Implementasi TSP dalam Menentukan Rute Tersingkat untuk Melewati Kota-Kota di Jawa Barat

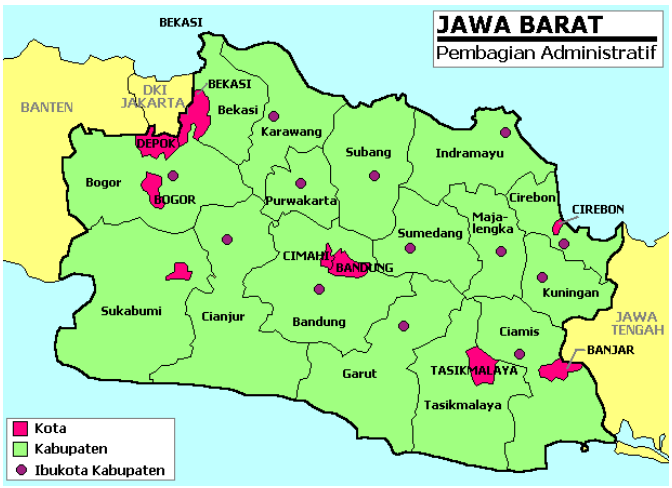
Hutama Tefotuho Hulu / 13515045
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha Nomor 10, Bandung 40132, Indonesia

***Abstrak*—Sebagai daerah satelit dari Ibu Kota DKI Jakarta, Jawa Barat merupakan salah satu daerah penting dalam pertumbuhan ekonomi Indonesia. Ibu Kota Provinsi Jawa Barat, yaitu Bandung, adalah salah satu kota terpenting di Indonesia, sekaligus menopang perekonomian daerah-daerah di sekitarnya. Tulisan ini akan membahas mengenai penentuan rute tersingkat untuk melewati kota-kota di Jawa Barat dengan menggunakan implementasi persoalan TSP. Penyelesaian TSP yang digunakan pada persoalan ini adalah penyelesaian dengan algoritma Brute Force dan Bobot Tur Lengkap (bagian dari Branch and Bound). Dari kedua algoritma yang ada, akan dianalisis yang mana yang menghasilkan kemampuan terbaik untuk menentukan rute tersingkat antar daerah-daerah di Provinsi Jawa Barat.**

***Kata Kunci*—TSP, Jawa Barat, Kota, Graf, Jarak, Brute Force, Bobot Tur Lengkap.**

I. PENDAHULUAN

Jawa Barat adalah sebuah provinsi di Indonesia, dengan ibu kotanya berada di Bandung. Provinsi Jawa Barat berada di bagian barat Pulau Jawa. Wilayahnya berbatasan dengan Laut Jawa di utara, Jawa Tengah di timur, Samudera Hindia di selatan, serta Banten dan DKI Jakarta di Barat.



Gambar 1 – Peta Provinsi Jawa Barat. sumber : https://id.wikipedia.org/wiki/Berkas:West_Java_province.png

Jawa Barat merupakan salah satu provinsi terluas di Indonesia, dengan luas 35.222,18 km^2. Penduduk Jawa Barat juga merupakan salah satu provinsi yang terbanyak di Indonesia, dengan jumlah penduduk 52.476.473. Provinsi Jawa Barat terbagi menjadi 27 daerah, yang terbagi atas 18 kabupaten dan 9 kota. 9 kota tersebut terbagi atas Bandung, Banjar, Bekasi, Bogor, Cimahi, Cirebon, Depok, Sukabumi, dan Tasikmalaya.[1]

Makalah ini akan membahas jarak antara kota-kota besar di Jawa Barat yang sudah tertulis diatas. Untuk mempermudah penjelasan, akan terdapat suatu tabel yang menjelaskan jarak antar daerah-daerah yang ada di Jawa Barat. Selain itu, juga akan terdapat tabel heuristik yang berisi tentang jarak antar titik dalam peta.

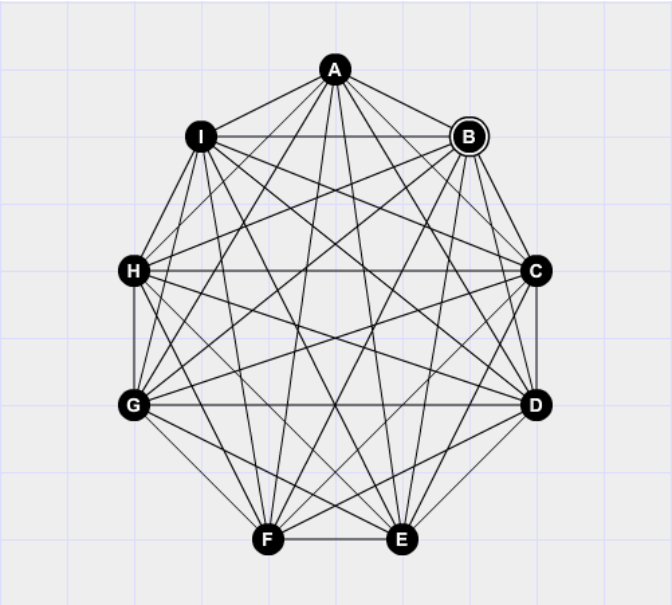
	A	B	C	D	E	F	G	H	I
A		146	133	183	13	215	165	97	113
B	146		274	325	160	114	306	239	43
C	133	274		57	124	200	39	113	240
D	183	325	57		174	250	43	65	290
E	13	160	124	174		206	155	88	124
F	215	114	200	250	206		233	276	110
G	165	306	39	43	155	233		100	276
H	97	239	113	65	88	276	100		204
I	113	43	240	290	124	110	276	204	

Tabel 1 – Jarak Perjalanan Darat Antar Kota di Provinsi Jawa Barat. Tabel diolah sendiri dengan bantuan situs http://www.distancefromto.net/

Berdasarkan tabel diatas, akan ditentukan jarak antar kota dengan menggunakan algoritma-algoritma yang diinginkan.

Catatan : Kota A – I dalam tabel menggunakan informasi sebagai berikut. A → Bandung, B → Banjar, C → Bekasi, D → Bogor, E → Cirebon, F → Cimahi, G → Depok, H → Sukabumi, I → Tasikmalaya. Apabila kita hendak melihat tabel 2 dengan elemen A,B, kita sedang memperhatikan jarak heuristik antara A (Bandung) dengan B (Banjar).

Agar semakin mudah memahami peta, peta akan dipermudah dalam sebuah bentuk graf. Simbol A – I memiliki artian yang sama dengan tabel diatas.



Gambar 2 –Graf Berarah yang melambangkan setiap kota di Provinsi Jawa Barat. Gambar diolah sendiri dengan bantuan situs http://illuminations.nctm.org/Activity.aspx?id=3550

II. LANDASAN TEORI

A. *Travelling Salesperson Problem*

Persoalan TSP mengandung seorang salesman dan sekelompok kota. Salesman harus mengunjungi semua kota dimulai dari kota pertama, dan kembali ke kota yang sama. Tantangan dari permasalahan ini adalah bagaimana seorang pedagang keliling ini meminimasi total perjalanan.[2]

TSP dapat dideskripsikan sebagai berikut :

TSP = {(*G*, *f*, *t*): G = (*V*, *E*) adalah sebuah graf lengkap
f adalah fungsi V x $V \rightarrow$ Z,
$t \in$ Z,
Z adalah graf yang mengandung cost yang tidak melebihi *t}*

B. *Algoritma Brute Force*

Brute force adalah sebuah pendekatan yang lempang (*straightforward*) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (*obvious way*).

Algoritma *brute force* umumnya tidak "cerdas" dan tidak mangkus, karena ia membutuhkan jumlah langkah yang besar dalam penyelesaiannya, terutama bila masalah yang dipecahkan berukuran besar (dalam hal ini ukuran masukannya). Kadang-kadang algoritma *brute force* disebut juga algoritma **naif** (naïve algorithm).

Algoritma *brute force* seringkali merupakan pilihan yang kurang disukai karena ketidakmangkusannya itu, tetapi dengan mencari pola-pola yang mendasar,keteraturan, atau trik-trik khusus, biasanya akan membantu kita menemukan algoritma yang lebih cerdas dan lebih mangkus.

Untuk masalah yang ukurannya kecil, kesederhanaan *brute force* biasanya lebih diperhitungkan daripada ketidakmangkusannya. Algoritma *brute force* sering digunakan sebagai basis bila membandingkan beberapa alternatif algoritma yang mangkus.

Meskipun *brute force* bukan teknik pemcahan masalah yang mangkus, namun teknik *brute force* dapat diterapkan pada sebagian besar masalah. Agak sukar menunjukkan masalah yang tidak dapat dipecahkan dengan teknik *brute force*. Beberapa pekerjaan mendasar di dalam komputer dilakukan secara *brute force*, seperti menghitung jumlah dari *n* buah ilangan, mencari elemen terbesar di dalam tabel, dan sebagainya.

Selain itu, algoritma *brute force* seringkali lebih mudah diimplementasikan daripada algoritma yang lebih canggih, dank arena kesederhanaannya, kadang-kadang algoritma *brute force* dapat lebih mangkus (ditinjau dari segi implementasi).[3]

C. *Algoritma Branch and Bound*

Algoritma *Branch and Bound* (*B&B*) adalah metode pencarian didalam ruang solusi secara sistematis. Ruang solusi diorganisasikan ke dalam pohon ruang status. Pembentukan pohon ruang status pada algoritma *B&B* berbeda dengan pembentukan pohon pada algoritma runut-balik. Bila pada algoritma runut-balik ruang solusi dibangun secara dinamis berdasarkan skema *DFS*, maka pada algoritma *B&B*, ruang solusi dibangun dengan skema *BFS*. Untuk mempercepat pencarian ke simpul solusi, maka setiap simpul diberi sebuah nilai ongkos (*cost*). Simpul berikutnya yang akan diekspansi tidak lagi berdasarkan urutan pembangkitannya (sebagaimana pada *BFS* murni), tetapi simpul yang memiliki ongkos paling kecil diantara simpul-simpul hidup lainnya (*least cost search*). Nilai ongkos pada setiap simpul *i* menyatakan taksiran ongkos termurah lintasan dari simpul *i* ke simpul solusi (*goal node*).

c(i) = nilai taksiran termurah dari simpul status *i* ke status tujuan

Dengan kata lain, c(i) menyatakan batas bawah (*lower bound*) dari ongkos pencarian solusi dan status *i*. Ongkos ini dihitung dengan suatu fungsi pembatas. Sebagaimana pada algoritma runut-balik, fungsi pembatas digunakan untuk membatasi pembangkitan simpul yang tidak mengarah ke simpul solusi. [3]

III. METODE PENYELESAIAN MASALAH

A. *Spesifikasi Persoalan*

Kita akan mencari suatu rute perjalanan yang diasumsikan dimulai dari Bandung, lalu memutari seluruh kota di Jawa Barat, kemudian kembali ke Bandung. Setiap kota diberikan

simbol sesuai dengan yang sudah dituliskan diatas, yaitu A → Bandung, B → Banjar, C → Bekasi, D → Bogor, E → Cirebon, F → Cimahi, G → Depok, H → Sukabumi, I → Tasikmalaya. Akan diberikan dua metode penyelesaian masalah, yaitu dengan Brute Force serta dengan Branch and Bound. Kedua penyelesaian tersebut dapat diperhatikan. Nantinya, apabila berhasil, kedua persoalan akan menghasilkan suatu nilai yang sama dengan rute yang sama.

B. Penyelesaian dengan Algoritma Brute Force

Algoritma Brute Force meninjau seluruh kemungkinan penyelesaian masalah. Secara tidak langsung, kita memperhatikan sirkuit Hamilton yang mungkin dibuat dalam sekali rute perjalanan.

Untuk setiap *n* kota, terdapat (*n – 1*)! kemungkinan penyelesaian masalah. Sehingga, untuk 9 kota di Jawa Barat, terdapat (9 – 1)! = 8! = 40320 sirkuit yang harus dienumerasikan, sehingga mendapatkan solusi penyelesaian yang paling mangkus.

Namun, penyelesaian ini dapat disederhanakan, dengan memperhatikan bahwa rute seperti A – B – C – D – E – F – G – H – I – A sama dengan rute A – I – H – G – F – E – D – C – B – A, sehingga jumlah graf yang harus dihitung menjadi 40320/2 = 20160 sirkuit Hamilton yang harus ditinjau.

Cara ini sangat tidak efisien, karena orde waktu yang digunakan tidak polynomial lagi (O(n) = n!). Namun, dengan metode Brute Force, dapat ditemukan bahwa jarak tersingkat perjalanan adalah Bandung – Cirebon – Sukabumi – Bogor – Depok – Bekasi – Cimahi – Banjar – Tasikmalaya – Bandung, dengan total rute 718 km.

C. Penyelesaian dengan Algoritma Branch and Bound

Untuk menyelesaikan persoalan dengan algoritma Branch and Bound, kita akan menggunakan metode bobot tur lengkap untuk menentukan pohon penyelesaian yang akan digunakan, sehingga dapat terpilih suatu pohon yang menghasilkan cost terkecil.

Awalnya, tentu saja kita akan menggunakan simpul "A" sebagai awal dari pohon pencarian yang akan kita buat. Untuk itu, kita akan menuliskan pohon dengan simpul A, beserta nilai boundary functionnya.

Berikut ini adalah pohon dengan akar "A"

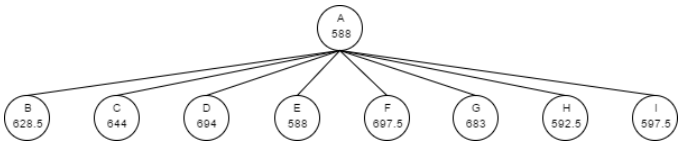
Rute saat ini : A

Gambar 3 – Sebuah pohon penyelesaian yang dimulai dengan akar A. Gambar diolah sendiri dengan bantuan situs draw.io.

Kemudian, akan dilakukan analisis terhadap kota-kota selain A. Pilih kota yang memiliki nilai boundary function terkecil/sama dengan kota A. Dengan melakukan analisis terhadap jarak kota-kota yang lain, didapatkan bahwa kota selanjutnya yang akan kita kunjungi selanjutnya adalah kota E (boundary function = 588).

Berikut ini adalah pohon dengan akar "A" dengan cabang yaitu daun-daun sisanya.

Rute saat ini : AE

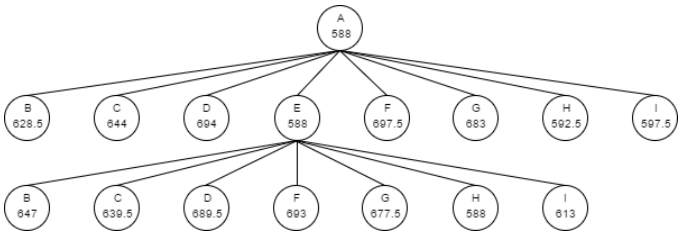


Gambar 4 – Sebuah pohon penyelesaian yang dimulai dengan akar A, kemudian dicari kota-kota lainnya. Gambar diolah sendiri dengan bantuan situs draw.io.

Kemudian, akan dilakukan analisis terhadap kota-kota selain A dan E. Pilih kota yang memiliki nilai boundary function terkecil/sama dengan kota E. Dengan melakukan analisis terhadap jarak kota-kota yang lain, didapatkan bahwa kota selanjutnya yang akan kita kunjungi selanjutnya adalah kota H (boundary function = 588).

Berikut ini adalah pohon dengan akar "A", daun "E", dengan cabang lanjutan yaitu daun-daun sisanya.

Rute saat ini : AEH

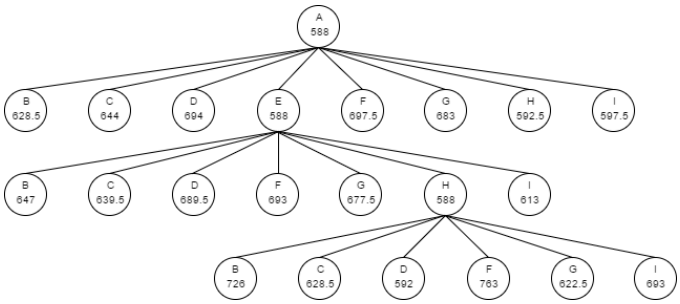


Gambar 5 – Sebuah pohon penyelesaian yang dimulai dengan akar A, E, kemudian dicari kota-kota lainnya. Gambar diolah sendiri dengan bantuan situs draw.io.

Kemudian, akan dilakukan analisis terhadap kota-kota selain A, E, dan H. Pilih kota yang memiliki nilai boundary function terkecil/sama dengan kota H. Dengan melakukan analisis terhadap jarak kota-kota yang lain, didapatkan bahwa kota selanjutnya yang akan kita kunjungi adalah kota D (boundary function = 592).

Berikut ini adalah pohon dengan akar "A", daun "E", daun "H", dengan cabang lanjutan yaitu daun-daun sisanya.

Rute saat ini : AEHD

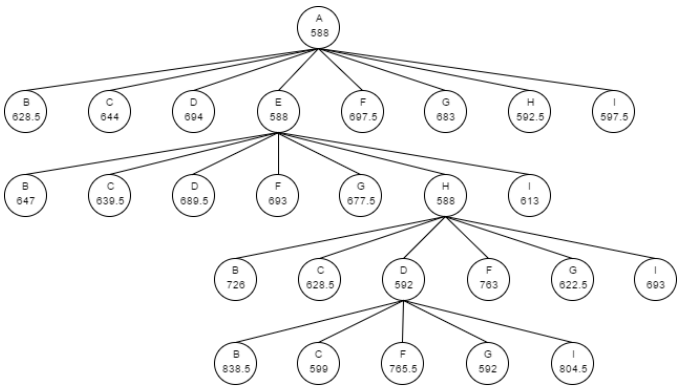


Gambar 6 – Sebuah pohon penyelesaian yang dimulai dengan akar A, E, H, kemudian dicari kota-kota lainnya. Gambar diolah sendiri dengan bantuan situs draw.io.

Kemudian, akan dilakukan analisis terhadap kota-kota selain A, E, H, dan D. Pilih kota yang memiliki nilai boundary function terkecil/sama dengan kota D. Dengan melakukan analisis terhadap jarak kota-kota yang lain, didapatkan bahwa kota selanjutnya yang akan kita kunjungi adalah kota G (boundary function = 592).

Berikut ini adalah pohon dengan akar "A", daun "E", daun "H", daun "D", dengan cabang lanjutan yaitu daun-daun sisanya.

Rute saat ini : AEHDG

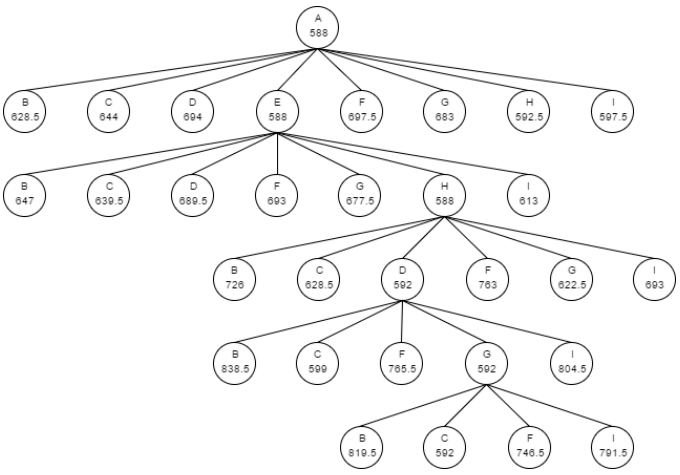


Gambar 7 – Sebuah pohon penyelesaian yang dimulai dengan akar A, E, H, D, kemudian dicari kota-kota lainnya. Gambar diolah sendiri dengan bantuan situs draw.io.

Kemudian, akan dilakukan analisis terhadap kota-kota selain A, E, H, D, dan G. Pilih kota yang memiliki nilai boundary function terkecil/sama dengan kota G. Dengan melakukan analisis terhadap jarak kota-kota yang lain, didapatkan bahwa kota selanjutnya yang akan kita kunjungi adalah kota C (boundary function = 592).

Berikut ini adalah pohon dengan akar "A", daun "E", daun "H", daun "D", daun "G", daun "C", dengan cabang lanjutan yaitu daun-daun sisanya.

Rute saat ini : AEHDGC

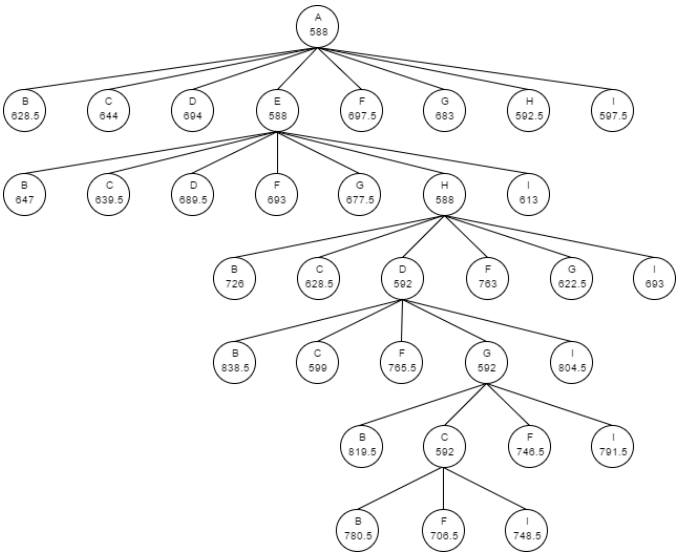


Gambar 8 – Sebuah pohon penyelesaian yang dimulai dengan akar A, E, H, D, G, kemudian dicari kota-kota lainnya. Gambar diolah sendiri dengan bantuan situs draw.io.

Kemudian, akan dilakukan analisis terhadap kota-kota selain A, E, H, D, G, dan C. Pilih kota yang memiliki nilai boundary function terkecil/sama dengan kota C. Dengan melakukan analisis terhadap jarak kota-kota yang lain, didapatkan bahwa kota selanjutnya yang akan kita kunjungi adalah kota F (boundary function = 592).

Berikut ini adalah pohon dengan akar "A", daun "E", daun "H", daun "D", daun "G", daun "C", daun "F", dengan cabang lanjutan yaitu daun-daun sisanya.

Rute saat ini : AEHDGCF



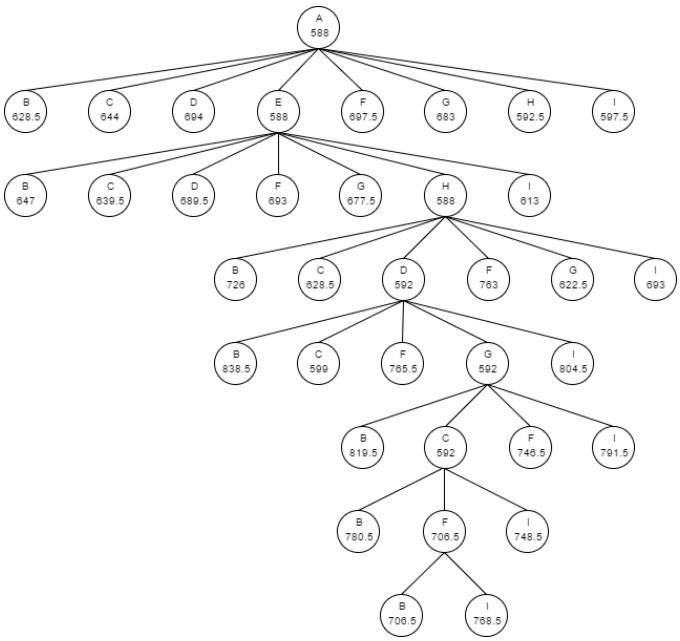
Gambar 9 – Sebuah pohon penyelesaian yang dimulai dengan akar A, E, H, D, G, C, kemudian dicari kota-kota

lainnya. Gambar diolah sendiri dengan bantuan situs draw.io.

Kemudian, akan dilakukan analisis terhadap kota-kota selain A, E, H, D, G, C, dan F. Pilih kota yang memiliki nilai boundary function terkecil/sama dengan kota F. Dengan melakukan analisis terhadap jarak kota-kota yang lain, didapatkan bahwa kota selanjutnya yang akan kita kunjungi adalah kota B (boundary function = 706.5).

Berikut ini adalah pohon dengan akar "A", daun "E", daun "H", daun "D", daun "G", daun "C", daun "F", daun "B", dengan cabang lanjutan yaitu daun-daun sisanya.

Rute saat ini : AEHDGCFB

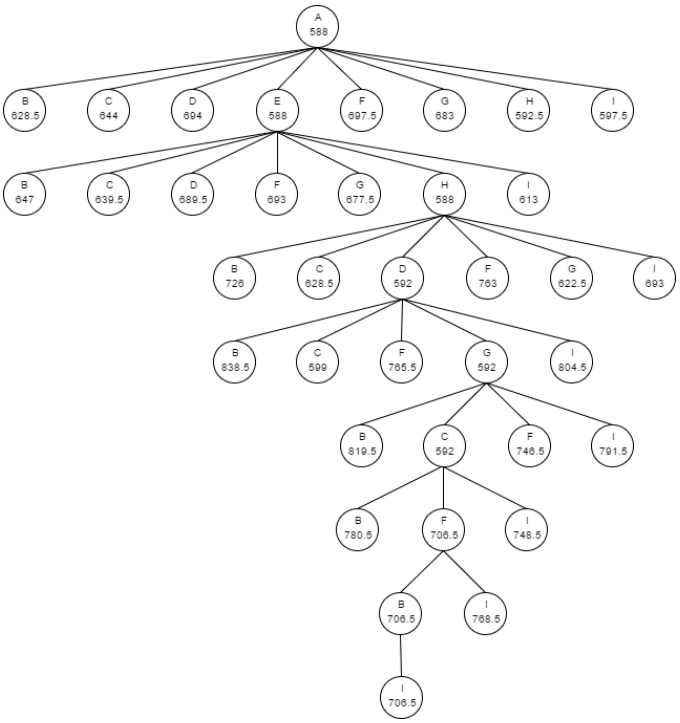


Gambar 9 – Sebuah pohon penyelesaian yang dimulai dengan akar A, E, H, D, G, C, F, kemudian dicari kota-kota lainnya. Gambar diolah sendiri dengan bantuan situs draw.io.

Kemudian, akan dilakukan analisis terhadap kota-kota selain A, E, H, D, G, C, F, dan B. Pilih kota yang memiliki nilai boundary function terkecil/sama dengan kota B. Dengan melakukan analisis terhadap jarak kota-kota yang lain, didapatkan bahwa kota selanjutnya yang akan kita kunjungi adalah kota I (boundary function = 706.5).

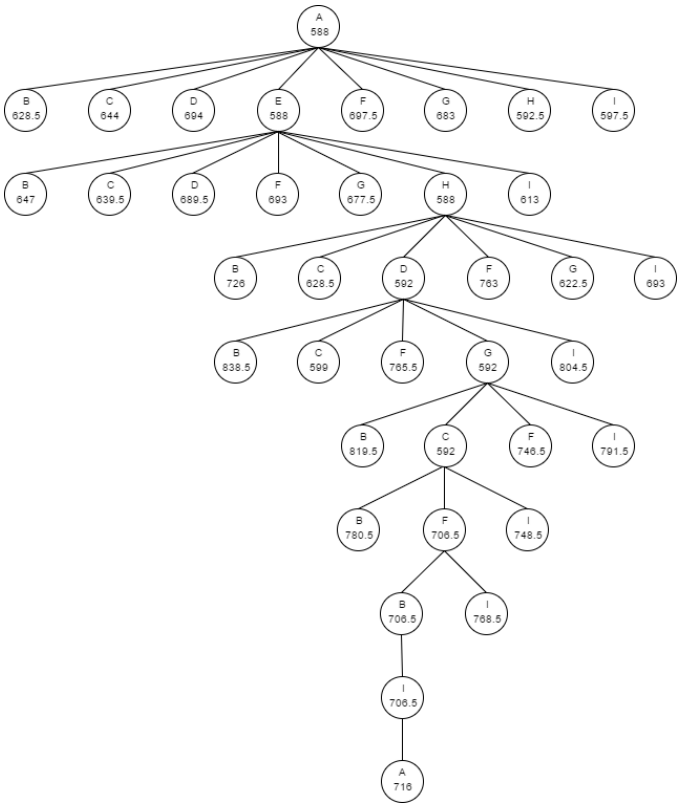
Berikut ini adalah pohon dengan akar "A", daun "E", daun "H", daun "D", daun "G", daun "C", daun "F", daun "B", daun "I", dengan cabang lanjutan yaitu daun-daun sisanya.

Rute saat ini : AEHDGCFBI



Gambar 10 – Sebuah pohon penyelesaian yang dimulai dengan akar A, E, H, D, G, C, F, B, kemudian dicari kota-kota lainnya. Gambar diolah sendiri dengan bantuan situs draw.io.

Yang terakhir, tentu saja, mengembalikan I ke A. Tetapi pada proses itu, tidak perlu lagi dilakukan perhitungan dengan pohon. Rute yang terbentuk adalah AEHDGCFBI, yaitu Bandung – Cirebon – Sukabumi – Bogor – Depok – Bekasi – Cimahi – Banjar – Tasikmalaya – Bandung, dengan total rute 718 km.



Gambar 11 – Sebuah pohon lengkap yang menunjukkan kemungkinan penyelesaian dari rute Gambar diolah sendiri menggunakan situs draw.io

IV. Analisis dan Perbandingan Kemampuan Algoritma

A. Waktu Penyelesaian

Algoritma Brute Force adalah algoritma yang memakan waktu cukup lama. Hal ini dikarenakan terdapat sangat banyak kemungkinan penyelesaian yang harus diperiksa satu persatu oleh algoritma tersebut (dalam soal ini, kita akan memeriksa total 20160 kemungkinan penyelesaian jawaban). Sehingga, kompleksitas algoritma ini mencapai O(n!). Namun, meskipun memakan waktu yang lama, solusi pasti dapat ditemukan. Dapat dilihat, jumlah penyelesaian dari algoritma ini adalah sebanyak

$$Penyelesaian = \frac{1}{2}(n-1)!$$

Sedangkan, untuk Algoritma Branch and Bound dalam bentuk Bobot Tur Lengkap, jumlah perhitungan jauh lebih sedikit. Hal ini disebabkan karena hanya sedikit perhitungan dan perbandingan yang dilakukan. Dalam hal ini, hanya terdapat sedikit pertambahan. Sehingga, waktu yang dibutuhkan untuk menyelesaikan persoalan ini jauh lebih cepat daripada penyelesaian menggunakan algoritma Brute Force.

B. Alokasi Memori

Algoritma Brute Force tidak menggunakan rekursi, karena dia hanya benar-benar menghitung berdasarkan apa yang diminta, tanpa memperhatikan rekursifitasnya.

Sedangkan, Branch and Bound melakukan perhitungan dengan cara memperhatikan rekursi. Apabila diperhatikan, dalam perhitungan menggunakan rekursif, alokasi memori yang digunakan banyak. Selain itu, jumlah simpul yang dibangkitkan juga banyak. Hal ini tentu saja menyebabkan pemakaian memori yang boros.

Selain itu, yang membuat Algoritma Branch and Bound dengan representasi bobot tur lengkap ini memakan memori yang banyak dikarenakan struktur data pohon yang memiliki maksimal 8 buah simpul yang diturunkan.

C. Hasil Perhitungan

Meskipun memakan waktu yang lama dan dianggap tidak efektif, Algoritma Brute Force dipastikan memberikan hasil yang optimal. Hal ini dikarenakan Brute Force menguji satu persatu kemungkinan penyelesaian, sehingga pasti didapatkan suatu angka yang terkecil.

Sedangkan, pada Algoritma Branch and Bound dengan representasi Bobot Tur Lengkap, tidak selalu menjamin didapatkannya suatu angka yang benar-benar sesuai. Hal ini disebabkan karena perhitungan dari algoritma ini selalu beranjak dari dua nilai terkecil, serta memperhatikan kondisi cost-nya. Sehingga, bisa saja terjadi suatu kondisi dimana cost dari suatu simpul yang ditimbulkan paling kecil, namun apabila di back-track, bukan menjadi solusi yang paling optimal.

Perhitungan di dalam makalah ini menghasilkan dua buah hasil yang sama, dengan menggunakan dua algoritma yang berbeda. Untuk melakukan perhitungan, dihitung terlebih dahulu menggunakan algoritma Brute Force. Setelah mendapatkan suatu angka yang dipastikan merupakan rute paling pendek, dilakukan lagi perhitungan menggunakan algoritma Branch and Bound, dengan representasi bobot tur lengkap. Setelah dilakukan backtracking dari simpul-simpul yang ditimbulkan, didapatkan kesimpulan bahwa kedua hasil tidak memiliki perbedaan.

V. Pengembangan dan Aplikasi Lain

Kedua algoritma, baik Brute Force maupun Branch and Bound, dapat dipergunakan untuk menyelesaikan persoalan-persoalan TSP yang lain. Pengembangan yang dapat dilakukan dalam persoalan TSP ini adalah dengan menambah jumlah kota yang hendak dihitung jaraknya. Hasil perhitungan dalam makalah ini yang menggunakan 9 kota dapat menghasilkan suatu angka eksak. Kedepannya, apabila kita hendak menggunakan kedua algoritma tersebut, kita dapat menambah jumlah kota, atau meninjau ibukota-ibukota kabupaten yang terdapat di Jawa Barat. Hal ini akan membuat perhitungan semakin detail.

Hal lain yang dapat dikembangkan adalah dengan menambahkan variabel waktu, serta memperhatikan faktor-faktor lain yang bukan semata-mata jalan, misalkan kemacetan dan waktu tak terduga. Apabila kita berhasil menangani kedua faktor tersebut, perhitungan akan menjadi semakin akurat.

Namun, variabel waktu ini harus terus-menerus diperbarui, tidak seperti jarak yang tidak selalu diperbarui. Hal ini dikarenakan faktor-faktor tidak terduga seperti kemacetan, kecelakaan, dan lain-lain.

VI. Kesimpulan dan Saran

A. *Hasil Perhitungan*

Berdasarkan perhitungan dengan menggunakan kedua algoritma, didapatkan bahwa rute tercepat yang dapat ditempuh bila hendak melewati kota-kota di Jawa Barat adalah sepanjang 718 km.

Rute yang ditempuh adalah Bandung – Cirebon – Sukabumi – Bogor – Depok – Bekasi – Cimahi – Banjar – Tasikmalaya – Bandung.

B. *Perbandingan Algoritma*

Berdasarkan analisis seluruh algoritma yang diberikan, algoritma yang lebih baik dipergunakan adalah algoritma Branch and Bound dengan representasi Bobot Tur Lengkap, karena waktu yang digunakan jauh lebih cepat, dan komputer masih dapat menangani jumlah memori yang dialokasikan.

C. *Saran*

Supaya hasil perhitungan algoritma menjadi semakin jelas, perhitungan dapat dibandingkan dengan menggunakan algoritma pencarian TSP yang lain, seperti Pemrograman Dinamis. Selain itu, saran-saran pengembangan yang sudah penulis sampaikan pada Bab V dapat diimplementasikan dalam percobaan-percobaan analisis kedepannya.

Ucapan Terima Kasih

Penulis ingin mengucapkan terima kasih kepada Tuhan Yang Maha Esa, karena rahmat dan berkat-Nya lah, penulis dapat menyelesaikan karya tulis ini. Selain itu, penulis juga ingin mengucapkan terima kasih kepada dosen IF2211 Bapak Dr. Rinaldi Munir atas bimbingannya selama satu semester dalam menjalani mata kuliah IF2211. Penulis juga ingin mengucapkan terima kasih kepada kedua orang tua penulis, serta rekan-rekan penulis yang tidak dapat penulis tuliskan satu persatu atas dukungan yang diberikan selama keberjalanan penulisan makalah ini.

References

[1] Munir, Rinaldi, *Diktat Kuliah IF2211 Strategi Algoritma*, Program Studi Teknik Informatika, STEI, ITB 2009

[2] Wikipedia, *Jawa Barat*, https://id.wikipedia.org/wiki/Jawa_Barat, diakses 16 Mei 2017

[3] Wikipedia, *Daftar Kabupaten dan Kota di Jawa Barat*, https://id.wikipedia.org/wiki/Daftar_kabupaten_dan_kota_di_Jawa_Barat, diakses 16 Mei 2017

[4] http://illuminations.nctm.org/Activity.aspx?id=3550, waktu akses 16 Mei 2017

[5] draw.io, waktu akses 16, 17, dan 18 Mei 2017

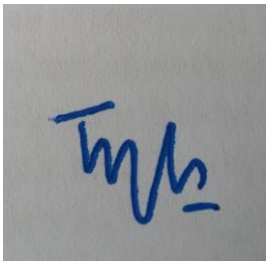
[6] Research Gate, *The Traveling Salesman Problem*, waktu akses 16 Mei 2017

[7] http://www.distancefromto.net/, waktu akses 16 dan 17 Mei 2017

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 Mei 2017



Hutama Tefotuho Hulu/13515045