Pendekatan Branch and Bound secara Breadth First dan Depth First

Raden Fitri Indriani[1], Dini Armyta[2], Rika Safrina[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if13020@students.if.itb.ac.id[1],
if13025@students.if.itb.ac.id[2], if13053@students.if.itb.ac.id[3]

Abstrak

Suatu masalah terkadang lebih mangkus jika dikerjakan dengan suatu metode walaupun secara keseluruhan metode tersebut sangatlah naif. Hal itu disebabkan karena setiap metode atau algoritma memiliki ciri khas nya masing-masing. Kali ini Kami ingin membahas keunikan dari algoritma Branch and Bound yang dapat mengimplementasikan dua buah algoritma *search* (traversal) yaitu *Breadth First* Search dan *Depth First Search*. Walaupun lingkup permasalahan yang dikuasai oleh kedua metode traversal ini sama, namun ada faktor lain yang membuat salah satu dari keduanya tersebut lebih mangkus.

***Kata kunci**: Branch and Bound, Breadth First Search, Depth First Search*

1. Pendahuluan

Seorang *programmer* dituntut untuk selalu menghasilkan program yang mangkus dan sangkil dalam menyelesaikan suatu masalah. Untuk itu, alangkah bijaknya jika seorang programmer mengetahui berbagai macam algoritma, sehingga dapat menggunakan algoritma yang tepat untuk menyelesaikan suatu masalah. Karena biasanya, setiap algoritma diimplementasikan dengan suatu masalah tertentu sesuai dengan keunikan yang dimilikinya.

Begitu pula dengan algoritma *Branch and Bound*, yang dapat menggunakan metode *Breadth First Search (BFS)* dan *Depth First Search (DFS).* Walaupun kedua metode tersebut memiliki kelebihan masing-masing, namun selalu ada yang lebih baik.

2. Branch and Bound

Algoritma *Branch and Bound*, atau yang biasa disingkat dengan B&B merupakan metode pencarian solusi di dalam ruang solusi secara sistematis, yang diimplementasikan ke dalam suatu pohon ruang status dinamis.

Setiap status tersebut diberikan biaya (*cost*) dalam pencapaian solusi. Semakin kecil *cost*, semakin cepat solusi ditemukan. Keunikan tersendiri dari algoritma B&B tersebut itulah yang membuat penyelesaian masalah menggunakan algoritma ini lebih mangkus dibandingkan *Brute Force*, khususnya untuk permasalahan kombinatorial, seperti *Travelling Salesperson Problem* (TSP), *Quadratic Assignment Problem* (QAP), *Job Shop Schedulling Problem*, dan *15-Puzzle*.

Rumus perhitungan *cost* dibentuk secara heuristik, tanpa aturan tertentu.

2.3 Pembangkitan Status Secara Breadth First Search

Prinsip BFS adalah selalu membangkitkan semua anak dari simpul pada pohon ruang status. Metode BFS pada *Branch and Bound* sering disebut juga *Best First Search* (BeFS).

Pada implementasi *Branch and Bound* :

1. Bangkitkan seluruh anak dari simpul awal
2. Masukkan setiap simpul tersebut ke dalam antrian sesuai urutan *cost*-nya dari kecil ke besar
3. Keluarkan simpul yang terletak pada head antrian (simpul dengan *cost* terkecil)
4. Ulangi langkah 1

2.2 Pembangkitan Status Secara Depth First Search

Prinsip DFS adalah selalu membangkitkan anak pertama dari simpul pada pohon ruang solusi.

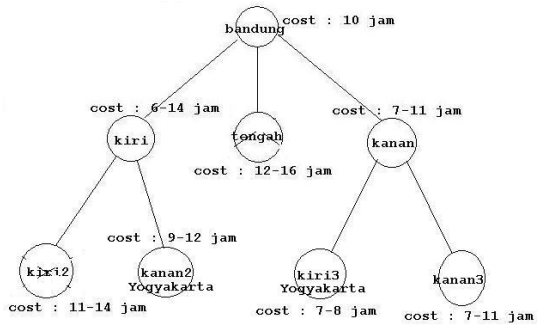
Pada implemetasi Branch and Bound :

1. *Cost* dari simpul awal (mis. x) akan menjadi *cost* patokan awal (g(x)).

2. Bangkitkan anak pertama yang masih hidup dari x (mis. y). Jika tidak ada lagi anak yang mungkin dibangkitkan, maka backtrack ke simpul bapaknya.
3. Jika g(x) ≥ g(y) maka ulangi langkah 1. Tapi jika tidak, matikan simpul y dan kembali ke simpul x (*backtrack*/runut balik), lalu ulangi langkah 2
4. Begitu seterusnya hingga menemukan status solusi

3. Contoh Aplikasi

Andaikan seorang warga Bandung hendak melakukan perjalanan dari Bandung ke Yogyakarta dengan kendaraan pribadi. Sebelumnya pengendara tersebut tidak pernah melakukan perjalanan tersebut, sehingga tidak tahu rute mana yang harus diambil agar dapat sampai ke Yogyakarta secepatnya.
Walaupun begitu, pengendara tersebut tahu bahwa perjalanan ke Yogyakarta tidak lebih dari 10 jam. Selain itu Ia punya peta yang menggambarkan tempo Bandung-Yogyakarta dapat ditempuh melalui jalan tersebut.

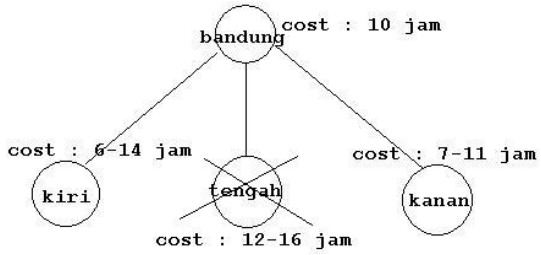


Peta Bandung-Yogyakarta

Pada persimpangan pertama, terdapat 3 arah ke Yogyakarta dengan waktu berbeda-beda.
Ke arah kiri : 6-14 jam
Ke arah tengah : 12-16 jam
Ke arah kanan : 7-11 jam

3.1 Penggunaan BFS

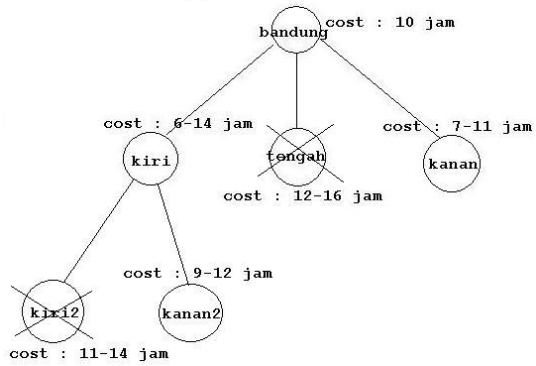
Jika kita menggunakan BFS, maka semua anak akan dibangkitkan, sebagai berikut.



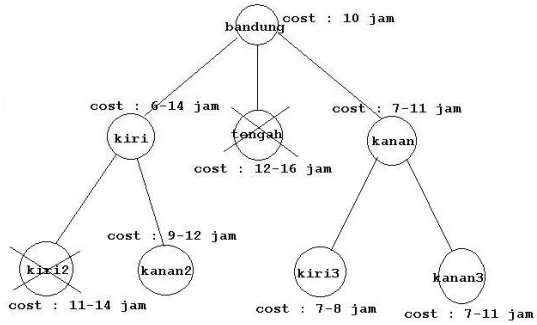
Simpul-simpul tersebut dimasukkan ke antrian dengan urutan [kiri, kanan] berdasarkan *cost* menaik. Sedangkan simpul tengah dijadikan simpul mati, karena tidak akan menghasilkan solusi (*cost* lebih besar daripada solusi yang dicari : 10 jam).

Selanjutnya, ambil kiri dari antrian (sehingga antrian menjadi [kanan]), bangkitkan anak-anak dari kiri. Yang berarti coba selidiki jalan ke arah kiri.
Di sana kita akan menemukan satu persimpangan lagi, sebagai berikut.
Ke arah kiri (kiri2) : 11-14 jam
Ke arah kanan (kanan2) : 9-12 jam

Masukkan simpul tersebut ke antrian sehingga antrian menjadi [kanan, kanan2]. Sedangkan kiri2 dijadikan simpul mati karena tidak akan menghasilkan solusi.



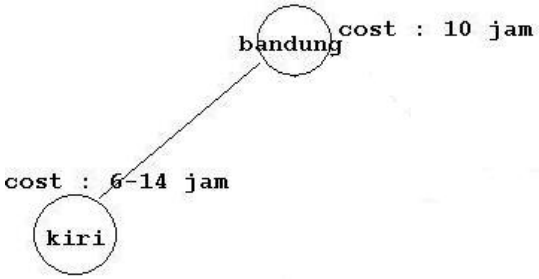
Kemudian, ambil head dari antrian, yaitu kanan (sehingga antrian menjadi [kanan2]). Bangkitkan anak-anak dari kanan, yang berarti telusuri jalan ke arah kanan.
Di sana kita akan menemukan suatu persimpangan.
Ke arah kiri (kiri3): 7-8 jam
Ke arah kanan (kanan3): 7-11 jam



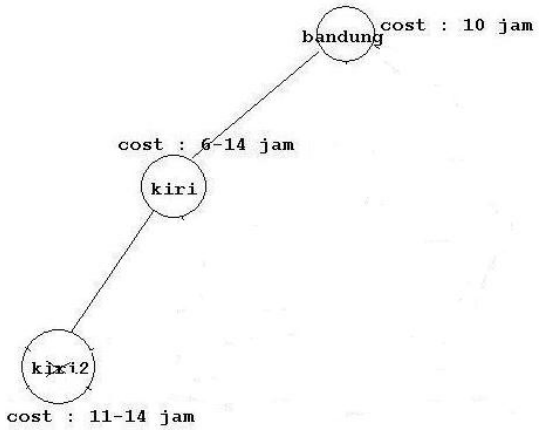
Masukkan semua simpul ke antrian sehingga antrian menjadi [kiri3,kanan2,kanan3].
Ambil head dari antrian, yaitu kiri3. Karena kiri3 adalah solusi (Bandung-Yogyakarta dalam waktu kurang dari atau sama dengan 10 jam), maka solusi pertama telah ditemukan.

3.2 Penggunaan DFS

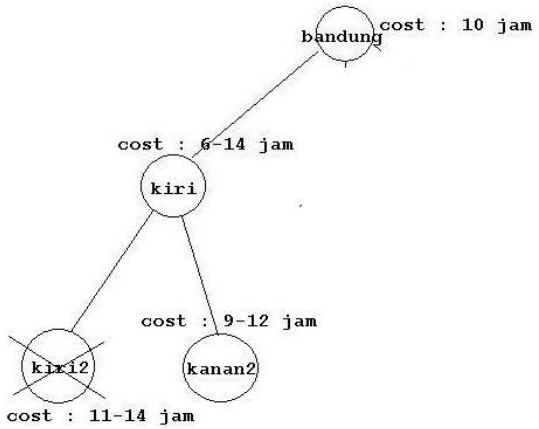
Jika kita menggunakan DFS, maka anak pertama akan dibangkitkan terlebih dahulu.



Kemudian, bangkitkan anak pertama dari kiri. *Cost* sekarang bernilai 6-14 jam.

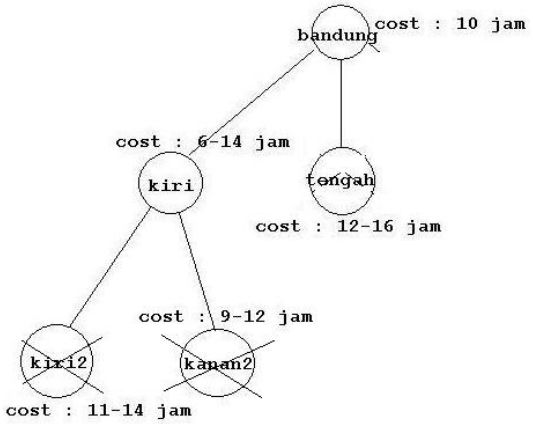


Karena *cost* anak pertama dari kiri (yaitu kiri2) lebih besar dibandingkan *cost* bapaknya, maka backtrack. Simpul kiri membangkitkan anak selanjutnya.

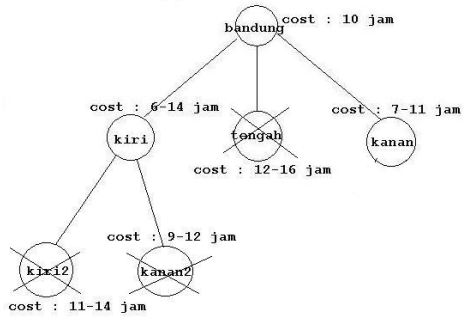


Karena *cost* simpul kanan2 lebih besar dibandingkan simpul bapaknya, maka harus backtrack ke simpul kiri. Karena simpul kiri tidak memiliki anak lagi yang mungkin dibangkitkan, maka backtrack lagi ke simpul awal, bandung.

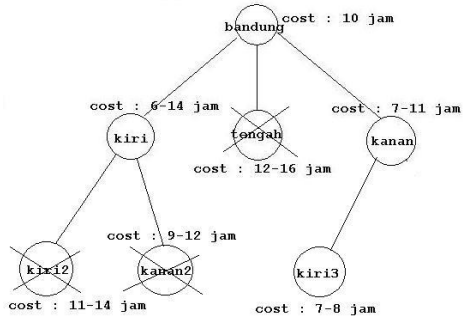
Kemudian, kita bangkitkan anak lain dari simpul Bandung, yaitu tengah.



Karena *cost* simpul tengah lebih besar dibandingkan *cost* bapaknya (bandung), maka backtrack lagi.
Kemudian, bangkitkan anak simpul bandung selanjutnya.



Kemudian, bangkitkan anak pertama dari simpul kanan, yaitu kiri3.



Karena kiri3 adalah solusi (Bandung-Yogyakarta dalam waktu kurang dari atau sama dengan 10 jam), maka solusi pertama telah ditemukan.

4. Kesimpulan

Penggunaan BFS dan DFS pada pendekatan Branch and Bound mempunyai kelebihan dan kekurangan masing-masing.

BFS membangkitkan seluruh anak yang mungkin dari suatu simpul dan memilih *cost* yang terbaik. Cara ini memberi kelebihan dalam menghemat waktu. Karena ia telah memikirkan masak-masak akan langkah yang ia ambil. Namun, metode ini lebih menghabiskan memori (ruang status).

Sedangkan DFS lebih ke arah untung-untungan. Jika solusi merupakan turunan dari anak pertama, maka metode ini akan sangat menghemat memori dan waktu, namun jika solusi terletak pada turunan anak terakhir, maka metode ini selain menghabiskan memori, juga akan sangat menghabiskan waktu.

Hingga tak heran bahwa banyak pakar algoritma lebih menyukai BFS daripada DFS yang sering disebut juga algortima malas (lazy algorithm) [2].

Daftar Pustaka

[1] Armand E. Prieditis, "Computational Intelligence Volume 14 Issue 2 Page 188 - May 1998", Department of Computer Science, University of California, Davis, 1998.

[2] J. Clausen and M. Perregaard, "Annals of Operations", Kluwer Academic Publisher, 1999.

[3] K. Malysiak and B. D. McKay, "Combinatorial Optimisation on the AP1000," Proc. Second Fujitsu-ANU CAP Workshop, Australian National University, 1991.

[4] K. Malysiak and B. D. McKay, "Extended Abstract - The Graph Isomorphism Problem: A Case of Parallelism in Backtrack Search", Proc. Fujitsu-ANU CAP Workshop, Australian National University, 1994.

[5] Munir, Rinaldi. "Strategi Algoritmik", Lab Ilmu dan Rekayasa Komputasi, Departemen Teknik Informatika ITB. 2005.

[6] Ten-Hwang Lai+ and Sartaj Sahni, "Anomalies In Parallel Branch-and-Bound Algorithms", University of Minnesota.

[7] Wright, Mike; Eglese, Richard W and Fu, Zhuo, "A Branch-and-bound Algorithm for Finding All Optimal Solutions of the Assignment Problem", Lancaster University Management School, 2004.

[8] http://www.isi.edu/~modi
Diakses tanggal 16 Mei 2005 pukul 11.00 WIB.

[9] *http://wwwcs.unipaderbond.de/pc2/talks/main.htm*
Diakses tanggal 16 Mei 2005 pukul 11.00 WIB.

[10] *http://www.brpreiss.com/books/opus4/html/page453.html* Diakses tanggal 16 Mei 2005 pukul 11.00 WIB.