Penggunaan Program Dinamis untuk Penyelesaian Persoalan Jalan Terpendek pada Jalan-Jalan di Jakarta

Marhadiasha Kusumawardhana / 13508091
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
marhadiasak@gmail.com

***Abstrak*—Jakarta adalah kota terbesar di Indonesia. Tentunya, kota ini punya jumlah jalan yang paling banyak di Indonesia. Untuk berpindah dari titik A menuju titik B tidaklah mudah, kita harus tahu rute jalan yang terbaik. Kadang dari A ke B ada 2-4 rute, bahkan terdapat jalan-jalan pintas yang tidak tertera pada papan penunjuk jalan. Tidak hanya itu saja, kadang pada saat-saat tertentu, apabila kita salah mengambil rute, kita bisa terjebak macet. Pada makalah ini, penulis ingin mencoba menyelesaikan masalah ini dengan mengaplikasikan program dinamis, yaitu dengan algoritma Djikstra. Makalah ini juga memperlihatkan bahwa program dinamis juga berguna untuk sistem busway.**

***Kata kunci*—Program Dinamis, Persoalan jalan terpendek, Jakarta, jalan, rute, macet, Algoritma Djikstra, *single-source*.**

I. Pendahuluan

Penulis, sebagai penduduk Jakarta selama 17 tahun, telah merasakan asam garamnya hidup di kota ini. Kemacetan, kriminalitas, sampah, kesenjangan sosial, buruknya sistem lalu lintas, dan lain sebagainya. Dari beribu masalah yang menumpuk, masalah paling memusingkan kepala adalah kemacetan. Ada istilah yang cukup ironis tentang masalah ini. "Jalan dari Blok M ke Ciputat lebih jauh daripada jalan dari Jakarta ke Surabaya".

Pemerintah sudah berupaya untuk memecahkan masalah ini. Contohnya, dengan menggunakan sistem busway. Solusi-solusi lain yang rencananya kaan digunakan di masa depan adalah monorail, subway, dan sebagainya. Namun, sepertinya setiap masyarakat Jakarta punya prinsip, "Aturan dibuat untuk dilanggar". Busway sudah ada, namun tetap saja motor-motor dan mobil pribadi menggunakannya. Akhirnya, masalah macet tidak dapat dihindari. Fauzi Bowo, Gubernur Jakarta saat makalah ini ditulis, dinilai sudah "menyerah" dalam menyelesaikan masalah ini.

Pada akhirnya, tidak ada cara lain bagi penduduk Jakarta biasa selain bergantung pada pengalamannya dalam memilih rute untuk menghindari kemacetan ini. Penulis pernah berhasil dalam mencoba trik ini dan pernah berhasil dan sukses menyimpan cukup waktu dan uang.

Untuk pergi dari Kampung Rambutan ke Pulo Gadung, kita bisa melewati berbagai jalan, atau memilih berbagai trayek bis. Pertanyaannya adalah jalan mana yang terbaik dalam segi jarak **dan** terhindarnya dari macet? Trayek manakah yang paling baik diambil? Rute mana yang seharusnya dilewati busway dari halte Blok M ke halte Senayan?

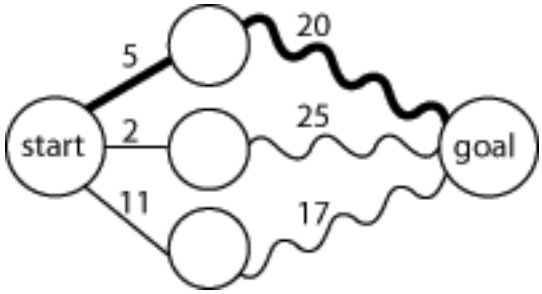
Berbagai algoritma seperti *Bruteforce* dan *Greedy* bisa digunakan untuk memecahkan masalah ini. Namun ada algoritma lain yang efektif dan akurat, yaitu dengan Program Dinamis. Pada makalah ini, penulis akan membahas penyelesaian masalah ini dengan Program Dinamis.

II. Konsep Program Dinamis

Pada subab ini penulis akan menjelaskan secara singkat tentang program dinamis dan persoalan jalan terpedek (*shortest path problem*).

Program dinamis merupakan strategi algoritma dengan memanfaatkan sebuah prinsip penting, yaitu **prinsip optimalitas**. Prinsip ini menyatakan nahwa "Jika solusi total optimal, maka solusi sampai tahap ke-k juga optimal"[1]. Program dinamis dapat mengubah masalah yang kompleks menjadi simpel dengan menguraikan masalah ini menjadi sejumlah tahap-tahap.

Dengan memgang teguh prinsip optimalitas, kita menggunakan solusi optimal dari tahap sebelumnya, untuk mendapatkan solusi optimal tahap berikutnya. Mari kita misalkan pada masalah jalan terpendek. Misal kita harus mengetahui jalan optimal dari A → F. Titik-titik yang berhubungan dengan F adalah D dan E. Jalan mana yang harus dilewati? Kita gunakan informasi apakah A → D atau A → E yang terpendek. Untuk mengetahui yang mana yang terpendek, kita gunakan informasi pada tahap sebelumnya. Misal, titik-titik yang berhubungan dengan D dan E adalah B dan C, jalan mana yang harus dilewati? Kembali lagi kita guankan informasi apakah A → B atau A → C yang terbaik. Begitu seterusnya.



Gambar 1. Uraikan masalah dengan bertahap dan gunakan nilai yang sudah dihitung sebelumnya.

III. Persoalan Jalan Terpendek dan Algoritma Djikstra

Pada paragraf sebelumnya kita telah menyentuh sedikit konsep program dinamis untuk menyelesaikan masalah jalan terpendek. Sekarang mari kita bahas lebih dalam masalah ini. Ada 4 jenis macam persoalan ini[2], yaitu:

- *Single-pair*, yaitu bagaimana menentukan jalan terpendek dari satu simpul ke satu simpul lain lewat beberapa simpul.
- *Single-source*, bagaimana menentukan jalan terpendek dari satu simpul ke semua simpul lain
- *Single-destination*, bagaimana menentukan jalan terpendek dari semua simpul ke satu simpul.
- *All-pair*, bagaimana menentukan jalan terpendek dari semua simpul ke semua simpul.

Sejauh ini, tidak ada algoritma yang khusus single-pair yang lebih baik dari pada algoritma khusus single-source. Caranya hanya lakukan algoritma khusus single-source, lalu ambil jalan dengan tujuan simpul yang diinginkan. Algoritma single-destination juga sebenarnya sama dengan single-source, hanya tinggal dibalik perannya saja. Jadi sebenarnya jenis persoalan bisa direduksi menjadi dua, yaitu single-source dan all-pair. Sebenarnya, algoritma all-pair juga berguna dan bisa digunakan untuk menyelesaikan masalah jalan terpendek, namun pada makalah ini penulis akan menggunakan algoritma khusus single-source.

Penyelesaian masalah single-source dengan program dinamis salah satunya dalah dengan Algoritma Djikstra. Algoritma ini dikemukakan oleh Djikstra, Ilmuwan sains komputer dari Belanda. Djikstra menyatakan bahwa lagoritma ini *menggunakan informasi yang didapatkan dari tahap sebelumnya untuk menentukan jalan terpendek pada tahap berikutnya*[3]. Prinsip ini sebenarnya adalah prinsip optimalitas program dinamis. Artinya **Algoritma Djikstra adalah salah satu algoritma program dinamis**. Oleh karena itu, kita akan gunakan algoritma ini untuk menyelesaikan masalah jalan terpendek pada makalah ini.

Sekarang mari kita simak bagaimana algoritma Djikstra tersebut[4].

1. Beri semua simpul sebuah nilai. Nilai ini disebut **jarak tentatif**. Nilai ini nanti akan menjadi jarak dari **simpul awal** ke simpul tersebut. Untuk pertama kali, berikan nilai tak terhingga untuk semua node kecuali simpul awal dan 0 untuk simpul awal.
2. Tandai semua simpul "perawan", atau "belum dikunjungi".
3. Jadikan simpul awal **simpul sekarang**.
4. Hitung jarak tentatif semua simpul tetangga yang belum dikunjungi. Jarak tentatifnya adalah jarak tentatif simpul sekarang ditambah jarak antara simpul sekarang dengan simpul tersebut (busur antar kedua simpul). Bila jarak tentatif yang baru dihitung lebih kecil dari jarak tentatif simpul tetangga tersebut sekarang, ganti dengan yang baru dihitung tersebut, bila tidak, buang hitungan kita tadi.
5. Setelah selesai, tandai simpul sekarang "dikunjungi".
6. Apabila semua simpul sudah "dikunjungi", maka **selesai**. Jika belum, jadikan simpul dengan jarak tentatif terkecil sebagai simpul sekarang, dan kembali ke langkah 4.

Mari kita perhatikan langkah 4. "Jarak tentatifnya adalah jarak tentatif simpul sekarang ditambah jarak antara simpul sekarang dengan simpul tersebut (busur antar kedua simpul)." Artinya untuk mencari jarak tentatif tahap berikutnya, diperlukan nilai jarak tentatif tahap sebelumnya. Hal ini adalah prinsip optimalitas.

IV. Memodelkan Peta Jakarta Menjadi Graf

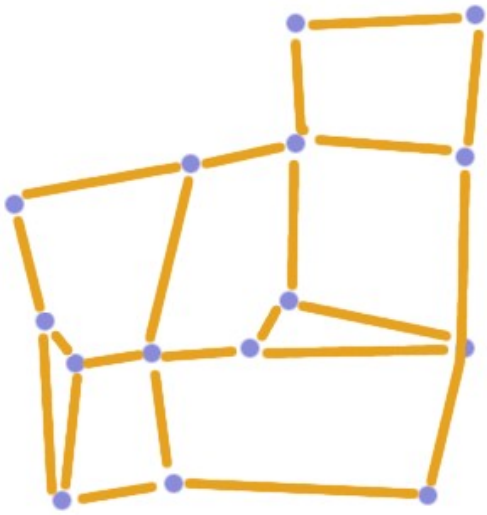
Gambar 2. Potongan Peta Jakarta

Setelah mengetahui cara menyelesaikan jalan terpendek

pada graf, sekarang bagaimana kita mengkonversi peta Jakarta menjadi sebuah graf? Jawabannya cukup simpel, jadikan jalan sebagai busur dan persimpangan jalan (baik itu pertigaasn, perempatan, atau lebih) menjadi busur. Mari kita coba memodelkan sebuah potongan peta Jakarta menjadi sebuah graf.

Pada gambar 2, sebuah potongan peta Jakarta, telah tertanda tiap persimpangan oleh lingkaran berwana biru keunguan, dan jalan berwarna kuning. Persimpangan Pulo Gadung, bertanda "A" sebagai awal lokasi kita.

Seperti yang sudah dituliskan di atas, kita ubah jalan menjadi busur dan persimpangan menjadi simpul. Sehingga kira-kira dari peta diatas kita mendapatkan graf pada gambar 3.



Gambar 3. Hasil pemodelan peta gambar 2 menjadi graf.

Yang akan kita lakukan pada graf tersebut adalah menghitung jarak terpendek dari A (misal, Pulo Gadung), menuju semua simpul. Yang menjadi jarak antar simpul adalah jarak fisik antar kedua persimpangan/ Pada makalah ini, penulis menggunakan fitur penggaris pada Google Earth sebagai alat bantu penghitung jarak fisik.

V. Koefisien Kemacetan

Nah sekarang kembali ke realita, apakah yang bisa kita jadikan jarak antar simpul (berat busur) pada jalan-jalan di Jakarta? Jawaban simpelnya adalah jarak fisik (dalam meter). Namun Jakarta bukanlah kota yang simpel. Kadang jalan yang pendek namun macet lebih menghabiskan waktu dibandingkan jarak yang jauh tapi tidak macet. Oleh karena itu, pada makalah ini penulis akan menggunakan salah satu metode untuk memasukkan hitungan kemacetan suatu jalan, yaitu dengan **koefisien kemacetan**.

Tentu saja kemacetan suatu jalan tidak sama setiap waktu. Kadang untuk pagi dari A ke B macet sekali, namun pada sore hari lancar. Kemacetan juga berubah tergantung arah, misal pada pagi hari A ke B macet, namun B ke A lancar. Selain itu, kemacetan juga bergantung pada hari, apakah itu akhir pekan atau hari biasa, dan lain sebagainya. Pada makalah ini, penulis akan menggunakan 3 faktor tersebut dari berbagai faktor lainnya untuk menentukan koefisien kemacetan ini.

Seperti namanya koefisien, maka tidak ada fungsi dan persamaan khusus untuk menentukannya. Nilai koefisien ini hanya bisa ditentukan dengan eksperimen secara empiris. Pada makalah ini, penulis menggunakan campuran pengalaman dan perkiraan untuk menentukan koefisien kemacetan tersebut.

Seperti yang sudah dikemukakan sebelumnya, ada 3 faktor yang penulis gunakan, yaitu waktu, arah dan hari. Sehingga koefisien kemacetan tergantung sebagai 4 parameter berbeda:

$$k(s, d, t, w)$$

s adalah simpul awal, d adalah simpul tujuan, t adalah waktu, w adalah hari. Pada makalah ini, penulis hanya membedakan 2 jenis waktu, yaitu pagi dan sore dan 2 jenis hari yaitu hari biasa dan akhir minggu.

Karena koefisien maka ada konvensi untuk menentukan nilainya. Penulis menentukan bahwa koefisien ini $k \geqslant 1$. Dengan 1 apabila dianggap tidak ada kemacetan dan makin tinggi nilai k, maki nparah kemacetannya.

Koefisien ini akan dikalikan dengan jarak pada di jalan tersebut dan dijadikan busur pada graf. Misal jarak dari Monas ke Senayan adalah 2000m dengan koefisien kemacetan 1,7 / m maka berat busur graf nya adalah 3400.

Setrlah memodelkan peta tersebut menjadi graf, kita gunakan algoritma Djikstra untuk mencari jalan terpendek dari suatu simpul ke simpul lain. Pada subbab-subbab selanjutnya, makalah ini melakukan studi kasus dengan menggunakan program dinamis, terutama algoritma djikstra, untuk menyelesaikan masalah.

VI. Studi Kasus 1: Busway, Tanpa Menghitung Kemacetan

Busway dibuat agar orang meninggalkan mobil pribadi dan menggunakan angkutan umum. Bagaimana orang bisa tertarik menggunakan angkutan umum? Pemerintah Jakarta membuat busway ini sedemikian rupa sehingga orang tertarik menggunakannya. Selain nyaman dan aman, busway juga memiliki jalur sendiri yang **bebas macet**.

Namun teori dan realita berbeda, busway di Jakarta tidaklah bebas macet, karena buruknya penegakan hukum, orang bisa sembarangan masuk jalur busway, sehinggu busway kebal macet bukanlah kenyataan. Namun, sebenarnya busway pernah bebas macet. Yaitu sekitar awal penggunaan busway hingga 2007. Setelah itu, busway bebas macet sudah hilang. Pada makalah ini, penulis menganggap kita ada di tahun 2006, yaitu saat

busway masih baru dan memang benar-benar bebas macet.

Pada subbab ini kita akan menyelesaikan masalah busway, yaitu tanpa menghitung koefisien kemacetan. Atau koefisien kemacetan selalu 1 pada tiap jalan.

Pada contoh di makalah ini, kita berada pada kondisi fiksi (khayalan), bahwa setelah halte Blok M, busway akan menuju halte Kampung Melayu. Namun, pemerintah DKI Jakarta bingung, jalan mana yang harus dilewati busway antar Blok M dan Kampung Melayu paling cepat agar menghemat biaya bensin. Pemda hanya memikirkan jalan yang paling cepat (bukan yang paling banyak mengangkut penumpang) karena memang busway hanya boleh berhenti pada halte, tidak seperti angkutan umum lainnya.

Sekarang mari kita lihat potongan peta Jakarta di daerah Blok M - Kampung Melayu di Gambar 4.

Pada gambar 5, walaupun tidak semua persimpangan dijadikan simpul, persimpangan-persimpangan penting yang layak dilewati busway diubah menjadi simpul. Simpul O adalah Blok M dan simpul H adalah Kampung Melayu. Dengan busur adalah garis-garis kuning antar simpul. Pertanyaannya adalah rute manakah yang tercepat dari simpul O ke H?

Sebelum kita menjawab pertanyaan tersebut, ada yang kurang dari graf di atas, yaitu jarak antar simpul. Seperti yang sudah ditulis di atas, penulis menggunakan fitur penggaris di Google Earth untuk mendapatkan jarak fisik antar persimpangan. Karena pada kasus ini koefisien kemacetan adalah 1, maka kita cukup menjadikan jarak fisik sebagai berat busur. Pada tabel 1, tertera berat busur yang sudah dihitung penulis.

Gambar 4. Potongan Peta Jakarta sekitar Blok M- Kampung Melayu.

Blok M ada di sekitar tulisan "..ramat Pela" di kiri bawah pada gambar dan Kampung Melayu ada pada tulisan "Bukit Duri" di kanan atas pada peta.

Sekarang mari kita modelkan potongan peta di atas menjadi graf dengan teknik subab III. Pertama kita ubah persimpangan menjadi simpul graf. Lihat gambar 5.

Gambar 5. Mengubah tiap persimpangan dipeta menjadi simpul graf.

Tabel 1. Jarak fisik (berat busur) antar simpul

Busur antar	Jarak fisik (m)
O – A	3417
O – B	3022
A – B	2819
A – D	1732
A – E	1723
B – C	830
C – D	2255
D – F	1655
E – F	2352
E – G	2755
F – H	2127
G – H	2232

Jarak fisik di tabel 1 kita ubah menjadi berat busur. Seakarang mari kita pecahkan masalah ini. Rute manakah yang terpendek dari Blok M ke Kampung Melayu? Tentu kita gunakan program dinamis yaitu algoritma Djikstra, sebagai alat, kita gunakan tabel data untuk tiap simpul.

Kolom pertama adalah simpul ynag bersangkutan. Kolom kedua menunjukkan apakah simpul tersebut sudah dikunjungi atau belum. Kolom 3 adalah jarak tentatif simpul tersebut dan kolom terakhir, Link, menunjukkan simpul dimana jarak tentatif simpul tersebut dihitung (simpul yang mana jarak tentatifnya ditambah dengan jarak simpul sekarang dengan simpul tersebut menjadi jarak tentatif simpu sekarang). Lebih baik Anda buat tabel ini sendiri pada secarik kertas atau program Anda untuk mengikuti algoritma yang akan dipaparkan di bawah ini.

Tabel 2 adalah tabel tersebut pada tahap Algoritma Djikstra awal (Lihat subbab III). Sesuai algoritma Djikstra, kita jadikan Blok M (simpul O) menjadi **simpul awal**, lalu jadikan simpul tersebut menjadi simpul sekarang.

Tabel 2. Tabel simpul pada tahap awal.

Simpul	Dikunjungi	Jarak Tentatif	Link
O (skrg)	Belum	0	O
A	Belum	Tak hingga	
B	Belum	Tak hingga	
C	Belum	Tak hingga	
D	Belum	Tak hingga	
E	Belum	Tak hingga	
F	Belum	Tak hingga	
G	Belum	Tak hingga	
H	Belum	Tak hingga	

Lalu, kita lihat tetangga simpul O, yaitu A dan B, dan hitung jarak tentatifnya. A adalah 0 + 3417 (jarak tentatif simpul sekarang (0), O, dan jarak antar simpul sekarang dengan simpul tetatangga yang mau dihitung (3417, lihat Tabel 1, jarak O – A)) dan B adalah 0 + 3022, sehinggu masing-masing jarak tentatifnya adalah adalah 3417 dan 3022 dengan Link O. Lalu kita tandai O sudah dikunjungi.

Setelah itu kita ambil simpul dengan jarak tentatif terkecil dan belum dikunjungi, yaitu B, sebagai simpul sekarang. Lakukan langkah 4, yaitu menilai tetangga B, C dan A dengan jarak tentatif 3022 + 850 = 3872 dengan link B untuk C. Untuk A, kita tidak perlu mengganti jarak tentatif, karena jarak tentatif yang baru, 3022 + 2819 = 5841 lebih besar dari jarak tentatif sebelumnya, 3417. Lalu, pilih simpul A sebagai simpul sekarang karena jarak tentatif terkecil dan tandai B sudah dikunjungi. Sekarang mari kita lihat tabel simpul sekarang pada tabel 3.

Tabel 3. Tabel simpul pada suatu tahap.

Simpul	Dikunjungi	Jarak Tentatif	Link
O	**Sudah**	0	O
A (skrg)	Belum	3417	O
B	**Sudah**	3022	O
C	Belum	3872	B
D	Belum	Tak hingga	
E	Belum	Tak hingga	
F	Belum	Tak hingga	
G	Belum	Tak hingga	
H	Belum	Tak hingga	

Mari kita lanjutkan algoritma ini. Setelah A dijadikan simpul sekarang, kita hitung jarak tentatif tetangganya (D dan E). Jarak tentatif D = 3417 + 1732 = 5149, Link A, dan E = 3417 + 1723 = 5140, link A. Tandai A dikunjungi, jadikan C simpul sekarang.

Seakarang kita hitung jarak tentatif tetangga C, yaitu D. Jarak tentatif yang baru = 3872 + 2255 = 6127. Ternyata jarak tentatif yang baru ini lebih besar dari yang lama (5149). Maka penghitungan ini dibuang dan jarak tentatif dan link D tidak berubah. Lalu tandai C dikunjungi, jadikan E sebagai simpul sekarang.

Mari kita hitung jarak tentatif tetangga E, yaitu F dan G. Jarak tentatif F = 5140 + 2352 = 7492, link E dan jarak tentatif G = 5140 + 2755 = 7895, link E. Tandai E sudah dikunjungi, dan jadikan D simpul sekarang. Mari kita lihat tabel 4 sudah sejauh mana kita sekarang.

Tabel 4. Tabel simpul pada suatu tahap.

Simpul	Dikunjungi	Jarak Tentatif	Link
O	**Sudah**	0	O
A	**Sudah**	3417	O
B	**Sudah**	3022	O
C	**Sudah**	3872	B
D (skrg)	Belum	5149	A
E	**Sudah**	5140	A
F	Belum	7492	E
G	Belum	7895	E
H	Belum	Tak hingga	

Mari kita lanjutkan. Kita hitung jarak tentatif tetangga D yang belum dikunjungi yaitu F. Jarak tentatif F yang baru = 5149 + 1655 = 6804, link D. Jarak yang baru jauh lebih kecil dari yang lama. Mari kita ganti jarak tentatif F dengan yang baru. Selanjutnya, kita tandai D sudah dikunjungi, dan jadikan F simpul sekarang (jarak tentatif terkecil yang belum dikunjungi, 6804).

Setalah itu, kita hitung jarak tentatif tetangga F, yaitu H.Jarak tentatif H = 6804 + 2127 = 8931, link F. Setalah itu kita jadikan G simpul sekarang dan tandai F sudah dikunjungi.

Satu-satunya tetangga G, H, kita hitung jarak tentatif barunya, = 7895 + 2232 = 10127, link G. Ternyata lebih besar dari yang sebelumnya, 8391. Maka kita biarkan. Tandai G dikunjungi. Karena simpul yang belum dikunjungi tinggal satu, H, kita bisa langsung menandai H dikunjungi. Selesai sudah rentetan algoritma Djikstra kita. Mari kita lihat akhir tabel simpul di tabel 5.

Tabel 5. Tabel simpul akhir.

Simpul	Dikunjungi	Jarak Tentatif	Link
O	**Sudah**	0	O
A	**Sudah**	3417	O
B	**Sudah**	3022	O
C	**Sudah**	3872	B
D	**Sudah**	5149	A
E	**Sudah**	5140	A
F	**Sudah**	6804	D
G	**Sudah**	7895	E
H	**Sudah**	8391	F

Sari tabel tersebut kita bisa menentukan jalan terpendek dari A ke suatu simpul dan rutenya.

Untuk mengetahui rute butuh teknik seperti "*Linked list*" dalam menjabarkan rutenya (oleh karena itu kolom ke-empat disebut link). Caranya adalah pertama tentukan simpul yang dituju, misalnya X. Lalu lihat simpul pada link simpul tersebut, misal linknya adalah simpul X', bentuk rute X' → X, lalu link pada simpul X' adalah X", buat rute X" → X' → X, dan begitu seterusnya hingga sampai pada simpul awal.

Mari kita lakukan penjabaran rute in pada tabel diatas. Misal kita ingin mendapatkan rute dari O → G. Pertama kita lihat link di simpul G. Ternyata adalah E. Terbentuk rute E → G. Lalu link pada E adalah A. Terbentuk rute A → E → G. Lalu link pada A adalah O. Maka terbentuk rute O → A → E → G. Rute tersebut lah yang menjadi rute terpendek jalan dari O → G dengan jarak 7895 m.

Nah kita gunakan cara di atas untuk menjawab pertanyaan kita, yaitu rute manakah yang terbaik pada Blok M → Kampung Melayu, atau dari O → H? Pertama

Gambar 6. Rute terpendek dari Blok M ke Kampung Melayu

kita lihat link di simpul H, ternyata dalah F. Terbentuk rute F → H. Lalu, link di F adalah D. Terbentuk rute D → F → H. Lalu link di D adalah A. Terbentuk rute A → D → F → H. Lalu, yang terakhir link A adalah O, maka terbentuklah rute penuh dari O → H, yaitu **O → A → D → F → H** dengan jarak **8391** m.

Dengan begitu, kita sudah memecahkan pertanyaan tadi, yaitu O → A → D → F → H. Sekarang mari kita kembalikan dari model graf ke model peta, rute terbaik dari Blok M ke kampun melayu adalah lewat O → A yaitu Jalan Trunojoyo, lalu ke Jalan Wolter Mongsidi dan belok kiri ke Jalan HR Rasuna Said (A → D), lalu belok kanan masuk ke Jalan Dr. Satrio (masuk ke D → F), dan lewati persimpangan empat (simpul F). Akhirnya sampailah pada Kampung Melayu dengan jalan terpendek.

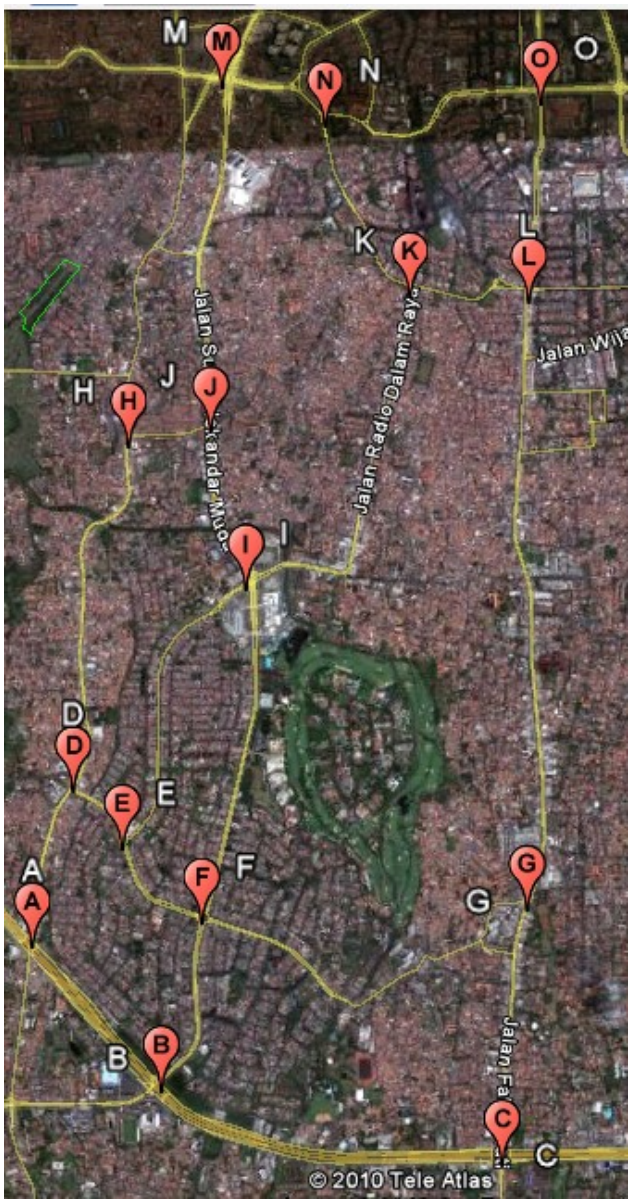
VII. Studi Kasus 2: Mobil Pribadi: Menghitung Kemacetan

Pada kasus pertama, kita menggunakan busway sebagai kendaraan yang, secara teori, bebas macet. Artinya koefisien kemacetan selalu 1. Sekarang kita menuju situasi yang lebih realistis, ada jalan yang macet pada saat tertentu dan ada yang tidak macet pada saat tertentu. Seperti yang telah dikemukakan pada subbab V, kita akan menggunakan koefisien kemacetan sebagai ukuran seberapa macet suatu jalan.

Gambar 7. Potongan peta Jakarta antara Lb. Bulus dan Blok M

Pada studi kasus kali ini kita menjadi seorang ibu yang harus mengantar anaknya pergi ke sekolah. Anggaplah rumah mereka ada di Ciputat dan sekolahnya ada di daerah Kebayoran Baru, tepatnya daerah Blok M. Sekarang mereka ada di Lebak Bulus dan ingin mencari jalan tercepat ke sekolah. Ternyata, ada banyak jalan menuju (Roma) ke sekolah sang anak, Lihat gambar 7. Lb. Bulus adalah titik biru sebelah kiri bawah peta dan Blok M adalah titik biru di sebelah kanan atas peta. Manakah jalan yang terbaik yang sebaiknya dilewati sang ibu?

Pertama-tama kita ubah dulu peta tersebut menjadi graf dengan mengubah persimpangan menjadi simpul dan jalan menjadi busur. Gambar 8 adalah hasil pengubahan tersebut.



Gambar 8. Persimpangan jalan menjadi simpul dan jalan menjadi busur.

Kita harus mencari jalan terbaik dari simpul A → O. Sekarang mari kita lihat data pada jalan-jalan tersebut. Tabel 6 adalah jarak fisik antar simpul.

Table 6. Tabel jarak fisik suatu jalan.

Busur Antar	Jarak (m)
A – B	983
A – D	895
B – C	1985
B – F	977
C – G	1398
D – E	238
D – H	2044
E – F	696
E – I	1751
F – G	2168
F – I	1772
G – L	3375
H – J	445
H – M	2155
I – J	917
I – K	2142
J – M	1882
K – L	664
K – N	1099
L – O	1067
M – N	584
N – O	1343

Setelah itu, perlu juga data koefisien kemacetan untuk mengetahui jalan mana yang rawan macet dan mana yang tidak terlalu macet sebagai penghitungan. Karena pada kasus ini adalah pagi dan hari biasa (mengantar anak ke sekolah), maka kita siapkan data koefisien kemacetan untuk pagi dan hari biasa. Untuk data yang lengkap butuh koefisien kemacetan pada sore hari dan *weekend* juga.

Data koefisien kemacetan yang kita siapkan juga cukup dengan elemen arah yang "*reasonable"*. Contohnya, kita tidak perlu memberikan data koefisien kemacetan untuk O → L karena jelas berkebalikan arah dengan tujuan, yaitu ke simpul O, tapi kita berikan data untuk L → O. Data tersebut ada pada tabel 7.

Data ini diambil dari kiraan "empiris" penulis yang sudah tinggal di Jakarta selama 17 tahun. Namun, pada praktik sebenarnya, data ini bisa saja didapatkan dengan cara menghitung waktu perjalanan, dan lain sebagainya.

Tabel 7. Koefisien kemacetan suatu jalan dengan arah tertentu, pagi hari, dan hari biasa.

Busur Antar	Koefisien
A → B	1
A → D	1,7
B → C	1
B → F	1,6
C → G	1,4
D → E	1,1
D → H	1,9
E → F	1,1
E → I	1,2
F → G	1,2
F → I	1,9
G → L	1,8
H → J	1,2
H → M	1,4
I → J	1,9
I → K	1,8
J → M	2
K → L	1,5
K → N	1,3
L → O	1,6
M → N	1,2
N → O	1,6

Yang dihitung sebagai "biaya" atau *cost* untuk melewati suatu busur adalah campuran dari kedua faktor di atas, yakni kemacetan dan jarak fisik. Oleh karena itu, untuk menggabungkan keduanya, mari kita kalikan kedua nilai tersebut. Tabel 8 menampilkan hasil perkalian tersebut. Nilai perkalian ini kita sebut biaya busur.

Tabel 8. Tabel biaya busur.

Busur Antar	Biaya Busur
A → B	983
A → D	1521,5
B → C	1985
B → F	1563,2
C → G	1957,2
D → E	261,8
D → H	3383,6
E → F	765,6
E → I	2101,2
F → G	2601,6
F → I	3366,8

G → L	6075
H → J	534
H → M	3017
I → J	1742,3
I → K	3855,6
J → M	3764
K → L	996
K → N	1428,7
L → O	1707,2
M → N	700,8
N → O	2148,8

Nilai-nila pada tabel 8 lah yang akan kita jadikan bobot busur graf karena selain sudah menghitung jarak fisik sebagai faktor, faktor kemacetan juga dihitung.

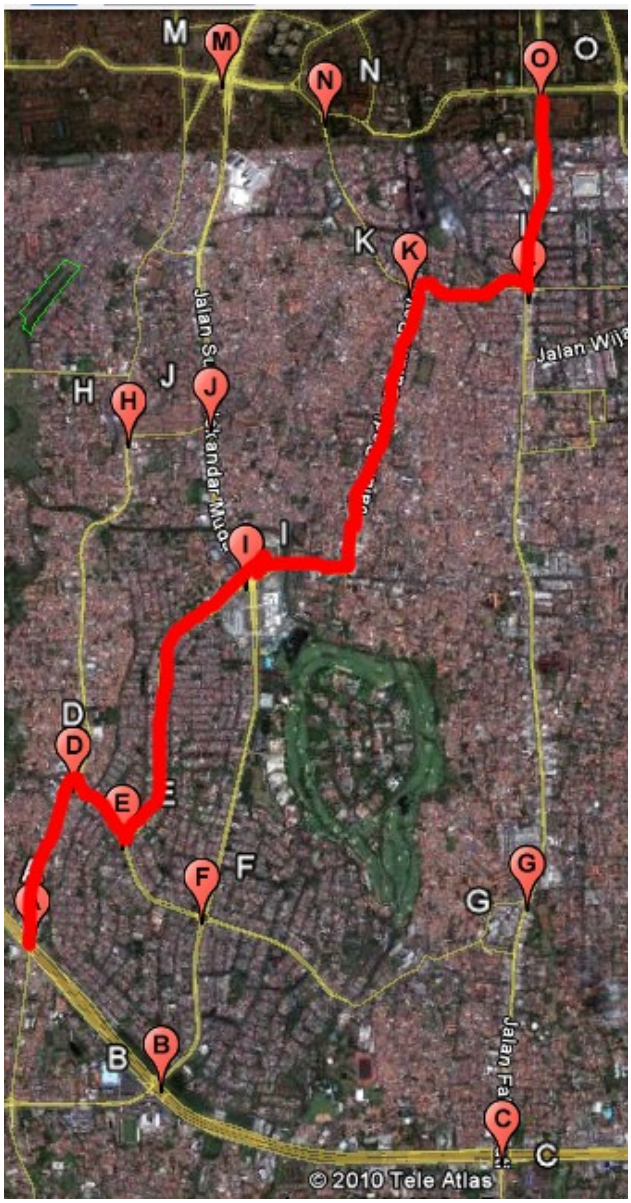
Dari nilai-nilai di atas, lengkaplah komponen-komponen sebuah graf, yaitu simpul, direpresentasikan dengan persimpangan, busur, direpresentasikan dengan jalan, dan bobot busur, direpresentasikan dengan biaya busur. Oleh karena itu, kita sudah bisa memulai mencari rute terbaik dengan algoritma Djikstra.

Cara menghitung dan mendapatkan jarak terpendek persis sama dengan studi kasus sebelumnya. Oleh karena itu penulis tidak menjabarkan detil *step-by-step* penggunaan algoritma Djikstra untuk mendapatkan rute terbaik pada makalah ini. Penulis hanya menyediakan kondisi akhir tabel simpul setelah melaksanakan algoritma Djikstra. Pembaca bisa mencobanya sendiri dan mencocokkan tabel simpul yang dibangun sendiri dengan tabel simpul yang tersedia. Tabel 9 adalah tabel simpul akhir tersebut.

Tabel 9. Tabel simpul akhir dari graf.

Simpul	Dikunjungi	Jarak Tentatif	Link
A	**Sudah**	0	A
B	**Sudah**	983	A
C	**Sudah**	2968	B
D	**Sudah**	1521,5	A
E	**Sudah**	1783,3	D
F	**Sudah**	2546,2	B
G	**Sudah**	4925,5	C
H	**Sudah**	4905,1	D
I	**Sudah**	3884,5	E
J	**Sudah**	5439,1	H
K	**Sudah**	7740,1	I
L	**Sudah**	8736,1	K
M	**Sudah**	7922,1	H
N	**Sudah**	8622,9	M
O	**Sudah**	10443,3	L

Sekarang mari kita bangun rute terbaik dari data tabel simpul akhir di atas. Tentu saja caranya dengan menghubungkan link-link yang ada di baris tabel mulai dari baris O. Rute terbaik tersebut adalah **A → D → E → I → K → L → O**. Gambar 9 memperlihatkan dengan jelas rute terbaik tersebut.



Gambar 9. Rute terbaik dari Lebak Bulus ke Kebayoran Baru.

Jadi, ibu tadi yang ingin mengantar anaknya sekolahpagi hari dan di hari biasa lebih baik mengitu rute seperti Gambar 9 di atas. Selain jaraknya pendek, Jalur ini juga kemacetannya bukan yang paling parah.

VIII. Daftar Referensi

[1] *Dynamic Programming*. http://www.seas.gwu.edu/~ayoussef/cs212/dynamicprog.html Diakses 28-29 November 2010.

[2] *Dynamic Programming*. http://www.cs.umass.edu/~barring/cs611/lecture/8.pdf Diakses 28-29 November 2010.

[3] Djikstra, 1959. *A Note on Two Problem in Connexion With Graphs*. http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf Diakses 28-29 November 2010.

[4] *Djikstra's Algorithm*. http://en.wikipedia.org/wiki/Dijkstra's_algorithm Diakses 28-29 November 2010.

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, 29 November 2010

Marhadiasha Kusumawardhana / 13508091