APLIKASI ALGORITMA *BRANCH AND BOUND* UNTUK MENYELESAIKAN INTEGER PROGRAMMING

Shieny Aprilia
13505089

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

E-mail : if15089@students.if.itb.ac.id

ABSTRAK

***Integer Programming* merupakan salah satu pemodelan untuk pencarian suatu solusi yang optimum dari suatu masalah. Banyak sekali kasus di sekitar kita yang memerlukan implementasi *integer programming* agar didapatkan suatu solusi paling optimal yang pada akhirnya menambah keuntungan. Ada banyak cara untuk menyelesaikan *integer programming*, misalnya dengan cara program linear standar yaitu dengan menggunakan grafik. Cara lainnya adalah dengan mengaplikasikan algoritma *Branch and Bound*. Makalah ini mencoba untuk menjelaskan bagaimana suatu *integer programming* diselesaikan dengan pengaplikasian algoritma *Branch and Bound*.**

Kata kunci : *Branch and Bound*, *Integer Programming*, BFS, DFS

1. PENDAHULUAN

Suatu permasalahan biasanya menuntut solusi yang optimum agar dapat diperoleh keuntungan yang sebesar – besarnya. Salah satu model untuk merepresentasikan suatu permasalahan adalah Program Linear (*Linear Programming*). Ada 2 jenis Program Linear :

a. *Integer Programming* : program linear dengan variabel bertipe Integer.
b. *Non-Integer Programming* : program linear dengan variabel bertipe Non-Integer, misal : bilangan real.

Model *Integer Programming* biasanya dipilih untuk permasalahan yang variabel – variabelnya tidak dimungkinkan bertipe bilangan tidak bulat, misalnya : variabel jumlah orang.

Integer Programming dapat diselesaikan dengan banyak cara, antara lain : dengan menggunakan grafik, dengan metode eliminasi dan substitusi, dan sebagainya. Salah satu cara yang cukup efektif untuk menyelesaikan *Integer Programming* adalah dengan mengaplikasikan algoritma *Branch and Bound*.

2. ALGORITMA *BRANCH AND BOUND*

Algoritma *Branch and Bound* adalah metode algoritma umum untuk mencari solusi optimal dari dari berbagai permasalahan optimasi, terutama untuk optimasi diskrit dan kombinatorial. Sebagaimana pada algoritma runut-balik, algoritma *Branch and Bound* juga merupakan metode pencarian di dalam ruang solusi secara sistematis. Ruang solusi diorganisasikan ke dalam pohon ruang status. Yang membedakan keduanya adalah bila pada algoritma runut-balik, ruang solusi dibangun secara dinamis berdasarkan skema DFS (*Depth First Search*), maka pada algoritma *Branch and Bound* ruang solusi dibangun dengan skema BFS (*Breadth First Search*).

Pada algoritma ini, permasalahan dibagi – bagi menjadi *subregion* – *subregion* yang mungkin mengarah ke solusi. Inilah yang disebut dengan *branching*, mengingat prosedur ini akan dilakukan berulang – ulang secara rekursif untuk setiap *subregion* dan setiap *subregion* yang dihasilkan akan membentuk sebuah struktur pohon yang disebut sebagai pohon pencarian atau pohon *branch-and-bound* di mana simpul – simpulnya membangun *subregion* – *subregion*. Selain *branching*, algoritma

ini juga melakukan apa yang disebut dengan *bounding* yang merupakan cara cepat untuk mencari batas atas dan bawah untuk solusi optimal pada *subregion* yang mengarah ke solusi.
Algoritma *Branch and Bound* banyak digunakan untuk memecahkan berbagai macam permasalahan antara lain : persoalan *Knapsack* 0/1, *Travelling Salesman Problem* (TSP), *The N-Queens Problem* (Persoalan N-Ratu), *Graph Colouring* (Pewarnaan Graf), Sirkuit Hamilton, *Integer Programming*, *Nonlinear Programming*, *Quadratic Assignment Problem* (QAP), *Maximum Satisfiability Problem* (MAX-SAT), dan lain sebagainya.

3. *INTEGER PROGRAMMING*

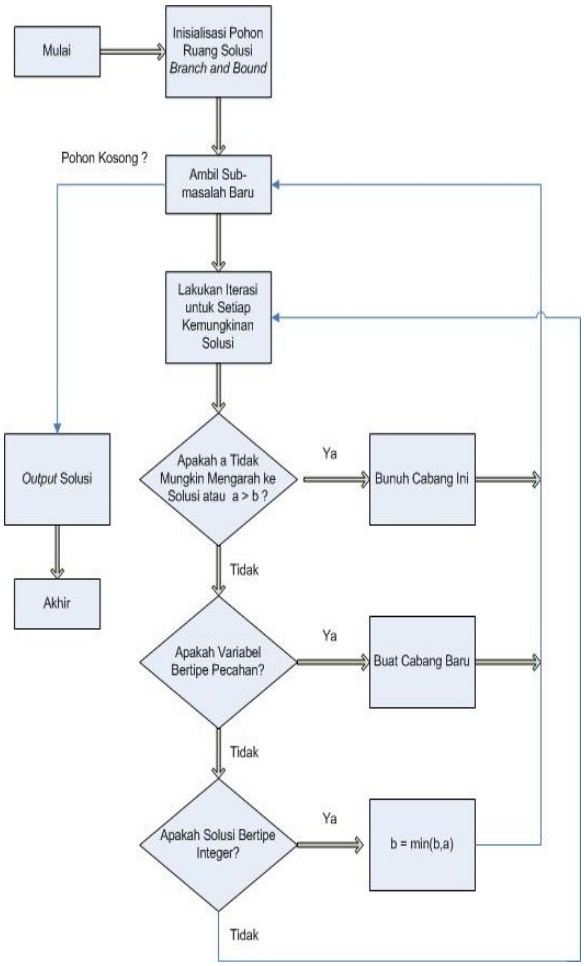
Integer Programming adalah program linear (*Linear Programming*) di mana variabel – variabelnya bertipe integer. *Integer Programming* digunakan untuk memodelkan permasalahan yang variabel – variabelnya tidak mungkin berupa bilangan yang tidak bulat (bilangan *real*), seperti variabel yang merepresentasikan jumlah orang, karena jumlah orang pasti bulat dan tidak mungkin berupa pecahan. *Integer Programming* juga biasanya lebih dipilih untuk memodelkan suatu permasalahan karena program linear dengan variabel berupa bilangan *real* kurang baik dalam memodelkan permasalahan yang menuntut solusi berupa bilangan integer, misalnya keuntungan produksi 3 pesawat dibandingkan dengan keuntungan produksi 3.5 pesawat akan menghasilkan selisih keuntungan yang signifikan.

4. APLIKASI ALGORITMA *BRANCH AND BOUND* DALAM PENYELESAIAN *INTEGER PROGRAMMING*

Telah dijelaskan sebelumnya bahwa algoritma *Branch and Bound* dapat digunakan untuk menyelesaikan *Integer Programming*. Gambar 1 adalah *flowchart* aplikasi algoritma *Branch and Bound* dalam menyelesaikan *Integer Programming* dengan optimasi minimum. Gambar 2 adalah *flowchart* aplikasi algoritma *Branch and Bound* dalam menyelesaikan *Integer Programming* dengan optimasi maksimum.

Berikut ini adalah contoh aplikasi algoritma *Branch and Bound* untuk menyelesaikan *Integer Programming* :

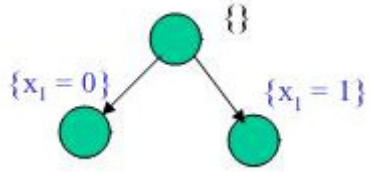
Persoalan :
Maksimum $Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$
Dengan batasan :
1. $6x1 + 3x2 + 5x3 + 2x4 <= 10$
2. $x3 + x4 <= 1$
3. $–x1 + x3 <= 0$
4. $–x2 + x4 <= 0$
5. $xi <= 1, xi >= 0$, xi **integer**



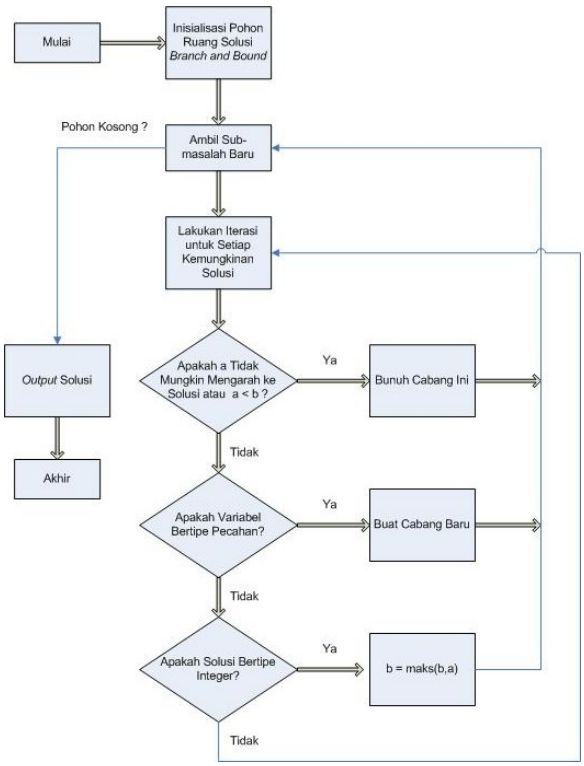
Gambar 1. *Flowchart* Algoritma *Branch and Bound* untuk *Integer Programming Optimasi Minimum*

Langkah 1 : Inisialisasi Pohon
Solusi terbaik sampai saat ini : {}
Nilai Z maksimum : ∞
Pohon :

Langkah 2
Solusi terbaik sampai saat ini : {}
Nilai Z maksimum : ∞
Pohon :



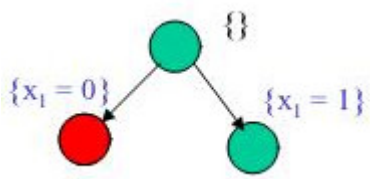
Langkah 3
Solusi terbaik sampai saat ini : {}
Nilai Z maksimum : ∞



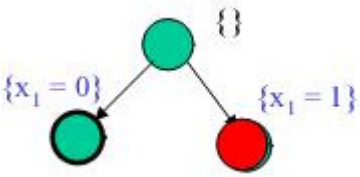
Gambar 2. *Flowchart* Algoritma *Branch and Bound* untuk *Integer Programming* Optimasi Maksimum

Keterangan : Iterasi anak kiri aras 1

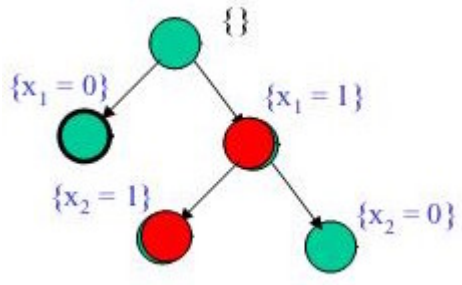
Pohon :



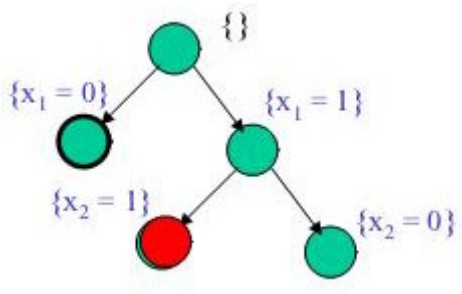
Langkah 4
Solusi terbaik sampai saat ini : {0,1,0,1}
Nilai Z maksimum : 9
Keterangan : Iterasi anak kanan aras 1
Pohon :



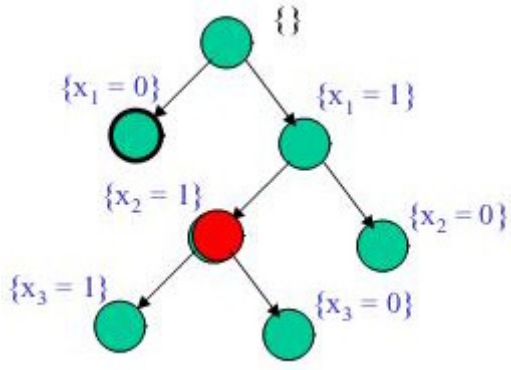
Langkah 5
Solusi terbaik sampai saat ini : {0,1,0,1}
Nilai Z maksimum : 9
Keterangan : Iterasi anak kanan aras 1 dan anak kiri aras 2
Pohon :



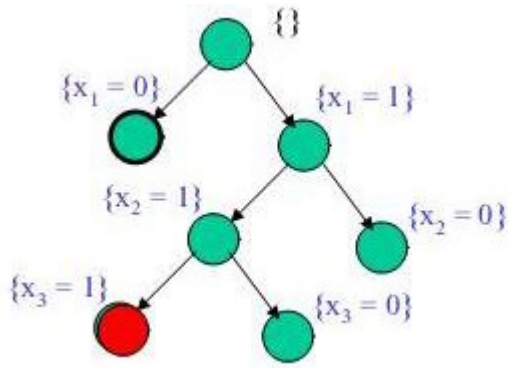
Langkah 6
Solusi terbaik sampai saat ini : {0,1,0,1}
Nilai Z maksimum : 9
Keterangan : Iterasi anak kiri aras 2 dengan solusi terbaik
Pohon :



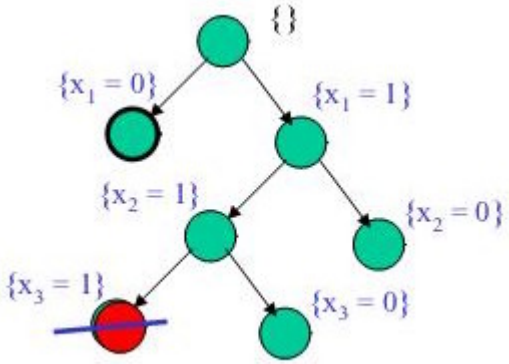
Langkah 7
Solusi terbaik sampai saat ini : {0,1,0,1}
Nilai Z maksimum : 9
Keterangan : Perluasan anak kiri aras 2
Pohon :



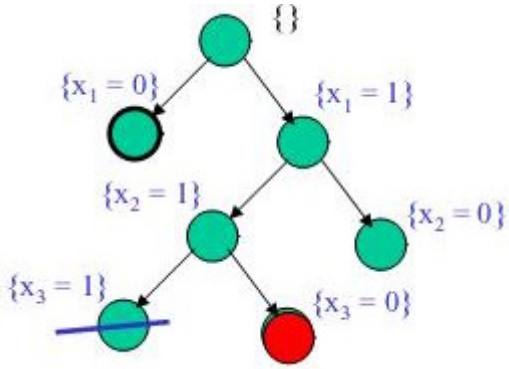
Langkah 8
Solusi terbaik sampai saat ini : {0,1,0,1}
Nilai Z maksimum : 9
Keterangan : Iterasi anak kiri aras 3
Pohon :



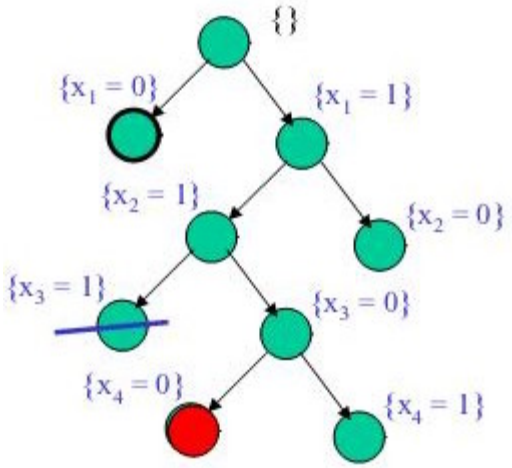
Langkah 9
Solusi terbaik sampai saat ini : {0,1,0,1}
Nilai Z maksimum : 9
Keterangan : Bunuh anak kiri aras 3 karena tidak mungkin mengarah ke solusi (melanggar batasan ke 2 dan 4)
Pohon :



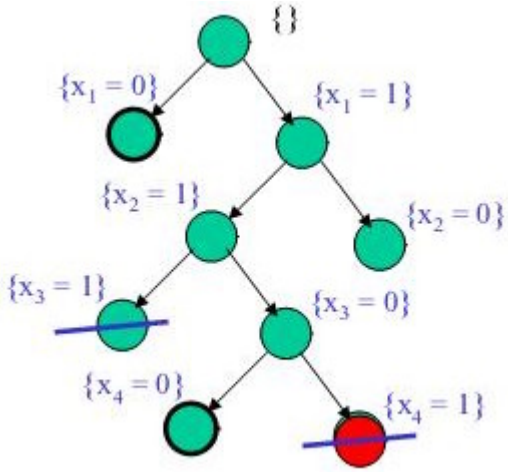
Langka
h 10
Solusi terbaik sampai saat ini : {0,1,0,1}
Nilai Z maksimum : 9
Keterangan : Iterasi anak kanan aras 3
Pohon :



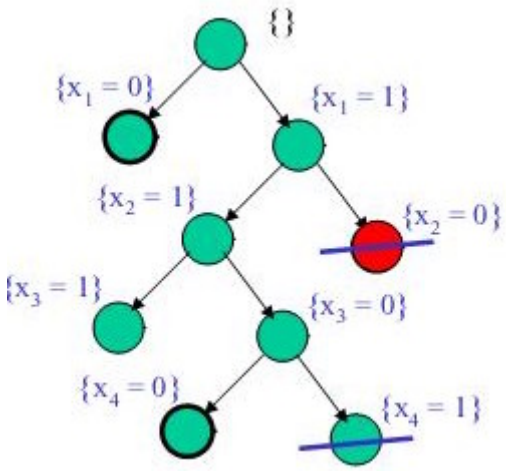
Langkah 11
Solusi terbaik sampai saat ini : {1,1,0,0}
Nilai Z maksimum : 14
Keterangan : Perluasan anak kanan aras 3 dan iterasi anak kiri aras 4
Pohon :



Langkah 12
Solusi terbaik sampai saat ini : {1,1,0,0}
Nilai Z maksimum : 14
Keterangan : Bunuh anak kanan aras 4 karena tidak mungkin mengarah ke solusi (melanggar batasan ke 1)
Pohon :



Langkah 13
Solusi terbaik sampai saat ini : {1,1,0,0}
Nilai Z maksimum : 14
Keterangan : Bunuh anak kanan aras 2 karena jika anak kanan aras 4 dibunuh, aras 2 tidak mungkin mengarah ke solusi
Pohon :



Setelah menjalani berbagai langkah, diperoleh solusi optimal dari permasalahan di atas.
Solusi : {1,1,0,0}
Nilai Z maksimum : 14

5. KESIMPULAN

Algoritma *Branch and Bound* cukup efektif untuk menyelesaikan *Integer Programming*. Dengan salah satu langkahnya yang tidak akan memperluas dan akan "membunuh" simpul yang tidak mungkin mengarah ke solusi, algoritma ini menjadi algoritma yang cukup efisien untuk menyelesaikan *Integer Programming*. Tetapi dalam menyelesaikan permasalahan ini, algoritma *Branch and Bound* mempunyai kelemahan, yaitu : algoritma ini tetap menghitung kemungkinan solusi dengan tipe variabel bilangan real walaupun pada akhirnya kemungkinan solusi ini tidak akan dipertimbangkan. Tetapi hal ini menyebabkan waktu komputasi bertambah lama.

REFERENSI

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

[2] http://dspace.mit.edu/html
Tanggal Akses : 18 Mei 2007 pukul 19.23

[3] http://en.wikipedia.org/wiki/Branch_and_bound
Tanggal Akses : 18 Mei 2007 pukul 19.14

[4] http://ieeexplore.ieee.org/Xplore
Tanggal Akses : 18 Mei 2007 pukul 19.13

[5] http://www.columbia.edu/
Tanggal Akses : 18 Mei 2007 pukul 19.32

[6] http://www.ingentaconnect.com
Tanggal Akses : 18 Mei 2007 pukul 19.23

[7] http://www.sce.carleton.ca/
Tanggal Akses : 18 Mei 2007 pukul 19.24

[8]http://www2.isye.gatech.edu/
Tanggal Akses : 18 Mei 2007 pukul 19.26