Pemecahan Masalah Knapsack dengan Menggunakan Algoritma Branch and Bound

Anggi Shena Permata / 13505117

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Insitut Teknologi Bandung
Jl. Ganesha 10, Bandung
Email : if15117@students.if.itb.ac.id

ABSTRAK

Permasalahan knapsack atau yang biasa kita kenal dengan sebutan 0/1 knapsack merupakan salah satu dari persoalan klasik yang banyak ditemukan pada literatur-literatur lama dan hingga kini permasalahan ini masih banyak ditemukan dalam kehidupan sehari-hari. Contoh kongkret permasalahan ini dalam dunia nyata adalah penjualan beberapa jenis keperluan rumah tangga oleh pedagang keliling dengan menggunakan gerobak ataupun alat pengangkut lainnya yang hanya memiliki kapasitas angkut maksimum sebesar w kg. keperluan rumah tangga yang akan dijual hanya berjumlah satu untuk tiap jenisnya dan tiap jenis barang memiliki berat w1,w2,w3,w4,…..wn dengan keuntungan yang diperoleh untuk tiap jenisnya adalah p1,p2,p3,p4,….pn. tidak semua jenis keperluan rumah tangga yang akan dijual oleh pedagang keliling tersebut dapat dimasukan kedalam alat pengangkut. Maka akan dipilih jenis-jenis keperluan rumah tangga yang akan dijual untuk setiap harinya oleh pedagang keliling tersebut agar diperoleh keuntungan yang maksimal dari penjualan barang-barang keperluan rumah tangga tersebut. Telah banyak strategi yang diterapkan untuk menyelesaikan permasalah tersebut diantaranya adalah dengan menggunakan algoritma exhaustive search, greedy dan juga branch and bound.

Kata kunci: *knapsack* ,keuntungan maksimum, fungsi objektif, *branch and bound*, pohon

1. PENDAHULUAN

1.1 *Knapsack problem*

Knapsack problem adalah suatu masalah bagaimana cara menentukan pemilihan barang dari sekumpulan barang di mana setiap barang tersebut mempunyai berat dan profit masing – masing, sehingga dari pemilihan barang tersebut didapatkan profit yang maksimum. Penyelesaian masalah dengan menggunakan algoritma exhaustive search adalah mengenumerasikan semua kemungkinan barang-barang yang layak atau memenuhi syarat yaitu tidak melebihi batas daya angkut gerobak untuk dijual setiap harinya , kemudian menghitung tiap-tiap keuntungan yang diperoleh dan memilih solusi yang menghasilkan keuntungan terbesar. Berbeda dengan algoritma exhaustive search yang cukup memakan waktu dan dapat menghasilkan solusi yang optimum, penyelesaian masalah dengan menggunakan algoritma greedy dilakukan dengan memasukan objek satu persatu kedalam gerobak dan tiap kali objek tersebut telah dimasukan kedalam gerobak maka objek tersebut tidak dapat lagi dikeluarkan dari gerobak. Pencarian solusi akan dilakukan dengan memilih salah satu jenis greedy (greedy by weight, greedy by profiit or greedy by density) yang diperkirakan dapat menghasilkan solusi yang optimum. Algoritma Branch and Bound juga merupakan salah satu strategi yang dapat digunakan dalam pencarian solusi optimum dari permasalahan knapsack ini. Dengan penentuan keuntungan maksimal pada tiap simpulnya, proses pencarian akan membawa kita pada solusi yang optimum . karena tidak semua objek pada permasalahan ini dapat dimasukan kedalam knapsack, maka kemungkinan bahwa kita akan sampai pada keadaan dimana tidak ada lagi simpul yang dapat dibangkitkan kerena telah melewati batas kapasitas daya angkut membuat kita harus menentukan solusi omtimum dengan membandingkan lintasan-lintasan mana yang berakhir didaun pada pohon yang akan manghasilkan keuntungan paling besar maka objek-objek tersebutlah yang akan dipilih untuk dimasukan kedalam knapsack/gerobak.

1.2 Algoritma Branch and Bound

Sebagaimana pada algortima runut-balik, algoritma *Branch & Bound* juga merupakan metode pencarian di dalam ruang solusi secara sistematis. Ruang Solusi diorganisasikan ke dalam pohon ruang status. Pembentukan pohon ruang status. Pembentukan pohon ruang status pada algoritma B&B berbeda dengan pembentukan pohon pada algoritma runutbalik. Bila pada algoritma runut-balik ruang solusi dibangun secara *Depth-First Search*(DFS), maka pada algoritma B&B

ruang solusi dibangun dengan skema *Breadth-First Search* (BFS).

Pada algoritma B&B, pencarian ke simpul solusi dapat dipercepat dengan memilih simpul hidup berdasarkan nilai ongkos (*cost*). Setiap simpul hidup diasosiasikan dengan sebuah ongkos yang menyatakan nilai batas (*bound*). Pada prakteknya, nilai batas untuk setiap simpul umumnya berupa taksiran atau perkiraan. Fungsi heuristik untuk menghitung taksiran nilai tersebut dinyatakan secara umum sebagai :

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

yang dalam hal ini,

$\hat{c}(i)$ = ongkos untuk simpul i

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

$\hat{g}(i)$ = ongkos mencapai simpul tujuan dari simpul akar i (perkiraan)

Nilai $\hat{c}$ digunakan untuk mengurutkan pencarian. Simpul berikutnya yang dipilih untuk diekspansi adalah simpul yang memiliki $\hat{c}$ minimum (Simpul-E). Strategi memilih simpul-E seperti ini dinamakan strategi pencarian berdasarkan biaya terkecil (least cost search).

Prinsip dari algoritma branch and bound ini adalah :

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 $\hat{c}(i)$ 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 $\hat{c}(j)$, dan masukkan semua anak-anak tersebut ke dalam antrian *Q*.
6. Kembali ke langkah 2.

1.3 Pohon

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Dengan setiap kemungkinan solusi dianggap sebagai sebuah simpul dan akar dari pohon, banyak algoritma yang menggunakan ruang solusi berupa pohon ini karena akan lebih memudahkan dalam penelusuran solusi yang ada berupa simpul-simpul pohon.

2. METODE

Untuk lebih memahami tahap-tahap penyelesaian permasalahan knapsack ini, kita ambil contoh persoalan seperti yang dituliskan pada bagian Abstrak yaitu dimana seorang pedagang keperluan rumah tangga keliling harus memilih barang-barang yang akan dijual setiap harinya dengan batas daya angkut gerobak yang dimilikinya. Untuk mempermudah, kita misalkan pedagang keliling tersebut hanya memiliki 4 jenis barang untuk dijual dengan berat dan keuntungan penjualan yang berbeda-beda untuk tiap jenisnya. Gerobak yang akan dipakai untuk mengangkut barang-barang tersebut hanya mampu menampuk beban seberat **16 kg**. Berikut merupakan tebel penggambaran berat dan keeuntungan yang akan diperoleh untuk tiap penjualan barang tersebut.

Barang ke-	Berat (W)	Keuntungan (P)	P/W
1	2	12	6
2	5	15	3
3	10	50	5
4	5	10	2

Gambar 1. tabel keterangan berat,keuntungan dan P/W tiap jenis barang

Dalam algoritma ini , kita akan menentukan cost dari tiap tiap simpul anak untuk dapat menentukan simpul mana yang kelak akan dibangkitkan yaitu simpul dengan cost tertinggi dalam penelusuran pohon unutk mencapai solusi dari permasalahan ini. Dalam permasalahan ini, kita akan mencari simpul-simpul yang akan membawa kita pada keuntungan terbesar oleh karena itu urutan pembangkitan simpul akan ditentukan oleh simpul mana yang memiliki cost tertinggi.

Cost dari tiap simpul akan ditentukan dengan perhitungan :

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

yang dalam hal ini,

$\hat{c}(i)$ = cost untuk simpul i

$f(i)$ = cost untuk sampai ke simpul I, dalam hal ini merupakan keuntungan dari simpul akar ke simpul i

$\hat{g}(i)$ = cost dari simpul i untuk sampai ke simpul tujuan, dalam hal ini dapat diperoleh dengan menggunakan rumus :

(P/W)max * daya angkut yang tersisa

pada tahap awal kita akan melakukan perhitungan dengan menggunakan rumus diatas untuk memperoleh batas awal atau akar dari pohon yang juga merupakan

simpul pertama. Pada keadaan ini, batas dihitung dengan pemikiran bahwa belum ada satupun barang yang dimasukan kedalam alat pengangkut maka kita dapat memilih 6 sebagai (P/W) terbesar karena belum ada satu barangpun yang dimasukan kedalam alat pengangkut dan kapasitas daya angkutpun masih utuh yaitu seberat 16 kg.

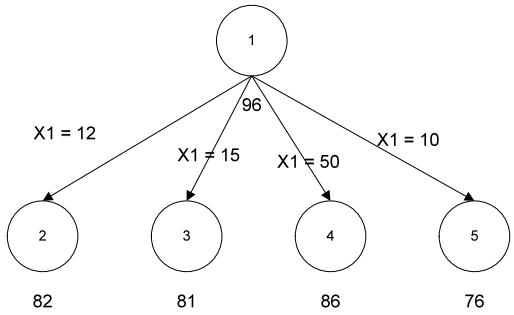
$\hat{c}(i) = f(i) + \hat{g}(i)$
$\hat{c}(1)$ = keuntungan yang diperoleh sampai disimpul awal + (P/W)max * daya angkut yang tersisa
= 0 + 6 *
= 96
Maka kita memperoleh 96 batas awal atau cost dari simpul awal.

96

Gambar 2. Akar Pohon *Knapsack problem* dengan n=4

Selanjutnya kita akan membangkitkan simpul-simpul anak dari akar pohon yaitu dengan membangkitkan simpul 1, simpul 2, simpul 3 dan simpul 4 sebagai gambaran dari 4 pilihan barang yang akan dimasukan pertama kali pada alat pengangkut dengan x1 merupakan keuntungan yang akan diperoleh pada penjualan tiap barang tersebut. Kemudian kita akan menghitung cost dari tiap simpul anak yang hidup dan juga kelayakannya untuk tetap hidup atau harus dibunuh. Dalam hal ini, simpul yang jumlah dari lintasannya tidak bisa lagi dibangkitkan (jika ditambah barang lagi kedalam alat pengangkut maka beratnya akan melebihi daya angkut) akan dibunuh.

$\hat{c}(2) = 12 + 5*(16-2) = 82$
$\hat{c}(3) = 15 + 6*(16-5) = 81$
$\hat{c}(3) = 50 + 6*(16-10)=86$
$\hat{c}(4) = 10 + 6*(16-5)=76$

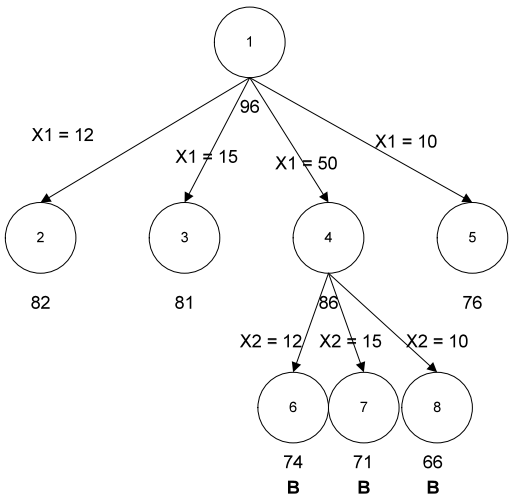


Gambar 3. pohon status knapsack problem tahap awal

Dari simpul-simpul yang telah dibangkitkan dan dihitung cost nya, maka diperoleh bahwa simpul **4** lah yang memiliki cost tertinggi oleh karena itu maka simpul 4 akan di perluas lagi. Simpul 6 ,7,8 akan dibangkitkan sebagai perluasan dari simpul 4 dengan barang yang mungkin dimasukan kedalam alat pengangkut adalah barang ke 1,2 dan 4. kemudian kita akan mengkitung cost dari simpul 6,7dan 8.

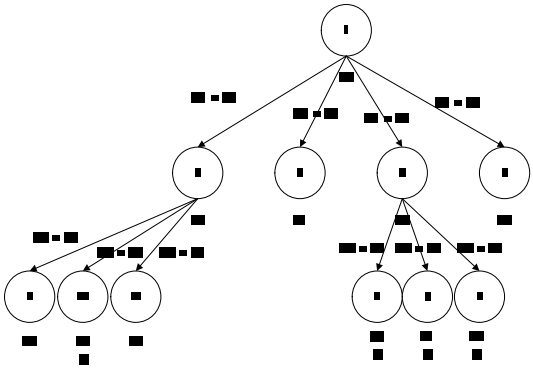
$\hat{c}(6) = (50+12) + 3*(16-10-2) = 74$
$\hat{c}(7) = (50+15) + 6*(16-10-5) = 71$
$\hat{c}(8) = (50+10) + 6*(16-10-5) = 66$

Setelah melakukan perhitungan, kita dapat melihat bahwa ketiga simpul yaitu simpul 6,7 dan 8 tidak bisa lagi diperluas oleh karena itu ketiga simpul ini akan dibunuh dengan menambahkan keterangan huruf **B** pada bagian bawah simpulnya. Simpul



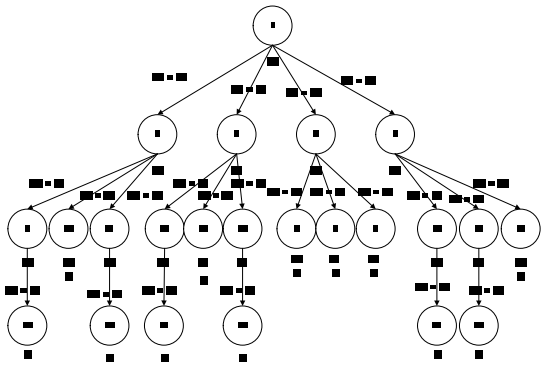
Gambar 4. pohon status knapsack problem tahap kedua

Sekarang kita hanya memiliki simpul 2,3 dan 5 sebagai simpul hidup. Diantara simpul-simpul tersebut, simpul 2 lah yang memiliki cost terbesar sehingga selanjutnya akan dilakukan perluasan pada simpul 2 dengan metode yang serupa dan diperoleh simpul 9,10, dan 11 dengan cost 72,74, dan 67. pada simpul 2, dapat dilihat bahwa simpul tersebut tidak dapat lagi diperluas oleh karena itu makan simpul ini akan dibunuh.



Gambar 5. pohon status knapsack problem tahap ketiga

Sekarang kita memiliki simpul 3,5,9 dan 11 sebagai simpul hidup dan diantar simpul-simpul tersebut, simpul 3 lah yang memiliki cost terbesar sehingga simpul tersebut akan diperluas dengan metode yang diterapkan pada perluasan simpul-simpul sebelumnya. Perluasan simpul-simpul hidup terus dilakukan sampai tidak ada lagi simpul hidup yang ditemukan dan dapat diperluas atau semua simpul telah dibunuh. Hal ini dilakukan karena pencarian solusi pada permasalahan ini tidak dapat diketahui kapan kita harus berhenti karena telah sampai kepada solusi tetapi penyelesaian masalah akan dilakukan dengan membandingkan solusi pada tiap daun. Dalam persoalan ini, akan dibandingkan jumlah keuntungan yang diperoleh pada tiap daun dan solusi permasalah merupakan lintasan yang menuju daun yang memiliki keuntungan terbesar.



Gambar 6. pohon status knapsack problem

Dari gambar diatas dapat disimpulkan bahwa solusi permasalahan dibentuk oleh lintasan yang menuju simpul daun 13 dan 7 dimana keduanya merupakan solusi yang serupa yaitu memilih barang 2 dengan berat 5 kg , keuntungan 15 dan barang 3 dengan berat 10 kg, keuntungan 50 sebagai solusi permasalahan dan akan menghasilkan keuntungan yang maksimal pada penjualan yaitu sebesar 65.

IV. KESIMPULAN

Algoritma Branch and Bound merupakan salah satu algoritma yang dapat diterapkan dalam mencari penyelesaian pada permasalahan knapsack. Dengan batas-batas yang ditentukan , algoritma ini mampu mencapai solusi yang optimum seperti yang berhasil dilakukan oleh algoritma exhautive search, namun dengan membunuh semua simpul yang tidak mungkin diperluas, algoritma branch and bound ini tidak memperhitungkan semua kemungkinan yang ada sehingga akan lebih mengefisienkan waktu yang digunakan untuk pemecahan masalah ini.

Menurut saya, algoritma ini masih kurang mangkus untuk diterapkan sebagai strategi pencarian solusi pada permasalahan knapsack karena tidak diketahuinya batasan kapan kita sampai pada tahap solusi. Dibandingkan penerapan algoritma branch and bound pada permasalahan n ratu atau tsp yang dapat dengan jelas diketahui keadaan dimana sudah tercapai solusi tanpa harus terus memperluas semua simpul hidup sampai tidak ada lagi simpul hidup yang tersisa, algoritma ini terasa kurang efisien untuk digunakan dalam mencari solusi pada permasalahan ini karena cukup rumit dan memakan banyak waktu meskipun tidak selama waktu yang dibutuhkan algoritma brute force untuk menyelesaikan permasalahan ini.

REFERENSI

[1] Munir, Rinaldi, *Strategi Algoritmik*,Bandung, 2007.