Pemanfaatan Algoritma Branch & Bound untuk Pencarian Rute Angkot

Ilma Alifia Mahardika (*Author*)
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
ilma.mahardika@gmail.com

Pencarian rute angkot dari satu tempat ke tempat lain memerlukan implementasi algoritma yang berbeda daripada pencarian rute jalan biasa. Jalur yang dipilih tidak hanya mempertimbangkan jarak tempuh terpendek, tetapi juga mempertimbangkan apakah jalur tersebut dilewati oleh rute angkot atau tidak. Oleh karena itu, algoritma Branch & Bound akan diterapkan untuk permasalahan ini.

Kata Kunci—rute; angkot; Branch & Bound; optimal; API

I. PENDAHULUAN

Angkot merupakan salah satu moda transportasi yang umum digunakan oleh masyarakat Indonesia sejak zaman dahulu. Hampir semua daerah di Indonesia memiliki transportasi ini. Bahkan banyak daerah yang memiliki lebih dari satu trayek. Hal tersebut akan menimbulkan kebingungan, terutama bagi pendatang baru, ketika ingin pergi dari satu tempat ke tempat lain menggunakan transportasi angkot. Pencarian rute angkot yang paling pendek diperlukan agar perjalanan yang dilalui pengguna transportasi ini memakan waktu yang singkat.

II. TEORI DASAR

A. Algoritma Branch & Bound

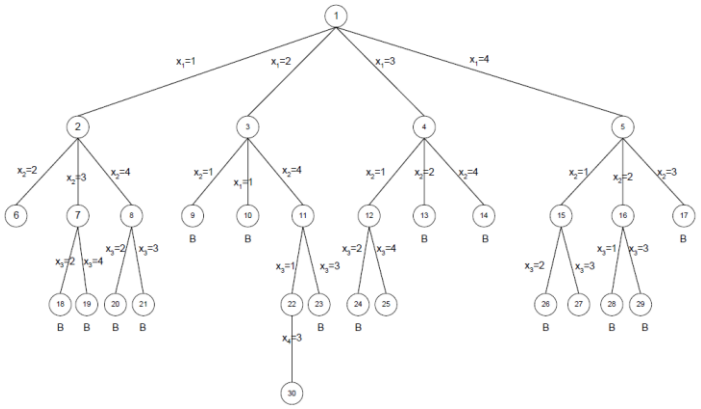
Algoritma Branch & Bound merupakan salah satu strategi pencarian solusi masalah dengan memanfaatkan pencarian di dalam ruang solusi secara sistematis. Ruang solusi adalah himpunan dari semua status yang menyatakan solusi dari persoalan.

Algoritma Branch and Bound membagi suatu persoalan menjadi sub persoalan yang lebih kecil dan mengarah ke solusi. Caranya dengan pencabangan (branching) dan melakukan pembatasan (bounding) untuk mencapai solusi optimal. Oleh karena itu algoritma ini bernama Branc and Bound.

Pencabangan (branching) yaitu proses membentuk persoalan ke dalam bentuk struktur pohon pencarian atau ruang solusi. Proses Pencabangan dilakukan untuk membangun semua cabang pohon yang menuju solusi, sedangkan proses pembatasan dilakukan dengan menghitung estimasi nilai (*cost*) simpul dengan memperhatikan batas. algoritma Branch and Bound pertama kali dikemukakan oleh A. H. Land dan A. G. Doig pada tahun 1960. [2]

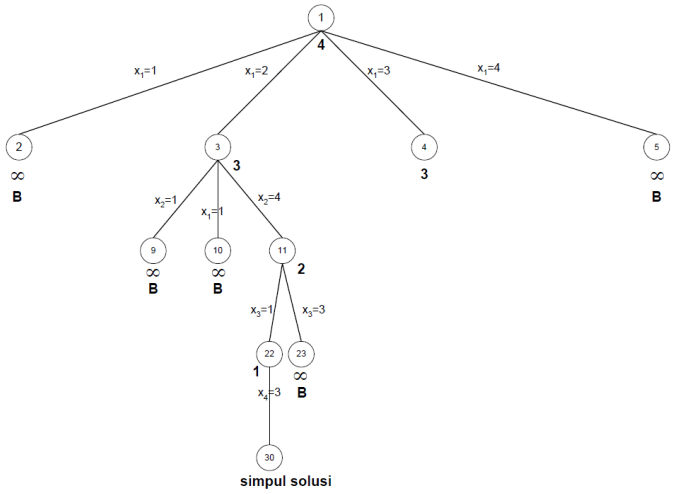
Skema dasar yang digunakan pada algoritma ini adalah BFS (*Breadth-first search*). BFS atau pencarian melebar yaitu melakukan ekspansi ruang solusi berdasarkan urutan pembangkitannya, atau biasa disebut FIFO (*first in first out*).

Yang membedakan algoritma Branch & Bound dan algoritma BFS adalah algoritma Branch & Bound digunakan untuk persoalan optimasi. Persoalan optimasi yaitu meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan atau *constraint* persoalan terkait. Setiap simpul pada ruang solusi di-*assign* dengan sebuah nilai ongkos (*cost*). Oleh karena itu pembangkitan ruang solusi pada algoritma Branch & Bound bukan lagi berdasarkan urutan pembangkitan atau FIFO, melainkan berdasarkan nilai ongkos yang paling kecil (*least cost search*). Berikut ini adalah perbandingan ruang solusi algoritma BFS murni dengan Branch & Bound:



Gambar 2.1 Pohon ruang status persoalan 4-ratu dengan algoritma BFS murni

Sumber: Diktat Rinaldi Munir



Gambar 2.2 Pohon ruang status persoalan 4-ratu dengan algoritma Branch & Bound

Sumber: Diktat Rinaldi Munir

Nilai yang di-*assign* pada setiap simpul menyatakan taksiran ongkos termurah lintasan dari simpul ke i ke simpul solusi. Nilai taksiran tersebut dapat berupa batas atas (*upper bound*) atau batas bawah (*lower bound*). Nilai ini bisa ideal apabila letak simpul solusi telah diketahui, namun umumnya pada banyak persoalan, letak simpul solusi tidak diketahui, nilai batas untuk setiap simpul hanya berupa taksiran.

Hal penting lain mengenai algoritma Branch & Bound adalah simpul yang tidak mengarah ke solusi akan dimatikan atau dipangkas. Proses pemangkasan ini dengan cara memanfaatkan suatu fungsi pembatas.

Berikut adalah fungsi untuk menghitung taksiran ongkos (*cost*):

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

$c(i)$: ongkos untuk simpul i

$f(i)$: ongkos mencapai simpul i dari akar

$g(i)$: ongkos mencapai simpul tujuan dari simpul i

Untuk permasalahan yang lebih kompleks, persoalan digambarkan dengan matriks A ukuran N × N agar dapat dipilih simpul mana yang akan dieksplorasi. Simpul yang dieksplorasi adalah simpul dengan nilai batas terkecil. Nilai batas didapatkan dari reduksi baris dan kolom matriks A yang merepresentasikan graf.

Proses reduksi dilakukan dengan mengurangi nilai *cij* pada baris atau kolom dengan nilai *cij* terkecil pada baris atau kolom tersebut, sedemikian sehingga didapatkan matriks tereduksi $A(t)$ dengan sebuah nilai nol pada setiap baris dan kolom.

Sebuah matriks dapat dikatakan tereduksi jika setiap kolom dan barisnya mengandung minimal satu nilai nol dan semua nilai lainnya bukan negatif. Selanjutnya total nilai pereduksi menjadi nilai batas simpul akar. Untuk setiap simpul anak yang dibangkitkan dengan mengunjungi $A(i,j)$, dilakukan reduksi matriks $A(t)$ untuk mendapatkan matriks tereduksi $A(j, \textit{simpul awal})$ diubah menjadi ∞.

Berdasarkan persamaan fungsi heuristik untuk menghitung nilai estimasi yang telah dijelaskan sebelumnya, maka nilai batas simpul anak dihitung dengan rumus:

$$\hat{c}(S)=\hat{c}(R)+A(i,j)+r$$

$\hat{c}(S)$: nilai perjalanan minimum yang melalui simpul S, dimana S adalah anak dari simpul R

$\hat{c}(R)$: Nilai perjalanan minimum yang melalui simpul R, dimana R adalah simpul akar

$A(i,j)$: bobot sisi (i,j) pada matriks tereduksi

r : jumlah semua pereduksi pada proses reduksi matriks untuk simpul S

Berikut ini adalah langkah-langkah yang dilakukan untuk penerapan algoritma Branch and Bound: [1]

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (*goal node*), maka solusi telah ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. Stop.

3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai $c(i)$ paling kecil. Jika terdapat beberapa simpul i yang memenuhi, pilih satu secara sembarang.

4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan, stop. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.

5. Untuk setiap anak j dari simpul i, hitung $c(j)$, dan masukkan semua anak-anak tersebut ke dalam Q.

6. Kembali ke langkah 2.

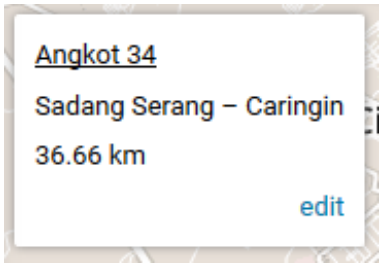
Beberapa persoalan yang dapat diselesaikan menggunakan algoritma Branch and Bound yaitu persoalan N-Ratu (peletakan ratu pada petak-petak papan berukuran N × N sedemikian sehingga tidak ada dua ratu atau lebih yang terletak pada satu baris yang sama, atau satu kolom yang sama, atau satu diagonal yang sama), persoalan 15-puzzle (sebuah permainan papan untuk mengurutkan lima belas angka dengan cara menggeser 'kotak kosong'), persoalan TSP (*Travelling Sales Person*), dan persoalan *knapsack.*

B. Rute Angkot

Rute angkot memiliki jalur yang bermacam-macam. Selain itu tidak semua jalan raya dilewati angkot. Hal ini menjadi suatu tantangan dalam menemukan angkot mana yang paling efektif dari satu tempat ke tempat yang lain. Selain itu jalur angkot yang tumpang tindih juga menjadi tantangan lain dalam menentukan angkot mana yang terbaik.

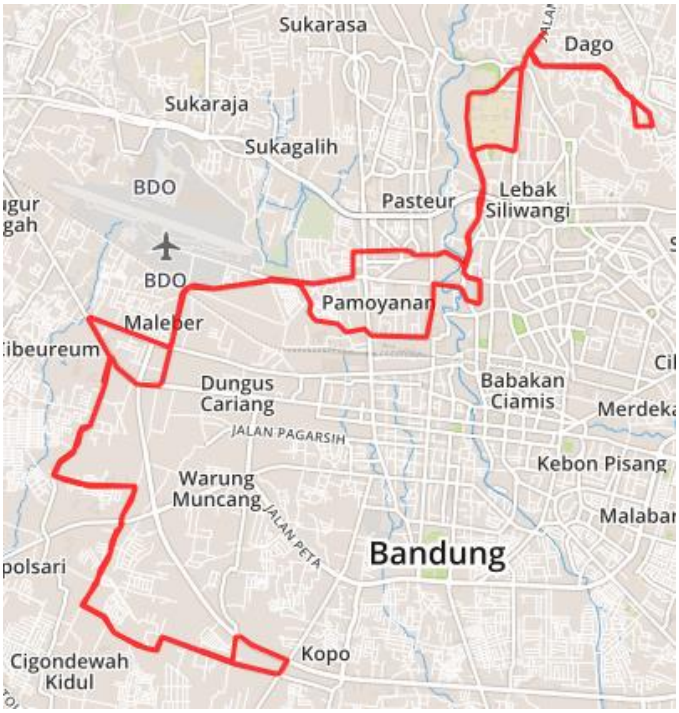
Data rute angkot dapat diambil pada website https://angkot.web.id/. [3]

Pada website tersebut terdapat informasi mengenai tujuan awal, tujuan akhir, serta gambar rute angkot terkait. Berikut ini salah satu contohnya:



Gambar 2.3 Data informasi rute angkot

Sumber: https://angkot.web.id/



Gambar 2.4 Rute angkot pada peta

Sumber: https://angkot.web.id/

C. Google Maps API

API atau *Application Programming Interface* merupakan sekumpulan *method* dan *tools* yang dapat digunakan untuk membangun sebuah perangkat lunak. [6]

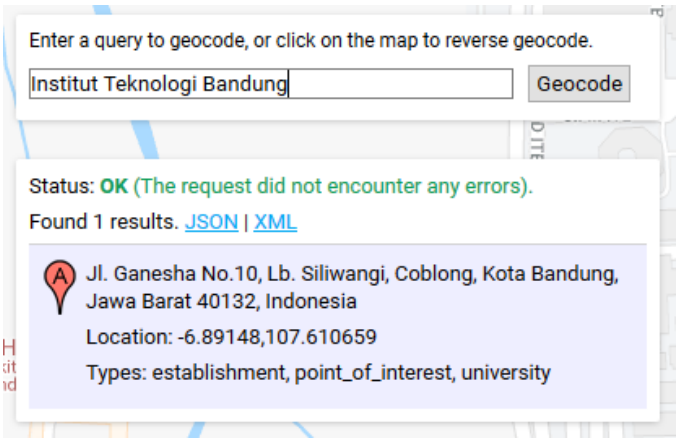
Google Maps API merupakan fungsi yang dikembangkan Google untuk para developer yang ingin mengintegrasikan fasilitas Google Maps dengan program yang dikembangkan. Jenis API yang sesuai untuk persoalan pencarian rute angkot yang efektif yaitu Places API dan Geocoding API. [3]

Places API akan memudahkan untuk menemukan tempat yang menjadi *point of interest*. Tempat tersebut dapat dideteksi data-data pentingnya seperti nama, alamat, penilaian, ulasan, dan kontak. Nama dan alamat akan diperlukan dalam penyelesaian persoalan pencarian rute angkot ini. Penggunaan Google Places API di sini sebagai penyederhanaan atau abstraksi dari titik awal dan titik tujuan. Beberapa *requests* yang cukup penting dan bisa dilakukan adalah *Place Search* (mencari tempat tertentu) dan *Place Autocomplete* (memberikan saran lokasi berdasarkan input pengguna). [4]

Gambar 2.5 Contoh Pemakaian Places API

Sumber: https://cloud.google.com/maps-platform/places/

Geocoding API memiliki dua *requests* yang bisa dilakukan *geocoding dan reverse geocoding*. *Geocoding* berguna untuk melakukan konversi data alamat suatu tempat menjadi koordinat geografis. *Reverse geocoding* melakukan hal sebaliknya. [3]



Gambar 2.6 Contoh Pemakaian Geocoding API (*Geocoding*)

Sumber: https://developers.google.com/maps/documentation/geocoding/

Gambar 2.7 Contoh Pemakaian Geocoding API (*Reverse Geocoding*)

Sumber: https://developers.google.com/maps/documentation/javascript/examples/geocoding-reverse

III. Metode Pemecahan Masalah

Tantangan yang terdapat dalam persoalan pencarian rute angkot yang paling efektif telah disebutkan pada bagian teori dasar sebelumnya, yaitu rute angkot tidak melalui setiap jalan raya dan beberapa rute angkot sangat mungkin untuk saling tumpang tindih jalur.

Persoalan ini dapat didekomposisi menjadi dua bagian:

- Jalan yang Tidak Dilewati Rute Angkot

  Maksud dari jalan yang tidak dilewati rute angkot adalah sebuah jalan tetapi tidak bisa menjadi tempat untuk naik angkot karena tidak ada rute angkot manapun pada jalan tersebut.

  Pada kasus ini maka pengguna harus berjalan kaki untuk menuju titik yang dilewati rute angkot.

- Jalan yang Dilewati Rute Angkot

  Maksud dari jalan yang dilewati rute angkot adalah sebuah jalan yang bisa menjadi tempat untuk naik angkot.

  Pada kasus ini maka pengguna menaiki angkot yang sudah terpilih.

Untuk setiap simpul pada ruang solusi yang dibangkitkan merupakan persimpangan jalan.

A. Jalan yang Tidak Dilewati Rute Angkot

Untuk fungsi pembatas pada bagian ini ditetapkan sebagai berikut:

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

$c(i)$: ongkos untuk simpul i

$f(i)$: ongkos mencapai simpul i dari akar

$g(i)$: ongkos mencapai simpul yang dilewati rute angkot

Keterangan lebih lanjut:

- Simpul yang dilewati rute angkot akan memiliki nilai $g(i)$ = 0 (nol), merujuk ke nilai paling minimum. Simpul ini diberi nilai minimum karena berpotensi besar menjadi simpul solusi.

- Simpul selain simpul pada poin sebelumnya akan memiliki nilai $g(i)$ = jarak ke simpul yang dilewati rute angkot terdekat, apabila simpul akhir ditemukan maka nilainya merupakan jarak ke simpul akhir tersebut dan jalur yang telah dilalui sebelumnya merupakan solusi dari persoalan.

B. Jalan yang Dilewati Rute Angkot

Untuk fungsi pembatas pada bagian ini ditetapkan sebagai berikut:

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

$c(i)$: ongkos untuk simpul i

$f(i)$: ongkos mencapai simpul i dari akar

$g(i)$: ongkos mencapai simpul tujuan, dapat berupa titik akhir atau titik untuk melanjutkan perjalanan selanjutnya

Keterangan lebih lanjut:

- Jika rute angkot tidak berhenti pada simpul tujuan maka nilai $g(i)$ = jarak dari simpul i ke simpul tujuan. Simpul yang akhirnya terpilih kemudian menjadi simpul akar untuk dekomposisi persoalan bagian pertama. Metode pada bagian jalan yang tidak dilewati rute angkot kembali diulang.

IV. IMPLEMENTASI

Untuk titik awal dan akhir dapat memanfaatkan Places API untuk menyederhanakan persoalan dengan cara hanya memperhitungkan tempat-tempat yang menjadi *point of interest.*

Tahapan pembangkitan ruang solusi dapat disimpan dalam struktur data array. Untuk jalur solusinya sendiri disimpan di struktur data yang terpisah.

Fungsi dan prosedur yang diperlukan diuraikan sebagai berikut:

A. tambahPenanda(titik)

Sebuah fungsi yang mengembalikan informasi mengenai sebuah titik pada peta. Fungsi ini akan memanfaatkan Geocoding API serta Places API.

B. hitungJarak(titik1, titik2)

Sebuah fungsi yang mengembalikan nilai bertipe double. Algoritma untuk menghitung jarak tempuh. Fungsi ini memanfaatkan prosedur tambahPenanda() untuk mendapatkan informasi koordinat geografisnya lalu diproses untuk mendapatkan jarak tempuh.

C. hitungPembatasA()

Sebuah fungsi yang mengembalikan nilai bertipe double. Algoritmanya merupakan implementasi rumus fungsi pembatas untuk kasus jalan yang tidak dilewati rute angkot. Ongkos mencapai simpul i dari akar atau $f(i)$ dapat diitung dengan memanfaatkan Geocoding API. Ongkos mencapai simpul yang dilewati rute angkot atau $g(i)$ dapat diitung dengan Geocoding API yang dipadukan dengan informasi rute angkot yang diperoleh dari website https://angkot.web.id/.

D. hitungPembatasB()

Sebuah fungsi yang mengembalikan nilai bertipe double. Algoritmanya merupakan implementasi rumus fungsi pembatas pada bab III bagian B. Ongkos mencapai simpul i dari akar atau *f*(*i*) dapat diitung dengan memanfaatkan Geocoding API sama seperti hitungPembatasA(). Ongkos mencapai simpul tujuan (titik akhir) atau *g*(*i*) dapat diitung dengan Geocoding API yang dipadukan dengan informasi rute angkot yang diperoleh dari website https://angkot.web.id/. Ada penambahan kasus ketika simpul tujuan tidak ada, maka dipakai sebuah fungsi heuristik lain yaitu jarak antara simpul i dengan simpul tujuan.

E. cekSampai()

Sebuah fungsi yang mengembalikan nilai bertipe boolean. Boolean ini untuk pengecekan apakah pembangkitan ruang solusi sudah mencapai titik akhir.

F. cariRute(titik_awal, titik_akhir)

Sebuah prosedur yang akan memproses parameter titik_awal dan titik_akhir untuk ditemukan jalur angkot yang

paling optimal. Prosedur ini akan meng-iterasi ruang solusi yang telah dibangkitkan untuk diitung fungsi pembatasnya sesuai dengan jenis simpulnya (bab III). Selain itu fungsi boolena cekSampai() juga perlu dipanggil.

G. Program Utama

Program utama berisi implementasi struktur data yang dipilih untuk ruang solusi, struktur data untuk simpul-simpul, antamuka untuk input dan output bagi pengguna, dan pemanggilan fungsi atau prosedur yang telah diimplementasikan sebelumnya.

V. ANALISIS

Tantangan yang terdapat dalam persoalan pencarian rute angkot yang paling efektif telah disebutkan pada bagian teori dasar sebelumnya, yaitu rute angkot tidak melalui setiap jalan raya dan beberapa rute angkot sangat mungkin untuk saling tumpeng tindih jalur.

Dengan menerapkan algoritma Branch and Bound maka jalur angkot yang tidak mengarah ke simpul akhir akan otomatis dimatikan. Karena hal tersebut metode pemecahan masalah dengan algoritma Branch and Bound akan memberikan kompleksitas waktu yang lebih singkat.

VI. KESIMPULAN

Persoalan pencarian rute angkot yang paling pendek sehingga menjadikan waktu tempuh semakin singkat dapat dipecahkan dengan sebuah strategi yaitu pemanfaatan algoritma Branch & Bound dengan fungsi pembatas sebagaimana yang telah diuraikan pada bagian metode pemecahan masalah.

REFERENSI

[1] Munir, Rinaldi. 2018. *Diktat Kuliah IF2211 Strategi Algoritma. Institut Teknologi Bandung: Bandung*.

[2] Mussafi, Noor Saif Muhammad. 2014. *Aplikasi Algoritma Branch and Bound untuk Kebakaran Kota Yogyakarta*.

[3] https://projectkiri.github.io/#about-kiri (diakses tanggal 07 Mei 2018 pukul 14.46 WIB)

[4] https://cloud.google.com/maps-platform/ (diakses tanggal 07 Mei 2018 pukul 15.02 WIB)

[5] NUM-MLK-RN. 2018. "Algoritma Branch & Bound" dalam Bahan Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika – STEI ITB.

[6] https://www.w3schools.com/graphics/google_maps_intro.asp (diakses tanggal 12 Mei 2018)

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, 07 Mei 2018

Ilma Alifia Mahardika
13516036