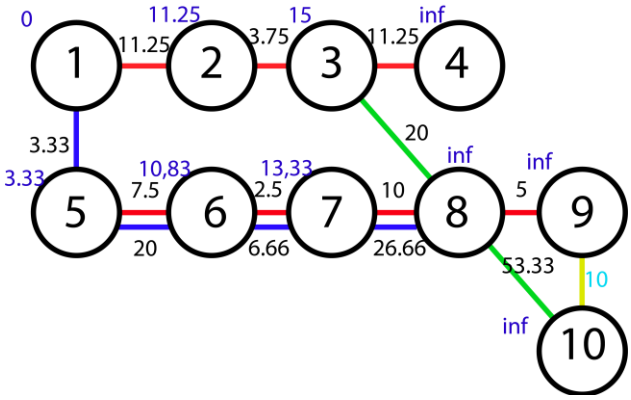
B. PENGAPLIKASIAN ALGORITMA DIJKSTRA DALAM MENCARI JALUR TERCEPAT

Pada subbab ini penulis ingin mendemonstrasikan cara kerja algoritma Dijkstra dalam mencari rute angkutan umum secara optimal. Misalkan saja ada satu orang yang ingin berpergian dari simpul 1 menuju simpul 10. Kita akan mencari lama perjalanan minimal serta rute angkutan umum-nya.

Maka kita akan menginisialisasi jarak tentatif semua simpul kecuali simpul 1 menjadi tak terhingga, sedangkan simpul 1 jarak-nya adalah 0. Dari simpul 1, kita akan memasukkan simpul 2 dan 5. Dengan demikian simpul 2 akan memiliki jarak tentatif 11.25, sedangkan simpul 5 memiliki jarak tentatif 3.33. Karena semua tetangga simpul 1 sudah dikunjungi, maka simpul 1 dimasukkan ke himpunan simpul yang sudah dikunjungi dan tidak akan ditinjau lagi. Selanjutnya, ada dua pilihan simpul untuk menjadi simpul yang akan dikunjungi selanjutnya. Sesuai dengan cara kerja algoritma Dijkstra, simpul 5 yang akan dikunjungi berikutnya karena simpul ini memiliki jarak tentative yang lebih kecil dibanding simpul 2. Simpul 5 hanya memiliki satu tetangga yang belum dikunjungi yaitu simpul 6. Namun, ada dua sisi yang bersisian dengan simpul 5 dan 6, bila menggunakan sisi atas, simpul 6 akan memiliki jarak tentatif 10,83 sedangkan bila memakai sisi bawah, simpul 6 akan memiliki jarak tentative 23,33. Sesuai dengan algoritma Dijkstra, yang dipilih adalah yang memiliki jarak tentative paling kecil, sehingga simpul 6 memiliki jarak tentatif 10,83. Sekarang masih terdapat dua simpul di dalam *heap* yaitu simpul 2 dengan jarak tentatif 11,25 dan simpul 6 dengan jarak tentatif 10,83. Simpul selanjutnya yang akan dikunjungi adalah simpul 6 sesuai dengan aturan simpul yang memiliki jarak tentatif lebih

kecil dikunjungi dahulu walaupun simpul 6 sudah melewati dua simpul dari simpul awal. Dari simpul 6, algoritma akan mengupdate nilai jarak tentatif simpul 7 menjadi 13,33 dengan jalur melewati sisi atas (busway koridor 2). Pada tahap selanjutnya, barulah simpul 2 yang akan dikunjungi. Pada iterasi ini, nilai jarak tentatif simpul 3 akan di-update menjadi 15.

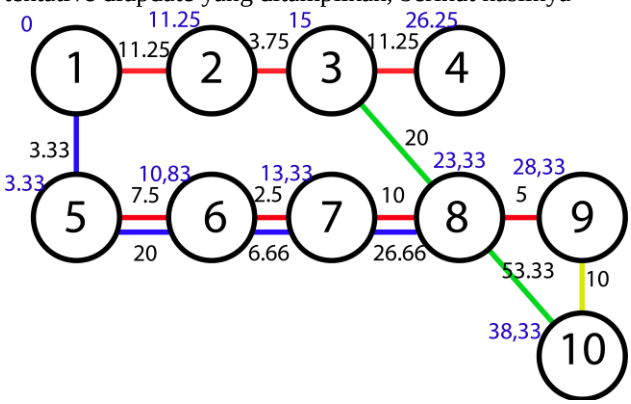
Sejauh ini, simpul-simpul yang masuk dalam himpunan simpul yang sudah dikunjungi adalah simpul 1, 2, 5, dan 6. Informasi mengenai jarak tentatif masing-masing simpul dapat dilihat pada gambar berikut



Gambar 7. Graf model dengan informasi jarak simpul berupa angka biru tua dekat simpul. Inf menandakan jarak tentative simpul tersebut masih tak terhingga

Selama iterasi sebelum-sebelumnya, setiap simpul hanya dikunjungi dari satu simpul lainnya, sehingga nilai jarak tentatif tiap simpul hanya berubah sekali. Pada kasus di atas dapat dilihat kasus dimana satu simpul bisa dikunjungi dari dua simpul lain. Pada gambar, simpul 8 dapat dikunjungi oleh simpul 7 dan simpul 3. Karena jarak tentatif simpul 7 lebih kecil, maka simpul 7 yang akan mencoba mengunjungi simpul 8 terlebih dahulu. Nilai jarak tentatif simpul 8 kini menjadi 23,33. Setelah simpul 7 ditandai sudah dikunjungi, kini giliran simpul 3 yang dikunjungi. Saat simpul 3 dikunjungi, jarak tentatif simpul 8 bila melewati simpul 3 adalah 35. Namun, karena nilai ini sudah lebih besar dari nilai jarak tentative simpul 8 saat ini yaitu 23,33, maka nilai ini tidak akan dipakai.

Untuk mempercepat pembahasan, maka hanya hasil akhir algoritma Dijkstra setelah semua jarak tentative diupdate yang ditampilkan, berikut hasilnya



Gambar 8. Jarak tentative semua simpul setelah algoritma selesai dijalankan.

Dari gambar sebelumya, telah didapat waktu yang dibutuhkan untuk pergi dari simpul 1 menuju simpul 10 yaitu 38,33 menit, sesuai dengan jarak tentative simpul 10 setelah algoritma selesai dijalankan.

C. RUNUT BALIK (BACKTRACK) UNTUK MENDAPATKAN RUTE ANGKUTAN UMUM

Dari algoritma sebelumnya, waktu optimal menggunakan angkutan umum sudah didapat. Namun, rute dan jenis angkutan umum apa saja yang perlu dinaiki untuk sampai ke tujuan belum didapat. Subbab ini akan menjelaskan algoritma runut balik untuk mendapatkan rute tersebut.

Untuk mendapatkan rute perjalanan dari simpul awal menuju simpul akhir, perlu diketahui jarak tentatif suatu simpul didapatkan dari simpul mana sebelumnya. Misalkan pada graf sebelumnya, simpul 10 dengan jarak tentatif 38,33. Ada dua kemungkinan rute berasal yaitu simpul 8 dan simpul 9. Untuk tujuan mengetahui simpul mana yang dikunjungi sebelum simpul 10 dikunjungi pada rute optimal, akan dilakukan pengecekan. Pengecekan dilakukan apakah jarak tentatif simpul 10 dikurang dengan bobot sisi sama dengan jarak tentatif simpul yang bersisian. Misalkan simpul 8 akan dicek, jarak tentatif simpul 10 dikurang bobot sisi (38,33 – 53,33 = -20) tidak sama dengan jarak tentatif sisi 8 yaitu 23,33. Sedangkan, jarak tentatif simpul 10 dikurang bobot sisi yang bersisian dengan simpul 9 (38,33 – 28,33 = 28,33) sama dengan jarak tentatif simpul 9. Hal ini membuktikan bahwa pada rute optimal, simpul yang dikunjungi sebelum simpul 10 adalah simpul 9.

Percabangan kembali ditemukan pada simpul 8, pada simpul ini terdapat tiga kemungkinan rute sebelum mengunjungi simpul 8 yaitu simpul 3, simpul 7 menggunakan busway koridor 2, dan simpul 7 menggunakan mikrolet jurusan M1. Kita dapat membuktikkan bahwa sebelum simpul 8, simpul 7 yang dikunjungi menggunakan busway koridor 2.

Sisa dari perjalanan tidak mengandung percabangan sehingga rute dapat ditentukan. Dengan begitu, rute optimal untuk pergi ke simpul 10 dari simpul 1 adalah simpul 1 – 5 – 6 -7 -8 – 9 10. Atau lebih rinci dapat ditulis menjadi

- Gunakan mikrolet M1 dari simpul 1 sampai simpul 5
- Gunakan Buswaw koridor 2 dari simpul 5 ke simpul 9
- Gunakan mikrolet M3 dari simpul 9 sampai simpul 10

Seluruh perjalanan dapat ditempuh dalam 39 menit.

Dengan penjelasan di atas, tulisan ini sudah dapat mengaplikasikan pencarian rute angkutan umum optimal menggunakan algoritma Dijkstra.

IV. KESIMPULAN

Aplikasi ini bila kedepannya sudah dibuat dan dapat digunakan akan dapat membantu banyak orang terutama wisatawan atau orang-orang yang tidak terbiasa dengan angkutan umum untuk mendapatkan rute angkutan umum yang cepat dengan mudah. Harapan penulis, dengan mudahnya mencari informasi tentang

angkutan umum, makin banyak orang yang beralih menggunakan angkutan umum sehingga kemacetan dan polusi dapat berkurang.

Untuk pengembangan algoritma ini, kedepannya data-data seperti tingkat kemacetan dan rata-rata kecepatan angkutan umum untuk setiap trayek atau koridor dapat ditingkatkan dan menjadi lebih detail, sehingga hasil perkiraan yang dihasilkan oleh aplikasi ini dapat menjadi lebih akurat dan terpercaya.

V. UCAPAN TERIMA KASIH

Pertama-tama penulis ingin bersyukur kepada Tuhan Yang Maha Esa, karena hanya oleh karena rahmat-Nya penulis dapat menyelesaikan tulisan ini. Penulis juga berterima kasih kepada dosen yang memberikan tugas ini Dr. Ir. Rinaldi Munir sekaliguse sebagai dosen pengajar penulis.atas bimbingan dan jasa beliau yang selama ini telah mengajar dan memberikan ilmu pada mata kuliah strategi algoritma dan sebelumnya membimbing dalam mata kuliah matematika diskrit, sehingga penulis mampu membuat tulisan ini. Tak lupa juga penulis berterima kasih atas rekan-rekan yang senantiasa memberikan dorongan dan semangat bagi penulis sehingga tulisan ini dapat selesai.

REFERENCES

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/
[2] Introduction to Algorithm, Thomas H. Cormen and friends
[3] http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1097-0118
[4] http://www.boost.org/doc/libs/1_58_0/libs/graph/doc/dijkstra_shortest_paths.html
[5] http://www.personal.kent.edu/~rmuhamma/GraphTheory/graphTheory.htm

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, 3 Mei 2015

Vincent Sebastian The / 13513057