Digital Repository Universitas Jember

PENERAPAN ALGORITMA *DYNAMIC PROGRAMMING* DAN ALGORITMA *BACKTRACKING* PADA PERMASALAHAN *MULTIPLE CONSTRAINTS KNAPSACK* 0-1

SKRIPSI

oleh

Fahma Hilviah
NIM 111810101044

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015

Digital Repository Universitas Jember

PENERAPAN ALGORITMA *DYNAMIC PROGRAMMING* DAN ALGORITMA *BACKTRACKING* PADA PERMASALAHAN *MULTIPLE CONSTRAINTS KNAPSACK* 0-1

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarjana Sains

oleh

Fahma Hilviah
NIM 111810101044

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015

Digital Repository Universitas Jember

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Kakek Zainullah dan Nenek Nurhayati serta Almarhum Kakek Sarrip dan Nenek Hasani yang senantiasa memberi semangat dan doa sepanjang waktu;
2. Ayahanda Sumarto dan Ibunda Latifah tercinta yang senantiasa memberi doa, inspirasi, semangat, dan kasing sayang;
3. Adikku Ahmad Rizqian Fadlianto yang selalu memberi semangat;
4. M. Ziaul Arif, S.Si., M.Sc selaku Dosen Pembimbing Utama dan Ahmad Kamsyakawuni S. Si, M. Kom selaku Dosen Pembimbing Anggota yang telah memberikan bantuan dan bimbingan secara intensif untuk penyempurnaan skripsi ini;
5. Seluruh guru dan dosen sejak taman kanak-kanak hingga perguruan tinggi yang telah memberi banyak ilmu dan membimbing dengan tulus;
6. Almamater Jurusan Matematika FMIPA Universitas Jember, SMA Negeri 3 Bondowoso, SMP Negeri 2 Bondowoso, SDN Grujugan Kidul 01, dan TK PGRI 01;
7. Seluruh keluarga besar KRAMAT'11, HIMATIKA "Geokompstat", sahabat-sahabat Kost R 36 dan I 33.

Digital Repository Universitas Jember

MOTTO

"Ilmu pengetahuan tanpa agama lumpuh, agama tanpa ilmu pengetahuan buta"

(Albert Einstein) *)

"Bukan bahagia yang membuat kita bersyukur, melainkan bersyukur yang membuat kita bahagia"

(Sastra Sekura)**)

*) Albert Einstein dalam kata mutiara www.duniakata.com/2014/11/45-kata-mutiara-tentang-pentingnya-ilmu.html

**) Mozaik Karya Sastra Sekura http://sastra-sekura.blogspot.co.id

Digital Repository Universitas Jember

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

nama : Fahma Hilviah

NIM : 111810101044

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Penerapan Algoritma *Dynamic Programming* dan Algoritma *Backtracking* Pada Permasalahan *Multiple Constraints Knapsack* 0-1" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan dalam institusi manapun dan juga bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya,tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, Desember 2015
Yang menyatakan,

Fahma Hilviah
NIM 111810101044

Digital Repository Universitas Jember

SKRIPSI

PENERAPAN ALGORITMA *DYNAMIC PROGRAMMING*
DAN ALGORITMA *BACKTRACKING*
PADA PERMASALAHAN *MULTIPLE CONSTRAINTS KNAPSACK* 0-1

oleh

Fahma Hilviah
NIM 111810101044

Pembimbing

Dosen Pembimbing Utama : M. Ziaul Arif, S.Si., M.Sc

Dosen Pembimbing Anggota : Ahmad Kamsyakawuni S. Si., M. Kom

Digital Repository Universitas Jember

PENGESAHAN

Skripsi berjudul "Penerapan Algoritma *Dynamic Programming* dan Algoritma *Backtracking* Pada Permasalahan *Multiple Constraints Knapsack* 0-1" telah diuji dan disahkan pada:

Hari, tanggal :

Tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji:

Ketua,	Sekretaris,
M. Ziaul Arif, S. Si., M. Sc	Ahmad Kamsyakawuni, S. Si., M. Kom
NIP. 198501112008121002	NIP. 197211291998021001
Anggota I,	Anggota II,
Prof. Drs. Kusno, DEA, Ph.D.	Dian Anggaraeni, S. Si., M. Si
NIP. 196101081986021001	NIP. 198202162006042002

Mengesahkan
Dekan,

Prof. Drs. Kusno, DEA, Ph.D.
NIP. 196101081986021001

Digital Repository Universitas Jember

RINGKASAN

Penerapan Algoritma *Dynamic Programming* dan Algoritma *Backtracking* Pada Permasalahan *Multiple Constraints Knapsack* 0-1; Fahma Hilviah, 111810101044; 2015; 44 Halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Knapsack merupakan suatu permasalahan bagaimana memilih objek dari sekian banyak objek dengan batasan objek tersebut sama atau lebih kecil dari kapasitas media penyimpannya sehingga diperoleh suatu penyimpanan yang optimal. *Knapsack* dapat diilustrasikan sebagai suatu kantong atau media penyimpanan. Perkembangan teknologi saat ini memungkinkan pencarian solusi optimal tersebut dilakukan oleh aplikasi komputer karena keterbatasan manusia untuk menyelesaikan masalah *knapsack*. Masalah *knapsack* dapat dilakukan dengan berbagai macam algoritma, diantaranya adalah menggunakan algoritma *Dynamic Programming* dan algoritma *Backtracking*. Tujuan penelitian ini adalah untuk menyelesaikan masalah *Multiple Constraints Knapsack Problem* (MCKP) 0-1 menggunakan algoritma *Dynamic Programming* dan algoritma *Backtracking* dan membandingkan kedua algoritma tersebut.

Penelitian ini dilakukan di industri Usaha Dagang (UD) Indo Handicraft yang terletak di Jalan Raya Tamanan, Desa Grujugan Kidul, Kecamatan Grujugan, Kabupaten Bondowoso. Pengambilan data dilakukan dengan metode wawancara dan data yang diambil berupa data harga beli per paket, harga jual per paket, dan berat barang per paket. Untuk menerapkan data tersebut dilakukan pengidentifikasian untuk mencari keuntungan (p_j). Tujuan dari peneliti adalah untuk mencari keuntungan maksimum di UD Indo Handicraft pada permasalahan *Multiple Constraints Knapsack Problem* (MCKP) 0-1 menggunakan algoritma *Dynamic Programming* dan algoritma *Backtracking*.

Digital Repository Universitas Jember

Berdasarkan hasil dan pembahasan menunjukkan bahwa setelah dilakukan pemilihan barang yang dibeli oleh UD Indo Handicraft, didapatkan keuntungan maksimum menggunakan algoritma *Dynamic Programming* adalah sebesar Rp. 2.238.000,- dengan berat maksimum 228 kg dan modal maksimum Rp. 6.578.000,- dengan waktu proses 29,2586 detik. Pada algoritma *Backtracking* diperoleh keuntungan maksimum sebesar Rp. 2.238.000,- dengan berat maksimum 228 kg dan modal maksimum Rp. 6.578.000,- dengan waktu proses 1119,8306 detik. Dari hasil kedua algoritma di atas dapat disimpulkan bahwa algoritma *Dynamic Programming* dan algortima *Backtracking* menghasilkan keuntungan yang sama, tetapi dalam segi lama waktu program dijalankan (*running time*), algoritma *Dynamic Programming* membutuhkan waktu yang lebih singkat daripada algoritma *Backtracking*. Dengan kata lain, algoritma *Dynamic Programming* lebih efektif dan efisien untuk menyelesaikan permasalahan *Multiple Constraints Knapsack* 0-1.

Digital Repository Universitas Jember

PRAKATA

Segala puji syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan segala berkat, rahmat, serta hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penerapan Algoritma *Dynamic Programming* dan Algoritma *Backtracking* pada Permasalahan *Multiple Constraints Knapsack* 0-1". Skripsi ini disusun untuk memenuhi salah satu syarat dalam menyelesaikan pendidikan strata 1 (S1) di Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak, baik bantuan secara langsung maupun tidak langsung. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kakek Zainullah dan Nenek Nurhayati serta Almarhum Kakek Sarrip dan Nenek Hasani yang senantiasa memberi semangat dan doa sepanjang waktu;
2. Ayahanda Sumarto dan Ibunda Latifah tercinta yang senantiasa memberi doa, inspirasi, semangat, dan kasing sayang;
3. Adikku Ahmad Rizqian Fadlianto yang selalu memberi semangat;
4. M. Ziaul Arif, S.Si., M.Sc selaku Dosen Pembimbing Utama dan Ahmad Kamsyakawuni S. Si, M. Kom selaku Dosen Pembimbing Anggota yang telah memberikan bantuan dan bimbingan secara intensif untuk penyempurnaan skripsi ini;
5. Prof. Drs. Kusno, DEA, Ph.D dan Dian Anggraeni S.Si, M.Si selaku Dosen Penguji yang telah memberikan kritik dan saran demi kesempurnaan skripsi ini;
6. Seluruh dosen dan karyawan Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam yang telah memberikan ilmu serta nasehat selama proses perkuliahan;

Digital Repository Universitas Jember

7. Almamater Jurusan Matematika FMIPA Universitas Jember, SMA Negeri 3 Bondowoso, SMP Negeri 2 Bondowoso, SDN Grujugan Kidul 01, dan TK PGRI 01;
8. Ivan Syaikhul Hadi, S.Si., yang telah menjadi *partner* terbaik selama skripsi, selalu memberikan semangat, selalu mendoakan, senantiasa mengisi hati serta sabar menemani;
9. Keluarga besar KRAMAT'11, sahabat-sahabat Kost R 36 dan I 33 yang senantiasa saling mendukung untuk menjadi yang terbaik;
10. Pengurus HIMATIKA "Geokompstat" periode 2012 – 2013 yang senantiasa mengajarkan banyak hal;
11. Semua pihak yang tidak dapat disebutkan satu per satu.

Skripsi ini juga tidak lepas dari kekurangan dan kesalahan baik isi maupun susunannya. Oleh karena itu, penulis menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap semoga skripsi ini dapat memberi manfaat bagi pembaca dan kehidupan.

Jember, Desember 2015 Fahma Hilviah

Digital Repository Universitas Jember

DAFTAR ISI

Halaman

Digital Repository Universitas Jember

Digital Repository Universitas Jember

DAFTAR TABEL

Halaman

Digital Repository Universitas Jember

DAFTAR GAMBAR

Halaman

Digital Repository Universitas Jember

DAFTAR LAMPIRAN

Halaman

Digital Repository Universitas Jember

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Setiap manusia pasti menginginkan keuntungan yang maksimal dengan sumber daya yang seminimal mungkin. Salah satu contoh dalam kehidupan sehari-hari adalah persoalan media penyimpanan yang terbatas tetapi diharuskan menyimpan beberapa objek ke dalam media untuk mendapatkan keuntungan yang maksimal. Dari permasalahan tersebut, munculah suatu permasalahan yang dikenal dengan "Masalah *Knapsack* (*Knapsack Problem*)". *Knapsack* dapat diilustrasikan sebagai suatu kantong atau media penyimpanan. Masalah *knapsack* merupakan suatu permasalahan bagaimana memilih objek dari sekian banyak objek dengan batasan objek tersebut sama atau lebih kecil dari kapasitas media penyimpannya sehingga diperoleh suatu penyimpanan yang optimal.

Perkembangan teknologi saat ini memungkinkan pencarian solusi optimal tersebut dilakukan oleh aplikasi komputer karena keterbatasan manusia untuk menyelesaikan masalah *knapsack*. Selain efisiensi waktu yang menjadi pertimbangan, perhitungan juga akan semakin sulit jika objek yang akan dipilih sangat banyak. Oleh karena itu, dibutuhkan suatu metode sekaligus program aplikasi penerapan metode tersebut yang dapat membantu menyelesaikan masalah *knapsack.*

Masalah *knapsack* terbagi menjadi tiga, yaitu *knapsack* 0-1, *bounded knapsack,* dan *unbounded knapsack*. Penelitian sebelumnya tentang *knapsack*, dilakukan oleh Arista (2007) mengkaji masalah *knapsack* 0-1 diselesaikan dengan menggunakan algoritma *Greedy* dan *Dynamic Programming*. Pada kajian tersebut disimpulkan bahwa berdasarkan keuntungan, algoritma *Dynamic Programming* lebih baik daripada algoritma *Greedy.* Pada penelitian lain, dilakukan oleh Ridho (2008) yang mengkaji masalah *knapsack* 0-1 diselesaikan dengan algoritma *Harmony*

Digital Repository Universitas Jember

Search (HS) dan *Dynamic Programming*. Pada kajian tersebut disimpulkan bahwa algoritma *Dynamic Programming* dan algoritma *Harmony Search* (HS) menghasilkan keuntungan yang sama tetapi langkah perhitungan algoritma *Dynamic Programming* lebih sederhana dan cepat.

Algoritma *Dynamic Programming* merupakan salah satu metode dari sekian banyak metode yang dapat digunakan untuk menyelesaikan permasalahan *knapsack*. Contoh metode lain yang dapat digunakan untuk menyelesaikan permasalahan *knapsack* adalah algoritma *Backtracking*. Algoritma *Backtracking* mempunyai prinsip dasar yang sama seperti *Brute-Force* yaitu mencoba segala kemungkinan solusi. Perbedaan utamanya adalah pada ide dasarnya, semua solusi dibuat dalam bentuk pohon solusi dan algoritma akan menelusuri pohon tersebut secara *DFS* (*Depth First Search)* sampai ditemukan solusi yang layak.

Knapsack memiliki beberapa variasi, diantaranya adalah *knapsack* tujuan ganda (*multi objective knapsack*), *knapsack* beberapa kendala (*multi constraints* atau *multidimensional knapsack*), *multiple knapsack* dan *quadratic knapsack*. Adanya suatu permasalahan yang kompleks ketika memilih objek dihadapkan pada lebih dari satu kendala, hal ini biasa disebut dengan permasalahan *knapsack* beberapa kendala atau *Multiple Constraints Knapsack Problem* (MCKP).

MCKP banyak diterapkan dalam berbagai bidang, salah satunya adalah di bidang bisnis ekonomi. Pada penelitian ini, data yang digunakan adalah data yang diambil dari Usaha Dagang (UD) Indo Handicraft yang terletak di Jalan Raya Tamanan, Desa Grujugan Kidul, Kecamatan Grujugan, Kabupaten Bondowoso. UD Indo Handicraft merupakan salah satu perusahaan yang menjual berbagai barang kerajinan tangan. Dalam proses pemilihan barang, UD Indo Handicraft selalu berusaha untuk memaksimalkan keuntungan usahanya tetapi terdapat kendala yang harus diperhatikan yaitu besar kapasitas maksimum muat barang dan modal untuk membeli barang. Dengan dua kendala yang harus dipertimbangkan maka proses optimasi akan semakin rumit. Berdasarkan masalah diatas, penulis tertarik untuk

Digital Repository Universitas Jember

mengangkat judul yang akan diteliti yaitu "Penerapan Algoritma *Dynamic Programming* dan Algoritma *Backtracking* pada Permasalahan *Multiple Constraints Knapsack* 0-1"

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka permasalahan yang akan dibahas adalah sebagai berikut:

a. bagaimana menyelesaikan masalah *Multiple Constraints Knapsack Problem* (MCKP) 0-1 menggunakan algoritma *Dynamic Programming* dan algoritma *Backtracking*?*;*
b. bagaimana perbandingan kedua algoritma tersebut berdasarkan hasil keuntungan maksimal yang didapat dan lama program dijalankan (*running time*)?.

1.3 Batasan Masalah

Adapun batasan permasalahan yang harus dibatasi pada penyelesaian permasalahan *Multiple Constraints Knapsack Problem* (MCKP) 0-1 adalah tingkat penjualan atau permintaan konsumen untuk masing-masing barang diasumsikan sama.

1.4 Tujuan

Tujuan dari penulisan skripsi ini adalah sebagai berikut.

a. menyelesaikan masalah *Multiple Constraints Knapsack Problem* (MCKP) 0-1 menggunakan algoritma *Dynamic Programming* dan algoritma *Backtracking;*
b. mengetahui perbandingan kedua algoritma tersebut berdasarkan hasil keuntungan maksimal dan lama waktu program dijalankan (*running time*).

Digital Repository Universitas Jember

1.5 Manfaat

Manfaat yang diperoleh dari penulisan skripsi ini adalah memberikan alternatif solusi dalam menyelesaikan permasalahan *Multiple Constraints Knapsack Problem* (MCKP) 0-1.

Digital Repository Universitas Jember

BAB 2. TINJAUAN PUSTAKA

2.1 Program Linier

Program linier dengan teknik optimasi linier adalah upaya menyelesaikan suatu masalah, dimana semua fungsi matematika yang digunakan dalam program linier merupakan fungsi linier. Program linier adalah salah satu teknik dalam riset operasi untuk memecahkan persoalan optimasi dengan menggunakan persamaan dan ketidaksamaan linier dalam rangka untuk mencari pemecahan yang optimum dengan memperhatikan pembatasan-pembatasan yang ada.

Optimasi berarti pencarian nilai terbaik dari beberapa fungsi yang diberikan pada suatu konteks. Secara umum optimasi adalah suatu proses untuk mencapai hasil yang optimal (nilai efektif yang dapat dicapai) pada masalah yang berhubungan dengan keputusan yang terbaik, maksimum, minimum, dan memberikan cara penentuan solusi yang memuaskan. Untuk dapat mencapai nilai optimal baik minimal atau maksimal tersebut, secara sistematis dilakukan pemilihan nilai variabel integer atau nyata yang akan memberikan solusi optimal. Berikut ini adalah beberapa persoalan yang memerlukan optimasi yang sering muncul dalam bentuk program linier (Munir, 2005):

a. penentuan pemilihan barang pada masalah *knapsack*;
b. menentukan lintasan terpendek dari suatu tempat ke tempat yang lain;
c. menentukan jumlah pekerja seminimal mungkin untuk melakukan suatu proses produksi agar pengeluaran biaya pekerja dapat diminimalkan dan hasil produksi tetap maksimal;
d. mengatur jalur kendaraan umum agar semua lokasi dapat dijangkau.

Digital Repository Universitas Jember

2.2 *Knapsack Problem*

Knapsack problem merupakan suatu permasalahan bagaimana memilih objek dari sekian banyak objek dan seberapa besar objek tersebut akan disimpan sehingga diperoleh suatu penyimpanan yang optimal dengan memperhatikan objek yang terdiri dari n objek $(1, 2, 3, \dots, n)$ dimana setiap objek memiliki bobot (w_i), nilai atau profit (p_i) dan kapasitas media penyimpanan (M). *Knapsack problem* merupakan sebuah persoalan yang sering digunakan pada bidang (jasa) pengangkutan barang seperti pengangkutan peti kemas dalam sebuah media pengangkut. Dalam usaha tersebut, diinginkan keuntungan yang maksimal untuk mengangkut barang dengan tidak melebihi kapasitas yang ada. Berdasarkan persoalan tersebut, diharapkan ada suatu solusi yang secara otomatis dapat mengatasi persoalan itu. Menurut Dimyati dan ahmad (2004), *knapsack* adalah permasalahan mengenai optimalisasi kombinatorial dimana kita harus mencari solusi terbaik dari banyak kemungkinan yang dihasilkan.

Jenis *knapsack problem* yang dikenal antara lain: *knapsack problem* 0-1 dimana objek yang dimasukkan ke dalam media penyimpanan dimensinya harus dimasukkan semua (1) atau tidak sama sekali (0), *bounded knapsack problem* dimana setiap objek tersedia sebanyak n unit dan jumlah objek yang dimasukkan ke dalam media penyimpanan terbatas dan *unbounded knapsack problem* dimana Setiap objek tersedia lebih dari 1 unit dan jumlah objek yang dimasukkan ke dalam media penyimpanan macamnya tidak terbatas (Pisinger, 1995).

Ada banyak variasi dari masalah *knapsack* yang muncul dengan aplikasi jumlah besar dari masalah dasar. Variasi utama terjadi dengan mengubah jumlah beberapa parameter masalah seperti jumlah kendala, jumlah tujuan, atau bahkan jumlah medianya. Adapun variasi permasalahan *knapsack* diatas antara lain (Kellerer *et al*, 2004):

a. *Multi Objective Knapsack Problem*

   Permasalahan yang memiliki fungsi tujuan lebih dari 1 untuk memaksimalkan keuntungannya;

Digital Repository Universitas Jember

b. *Multi Demensional* atau *Multiple Constrains Knapsack Problem*

   Permasalahan yang memiliki kendala lebih dari 1 untuk memaksimalkan keuntungannya;

c. *Multi Knapsack Problem*

   Permasalahan yang memiliki fungsi objektif/media penyimpanan lebih dari 1 dimana semua item harus dikemas untuk memaksimalkan keuntungannya;

d. *Quadartic Knapsack Problem*

   Permasalahan yang tujuannya memaksimalkan fungsi objektif dalam bentuk kuadratik untuk kendala kapasitas biner dan linear.

Knapsack yang akan dibahas pada skripsi ini adalah jenis *Multiple Constrains Knapsack Problem* (MCKP) 0-1 dimana variabel keputusan yang diperoleh adalah x_i bernilai 1 jika objek dipilih dan x_i bernilai 0 jika objek tidak dipilih.

2.2.1 *Knapsack* 0-1 *Problem*

Dalam persoalan ini, kita diberikan *n* buah objek dan sebuah media penyimpanan yang memiliki daya tampung maksimal senilai *M*. Setiap benda memiliki bobot (w_i) dengan nilai keuntungan profit (p_i). Permasalahannya adalah bagaimana memilih objek-objek yang dimasukkan ke dalam media penyimpanan sehingga tidak melebihi kapasitas dari media penyimpanan namun memaksimalkan total keuntungan yang diperoleh. Pada persoalan *knapsack* 0-1 barang yang diangkut dimensinya harus diangkut seluruhnya atau tidak sama sekali (Springer, 2005).

Permasalahan *knapsack* 0-1 memiliki solusi penyelesaian yang dinyatakan sebagai himpunan:

$$X = x_1, x_2, x_3, \dots, x_n$$

dimana $x_i = 1$ jika benda ke-i dimasukkan kedalam media penyimpanan, dan $x_i = 0$ jika benda ke-i tidak dimasukkan kedalam media penyimpanan. Misalkan solusi dari suatu permasalahan *knapsack* adalah $X = \{0,1,0,1\}$, itu berarti benda ke-1 dan ke-3 tidak dimasukkan ke dalam media penyimpanan sedangkan benda ke-2 dan ke-4

Digital Repository Universitas Jember

dimasukkan ke dalam media penyimpanan. Secara matematik, permasalahan *knapsack* bilangan bulat merupakan program linier bilangan bulat, maka formula permasalahan *knapsack* 0-1 menurut Kellerer *et al.* (2004) adalah sebagai berikut.

$$\text{Fungsi tujuan Maks/Min} \quad : Z = \sum_{i=1}^{n} p_i x_i$$

$$\text{Kendala} \quad : \sum_{i=1}^{n} w_i x_i \leq M, \quad x_i \in \{0,1\} \qquad (2.1)$$

dengan,

Z : nilai optimum dari fungsi tujuan

n : banyak barang

p_i : keuntungan barang ke-i , dimana $i = 1,2, \dots, n$

w_i : berat barang ke-i, dimana $i = 1,2, \dots, n$

x_i : variabel keputusan (1 jika dipilih, 0 jika tidak dipilih)

M : kapasitas media penyimpanan (*knapsack*)

2.2.2 *Multiple Constrains Knapsack Problem* (MCKP)

Multiple Constrains Knapsack Problem (MKCP) adalah suatu permasalahan *knapsack* yang sering disebut juga dengan *Multidimensional Knapsack Problem*. Pada *Multiple Constrains Knapsack Problem* (MCKP), tiap-tiap item pilihan memiliki lebih dari satu dimensi batasan (kendala) seperti berat, rasa, biaya, waktu, pekerja, dan seterusnya (Puchinger *et al*, 2007). Tujuannya untuk memperoleh solusi optimal dengan memilih kombinasi barang sedemikian rupa dimana semua batasan tidak melewati kapasitas yang tersedia. *Multiple Constrains Knapsack Problem* (MCKP) dapat dirumuskan sebagai berikut (Kellerer *et al*, 2004).

$$\text{Fungsi tujuan} \quad : Z = \sum_{j=1}^{n} p_j x_j$$

$$\text{Kendala} \quad : \sum_{j=1}^{n} w_{ij} x_j \leq c_i , \ i = 1, \dots, m \qquad (2.2)$$

$$x_j \in \{0,1\}, \quad j = 1, \dots, n$$

Digital Repository Universitas Jember

dengan,

Z	: nilai optimum dari fungsi tujuan
i	: sumber
j	: barang
p_j	: keuntungan tiap barang ke-j ($p_j > 0$)
x_j	: variabel keputusan (1 jika dipilih dan 0 jika tidak)
w_{ij}	: nilai tiap sumber ke- i ($w_{ij} \geq 0$)
c_i	: kapasitas tiap sumber ($c_i > 0$)
m	: banyak sumber
n	: banyak barang

Perumusan matematik *Multiple Constraints Knapsack Problem* (MCKP) diatas memformulasikan perolehan nilai optimal (Z) dari banyaknya barang (n), banyaknya sumber (m), dan keuntungan (p_j) dengan kapasitas c_i. Setiap barang ke-j memiliki nilai dari tiap sumber ke-i (w_{ij}). Variabel keputusan (x_j) 0 atau 1 mengindikasi barang-barang yang dipilih. Tujuan yang ingin diraih adalah memilih subset dari sekian banyak barang dengan total keuntungan (p_j) yang maksimal. Barang-barang yang dipilih tidak boleh melebihi kapasitas dari sumber (c_i), seperti didefinisikan pada kendala *knapsack* diatas.

2.3 Algoritma

Algoritma merupakan langkah-langkah penyelesaian masalah secara sistematis. Sebuah algoritma tidak hanya harus benar tetapi juga harus efisien. Algoritma dikatakan efisien jika waktu tempuh (*running time*) dan berapa ruang dalam memori yang dibutuhkan untuk menjalankan algoritma sangat cepat. Meskipun suatu algoritma memberikan hasil yang mendekati optimal tetapi waktu yang dibutuhkan sangat lama maka algoritma tersebut biasanya jarang dipakai

Digital Repository Universitas Jember

(Munir, 2005). Algoritma juga merupakan serangkaian kata atau instruksi untuk mendapatkan hasil khusus dalam beberapa langkah berhingga (Chartrand dan Oerllemann, 1993). Sifat-sifat algoritma meliputi hal-hal sebagai berikut:

a. jelas, yaitu setiap langkah pada setiap algoritma harus dinyatakan dengan jelas, tidak bermakna ganda dan ditetapkan dengan cermat;
b. logis (urut), yaitu algoritma dibuat berdasarkan aturan yang tetap, berdasarkan pada alur berfikirnya;
c. terhingga, yaitu sebuah algoritma harus berhenti setelah melakukan satu langkah atau lebih dalam suatu interval waktu tertentu. Tanpa sifat ini, sebuah algoritma tidak bisa diimplementasikan oleh manusia ataupun mesin;
d. menyelesaikan masalah, yaitu dengan input tertentu suatu algoritma akan menyelesaikan masalah dalam kelasnya;
e. efektif, yaitu sebuah algoritma harus menegaskan tindakan sederhana yang dapat dilakukan secara efektif. Sifat ini menjamin bahwa setiap langkah pada suatu algoritma secara nyata dapat dijalankan.

2.4 Algoritma *Dynamic Programming*

Dynamic programming adalah prosedur matematis yang terutama dirancang untuk memperbaiki efisien perhitungan masalah pemrograman matematis tertentu dengan menguraikannya menjadi bagian-bagian masalah yang lebih kecil atau sub-sub masalah sehingga perhitungannya lebih sederhana. Suatu masalah yang diselesaikan dengan *Dynamic Programming* dapat dibagi dalam tahap-tahap. Setiap tahap mewakili kemungkinan kapasitas dari masalah. Banyaknya tahap ditentukan pada awal *Dynamic Programming*. Selanjutnya kemungkinan kapasitas dimulai dengan mengambil kemungkinan kapasitas minimal (0) sampai kemungkinan kapasitas maksimal (kemungkinan kapasitas yang tersedia). Untuk mencari total keuntungan optimal, dicari terlebih dahulu keuntungan yang diperoleh dari tiap kelompok barang dengan kemungkinan kapasitasnya. Perhitungan di setiap

Digital Repository Universitas Jember

kelompok barang melalui persamaan rekursif dan selanjutnya menghasilkan penyelesaian optimal yang mungkin bagi seluruh barang (Taha, 1996).

Menurut Cormen *et al.* (2007), *Dynamic Programming* memecahkan setiap sub-sub masalah hanya sekali dan kemudian menyimpan jawabannya dalam sebuah tabel, sehingga menghindari pekerjaan komputasi berulang untuk memperoleh jawabannya setiap kali sub-sub masalah ditemukan. *Dynamic Programming* biasanya diterapkan untuk masalah optimasi. Dalam hal ini, bisa ada banyak terhingga kemungkinan solusi sehingga akan mungkin ada beberapa solusi yang mencapai solusi optimal.

2.4.1 Unsur-unsur *Dynamic Programming*

Adapun unsur-unsur yang harus dipahami pada *Dynamic Programming* menurut Taha (1996) yaitu:

a. Tahap

   Tahap dapat diartikan sebagai bagian dari masalah pada *Dynamic Programming* yang mengandung variabel keputusan yang akan dipilih salah satu atau lebih yang paling optimal.

b. Keadaan

   Keadaan merupakan kondisi yang memungkinkan pada suatu tahap tertentu yang menunjukan kaitan antara tahap satu dengan tahap berikutnya.

c. Variabel keputusan

   Variabel keputusan adalah variabel yang menguraikan secara lengkap keputusan-keputusan yang dibuat.

d. Nilai fungsi

   Nilai fungsi adalah nilai yang ditentukan dari variabel keputusan.

e. Keputusan optimal

   Keputusan optimal adalah keputusan yang merupakan solusi terbaik dari masalah.

Digital Repository Universitas Jember

2.4.2 Langkah- langkah algoritma *Dynamic Programming*

Adapun kangkah-langkah untuk algoritma *Dynamic Programming* adalah sebagai berikut :

a. Menentukan struktur dari masalah yaitu dengan menunjukkan bahwa masalah dapat dipisahkan kedalam submaslah-submasalah optimal.
b. Menentukan persamaan rekursif untuk mencari solusi optimal untuk submasalah.
c. Menghitung nilai dari solusi optimal, dimana solusi optimal pada langkah ini merupakan solusi optimal dari setiap submasalah dan solusi-solusi tersebut disimpan dalam suatu tabel yang dihitung dengan menggunakan persamaan rekursif yang sudah ditentukan pada langkah b.
d. Menentukan keputusan optimal dari masalah dimana keputusan optimal tersebut diambil dari tabel yang berisi solusi optimal dari setiap submasalah.

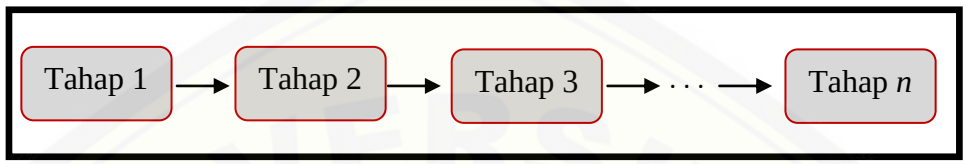
2.4.3 Prosedur Rekursif

Prosedur penyelesaian suatu masalah dengan *Dynamic Programming* yaitu dengan prosedur rekursif yang berarti bahwa setiap kali mengambil keputusan harus memperhatikan keadaan yang dihasilkan oleh keputusan optimal sebelumnya dan merupakan landasan bagi keputusan optimal berikutnya. Prosedur penyelesaian rekursif ada 2 macam yaitu prosedur maju (*forward procedure*) dan prosedur mundur (*backward procedure*). Prosedur maju (*forward procedure*) merupakan suatu prosedur perhitungan yang dimulai dari tahap pertama ke tahap akhir dimana tahapan tersebut dilakukan sebanyak *n* tahapan atau sebanyak jumlah barang. Perhitungan dimulai dengan mencari nilai keuntungan di tahap 1 lalu dilanjutkan mencari keuntungan di tahap 2 sampai tahap *n*. Setelah dilakukan perhitungan diperoleh keputusan optimal (lihat gambar 2.1). Sedangkan prosedur mundur (*backward procedure*) merupakan suatu prosedur perhitungan yang dimulai pada tahap terakhir dan berlanjut ke belakang ke tahap 1. Perhitungan dimulai dengan mencari nilai

Digital Repository Universitas Jember

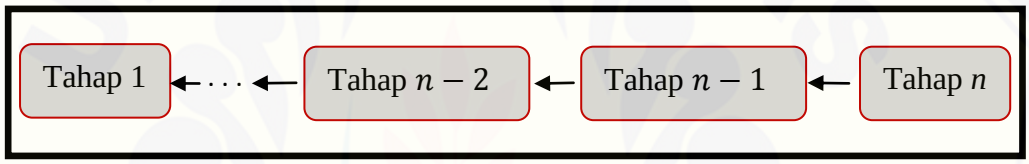
keuntungan di tahap n lalu dilanjutkan mencari nilai keuntungan $n-1$ sampai tahap 1. Setelah dilakukan perhitungan diperoleh keputusan optimal (lihat Gambar 2.2).

Model prosedur maju (*forward procedure*):



Gambar 2.1 Perhitungan Rekursif Maju Algoritma *Dynamic Programming*

Model prosedur mundur (*backward procedure*):



Gambar 2.2 Perhitungan Rekursif Mundur Algoritma *Dynamic Programming*

Dari ilustrasi di atas dimisalkan bahwa suatu masalah dapat dipecahkan dengan n tahapan. Pada ilustrasi Gambar 2.1, menggambarkan rekursif maju (*forward recursion*), sedangkan Gambar 2.2 di atas menggambarkan rekursif mundur (*backward recursion*). Perbedaan pokok antara rekursif maju dan rekursif mundur terletak pada cara mendefinisikan tahap yang dipakai, apakah digunakan tahap pertama atau tahap terakhir untuk memulai perhitungan. Secara matematik, perhitungan *Dynamic Programming* untuk kasus *Multiple Constraints Knapsack Problem* (MCKP) menggunakan persamaan 2.2.

2.4.4 Perhitungan Algoritma *Dynamic Programming* pada *Multiple Constraints Knapsack Problem* (MCKP) 0-1

Dynamic programming merupakan metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*)

Digital Repository Universitas Jember

sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Menurut Prasetiyowati dan Wicaksana (2013), penyelesaian masalah dengan metode *Dynamic Programming* ini terdapat beberapa persoalan, antara lain:

a. terdapat sejumlah berhingga pilihan yang mungkin;
b. solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya;
c. menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Penerapan algoritma *Dynamic Programming* rekursif maju (*forward recursion*) pada persoalan *Multiple Constraints Knapsack Problem* (MCKP) 0-1 adalah sebagai berikut (Kellerer *et al*, 2004):

a. tahap (k) adalah proses memasukkan objek ke dalam c_i,
b. status $(c_1, \dots, c_m)$ menyatakan kapasitas muat c_i yang tersisa setelah memasukkan objek pada tahap sebelumnya.
    1) Dari tahap ke-1, kita masukkan objek ke-1 ke dalam c_i untuk setiap satuan kapasitas c_i sampai batas kapasitas maksimumnya. Karena kapasitas c_i adalah bilangan bulat, maka pendekatan ini praktis.
    2) Misalkan ketika memasukkan objek pada tahap k, kapasitas muat c_i sekarang adalah $(c_1 - w_{1j}, \dots, c_m - w_{mj})$.
    3) Untuk mengisi kapasitas sisanya, kita menerapkan prinsip optimalitas dengan mengacu pada nilai optimum dari tahap sebelumnya untuk kapasitas sisa $c_1 - w_{1j}, \dots, c_m - w_{mj}$ yaitu $f_{k-1}(c_1 - w_{1j}, \dots, c_m - w_{mj})$,
    4) Selanjutnya kita bandingkan nilai keuntungan dari objek pada tahap k (yaitu p_k) ditambah nilai $f_{k-1}(c_1 - w_{1j}, \dots, c_m - w_{mj})$ dengan keuntungan pengisian hanya $k - 1$ macam objek, $f_{k-1}(c_1, \dots, c_m)$,

Digital Repository Universitas Jember

5) Jika $p_k + f_{k-1}(c_1 - w_{1j}, \dots, c_m - w_{mj})$ lebih kecil dari $f_{k-1}(c_1, \dots, c_m)$, maka ojek ke-k tidak dimasukkan ke dalam c_i, tetapi jika lebih besar, maka objek yang ke-k dimasukkan.

Relasi rekursif untuk persoalan ini adalah

$f_0(c_1, \dots, c_m) = 0, \quad (c_1, \dots, c_m) = 0,1,2, \dots, c_i$

$f_k(c_1, \dots, c_m) = -\infty, \quad (c_1, \dots, c_m) < 0$

$f_k(c_1, \dots, c_m) = f_{k-1}(c_1, \dots, c_m)$, Jika $c_i < w_{ij}$ untuk beberapa $i \in \{1, \dots, m\}$

$f_k(c_1, \dots, c_m) = \max\{f_{k-1}(c_1, \dots, c_m), p_k + f_{k-1}(c_1 - w_{1j}, \dots, c_m - w_{mj})\}$

Jika $c_i \geq w_{ij}$ untuk semua $i \in \{1, \dots, m\}$

yang dalam hal ini,

$f_0(c_1, \dots, c_m)$ = 0 adalah nilai dari persoalan *knapsack* kosong dengan kapasitas $c_1, \dots, c_m$,

$f_k(c_1, \dots, c_m)$ = keuntungan optimum dari persoalan *knapsack* pada tahap k untuk kapasitas c_i sebesar $c_1, \dots, c_m$,

$f_k(c_1, \dots, c_m)$ = $-\infty$ adalah nilai dari persoalan *knapsack* untuk kapasitas negatif. Solusi optimum dari persoalan *knapsack* adalah $f_n(c_i)$.

2.5 Algoritma *Backtracking*

Algoritma *Backtracking* disebut juga dengan algoritma runut-balik yang pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950. Algoritma *Backtracking* mempunyai prinsip dasar yang sama seperti *Brute-Force* yaitu mencoba segala kemungkinan solusi. Perbedaan utamanya adalah pada ide dasarnya, semua solusi dibuat dalam bentuk pohon solusi dan algoritma akan menelusuri pohon tersebut secara *DFS* (*Depth First Search)* sampai ditemukan solusi yang layak. Pada algoritma *Backtracking*, hanya pilihan yang mengarah ke solusi yang dieksplorasi,

Digital Repository Universitas Jember

pilihan yang tidak mengarah ke solusi tidak dipertimbangkan lagi sehingga waktu pencarian lebih efisien (Munir, 2005).

Pencarian algoritma *Backtracking* berbasis pada DFS (*Depth First Search*), yaitu mencari solusi dari akar ke daun (dalam pohon ruang solusi) dengan pencarian mendalam. Prinsip dasarnya adalah menganalisa segala jalan menuju suatu solusi yang ada kemudian memilih salah satu jalan yang akan mengarah ke solusi terbaik. Simpul-simpul yang sudah dilahirkan (diperiksa) dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E atau *Expand Node* (Putra *et al*, 2013).

Algoritma *Backtracking* dapat digunakan dalam menyelesaikan berbagai masalah. Adapun masalah-masalah yang dapat diselesaikan antara lain: *N-Qween Problem*, *Graph Coloring*, *Hamilton Cycles*, *Traveling Salesman Problem* (TSP) dan *Knapsack.*

2.5.1 Properti Umum Algoritma *Backtracking*

Menurut Munir (2005), untuk menerapkan algoritma *Backtracking*, properti berikut didefinisikan :

a. Solusi persoalan.

   Solusi dinyatakan sebagai vektor dengan n-tuple:

   $$X = (x_1, x_2, \dots, x_n), \qquad x_j \in S_i$$

   Dimana $S_1 = S_2 = \cdots = S_n$, Contoh: $S_i = \{0,1\}$, $S_j = 0$ atau $S_j = 1$.

b. Fungsi pembangkit (nilai x_n)

   Dinyatakan sebagai predikat:

   $$T(n)$$

   $T(n)$ membangkitkan nilai untuk x_n , yang merupakan komponen vektor solusi.

c. Fungsi pembatas (fungsi kriteria)

   Fungsi pembatas merupakan fungsi yang menentukan langkah selanjutnya berupa penerusan pencarian solusi atau melakukan *Backtracking*, dinyatakan sebagai:

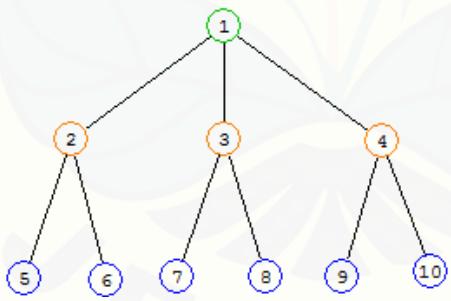
Digital Repository Universitas Jember

$$B(x_1, x_2, \ldots, x_n)$$

Fungsi pembatas menentukan apakah $(x_1, x_2, \ldots, x_n)$ mengarah ke solusi atau tidak. B bernilai benar jika $(x_1, x_2, \ldots, x_n)$ mengarah ke solusi. Jika benar, maka pembangkitan nilai untuk x_{n+1} dilanjutkan, tetapi jika salah, maka $(x_1, x_2, \ldots, x_n)$ dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

2.5.2 Pengorganisasian Solusi

Semua kemungkinan solusi dari persoalan disebut ruang solusi (*solution space*). Jika $x_j \in S_i$, maka $S_1 \times S_2 \times \cdots \times S_n$ disebut ruang solusi. Jumlah anggota diruang solusi adalah $|S_1| \cdot |S_1| \cdot \cdots \cdot |S_n|$. Berdasarkan Gambar 2.3, Ruang solusi diorganisasikan ke dalam struktur pohon. Tiap simpul pohon menyatakan status (*state*) persoalan, sedangkan sisi (cabang) menyatakan keputusan (x_i). Lintasan dari akar ke daun menyatakan solusi yang mungkin. Seluruh lintasan dari akar ke daun akan membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (*state space tree*).



Keterangan:

- : Simpul akar
- : Simpul dalam
- : Simpul daun

Gambar 2.3 Struktur Pohon Ruang Solusi Secara Umum

Nama *Backtracking* diperoleh dari sifat algoritma yang karakteristik himpunan solusinya sudah disusun menjadi suatu pohon ruang solusi. Lihat gambar 2.3, misalkan pohon diatas menggambarkan solusi dari suatu permasalahan. Untuk mecapai solusi (5), maka jalan yang ditempuh adalah (1,2,5), demikian juga dengan solusi-solusi yang lain. Algoritma *Backtracking* akan memeriksa mulai dari solusi

Digital Repository Universitas Jember

yang pertama yaitu solusi (5). Jika ternyata solusi (5) bukan solusi yang layak maka algoritma akan melanjutkan ke solusi (6). Jalan yang ditempuh ke solusi (5) adalah (1,2,5) dan jalan untuk ke solusi (6) adalah (1,2,6). Kedua solusi ini memiliki jalan awal yang sama yaitu (1,2). Jadi daripada memeriksa ulang dari (1) kemudian (2) maka hasil (1,2) disimpan dan langsung memeriksa solusi (6). Jika pada pengecekan status di *node* (2) sudah dapat dipastikan bahwa jalan ini tidak akan menghasilkan solusi yang layak maka penelusuran jalan langsung dibatalkan dan dalam kasus ini langsung menelusuri *node* (3). Pembatalan penelusuran pada *node* (2) secara otomatis akan menghilangkan pengecekan jalan (1,2,5) dan (1,2,6) yang merupakan faktor untuk mengurangi waktu yang diperlukan. Semakin cepat terdeteksi bahwa jalan yang ditempuh tidak akan mengarah ke suatu solusi yang layak maka program akan bekerja dengan lebih efisien.

2.5.3 Prinsip Pencarian Solusi dengan Algoritma *Backtracking*

Langkah-langkah pencarian solusi dengan metode *Backtracking* adalah sebagai berikut :

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti aturan *Depth-First Search* (DFS). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*Expand-node*).
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut “dibunuh” sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas (*bounding function*). Simpul yang sudah mati tidak akan pernah diperluas lagi.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian *Backtracking* ke simpul aras diatasnya. Lalu, teruskan dengan membangkitkan

Digital Repository Universitas Jember

simpul anak yang lainnya. Selanjutnya simpul ini menjadi simpul-E yang baru. Pencarian dihentikan bila kita telah sampai pada *goal node*.

2.5.4 Perhitungan Algoritma *Backtracking* pada *Multiple Constraints Knapsack Problem* (MCKP) 0-1

Pada permasalahan *knapsack* 0-1 dengan beberapa kendala atau *Multiple Constrains Knapsack Problem* (MCKP) 0-1, solusi persoalan dinyatakan sebagai vektor $(x_1, x_2, \dots, x_n)$ dengan $x_j = \{0,1\}$ dimana terdapat 2^n ruang solusi yang didapatkan dari permasalahan yang diberikan untuk mencapai solusi optimal. Untuk menerapkan algoritma *Backtracking* pada persoalan *Multiple Constraints Knapsack Problem* (MCKP) 0-1, berikut properti yang diperlukan (Munir, 2005):

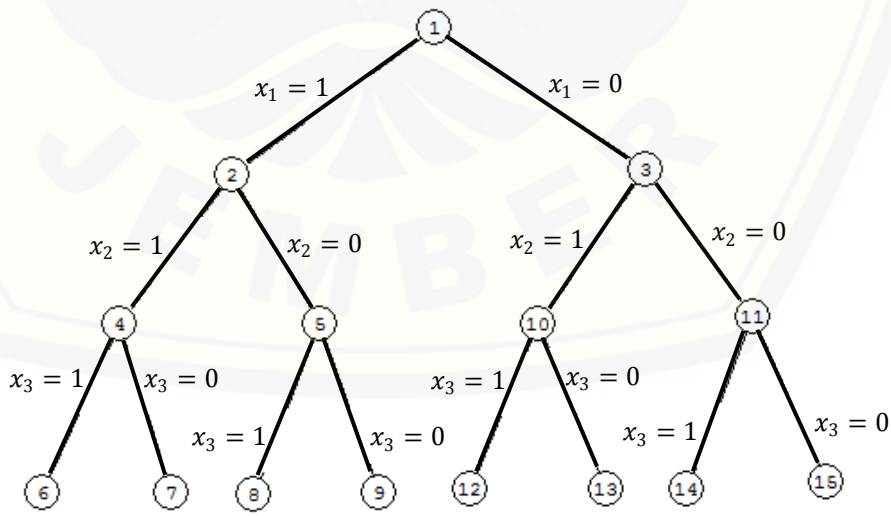
a. Solusi persoalan

Terdapat 2^n ruang solusi yang dinyatakan sebagai vektor:

$$X = (x_1, x_2, \dots, x_n), \qquad x_j \in \{0,1\}$$

$x_j = 0$ jika barang tidak di ambil, $x_j = 1$ jika barang di ambil

Misalkan $n = 3$, maka terdapat $2^n = 2^3 = 8$ ruang solusi persoalan (lihat Gambar 2.4) yaitu (0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1).



Gambar 2.4 Ruang Solusi untuk Persoalan MCKP 0-1 dengan $n = 3$

Digital Repository Universitas Jember

b. Fungsi pembangkit (nilai x_n)

Dinyatakan sebagai:

$$T(n)$$

$T(n)$ membangkitkan nilai untuk x_n , yang merupakan komponen vektor solusi dimana nilai-nilai yang mungkin dari $x_1, x_2, \dots, x_n$ siap dipilih.

c. Fungsi Pembatas (dapat dianggap sebagai fungsi *constraints*)

Fungsi pembatas untuk persoalan *Multiple Constraints Knapsack Problem* (MCKP) 0-1 terdapat pada persamaan 2.2. Jika benar, maka pembangkitan nilai untuk x_{n+1} dilanjutkan, tetapi jika salah, maka $(x_1, x_2, \dots, x_n)$ dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

Digital Repository Universitas Jember

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Pada penelitian ini, data yang digunakan adalah data primer sejumlah barang yang akan dibeli oleh Usaha Dagang (UD) Indo Handicraft. UD tersebut begerak dibidang perdagangan jenis kerajinan yang terletak di Jalan Raya Tamanan, Desa Grujugan Kidul, Kecamatan Grujugan, Kabupaten Bondowoso. UD Indo Handicraft menjual berbagai jenis barang kerajinan tangan di antaranya macam-macam guci, tas, aneka hiasan rumah dan sebagainya. Data yang didapat adalah data mentah bobot barang (w_i) berupa harga beli tiap paket barang, harga jual tiap paket barang, berat tiap paket barang dan biaya operasional yang meliputi biaya untuk bahan bakar, biaya pemeliharaan kendaraan serta biaya tenaga penggerak atau supir yang telah ditetapkan. Untuk menerapkan data ini pada permasalahan *Multiple Constraints Knapsack Problem* (MCKP) 0-1, peneliti perlu melakukan pengidentifikasian dari data mentah yaitu menggunakan data harga beli tiap paket barang, harga jual tiap paket barang dan berat tiap paket barang yang telah ditetapkan kemudian mencari keuntungan (p_i) dari masing-masing barang. Sedangkan biaya operasional yang meliputi biaya untuk bahan bakar, biaya pemeliharaan kendaraan, biaya tenaga penggerak atau supir diabaikan serta tidak ada pengaruh naik turunya harga barang. Data mentah yang di ambil dari UD Indo Handicraft tersebut disajikan dalam Tabel 3.1.

Tabel 3.1. Data Barang UD. Indo Handicraft

No	Nama Barang	Berat (kg)	Harga Beli (Rp)	Harga Jual (Rp)
1	Gerabah	18	250.000	425.000
2	Vas Bunga	12	200.000	300.000
3	Guci	18	875.000	1.150.000
4	Wayang	18	1.020.000	1.200.000

Digital Repository Universitas Jember

Lanjutan Tabel 3.1. Data Barang UD. Indo Handicraft

No	Nama Barang	Berat (kg)	Harga Beli (Rp)	Harga Jual (Rp)
5	Tas Rajut	9	300.000	412.000
6	Dompet Rajut	6	240.000	340.000
7	Topi Rajut	6	240.000	322.000
8	Alas Meja	24	300.000	450.000
9	Sandal Rajut	6	175.000	300.000
10	Tas Kulit	24	960.000	1.080.000
11	Sepatu Kulit	24	1.750.000	2.700.000
12	Dompet Kulit	6	246.000	345.000
13	Hiasan Dinding	12	72.000	115.000
14	Mainan Plastik	12	168.000	240.000
15	Kotak Tisu	6	228.000	315.000
16	Lampu Tidur	12	162.000	201.000
17	Tempat Lilin	9	156.000	228.000
18	Keranjang Buah	12	375.000	510.000
19	Figura	6	240.000	360.000
20	Miniatur mobil	12	540.000	720.000

3.2 Langkah-langkah Penelitian

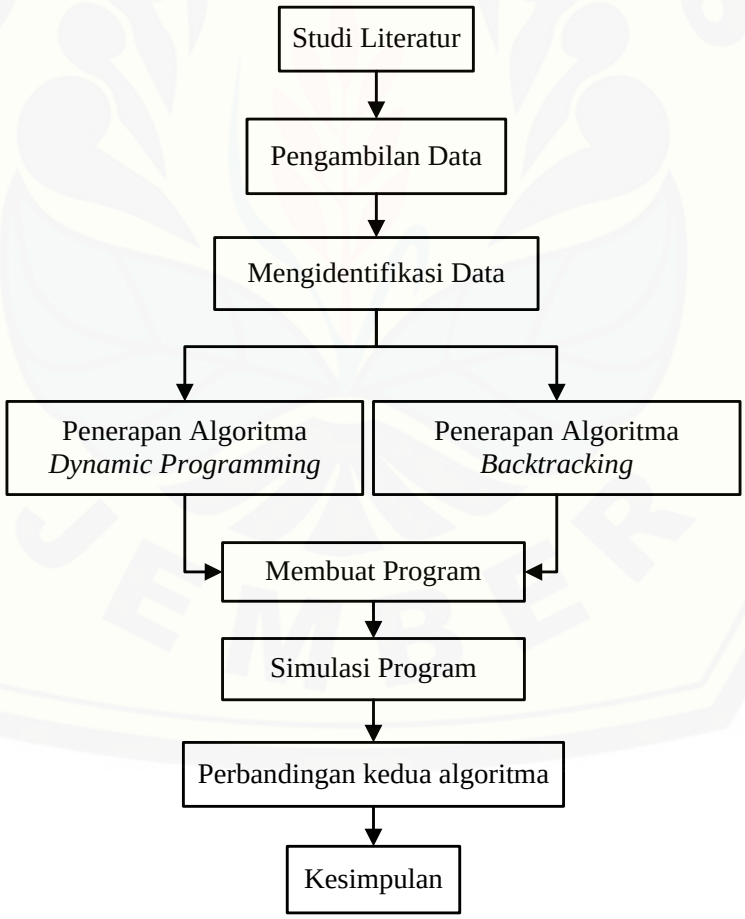
Langkah-langkah yang akan dilakukan dalam menyelesaikan permasalahan *Multiple Constraints Knapsack Problem* (MCKP) 0-1 adalah sebagai berikut.

a. Studi literatur yang dilakukan pada penelitian ini adalah mempelajari mengenai *knapsack*, *Multiple Constraints Knapsack Problem* (MCKP), algoritma *Dynamic Programming* dan algoritma *Backtracking*;
b. Pengambilan data pada Tabel 3.1 dilakukan dengan metode wawancara langsung kepada pihak pemilik Usaha Dagang (UD) Indo Handicraft yang terletak di Jalan Raya Tamanan, Desa Grujugan Kidul, Kecamatan Grujugan, Kabupaten Bondowoso;
c. Mengidentifikasi data dilakukan dengan memilih data yang akan digunakan untuk permasalahan MCKP 0-1 di UD Indo Handicraft, yaitu menggunakan data harga beli tiap paket barang, harga jual tiap paket barang dan berat tiap paket

Digital Repository Universitas Jember

barang kemudian dicari keuntungan (p_i) pada masing-masing barang yang diperoleh dari selisih harga beli dan harga jual yang telah ditetapkan.

d. Menerapkan algoritma *Dynamic Programming* dan algoritma *Backtracking* pada data yang sudah diidentifikasi;
e. Membuat program sesuai algoritma yang digunakan pada langkah (d) menggunakan *software* Matlab;
f. Membandingkan kedua algoritma menggunakan program yang telah dibuat berdasarkan hasil keuntungan maksimal dan lama program saat dijalankan (*running time*);
g. Membuat kesimpulan berdasarkan hasil langkah (d) yang telah dilakukan.



Gambar 3.1 Skema Langkah-langkah Penelitian