Aplikasi Algoritma Branch and Bound dalam Pencarian Solusi Optimum Job Assignment Problem

Calvin Aditya Jonathan 13513077
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13513077@students.itb.ac.id

***Abstract*—Algoritma *Branch and Bound* adalah metode pencarian solusi dalam ruang solusi secara sistematis. Algoritma tersebut menggunakan pohon ruang status dinamis dan menerapkan skema yang menyerupai BFS tetapi dengan batasan-batasan tertentu yang membuatnya lebih mangkus. Ide dasar dari algoritma tersebut adalah membuang simpul-simpul pada pohon ruang status yang tidak menuju ke solusi dengan menerapkan fungsi *bounding*. Salah satu aplikasi dari algoritma ini adalah pencarian solusi persoalan *job assignment*. Persoalan tersebut pada dasarnya adalah pencarian distribusi pekerjaan yang optimal. Kita dapat menggunakan algoritma B*ranch and Bound* untuk mencari kombinasi distribusi dengan biaya minimum. Solusi yang dihasilkan oleh algoritma *Branch and Bound* tidak memiliki bukti matematis yang menyatakan bahwa solusi tersebut optimum dan seringkali memerlukan perbandingan dengan algoritma lain, tetapi jika waktu lebih dipentingkan maka algoritma ini adalah pilihan yang baik.**

***Index Terms*—job assignment problem, branch, bound, optimum.**

I. PENDAHULUAN

Seiring dengan globalisasi, persaingan dalam bidang apa pun di dunia ini menjadi semakin ketat. Setiap orang dituntut untuk hidup secara efisien dan optimal agar dapat bertahan di dunia yang semakin hari semakin modern.

Tanpa disadari, kehidupan manusia menjadi semakin sistematis dan terstruktur. Untuk setiap aksi yang kita lakukan, kita memperhitungkan langkah-langkah terbaik yang harus diambil. Contohnya adalah mengkalkulasi keuntungan maksimum pada pasar saham, efektivitas maksimum penjadwalan pekerja, dan masih banyak lagi.

Persoalan-persoalan sistematis seperti yang sudah dicontohkan dapat diselesaikan dengan metode yang juga bersifat sistematis. Dalam dunia komputasi, metode tersebut dikenal sebagai algoritma.

Sejalan dengan perkembangan zaman, dunia komputasi pun menjadi semakin canggih. Muncul banyak persoalan-persoalan kompleks yang mencerminkan kehidupan nyata. Banyak peminat bidang komputasi di dunia berlomba-lomba untuk menciptakan algoritma yang mangkus untuk menyelesaikan persoalan tersebut.

Seorang *programmer* dituntut untuk selalu menciptakan program yang mangkus dan sangkil untuk menyelesaikan suatu persoalan. Karena itu, sebaiknya seorang *programmer* mengetahui berbagai macam algoritma agar dapat memilih algoritma yang paling tepat untuk menyelesaikan suatu persoalan. Salah satu algoritma tersebut adalah algoritma *branch and bound*.

Algoritma tersebut merupakan salah satu algoritma yang dapat digunakan untuk menyelesaikan masalah dasar yang sering muncul pada kehidupan sehari-hari, yaitu *job assignment problem*. Pada dasarnya, persoalan tersebut terkait dengan distribusi bobot pekerjaan agar biaya yang dikeluarkan minimum.

Pada makalah ini akan dibahas pengertian *job assignment problem* serta metode pencarian solusi optimum untuk persoalan tersebut menggunakan algoritma *Branch and Bound*. Selain itu, makalah ini juga membahas keuntungan dan kerugian algoritma *Branch and Bound*.

II. JOB ASSIGNMENT PROBLEM

Persoalan optimasi kombinatorial adalah persoalan dengan himpunan solusi terhingga. Meskipun secara prinsip persoalan tersebut dapat diselesaikan secara enumerasi, pada persoalan yang lebih kompleks metode tersebut tidak dapat diterima secara praktis karena batasan-batasan seperti waktu.

Job assignment problem merupakan salah satu contoh dari persoalan optimasi kombinatorial yang dapat didefinisikan sebagai berikut:

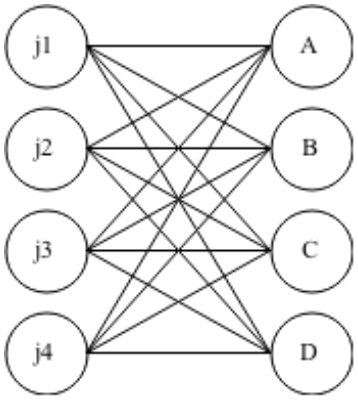
- Diberikan n buah agen dan m buah pekerjaaan , yang mana selalu belaku n >= m.
- Setiap pekerjaan memiliki biaya (*cost*) jika diberikan kepada agen tertentu.
- Berikan setiap pekerjaan kepada agen yang berbeda-beda. Satu agen hanya bisa mendapat satu pekerjaan.
- Carilah kombinasi dengan total *cost* minimum.

Adapun definisi matematisnya sebagai berikut:

- Diberikan dua himpunan P (pekerjaan) dan A (agen) yang memiliki kardinalitas |A| >= |P|.
- Terdapat nilai bobot (*cost*) untuk setiap anggota relasi P x A.

- Tentukan fungsi bijektif P dengan A sehingga nilai total *cost* minimum.

Persoalan ini dapat dimodelkan melalui graf bipartit lengkap. Contoh:



Gambar 1 Pemodelan dalam Graf Bipartit

Bagian kiri dari graf adalah himpunan pekerjaan dan bagian kanan adalah himpunan agen.

III. ALGORITMA BRANCH AND BOUND

Sebuah algoritma adalah sebuah cara atau metode penyelesaian masalah yang sistematis dan terstruktur. Salah satu contoh algoritma yang sering digunakan dalam bidang komputasi adalah *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. Algoritma ini pertama kali diperkenalkan oleh A.H. Land dan A.G. Doig pada tahun 1960. Algoritma *branch and bound* adalah teknik generalisasi dan optimasi yang mampu menyelesaikan permasalahan-permasalahan yang tidak dapat diselesaikan menggunakan algoritma *greedy* atau *dynamic programming*.

Meskipun cakupan persoalan yang dapat diselesaikan lebih luas, *Algoritma Branch and Bound* jauh lebih lambat jika dibandingkan dengan algoritma lainnya. Pada kasus terburuk, kompleksitas waktu algoritma *branch and bound* dapat berupa eksponensial. Namun, jika di-implementasi dengan baik dan hati-hati, secara rata-rata algoritma ini dapat berjalan dengan cepat.

Ide dasar dari algoritma ini adalah pendekatan enumerasi dengan cara mematikan (*pruning*) cabang-cabang yang tidak mengarah ke solusi. Pembangunan pohon ruang status menggunakan langkah yang sama dengan algoritma BFS (*Breadth First Search*), tetapi tidak semua simpul anak dibangkitkan. Simpul yang dibangkitkan hanyalah simpul-simpul yang memenuhi kriteria tertentu sesuai dengan permasalahan.

Pada algoritma *branch and bound*, terdapat dua proses utama yaitu *branching* dan *bounding*. *Branching* adalah proses pencarian solusi optimum secara rekursif sehingga akan terbentuk pohon secara otomatis. *Bounding* adalah proses pencarian nilai batas atas atau bawah untuk kemudian digunakan dalam proses *pruning* di mana hanya simpul dengan nilai batas atas terbesar atau batas bawah terkecil yang anak-anaknya dibangkitkan.

Setiap persoalan biasanya memiliki fungsi pencarian nilai batas yang berbeda-beda. Pemilihan fungsi tersebut biasanya dilakukan hanya dengan naluri dan pengalaman pembuat algoritma dan tidak ada bukti matematis bahwa fungsi tersebut sudah optimal untuk persoalan yang terkait, sehingga ketepatan dan kecepatan algoritma ini bergantung pada pemilihan fungsi tersebut. Selain itu, tidak menutup kemungkinan persoalan yang sama dapat memiliki fungsi pencarian nilai batas yang berbeda-beda.

Pada *job assignment problem*, setiap simpul diberikan ongkos (*cost*). Simpul berikutnya yang akan diekspansi tidak lagi berdasarkan urutan pembangkitannya (sebagaimana pada BFS murni), tetapi simpul yang memiliki ongkos yang paling kecil (*least cost search*). Nilai ongkos pada setiap simpul i menyatakan taksiran ongkos termurah lintasan dari simpul i ke simpul solusi (*goal node*):

$$\hat{c}(i) = \text{nilai taksiran lintasan termurah dari simpul i ke simpul tujuan}$$

Dengan kata lain, $\hat{c}(i)$ menyatakan batas bawah (lower bound) dari ongkos pencarian solusi dari simpul i.

Fungsi heuristik untuk menghitung taksiran nilai yang diberikan pada simpul dinyatakan sebagai berikut:

$$\hat{c}(i) = \hat{f}(i) + \hat{g}(i) \quad (1)$$

$\hat{c}(i)$ adalah ongkos untuk simpul i, $\hat{f}(i)$ adalah ongkos untuk mencapai simpul i dari akar pohon, dan $\hat{g}(i)$ adalah ongkos untuk mecapai simpul tujuan dari simpul i. Simpul selanjutnya yang dipilih untuk ekspansi (dibangkitkan simpul anaknya) adalah simpul yang memiliki nilai $\hat{c}$ terkecil.

Setiap simpul dimasukkan ke dalam antrian (*queue*). Setiap kali sebuah simpul di-ekspansi, simpul tersebut dikeluarkan dari antrian. Jika suatu saat proses *branching* mencapai simpul yang sudah tidak bisa di-ekspansi, nilai batas bawah simpul tersebut akan dibandingkan dengan nilai batas bawah simpul lainnya dalam antrian. Jika nilai batas bawah simpul dalam antrian tidak ada yang lebih kecil maka simpul yang tidak bisa di-ekspansi dianggap simpul solusi optimum. Jika masih ada yang nilai batas bawahnya lebih kecil, simpul dengan nilai tersebut akan di-ekspansi dan proses *branching* dimulai lagi dari simpul tersebut.

IV. PENCARIAN SOLUSI OPTIMUM JOB ASSIGNMENT PROBLEM

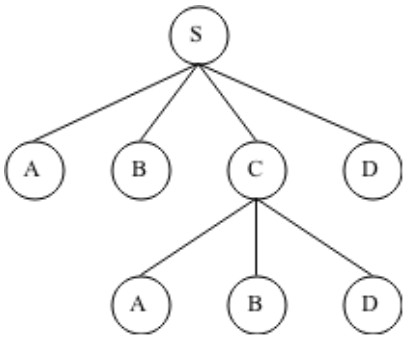
4.1 Representasi Pohon Ruang Status

Pada masalah *job assignment*, proses pemilihan simpul yang dibangkitkan dilakukan dengan memilih agen yang

mengerjakan setiap pekerjaan. Apabila seorang agen sudah pernah dipilih untuk sebuah pekerjaan maka dia tidak dapat dipilih lagi untuk pekerjaan lainnya.

Proses pemilihan agen dilakukan secara bertahap sesuai dengan aras pohon. Simpul-simpul pada aras pertama merepresentasikan pemilihan agen untuk pekerjaan pertama. Misalkan ada *n* buah agen maka ada *n* buah simpul pada aras pertama. Aras kedua merepresentasikan pemilihan agen untuk pekerjaan kedua. Jumlah simpul pada aras kedua adalah *n*-1 buah karena sebelumnya salah satu agen sudah dipilih untuk pekerjaan pertama. Aras ketiga akan berisi *n*-2 simpul dan seterusnya sampai aras ke-*n*.

Pada pohon ruang status, terdapat aras ke-0 yang berisi simpul *start* dan tidak merepresentasikan pemilihan agen. Berikut adalah contoh pohon ruang status untuk *job assignment problem*:



Gambar 2 Contoh Pohon Ruang Status

Pada contoh pohon tersebut, pekerjaan pertama diserahkan kepada agen C dan karena menggunakan algoritma *branch and bound*, hanya simpul C saja yang anaknya dibangkitkan.

4.2 Pencarian Nilai Batas

Pada permasalahan ini, kita dapat mencari nilai batas bawah setiap simpul menggunakan persamaan (1). Untuk mempermudah proses pencarian, kita dapat membuat matriks yang merepresentasikan *cost* pemberian pekerjaan kepada seorang agen. Misalkan ada 4 buah pekerjaan {j1, j2, j3, j4} dan 4 orang agen {A, B, C, D}, berikut adalah matriks yang dihasilkan:

j1	j2	j3	j4	
8	1	9	2	A
1	6	3	4	B
7	5	4	7	C
3	9	6	8	D

Gambar 3 Contoh Matriks *Cost*

Pada matriks tersebut, jika kita memberikan pekerjaan j1 kepada A maka *cost*-nya adalah 8 dan jika diberikan kepada B maka *cost*-nya adalah 1.

Dengan menggunakan persamaan (1), kita akan mencari nilai batas bawah untuk setiap simpul. Misal simpul *x* adalah simpul pada aras pertama yang memberikan pekerjaan j1 kepada agen A. Maka nilai batas bawah untuk simpul *x* adalah:

$$\hat{c}(x) = \hat{f}(x) + \hat{g}(x)$$
$$\hat{c}(x) = (0 + 8) + (5 + 3 + 4) = 20$$

Nilai $\hat{g}(x)$ didapat dengan menjumlahkan *cost* minimum dari sisa pekerjaan yang belum diserahkan ke agen tanpa memperhitungkan baris A. Dalam hal ini, j2 = 5, j3 = 3, dan j4 = 4.

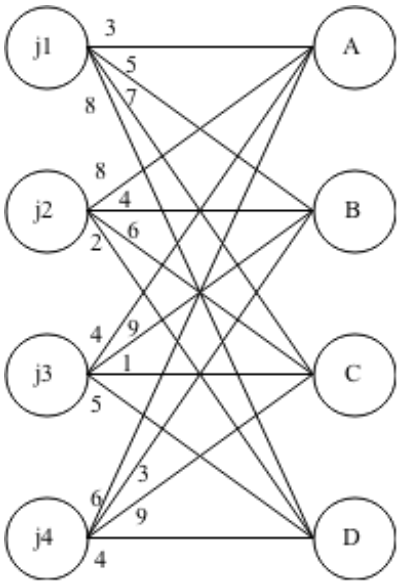
4.3 Proses Pencarian Solusi Optimum dan Contoh Kasus

Langkah-langkah utama dari proses pencarian solusi adalah:

1. Masukkan semua simpul ke dalam antrian Q
2. Cari nilai batas bawah $\hat{c}$ setiap simpul dalam Q
3. Lakukan ekspansi simpul dengan nilai $\hat{c}$ terkecil, jika ada dua simpul yang memenuhi, pilih salah satu
4. Keluarkan simpul tersebut dari Q
5. Ulangi langkah 1 untuk semua simpul anak yang baru dibangkitkan sampai tidak ada simpul anak yang bisa dibangkitkan lagi
6. Simpul terakhir adalah solusi

Untuk mendapat gambaran yang lebih jelas, diberikan contoh kasus:

Misalkan ada sebuah *job assignment problem* dengan 4 buah pekerjaan {j1, j2, j3, j4} dan 4 orang agen {A, B, C, D}. Jumlah agen dan pekerjaan disamakan untuk mempermudah pencarian solusi pada contoh kasus. *Cost* j1 untuk masing-masing agen adalah 3, 7, 5, 8 terurut dari A ke D. *Cost* j2 adalah 8, 4, 6, 2. *Cost* j3 adalah 4, 9, 1, 5. *Cost* j4 adalah 6, 3, 9, 3. Pemodelannya dalam graf bipartit adalah:



Gambar 4 Graf Bipartit dari Contoh Kasus

Langkah pertama adalah membuat matriks *cost* untuk persoalan tersebut:

j1	j2	j3	j4	
3	8	4	6	A
5	4	9	3	B
7	6	1	9	C
8	2	5	4	D

Gambar 5 Matriks *Cost* dari Contoh Kasus

Setelah itu, masukkan semua simpul ke antrian Q dan hitung nilai batas bawah dari setiap simpul (simpul 1 adalah simpul pemilihan agen A untuk pekerjaan j1):

$c\hat{}(1) = (0 + 3) + (2 + 1 + 3) = 9$
$c\hat{}(2) = (0 + 5) + (2 + 1 + 4) = 12$
$c\hat{}(3) = (0 + 7) + (2 + 4 + 3) = 16$
$c\hat{}(4) = (0 + 8) + (4 + 1 + 3) = 16$

Karena $c\hat{}(1)$ memiliki nilai paling kecil berarti pekerjaan j1 diserahkan kepada agen A dan simpul 1 dipilih untuk ekspansi. Simpul 1 dikeluarkan dari Q. Kemudian nilai pada baris agen A diubah menjadi ∞.

j1	j2	j3	j4	
∞	∞	∞	∞	A
5	4	9	3	B
7	6	1	9	C
8	2	5	4	D

Gambar 6 Ekspansi Simpul 1

Kemudian, untuk menentukan kepada agen mana kita serahkan pekerjaan j2, masukkan simpul 5, 6, serta 7 ke dalam Q lalu cari lagi nilai batas bawah untuk simpul-simpul tersebut:

$c\hat{}(5) = (3 + 4) + (1 + 4) = 12$
$c\hat{}(6) = (3 + 6) + (5 + 3) = 17$
$c\hat{}(7) = (3 + 2) + (1 + 3) = 9$

Karena $c\hat{}(7)$ memiliki nilai paling kecil maka pekerjaan j2 diserahkan kepada agen D dan simpul 7 dipilih untuk ekspansi. Simpul 7 dikeluarkan dari antrian.

j1	j2	j3	j4	
∞	∞	∞	∞	A
5	4	9	3	B
7	6	1	9	C
∞	∞	∞	∞	D

Gambar 7 Ekspansi Simpul 7

Masukkan simpul 8 dan 9 ke dalam Q. Ulangi lagi pencarian nilai batas bawah untuk pekerjaan j3.

$c\hat{}(8) = (5 + 9) + 9 = 23$
$c\hat{}(9) = (5 + 1) + 3 = 9$

Karena $c\hat{}(9)$ lebih kecil dari $c\hat{}(8)$ maka pekerjaan j3 diberikan kepada agen C dan simpul 9 dipilih untuk ekspansi. Simpul 9 dikeluarkan dari antrian.

j1	j2	j3	j4	
∞	∞	∞	∞	A
5	4	9	3	B
∞	∞	∞	∞	C
∞	∞	∞	∞	D

Gambar 8 Ekspansi Simpul 9

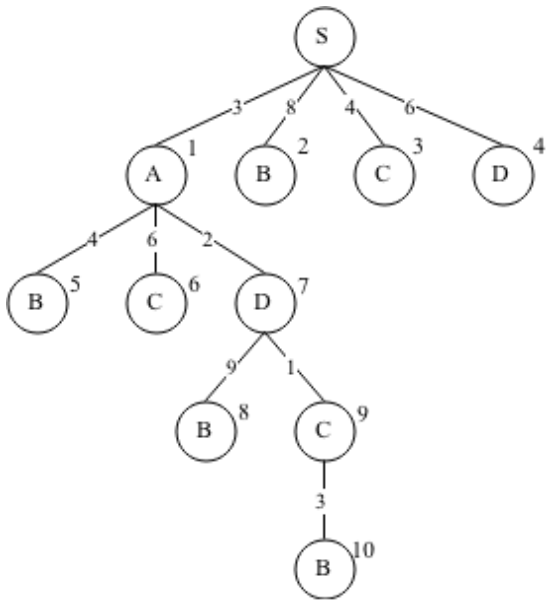
Simpul 9 hanya memiliki satu anak yaitu simpul 10 dengan nilai batas bawah $c\hat{}(10) = (6 + 3) + 0 = 9$.

j1	j2	j3	j4	
∞	∞	∞	∞	A
0	0	0	0	B
∞	∞	∞	∞	C
∞	∞	∞	∞	D

Gambar 9 Kondisi Akhir Matriks

Untuk mengecek apakah simpul 10 merupakan solusi optimum, bandingkan nilai batasnya dengan simpul-simpul dalam antiran (simpul 2, 3, 4, 5, 6, dan 8). Jika ada simpul yang memiliki nilai batas bawah lebih kecil dari 9 maka simpul tersebut akan di-ekspansi untuk menentukan solusi yang lebih optimum. Karena pada kasus ini tidak ada yang lebih kecil dari 9 maka simpul 10 dianggap optimum dan solusi *job assignment problem* adalah <A, D, C, B> yang berarti pekerjaan j1 diserahkan kepada A, j2 kepada D, j3 kepada C, dan j4 kepada B. Total *cost* dari solusi adalah 9.

Berikut adalah ilustrasi pohon ruang status yang dibangun selama proses pencarian:



Gambar 10 Pohon Ruang Status dari Contoh Kasus

Angka pada garis penghubung adalah *cost* untuk menuju ke simpul tersebut dan angka di kanan atas simpul menunjukkan simpul ke berapa.

4.4 Validasi Solusi Melalui Exhaustive Search

Exhaustive search adalah salah satu teknik pencarian *brute force* yang melakukan enumerasi semua kemungkinan solusi permasalahan dan melakukan pengecekan terhadap setiap kemungkinan untuk menentukan solusi optimum. Hasil dari *exhaustive search* selalu optimal karena ia menelusuri semua kemungkinan. Exhaustive search dapat diterapkan pada job assignment problem untuk mengecek apakah solusi yang didapat melalui algoritma *branch and bound* sudah optimum. Berikut adalah tabel hasil *exhaustive search*:

Solusi	Total *Cost*	Solusi	Total *Cost*
<A, B, C, D>	12	<C, A, B, D>	28
<A, B, D, C>	21	<C, A, D, B>	23
<A, C, B, D>	22	<C, B, A, D>	19
<A, C, D, B>	17	<C, B, D, A>	22
<A, D, B, C>	23	<C, D, B, A>	24
<A, D, C, B>	9	<C, D, A, B>	16
<B, A, C, D>	18	<D, A, B, C>	34
<B, A, D, C>	27	<D, A, C, B>	22
<B, C, A, D>	19	<D, B, A, C>	25
<B, C, D, A>	22	<D, B, C, A>	19
<B, D, A, C>	20	<D, C, A, B>	21
<B, D, C, A>	14	<D, C, B, A>	29

Tabel 1 Hasil Enumerasi *Exhaustive Search*

Solusi optimum yang didapat adalah <A, D, C, B> dengan total *cost* 9. Solusi tersebut sama dengan solusi yang didapat dari algoritma *branch and bound* sehingga dapat dikatakan algoritma branch and bound yang dipakai sudah optimal.

V. KESIMPULAN

Algoritma *branch and bound* merupakan algoritma yang sering digunakan untuk menyelesaikan persoalan optimasi, terutama yang menyangkut enumerasi. Algoritma *branch and bound* menggunakan skema BFS yang lebih pintar, yaitu menggunakan fungsi pembatas atau *bound* untuk menentukan simpul yang akan di-ekspansi berikutnya sehingga tidak perlu menelusuri simpul-simpul yang tidak mengarah ke solusi. Kasus terburuk dari algoritma ini sama seperti exchaustive search yaitu *n*!, namun hal tersebut sangat jarang terjadi karena algoritma ini menggunakan bantuan fungsi pembatas dalam pencarian solusi.

Job assigment problem adalah salah satu persoalan dasar pada bidang optimasi kombinatorial. Solusi optimum dari persoalan tersebut adalah pemberian pekerjaan terhadap agen sedemikian rupa agar total biaya yang diperlukan minimum. Kita dapat menemukan solusi optimum tersebut menggunakan algoritma *branch and bound*.

Ketepatan solusi optimum yang diperoleh dari algoritma *branch and bound* sangat tergantung pada pemilihan fungsi pencarian nilai batas. Selain itu, fungsi tersebut dipilih hanya berdasarkan naluri dan pengalaman, sehingga algoritma ini terkadang tidak memberikan hasil yang benar-benar optimal. Kekurangan algoritma *branch and bound* adalah seringkali, algoritma ini perlu algoritma lain untuk membuktikan bahwa solusi yang didapat sudah optimal karena algoritma *branch and bound* tidak memiliki bukti matematis yang kuat. Namun, pada prakteknya algoritma ini merupakan salah satu algoritma yang cukup mangkus dan dapat diaplikasikan terhadap banyak persoalan.

Selain *branch and bound* banyak algoritma lain yang dapat digunakan untuk mencari solusi optimum dari masalah *job assignment*, contohnya adalah *exhaustive search*. Namun, apabila aspek waktu lebih dipentingkan daripada ketepatan solusi, algoritma ini bisa menjadi pilihan dalam menyelesaikan masalah *job assignment* karena jika dibandingkan dengan algoritma lain (dalam hal ini *exhaustive search*), algoritma *branch and bound* lebih cepat.

VI. UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena hanya atas berkat-Nya penulis dapat menyelesaikan makalah yang berjudul "Aplikasi Algoritma Branch and Bound dalam Pencarian Solusi Optimum Job Assignment Problem" ini dengan tepat waktu. Tidak lupa penulis mengucapkan terima kasih kepada Dr. Ir. Rinaldi Munir, MT dan Masayu Leylia Khodra, ST., MT. atas pengajaran dan bimbingannya selama satu semester kuliah Strategi Algoritma karena ilmu yang didapat penulis sangat membantu dalam pengerjaan makalah ini. Selain itu, penulis juga mengucapkan terima kasih kepada keluarga dan teman-teman atas dukungan, masukan, dan doa yang telah mereka berikan.

DAFTAR PUSTAKA

[1] Munir, Rinaldi.*Strategi Algoritma*.Informatika Bandung, 2009
[2] http://www.seas.gwu.edu/~ayoussef/cs212/branchandbound.html -- diakses pada tanggal 4 Mei 2015
[3] http://www.me.utexas.edu/~jensen/models/network/net9.html -- diakses pada tanggal 4 Mei 2015
[4] http://stanford.edu/class/ee364b/lectures/bb_slides.pdf -- diakses pada tanggal 4 Mei 2015

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, 4 April 2015

Calvin Aditya Jonathan 13513077