APLIKASI PROGRAM DINAMIS DALAM OPTIMASI PRODUKSI PERMEN

Petra Novandi

Informatika Institut Teknologi Bandung
Labtek V Jl. Ganesha No. 10, Bandung
email : if15059@students.if.itb.ac.id, me@van-odin.net

ABSTRAK

Permasalahan optimasi merupakan bagian dari permasalahan kehidupan manusia sehari-hari. Dalam usaha untuk memenuhi kebutuhannya, manusia membutuhkan optimasi dalam pekerjaannya. Optimasi tersebut adalah meminimumkan biaya pengerjaan serta memaksimumkan pendapatan pengerjaan. Akan tetapi dalam pengerjaan tersebut, manusia selalu menghadapi batasan-batasan dalam usaha mengoptimasi.

Untuk memproduksi suatu barang, banyak sekali kendala yang dihadapi dalam memaksimumkan keuntungan penjualan barang tersebut dengan sumber daya dan modal yang terbatas. Permasalahan ini dapat dikategorikan ke ke dalam kelompok *integer knapsack problem* (*1/0 knapsack problem*), sebuah permasalahan kombinatorial yang bertujuan untuk memaksimumkan keuntungan yang diperoleh pada setiap objek dengan sumber daya yang terbatas yang menopang objek tersebut.

Akan tetapi pada kenyataannya banyaknya jenis sumber daya yang dibutuhkan untuk memaksimumkan hasil tidak hanya terbatas pada satu jenis saja, melainkan banyak jenis. *Knapsack* jenis ini kemudian disebut dengan *Integer Programming Problem* atau *Multiconstraint Knapsack* (MKP)

Dalam sejarah ilmu komputer, banyak sekali cara yang dapat ditempuh untuk mencari solusi dari *knapsack* : *branch and bound*, *dynamic programming*, *genetic algorithm, ant colony, heuristic*, dan sebagainya. Akan tetapi yang paling terkenal dan banyak dipakai adalah program dinamis.

Makalah ini membahas pendekatan program dinamis untuk mencari solusi pencarian keuntungan penjualan maksimum dalam produksi berbagai jenis permen dengan bahan-bahan dasar permen sebagai batasannya (*constraint*)

Kata kunci : *Knapsack, Multiconstrained knapsack, Integer programming knapsack, Dynamic programming*

1. PENDAHULUAN

Permasalahan optimasi merupakan bagian dari permasalahan kehidupan manusia sehari-hari. Dalam usaha untuk memenuhi kebutuhannya, manusia membutuhkan optimasi dalam pekerjaannya. Optimasi tersebut adalah meminimumkan biaya pengerjaan serta memaksimumkan pendapatan pengerjaan. Akan tetapi dalam pengerjaan tersebut, manusia selalu menghadapi batasan-batasan dalam usaha mengoptimasi.

Salah satu contoh permasalahan yang memberi batasan pada manusia untuk mengoptimasi adalah knapsack. Dalam permasalahan aslinya *knapsack* digambarkan sebagai usaha seorang pencuri yang ingin mengambil sebanyak-banyaknya barang yang ia curi sehingga muat dalam karung (*knapsack*) yang tidak mampu menampung berat seluruh barang. Oleh karena itu si pencuri ingin mengambil barang sehingga dapat muat dalam karung dan mencapai keuntungan terbanyak ketika barang tersebut dijual.

Permasalahan ini dikategorikan sebagai permasalahan kombinatorial yakni bagaimana menentukan kombinasi dari barang-barang yang ingin dimuat sehingga dapat mencapai keuntungan maksimum.

Diberikan n buah barang masing-masing x_1 sampai x_n. Setiap x_j mempunyai harga p_j dan bobot w_j . Kita mempunyai karung dengan kapasitasn C

Maksimumkan

$$\sum_{j=1}^{n} p_j x_j$$

Untuk

$$\sum_{j=1}^{n} w_j x_j \le C$$

$$x_j \in \{0,1\}, j = 1,....,n \qquad (1)$$

Akan tetapi dalam banyak kasus yang ada pada kenyataan, *knapsack* sesungguhnya tidak hanya memiliki batasan dalam berat saja melainkan banyak sekali batasan. Persoalan ini dinamakan *multi constrained knapsack.*atau juga disebut *Integer programming problem* (pemrograman bilangan bulat) dan *linear programming problem*.

Maksimumkan

$$\sum_{j=1}^{n} w_j x_j$$

Untuk

$$\sum_{j=1}^{n} w_{ij} x_j \le C_i, i = 1,...,M$$

$$x_j \in \{0,1\}, j = 1,....,n \qquad (2)$$

Seluruh permasalahan *knapsack* digolongkan ke dalam kelompok permasalahan *NP-hard problem* yakni sebuah kategori permasalahan yang diyakini tidak mempunyai penyelesaian dengan fungsi waktu asimptotik yang polinomial.

2. DESKRIPSI PERMASALAHAN

Permasalahan ini diinspirasikan oleh seorang rekan yang ingin membuka kerajinan rumah tangga penghasil kue. Dan penulis juga memikirkan sebuah ide yang serupa yakni membangun sebuah pabrik permen. Anggap seorang pengusaha membuka sebuah pabrik penghasil permen. Pabrik ini ditujukan untuk membuat banyak permen yang terdiri dari berbagai jenis permen yang disukai oleh anak-anak. Untuk menghasilkan sebuah jenis permen diperlukan beberapa bahan dasar. Adapun bahan dasar dari permen tersebut antara lain cokelat, krim susu, dan juga gula. Untuk setiap jenis permen, diperlukan takaran cokelat, krim susu, dan gula yang berbeda-beda. Hal ini dibuat agar pengkonsumsi permen ini tidak merasa bosan dengan satu jenis permen saja.

Sang pemilik pabrik ini menjual setiap jenis permennya dalam bentuk paket yang akan siap dibawa di dalam kontainer untuk dijual ke distributor dengan harga yang telah ditetapkan bersama untuk setiap paket jenis permennya. Dengan setiap paket yang dibuat dari kadar cokelat, krim susu, dan gula yang berbeda-beda.

Dalam penulisan sederhana, diberikan beberapa jumlah *constraint* C_1 untuk banyaknya cokelat yang disediakan, *C2* untuk krim susu, dan *C3* untuk gula serta diberikan banyaknya pesanan paket permen x_1, x_2, ... x_k di untuk setiap x_j, $j = 1, .., n$

3. PENYELESAIAN

Persoalan ini sebenarnya identik dengan permasalahan pemrograman linier yang sering menjadi dasar pertimbangan dalam mencari optimum dari sebuah permasalahan yang berhubungan dengan keterbatasan. Diberikan persamaan-persamaan produksi dan diminta untuk mencari solusi optimum dari persamaan-persamaan tersebut selama masih dalam selang batasan.

Sebuah persoalan pemrograman bilangan bulat dapat dinyatakan dengan

Maksimumkan

$$\sum_{j=1}^{n} w_j x_j$$

Untuk

$$\sum_{j=1}^{n} w_{ij} x_j \le C_i, i = 1,...,m$$

$$x_j \in \{0,1\}, j = 1,....,n \qquad (2)$$

Dimana p_j, w_{ij}, dan c_i adalah bilangan bulat tidak bertanda (cacah).

Menurut David Prisinger, jika pada (2) digunakan *constraint* yang yang lebih padat maka solusi dapat dicari dengan cara yang lebih cepat. *Constraint* tersebut dapat dipadatkan dengan menyelesaikan sederetan permasalahan *Subset-sum* : Untuk setiap *constraint* yang ada i = 1,..., *m*,

Maksimumkan

$$z_i = \sum_{j=1}^{n} w_{ij} x_j$$

Untuk

$$\sum_{j=1}^{n} w_{ij} x_j \le C_i$$

$$x_j \geq 0, j = 1,....,n \quad (4)$$

Dan jika zi < ci dalam sebuah solusi optimal, kita dapat memadatkan constraint dalam (2) menjadi

$$\sum_{j=1}^{n} w_{ij} x_j \leq z_i, i = 1,...,m \quad (5)$$

Jika dikerjakan menggunakan cara *brute force*, persoalan ini mewajibkan pengerjaan sebanyak n! untuk mencari seluruh kombinasi yang ada sekaligus memeriksa apakah kombinasi tersebut memenuhi syarat sekaligus menghitung berapa keuntungan yang diperoleh tiap kombinasi. Hal ini mungkin akan sangat melelahkan.

Permasalahan produksi permen ini dapat diselesaikan lebih cepat dengan menggunakan program dinamis maju, yakni dengan mengisikan jumlah bahan baku yang dapat dipakai kemudian membandingkannya dengan paket permen yang telah direncanakan untuk dibuat sebelumnya.

Pada persoalan ini

1. Tahap (k) adalah proses memasukkan paket permen yang akan dibuat ke dalam rencana. Dalam instansiasi persoalan ada 3 paket permen, sehingga ada 3 tahap.
2. Akan ada *k* buah status y_1 sampai y_k yang menyatakan kapasitas tersusa dalam setiap *constraint*. Dalam hal ini ada 3 buah status.

Pada tahap ke 1 kita masukkan objek ke-1 ke dalam karung untuk setiap satuan kapasitas tiap *constraint* sampai salah satu dari *constraint* tersebut penuh. Karena banyaknya bahan baku yang dibutuhkan tiap paket diasumsikan dipakai secara diskrit maka pendekatan ini tidak begitu sulit.

Ketika memasukkan tiap objek pada tahap *k* kita harus memastikan bahwa tidak ada pengisian yang menyebabkan salah satu dari bahan baku permen dipakai berlebihan. Misalkan pada tahap tersebut, sisa semua bahan baku permen adalah $y_i - w_{ik}$. Untuk menggunakan bahan baku yang tersisa, kita menerapkan prinsip optimalitas dengan mengacu nilai optimum dari tahap sebelumnya untuk bahan baku sisa $y_i - w_{ik}$ (yaitu $f_{ik-1}(y_i - w_{ik})$). Lalu bandingkan tiap keuntungan tiap permen pada tahap tersebut ditambah nilai keuntungan pengisian objek sebelumnya. Jika lebih kecil, maka permen tersebut tidak jadi dibuat.

Relasi rekurens tersebut adalah

$$\begin{aligned} &f_0(y_1, y_2, y_3) = 0, y_i = 0,1,2,...,C_i \\ &f_k(y_1, y_2, y_3) = -\infty, y_1 < 0 \cup y_2 < 0 \cup y_3 < 0 \\ &f_k(y_1, y_2, y_3) = \max(f_{k-1}(y_1, y_3, y_3), \\ &p_k + f_{k-1}(y_1 - w_{k1}, y_2 - w_{k2}, y_3 - w_{k3})) \end{aligned} \quad (6)$$

Untuk mendemonstrasikannya penulis mengambil suatu instansiasi permasalahan. Misalkan dalam suatu pagi pemilik pabrik permen membuka pabriknya dan mendapati stok cokelat, krim susu, dan gula sebanyak 5 ton, 6 ton, dan 5 ton. Si pemilik hari itu berencana ingin membuat dua jenis paket permen, yakni jenis permen A yang dibuat dari 3 ton cokelat, 1 ton krim susu, dan 2 ton gula; jenis permen B yang dibuat dari 2 ton cokelat, 1 ton krim susu, dan 1 ton gula; serta jenis permen C yang dibuat dari 2 ton cokelat, 2 ton krim susu dan 1 ton gula.

Paket permen A dijual dengan harga 6 juta rupiah, paket B dengan harga 4 juta rupiah dan paket C dengan harga 2 juta rupiah. Sang pemilik pabrik ingin mengetahui paket permen yang mana yang harus diproduksinya hari itu untuk mendapatkan keuntungan semaksimal mungkin.

Tabel 1 Daftar Biaya dan Keuntungan

Barang	Cokelat	Krim susu	Gula	Harga
Permen A	3 ton	1 ton	2 ton	6.000.000
Permen B	2 ton	1 ton	1 ton	4.000.000
Permen C	1 ton	2 ton	1 ton	2.000.000

Untuk lebih menyederhanakan model soal, dipakai tabel berikut ini

Tabel 2 Model Permasalahan

Barang	w_1	w_2	w_3	p
1	3	1	2	6
2	2	1	1	4
3	1	2	1	2

Tahap I

$$\begin{aligned} &f_1(y_1, y_2, y_3) = \max(f_0(y_1, y_2, y_3), \\ &p_1 + f_0(y_1 - w_{11}, y_2 - w_{12}, y_1 - w_{13})) \end{aligned} \quad (7)$$

Untuk menghemat tempat dan memudahkan perhitungan, tiap tabel pengisian dibedakan per *constraint* yang ada.

Tabel 3 Tabulasi Tahap I *constraint* C_1

Y_1			Optimum	
	$f_0(x)$	$6 + f_0(y_1-3)$	$f(y_1)$	(x_1, x_2, x_3)
0	0	-∞	0	(0, 0, 0)
1	0	-∞	0	(0, 0, 0)
2	0	-∞	0	(0, 0, 0)
3	0	6	**6**	(1, 0, 0)

4	0	6	**6**	(1, 0, 0)
5	0	6	**6**	(1, 0, 0)

Tabel 4 Tabulasi Tahap I *constraint* C_2

y_1			Optimum	
	$f_0(x)$	$6 + f_0(y_1\text{-}1)$	$f(y_1)$	(x_1, x_2, x_3)
0	0	-∞	0	(0, 0, 0)
1	0	-∞	0	(0, 0, 0)
2	0	-∞	0	(0, 0, 0)
3	0	6	**6**	(1, 0, 0)
4	0	6	**6**	(1, 0, 0)
5	0	6	**6**	(1, 0, 0)
6	0	6	**6**	(1, 0, 0)

Tabel 5 Tabulasi Tahap I *constraint* C_1

y_1			Optimum	
	$f_0(x)$	$6 + f_0(y_1\text{-}2)$	$f(y_1)$	(x_1, x_2, x_3)
0	0	-∞	0	(0, 0, 0)
1	0	-∞	0	(0, 0, 0)
2	0	-∞	0	(0, 0, 0)
3	0	6	**6**	(1, 0, 0)
4	0	6	**6**	(1, 0, 0)
5	0	6	**6**	(1, 0, 0)

Tahap II

$$f_2(y_1, y_2, y_3) = \max(f_1(y_1, y_2, y_3), p_2 + f_1(y_1 - w_{21}, y_2 - w_{22}, y_3 - w_{23})) \quad (8)$$

Tabel 6 Tabulasi Tahap II *constraint* C_1

y_1			Optimum	
	$f_1(x)$	$4 + f_1(y_1\text{-}2)$	$f_2(y_1)$	(x_1, x_2, x_3)
0	0	4 + (-∞) = -∞	0	(0, 0, 0)
1	0	4 + (-∞) = -∞	0	(0, 0, 0)
2	0	4 + 0 = **4**	**4**	(0, 1, 0)
3	**6**	4 + 0 = **4**	**6**	(1, 0, 0)
4	**6**	4 + 0 = **4**	**6**	(1, 0, 0)
5	6	4 + 6 = 10	**10**	(1, 1, 0)

Tabel 7 Tabulasi Tahap II *constraint* C_2

y_2			Optimum	
	$f_1(x)$	$4 + f_1(y_2\text{-}1)$	$f_2(y_2)$	(x_1, x_2, x_3)
0	0	4 + (-∞) = -∞	0	(0, 0, 0)
1	0	4 + (-∞) = -∞	0	(0, 0, 0)
2	0	4 + 0 = 4	0	(0, 1, 0)
3	**6**	4 + 0 = 0	**6**	(1, 0, 0)
4	**6**	4 + 6= **10**	**10**	(1, 1, 0)
5	**6**	4 + 6 = **10**	**10**	(1, 1, 0)
6	6	4 + 6 = **10**	**10**	(1, 1, 0)

Tabel 8 Tabulasi Tahap II *constraint* C_3

y_3			Optimum	
	$f_1(x)$	$4 + f_1(y_3\text{-}1)$	$f_2(y_3)$	(x_1, x_2, x_3)
0	0	4 + (-∞) = -∞	0	(0, 0, 0)
1	0	4 + (-∞) = -∞	0	(0, 0, 0)
2	0	4 + 0 = **4**	**4**	(0, 1, 0)
3	6	4 + 0 = 4	**6**	(1, 0, 0)
4	6	4 + 6 = **10**	**10**	(1, 1, 0)
5	6	4 + 6 = **10**	**10**	(1, 1, 0)

Tahap III

$$f_3(y_1, y_2, y_3) = \max(f_2(y_1, y_2, y_3), p_3 + f_2(y_1 - w_{31}, y_2 - w_{32}, y_3 - w_{33})) \quad (9)$$

Tabel 9 Tabulasi Tahap III *constraint* C_1

y_1			Optimum	
	$f_2(x)$	$2 + f_2(y_1\text{-}1)$	$F_3(y_1)$	(x_1, x_2, x_3)
0	0	2 + (-∞) = -∞	0	(0, 0, 0)
1	0	2 + (-∞) = -∞	0	(0, 0, 0)
2	4	2 + 0 = 2	**4**	(0, 1, 0)
3	6	2 + 4 = 6	**6**	(1, 0, 0)
4	6	2 + 6 = 8	**8**	(1, 0, 1)
5	10	2 + 6 = 8	**10**	(1, 1, 0)

Tabel 10 Tabulasi Tahap III *constraint* C_2

y_2			Optimum	
	$f_2(x)$	$2 + f_2(y_2\text{-}2)$	$f_3(y_2)$	(x_1, x_2, x_3)
0	0	2 + (-∞) = -∞	0	(0, 0, 0)
1	0	2 + (-∞) = -∞	0	(0, 0, 0)
2	0	2 + 0 = **2**	**2**	(0, 0, 1)
3	6	2 + 0 = **2**	**2**	(1, 0, 0)
4	6	2 + 0 = **2**	**2**	(1, 0, 0)
5	**10**	2 + 6 = 8	**10**	(1, 1, 0)
6	10	2 + 10 = **12**	**12**	(1, 1, 1)

Tabel 11 Tabulasi Tahap III *constraint* C_1

y_3			Optimum	
	$f_2(x)$	$2 + f_2(y_3\text{-}1)$	$f_3(y_3)$	(x_1, x_2, x_3)
0	0	$2 + (-\infty) = -\infty$	0	(0, 0, 0)
1	0	$2 + (-\infty) = -\infty$	0	(0, 0, 0)
2	**4**	2 + 0 = 2	**4**	(0, 1, 0)
3	**6**	2 + 0 = 2	**6**	(1, 0, 0)
4	**10**	2 + 6 = **8**	**10**	(1, 1, 0)
5	10	2 + 10 = **12**	**12**	(1, 1. 1)

Dari hasil di atas didapat hasil yang paling maksimum adalah 12. Hasil perhitungan tersebut bukanlah merupakan hasil yang paling maksimum yang optimal karena hasil tersebut tidak memenuhi salah satu *constraint* juga akibat penghitungan yang dipisahkan ke dalam tabel yang terpisah. Oleh karena itu yang menjadi solusi optimum dari permasalahan tersebut adalah nilai maksimum yang memenuhi seluruh constraint yang ada. Dalam hal ini keuntungan optimum yang dicapai adalah 10 juta rupiah yang didapat dari kombinasi paket A dan paket B.

Adapun maksimum pengerjaan yang dilakukan adalah sebanyak $O(n.m.r)$ untuk n adalah banyaknya paket permen yang ingin dibuat, m adalah banyaknya jenis bahan baku permen yang disediakan, serta r adalah jumlah maksimum yang bahan baku yang disediakan. Meski terlihat seperti polinomial, tetapi sebenarnya persoalan ini berkembang eksponensial seiring dengan banyaknya n, m, dan r yang diberikan. Kompleksitas ini sering dinamakan *pseudo-polynomial*.

4. KESIMPULAN

Permasalahan optimasi produksi permen dapat digolongkan ke dalam permasalahan *integer knapsack*. Akan tetapi karena banyaknya *constraint* yang harus dipenuhi dalam memproduksi permen maka permasalahan ini dapat digolongkan ke dalam kategori *knapsack* yang lebih general yakni *multiconstraint knapsack* atau *linear programming knapsack*. Permasalahan ini termasuk ke dalam golongan permasalah knapsack lainnya yakni *NP-hard problem* (*Non-deterministic polynomial problem*). Maksudnya adalah permasalahan ini tidak dapat ditentukan apakah dapat dikerjakan dalam kompleksitas waktu asimtotik yang polinomial.

Untuk menyelesaikan permasalahan optimasi produksi permen, program dinamis adalah salah satu algoritma yang mangkus. Algoritma ini menyusun permasalahan menjadi permasalahan yang lebih kecil dan kemudian membandingkan persoalan pada suatu tahap pengerjaan dengan persoalan sebelumnya.

Jika dibandingkan dengan penggunaan *brute force*, program dinamis mampu menyelesaikan persoalan dengan lebih cepat secara beruntun. Algoritma ini layak dipakai bagi seseorang yang ingin membuka pabrik permen ☺.

Kompleksitas penyelesaian algoritma ini adalah $O(n.m.r)$ yakni n untuk banyaknya paket permen yang ingin dibuat, m banyaknya jenis bahan baku permen, dan r adalah jumlah maksimum bahan baku permen. Meski terlihat sebagai polinomial, sebenarnya solusi ini masih terpaut dalam NP-hard karena untuk jumlah yang makin besar, pengerjaan yang dilakukan oleh algoritma ini membesar pula secara eksponensial.

REFERENSI

[1] Munir M.T. Rinaldi. 2005, "*Diktat Kuliah IF2251 Strategi Algoritmik*". Bandung : Institut Teknologi Bandung

[2] Wikipedia, "*Knapsack Problem*", terakhir akses 23 Mei 2007. http://en.wikipedia.org/wiki/Knapsack_Problem

[3] Wikipedia, "Candy", terakhir akses 23 Mei 2007. http://en.wikipedia.org/wiki/Candy

[4] "Knapsack Repository", terakhir akses 23 Mei 2007 http://tracer.lsi.upc.es/mknp/

[5] Eko Wibowo, "*Cook*", 2006, Bina Nusantara Programming Contest

[6] David Pisinger, "*Algorithms for Knapsack Problem*", Februari 2005, Ph.D. Thesis. Dept. of Computer Science, University of Copenhagen