Penerapan Metode *Sequensial Dynamic Programming* Untuk Optimasi Pemakaian Bahan Baku Pada Industri Manufaktur

Leo Willyanto Santoso
Jurusan Teknik Informatika Fakultas Teknologi Industri
Universitas Kristen Petra
Jl. Siwalankerto 121-131 – Surabaya 60236
Telp +62 31 8439040, Fax + 62 31 8417658
Email : leow@petra.ac.id

Abstrak

Manajemen produksi dan operasi merupakan usaha-usaha pengelolaan secara optimal penggunaan sumber daya (atau sering disebut faktor-faktor produksi) – tenaga kerja, mesin-mesin, peralatan, bahan mentah dan sebagainya – dalam proses transformasi bahan mentah dan tenaga kerja menjadi berbagai produk dan jasa. Para manajer produksi dan operasi mengarahkan berbagai masukan (*input*) agar dapat memproduksi sebagai keluaran (*output*) dalam jumlah, kualitas, harga, waktu dan tempat tertentu sesuai dengan permintaan dari konsumen. Pemakaian bahan baku yang tidak terkontrol akan sangat mempengaruhi kualitas proses, produk dan layanan dari industri manufaktur. Oleh karena itu, sangat dibutuhkan sebuah solusi untuk memecahkan ”cutting-stock problem” dengan hasil yang memuaskan.

Pada makalah ini dijelaskan penerapan sebuah solusi paralel dari metode *sequential dynamic programming* untuk memecahkan masalah 2D knapsack (*cutting-stock*) dimana metode ini dapat melakukan optimasi dengan meminimalkan sisa pemotongan/pemakaian bahan baku dengan baik. Perangkat lunak dalam penelitian ini dikembangkan dengan Visual Basic 6.0 dan sistem manajemen basis data Microsoft SQL Server 2000.

Uji coba perangkat lunak yang telah dilakukan diketahui bahwa program ini dapat menunjang proses optimasi pemakaian bahan baku sampai sebesar 60%. Selain itu, perangkat lunak ini mempunyai alur proses kerja yang jelas dan menyediakan fitur penjadwalan yang sangat membantu user dalam melakukan kontrol produksi.

Kata kunci: Optimasi, Sistem Informasi Produksi, dan *Knapsack Problem*

1. Pendahuluan

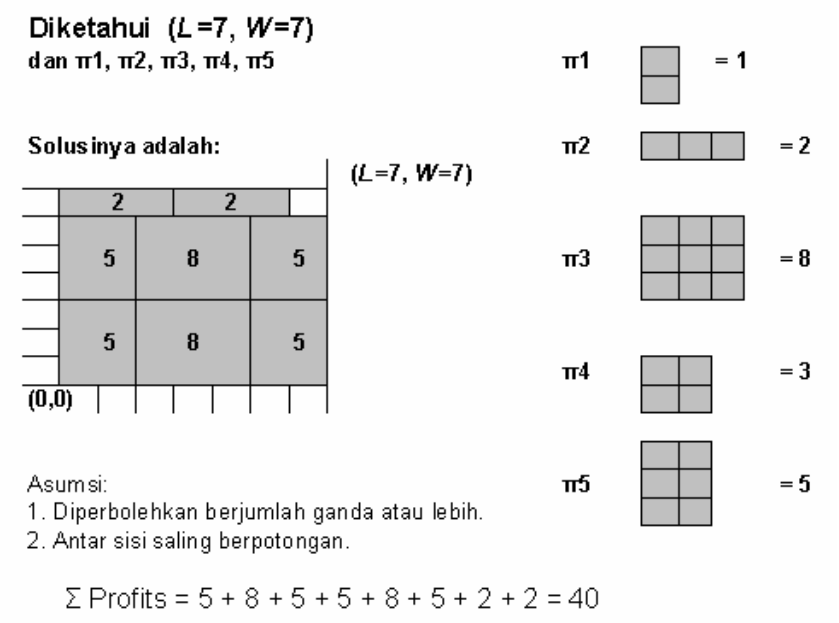
Pengendalian Aktivitas Produksi (*Production Activity Control* = PAC) bertujuan untuk mempertahankan keseimbangan antara sumber-sumber daya manufacturing yang tersedia dan permintaan total. Fungsi dari PAC, yang sering disebut sebagai: *shop floor control* (SFC) adalah melakukan aktivitas-aktivitas sebagaimana telah direncanakan, melaporkan hasil-hasil operasi, dan memperbaiki atau merevisi rencana-rencana yang diperlukan untuk mencapai hasil yang diinginkan. PAC melakukan umpan-balik melalui pengukuran output aktual dan membandingkannya dengan rencana-rencana. Dengan demikian, PAC merupakan komponen esensial dari *close-loop* MRP. PAC mencakup aktivitas-aktivitas keseluruhan dari *shop-floor scheduling and control* yang biasa disebut sebagai *shop-floor control* yang secara keseluruhan merupakan bagian dalam PAC, serta sebagian aktivitas-aktivitas

penjadwalan dan tindak lanjut terhadap pemasok (*supplier scheduling and follow-up*) yang merupakan bagian terbesar dalam PAC.

Banyak teknik PAC didesain untuk melaksanakan rencana-rencana material secara terperinci yang dihasilkan melalui sistem MRP. Perluasan dari definisi PAC dilakukan melalui pengembangan penggunaan komputer pada *shop floor* dan *electronic data interchange* (EDI) dengan pemasok. Selain itu, komputer juga banyak digunakan untuk membantu pencapaian efisiensi operasi (*operating efficiencies*), dimana ongkos-ongkos manufakturing harus diminimumkan guna memperoleh harga kompetitif. Pengendalian ongkos-ongkos membutuhkan operasi yang efisien dari keseluruhan organisasi. Elemen-elemen yang perlu diperhatikan dalam efisiensi operasi, adalah: supervisi pabrik dan tenaga kerja tidak langsung, dukungan dan keterlibatan pekerja, mesin dan peralatan yang andal, fasilitas pendukung yang efektif dan pengelolaan bahan baku menjadi bahan jadi yang efisien.

Dalam pengelolaan bahan baku sering terjadi proses pemotongan bahan baku menjadi beberapa bagian untuk diproses lebih lanjut. Pemotongan ini sering dilakukan secara manual tanpa melakukan perencanaan yang matang, sehingga pada setiap akhir proses produksi banyak terdapat sisa potongan bahan baku yang terbuang sia-sia. Hal ini sering menjadi faktor penting yang mendorong kita untuk melakukan optimasi dalam sisa pemotongan bahan baku. Permasalahan optimasi sisa pemotongan dikenal sebagai masalah knapsack.

Sebuah masalah knapsack memerlukan proses pencarian sebuah subset dari kumpulan obyek dengan memaksimumkan jumlah dari keuntungan obyek dan tidak boleh melebihi ukuran dari knapsack atau melanggar batasan yang ada. Sejumlah masalah muncul dalam bidang ilmu komputer dan riset operasional, yaitu aplikasi ”cutting stock”. Masalah 2D sangat menarik untuk dipelajari lebih jauh. Masalah knapsack 0-1 sudah dapat diselesaikan secara efisien dengan linier systolic array. Beberapa algoritma knapsack 2D memperhatikan sejumlah batasan masalah yang dikenal, dan masalah-masalah ini sering mudah untuk diselesaikan dengan menambahkan batasan-batasan atau menggunakan *approximation method*. Masalah yang dianalisis di sini termasuk dalam kelompok NP, dimana solusi terbaik hasil komputasi secara sekuensial adalah $O(LW(n+L+W))$, dimana n adalah jumlah obyek, dan L dan W adalah dimensi dari knapsack. *Running time* untuk algoritma ini adalah *pseudo-polynomial* karena dalam kaitan dengan ukuran input, kapasitas knapsack dikodekan hanya dalam $\log_2(L) + \log_2(W)$ bit. Gambar 1 menunjukkan sebuah solusi untuk masalah knapsack sederhana.



Gambar 1. Solusi Mungkin dari *Cutting Stock* 2D

2. Masalah *Knapsack* 2D

Masalah Knapsack 2D merupakan masalah pengisian daerah berdimensi (L, W) dengan n buah persegi dengan ukuran (l_i, w_i) dimana i = 1, 2, ..., n. Profit adalah suatu nilai yang positif, π_1, π_1, ..., π_n yang berkaitan dengan tiap persegi. Dengan parameter ini, keuntungan maksimum dari $\pi_1 z_1$, $\pi_1 z_2$, ..., $\pi_n z_n$ dihitung dimana z_i adalah suatu nilai positif dimana knapsack dibagi dalam z_i bentuk persegi i, yang mempunyai ukuran (l_i, w_i). Masalah pemotongan ini hanya mengijinkan terjadinya proses rekursi sisi per sisi, sehingga semua pemotongan dibuat tegak lurus dari satu sisi persegi terhadap yang lainnya. Obyek benda dapat mempunyai orientasi yang sudah tepat atau dapat diputar 90^0. Tambahan n obyek dengan dimensi (w_i, l_i) dengan keuntungan π_1 ditambahkan ketika proses rotasi memungkinkan. Salah satu algoritma untuk menyelesaikan masalah ini adalah dengan menerapkan *dynamic programming*.

Fungsi Knapsack $F(x, y)$, didapatkan dari teknik *dynamic programming*, dilakukan proses perhitungan sehingga untuk lokasi (x, y), $F(x, y)$ adalah keuntungan terbesar yang diperoleh dari persegi yang dibuat dari sisi X dan Y dan titik (x, y). Hal ini memenuhi pertidaksamaan matamatis berikut ini yang berhubungan dengan batasan pemotongan yang dapat dilakukan.

$$0 \le x \le L,$$
$$0 \le y \le W,$$
$$F(x, y) \ge 0,$$
$$F(x_1 + x_2, y) \ge F(x_1, y) + F(x_2, y),$$
$$F(x, y_1 + y_2) \ge F(x, y_1) + F(x, y_2),$$
$$F(l_i, w_i) \ge \pi_i, (i = 1, 2, \ldots, n)$$

3. *Sequential Dynamic Programming*

Sequential Dynamic Programming adalah suatu teknik matematis yang biasanya digunakan untuk membuat suatu keputusan dari serangkaian keputusan yang saling berkaitan. Tujuan utama model ini adalah untuk mempermudah penyelesaiann persoalan optimasi yang mempunyai karakteristik tertentu.

Ide dasar *dynamic programming* ini adalah membagi persoalan menjadi beberapa bagian yang lebih kecil sehingga memudahkan penyelesaiannya. Akan tetapi, berbeda dengan *linear programming*, pada persoalan *dynamic programming* ini tidak ada formulasi matematis yang standar. Karena itu, persamaan-persamaan yang terpilih untuk digunakan harus dikembangkan agar dapat memenuhi masing-masing situasi yang dihadapi. Dengan demikian, maka antara persoalan yang satu dengan persoalan lainnya dapat mempunyai struktur penyelesaian persoalan yang berbeda.

Keuntungan dari penggunaan *dynamic programming* adalah dapat memperoleh solusi dari suatu masalah tanpa adanya *exponential running time*. Setiap obyek berbentuk kotak akan diatur posisinya, dan sebuah tabel akan mencatat pemotongan terbaik untuk tiap lokasi. Setelah itu, semua isi tabel akan dibaca untuk menghitung jumlah panjang dan lebar dari obyek untuk membuat konfigurasi dari obyek yang menghasilkan keuntungan maksimum. Algoritma dari *sequential dynamic programming* dapat dilihat pada Segmen 1. Berikut ini adalah persamaan matematis relasi yang digunakan untuk mendapatkan algoritma *sequential*.

$F_0 (x, y) = \max \{0, \pi_j \mid l_j \le x \wedge w_j \le y\}$
$F_k (x, y) = \max \{ F_{k-1} (x, y), F_{k-1} (x_1, y) + F_{k-1} (x_2, y), F_{k-1} (x, y_1) + F_{k-1} (x, y_2) \}$
$0 < x_1 \le x_2, x_1 + x_2 \le x, 0 < y_1 \le y_2, y_1 + y_2 \le y.$

Pada step 1 algoritma, semua lokasi dalam knapsack diinisialisasi dengan nilai 0. Pada step 2 ini, setiap obyek mulai diperhatikan, meletakkan nilai keuntungan tertinggi pada semua lokasi dimana memungkinkan akan dilakukan. Pada step 3 ini dilakukan pembacaan isi tabel 2D dari baris yang paling rendah ke baris yang paling tinggi, menjumlahkan semua kemungkinan kombinasi yang dapat dilakukan baik secara vertikal atau horisontal dan menyimpan dua obyek yang mempunyai jumlah keuntungan tertinggi. Pada segmen 1 ditunjukkan bagaimana nilai $F(x, y)$ dihitung secara iterasi. Karena hanya potongan persegi yang digunakan, solusi parsial pada posisi (i, j) dipotong secara paralel menjadi dua bagian, x dan y. Karena simetris, hanya potongan x, mulai dari $x = 0$ sampai $i/2$, dan potongan y mulai dari 0 sampai $j/2$ yang dipertimbangkan untuk setiap (i, j). Algoritma *sequential* ini mempunyai waktu proses $O(LW(n+L+W))$, dimana n adalah jumlah obyek, dan L, W menyatakan dimensi dari knapsack.

Segmen 1. Algoritma *Sequential dynamic programming*.

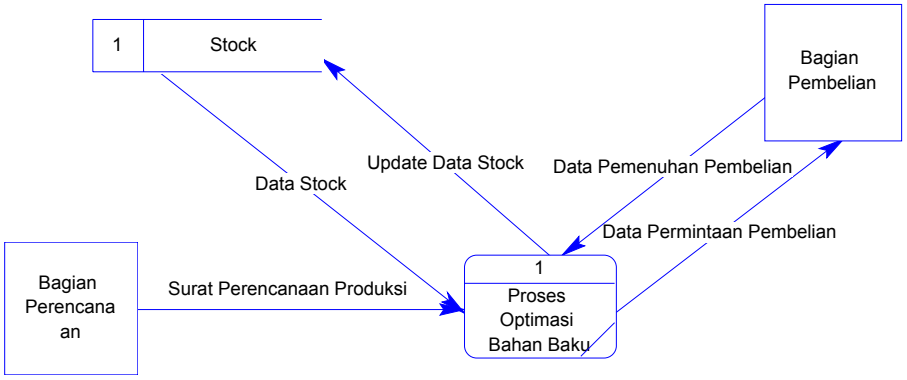
```
for i = 0 to L                                  // Step 1
    for j = 0 to W
        F_{i, j} = 0
for i = 0 to n                                  // Step 2
    for j = 0 to L
        for j = 0 to W
            if (l_k ≤ i dan w_k ≤ j dan π_k > F_{i, j})
        F_{i, j} = π_k
for i = 0 to L                                  // Step 3
    for j = 0 to W {
        for k = 0 to ⌊ i/2 ⌋
            { sum = F_{k, j} + F_{i-k, j}
              if (sum> F_{i, j}) F_{i, j} = sum}
        for k = 0 to ⌊ j/2 ⌋
            { sum = F_{k, i} + F_{i, j-k}
              if (sum> F_{i, j}) F_{i, j} = sum}}
```

4. Analisis, Desain dan Implementasi Perangkat Lunak

Pada tahapan ini, penulis mengambil studi kasus pada perusahaan meubel yang menggunakan bahan baku berupa kayu lapis untuk membuat perlengkapan rumah tangga dan kantor seperti meja, kursi dan almari. Perusahaan juga melayani pemesanan barang sesuai dengan keinginan konsumen, seperti bentuk dan ukurannya. Permasalahan-permasalahan yang sering terjadi adalah:

- Kesalahan-kesalahan pemotongan bahan baku yang menyebabkan kerugian bagi pihak perusahaan.
- Kekeliruan perhitungan dalam proses perencanaan produksi, yang mengakibatkan kurang optimalnya pemakaian bahan baku.

Untuk membantu permasalahan tersebut maka perusahaan membutuhkan program perencanaan produksi untuk membantu menghitung, menentukan dan mengoptimalkan pemakaian bahan baku yang dibutuhkan untuk proses produksi. Secara sederhana, *data flow diagram* (DFD) untuk proses optimasi-nya adalah seperti pada Gambar 2.



Gambar 2. DFD Proses Optimasi Bahan Baku

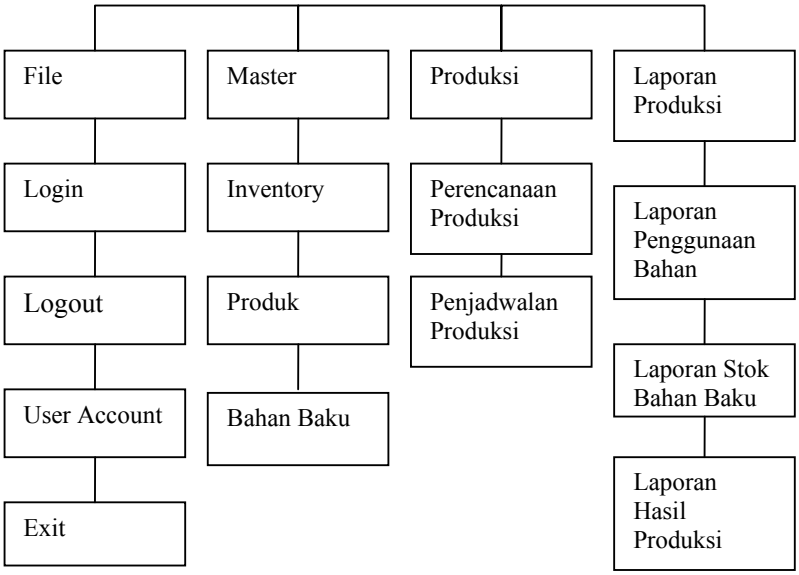
Implementasi perangkat lunak yang mengacu pada analisis dan desain sistem yang telah dirancang ini meliputi implementasi *database* dengan menggunakan Microsoft SQL Server 2000 dan implementasi program dengan menggunakan Microsoft Visual Basic 6.0. Fasilitas optimasi ini akan diintegrasikan dengan program sistem informasi produksi yang telah dibuat sebelumnya.

Desain Form utama nantinya mempunyai beberapa menu sesuai dengan fungsinya masing-masing, menu-menu tersebut antara lain: file, master, produksi, bahan baku dan laporan produksi. Kemudian tiap-tiap menu tersebut mempunyai beberapa *submenu*. Desain Menu perangkat lunak dapat dilihat pada Gambar 3. Berikut ini adalah penjelasan dari masing-masing *submenu* tersebut.

1. Menu File
   Memiliki empat *submenu*, yaitu:
   a. Login, Berfungsi untuk masuk dalam program.
   b. Logout, Berfungsi untuk keluar dari program.
   c. User Account, Berfungsi untuk menambah, mengubah atau menghapus data *user account*.
   d. Exit, Berfungsi untuk menutup program.
2. Menu Master
   Memiliki tiga *submenu*, yaitu:
   a. Inventory, Berfungsi untuk memasukkan data stok bahan baku.
   b. Produk, Berfungsi untuk memasukkan data produk.
   c. Bahan baku, Berfungsi untuk melakukan permintaan bahan baku, pengembalian sisa bahan baku, Pembelian bahan baku dan Penerimaan bahan baku.
3. Menu Produksi
   Memiliki dua *submenu*, yaitu:
   a. Perencanaan Produksi, Berfungsi untuk melakukan:
      - Perencanaan produksi dan perencanaan kebutuhan sumber daya
      - Penjadwalan produksi Induk (MPS) dan *Rough Cut Capacity Planning* (RCCP)
      - Perencanaan Kebutuhan Material dan optimasinya.
      - Pengendalian Aktivitas Produksi (PAC).
   b. Penjadwalan Produksi, Berfungsi untuk:
      - Menyediakan atau memberikan input utama kepada sistem perancanaan kebutuhan material dan kapasitas (*material and capacity requirements planning* = M&CRP),
      - Menjadwalkan pesanan-pesanan produksi dan pembelian (*production and purchase orders*) untuk item-item MPS,

- Memberikan landasan untuk penentuan kebutuhan sumber daya dan kapasitas.
- Memberikan basis untuk pembuatan janji tentang penyerahan produk (*delivery order*) kepada pelanggan.

4. Menu Laporan Produksi
   Memiliki tiga *submenu*, yaitu:
   a. Laporan Penggunaan Bahan, Berfungsi untuk mencetak Laporan Penggunaan Bahan tiap proses produksi.
   b. Laporan Stok Bahan Baku, Berfungsi untuk mencetak Laporan Stok Bahan Baku dari seluruh bahan baku yang ada di gudang.
   c. Laporan Hasil Produksi, Berfungsi untuk mencetak Laporan Hasil Produksi berdasarkan data proses produksi.



Gambar 3. Desain Menu Perangkat Lunak

5. Uji Coba Perangkat Lunak

Uji coba perangkat lunak yang dilakukan dengan menggunakan komputer Pentium IV-2,4 GHz dan Memory 256 MB dihasilkan data seperti pada Tabel 1 berikut ini.

Tabel 1. Hasil Uji Coba Perangkat Lunak

				Perbandingan	
Jenis Bahan	**m**	**n**	**obj**	**t lama**	**t**
ATP30	10	51	11961	312.93	105.13
ATP31	10	32	11408	402.13	192.4
ATP32	22	60	60170	19.60	4.02
ATP33	40	90	62459	168.20	56.9
ATP34	10	18	729	0.72	0.16

ATP35	20	50	9409	11.42	4.54
A1s	19	62	2950	0.27	0.11
A2s	20	53	3423	2.57	0.54
A3	20	46	5380	1.78	0.66
A4	19	35	5885	1.58	0.77
A5	20	45	12553	3.97	1.51

Dengan *m* menyatakan jumlah tipe pemotongan (*i*) dan *n* menyatakan jumlah pemotongan yang mungkin (*j*).

6. Penutup

Berdasarkan uji coba yang telah dilakukan, dapat diambil kesimpulan, yaitu:

1. Prosentase optimasi yang berhasil dicapai adalah sekitar 60% apabila dibandingkan dengan proses manual.
2. Waktu proses yang diperlukan untuk model sequential adalah $O(LW(n + L + W))$. Untuk mengurangi *running time*, dapat dilakukan proses secara parallel.
3. Proses perencanaan produksi akan semakin lengkap apabila dilengkapi dengan proses penentuan pemotongan bahan baku.

Daftar Pustaka

A.Lodi and M. Monaci, *Integer linear programming models for 2-staged two-dimensional knapsack problems*, Mathematical Programming, Ser. B 94 (2002), 257-278.

G. Belov and G Scheithauer, *A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two dimensional two-stage cutting*, Technical report, Dresden University, 2003, URL: www.math.tu-dresden.de/~capad.

G.L. Nemhauser and L.A. Wolsey, *Integer and combinatorial optimization*, John Wiley and Sons, New York, 1998.

H. Dyckhoff and U. Finke, *Cutting and packing in production and distribution*, Physica Verlag, Heidelberg, 1992.

ILOG optimization suite. Delivering a competitive advantage, White paper, ILOG Corporation, 2001, URL:http://www.ilog.com.

J.C. Herz, *Recursive computational procedure for two dimensional stock cutting*, IBM J. Res. Develop., 16:462-469, 1967.

J. Potter, J. Baker, S. Scott, A. Bansal, C. Leangsuksun, C. Astagiri, *MASC: An associative computing paradigma*, IEEE Computer, pages 19-25, November 1994.

M. Hifi and C. Roucairol, *Approximate and exact algorithm for constrained (un)weighted two-dimensional two-staged cutting stock problems*, Journal of combinatorial optimization 5 (2001), 465-494.

P.C. Gilmore and R.E. Gomory, *Multistage cutting stock problems of two or more dimensions*, Operations Research, 13: 94-120, 1965.

P. Y. Wang, *Two algorithms for constrained two dimensional cutting stock problems*, Operations Research, 31(3): 573-586, 1983.

Santoso, Leo Willyanto, Pembuatan Sistem Informasi Produksi Untuk Meningkatkan Kualitas Sistem Manufaktur dan Jasa, Seminar Nasional II Peningkatan Kualitas Sistem Manufaktur dan Jasa, Yogyakarta, April 2005