BAB 2

LANDASAN TEORI

2.1 Permasalahan *Cutting Stock* Satu Dimensi

Permasalahan *Cutting stock* merupakan suatu permasalahan yang muncul karena banyak dipakai aplikasinya dalam bidang perindustrian. Misalkan di dalam perindustrian kayu, bagaimana memanajemen pemotongan kayu supaya dapat meminimumkan sisa pemotongan yang dihasilkan dan dapat membentuk pola pemotongan yang optimal.

Dalam hal inilah permasalahan *cutting stock* dapat digunakan untuk menyelesaikan permasalahan di atas sebagai salah satu aplikasi dari permasalahan optimisasi, atau yang lebih spesifik adalah sebuah *permasalahan program linier integer*. Sebagai salah satu permasalahan program linier integer maka hasil yang diharapkan dalam suatu permasalahan adalah bilangan bulat.

Dikatakan sebagai permasalahan *cutting stock* jika terdapat permintaan ukuran dari pesanan dan adanya batasan yang ditetapkan, serta tujuannya untuk meminimumkan sisa. Adapun permasalahan *cutting stock* satu dimensi adalah terbatas hanya membahas satu kendala yang sesuai dengan kendala yang ditetapkan dan mengabaikan kendala lain.

Misalnya hanya untuk menentukan pola pemotongan optimal yang meminimumkan sisa pemotongan, dan diberikan satu kendala saja yaitu ukuran produk yang dihasilkan saja tanpa memperhitungkan kapasitas gudang penyimpanan, tingkat kualitas kayu, tingkat kualitas kayu, atau kendala lain yang berkaitan dengan permasalahan.

Secara spesifik, dimisalkan terdapat suatu ukuran panjang produk L yang dihasilkan *(raw)* dan akan dipotong dalam beberapa pola j dengan ukuran panjang pesanan x *(final)* dari tiap-tiap pesanan jenis i. Dan tentu saja, nilai-nilai tersebut terbatas pada bilangan integer i, $i = 1,2,\cdots,n.$

Permasalahan *cutting stock* ini biasanya diselesaikan dengan formula yang diperkenalkan oleh Gilmore-Gomory (1961, 1963). Pola pemotongan yang mungkin akan dienumerasikan sebelumnya. Pola-pola tersebut didefinisikan sebagai suatu vektor $\left(a_{1j},\cdots,a_{ij},\cdots,a_{mj}\right)$ dimana elemen a_{ij} menunjukkan jumlah berapa kali pesanan dengan ukuran panjang i dihasilkan dalam pola j. Misalkan x_j adalah variabel yang menandakan jumlah gelondongan kayu yang akan dipotong sesuai dengan pola j.

Dalam membentuk program linier sebagai permasalahan utama, maka fungsi objektif yang mungkin adalah meminimumkan sisa potongan dan meminimumkan jumlah total gelondongan *raw* yang dipotong yaitu meminimumkan $\sum_j x_j$. Dua bentuk formula ini adalah sama jika fungsi objektif dalam permasalahan *cutting stock* dimasukkan beberapa sisa gelondongan kayu yang ditunjukkan sebagai suatu variabel slack.

Permasalahannya adalah bagaimana menentukan pola-pola pemotongan optimal yang meminimumkan sisa. Secara matematika permasalahannya adalah diberikan suatu data d_i dan x_i bernilai positif dimana $i = 1,2,\cdots,m$ dan L, yang akan meminimumkan Z diberikan:

$$Z = \sum_{j \in J} x_j$$

kendala $\sum_{j \in J} a_{ij} x_j \geq d_i$

x_j integer dan $x_j \geq 0$

dimana J adalah himpunan dari pola pemotongan yang mungkin.

Diselesaikan sebuah permasalahan program linier dan pada umumnya merupakan suatu permasalahan optimisasi dan terdapat teknik untuk menyelesaikan. Salah satu metode adalah metode simpleks (dualitas) yang dimulai dari beberapa titik yang tidak optimal dan secara iterasi menemukan solusi yang baik dan lebih baik lagi. Algoritma ini menyimpulkan bahwa solusi optimal didapat ketika tidak ada lagi langkah yang dibuat untuk memperbaiki solusi.

2.2 Heuristik 'Largest In Least Empty' (LILE)

Sebelum membentuk program linier, akan didaftarkan pola-pola pemotongan yang mungkin dipotong dari *raw*. Dalam menentukan pola digunakan heuristik yang sederhana yaitu 'Largest In Least Empty'(LILE).

Heuristik ini dijelaskan dalam tahapan di bawah ini:

1. Cari nilai penyelesaian yang *final* yang menurun dan tentukan permintaan yang belum tepenuhi.
2. Urutkan *final* dalam permintaan yang belum terpenuhi dari yang terbesar sampai yang terkecil.
3. Lanjutkan proses ini sampai pengurutan daftar *final* dari permintaan yang belum terpenuhi ditempatkan secara lengkap.

Sangat memungkinkan pola dapat dibangkitkan dengan algoritma ini , ini adalah salah satu pola yang digunakan untuk mempermudah penyelesaian program integer. Algoritma LILE bertujuan untuk meminimumkan jumlah gelondongan kayu tambahan yang diperlukan, dan kembali memudahkan penyelesaian.

2.3 Pembangkitan Kolom (Column Generation)

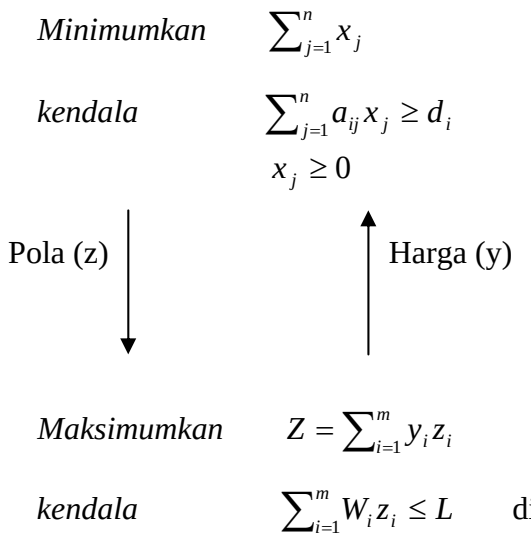
Menyelesaikan permasalahan *cutting stock* dalam bentuk program linier tidak selalu mudah, dimana ada satu permasalahan yaitu dalam formulasi membentuk program linier, dimana harus mendaftarkan semua pola pemotongan. Jika terdapat banyak ukuran panjang pesanan yang berbeda yang akan dipotong maka akan sangat sulit untuk membentuk formulasi program liniernya karena akan begitu banyak variabel yang akan dibuat.

Maka alternatif yang akan digunakan adalah sebuah *pendekatan pembangkitan kolom yang tertunda.* Metode ini menyelesaikan permasalahan *cutting stock* dengan memulainya hanya dengan sedikit pola. Dan akan dibangkitkan pola tambahan ketika dibutuhkan. Pola yang baru akan didapat dengan menyelesaikan submasalah optimisasi lain yaitu permasalahan *knapsack*. Penggunaan metode pembangkitan kolom yang tertunda ini lebih efisien daripada pendekatan biasa.

Akan dipilih kumpulan *solusi pola inisial* untuk dimasukkan dalam model dan penyelesaian program linier. Solusi pola inisial ini merupakan cara sederhana untuk memasukkan suatu pola dari setiap *final*, dimana *setiap pola inisial mengandung nilai paling maksimum yang dapat dipotong dari raw*.
Dengan memilih pola yang mengikutkan semua ukuran *final*, solusi awal akan layak tetapi tidak optimal. Walaupun hal ini sepertinya tidak memungkinkan untuk memilih kumpulan pola yang benar, digunakan informasi variabel dual dari program linier untuk membangkitkan pola yang baru, dimana akan diselesaikan beberapa submasalah dari variabel dual.

Digunakan hubungan dualitas yang akan mengubah fungsi tujuan dari meminimumkan sisa menjadi memaksimumkan jumlah *final* yang dapat dihasilkan.Akan terdapat beberapa submasalah (permasalahan *knapsack*) yang akan diselesaikan untuk membangkitkan pola yang baru. Terdapat dua permasalahan yang paling utama yaitu permasalahan program linier dan permasalahan *knapsack* yang akan diselesaikan secara iterasi sampai tidak ada lagi pola yang dapat dibangkitkan, yang dapat mengurangi jumlah gelondongan kayu yang dipotong.

Hubungan dualitas dalam permasalahan *cutting stock* ini dapat digambarkan sebagai berikut:



Gambar 2.1 Hubungan Dualitas

2.3.1 Pembangkitan Kolom yang Tertunda (*Delayed Column Generation)*

Misalkan a_{ij} adalah jumlah berapa kali pesanan dengan ukuran panjang i dihasilkan dalam pola j. Misalkan x_j adalah jumlah berapa kali pola ke-j dipotong. Dan d_i adalah permintaan untuk panjang pesanan ke-i dan n adalah jumlah dari pola yang ada dalam model. Permasalahan dapat diformulasikan sebagai berikut:

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

kendala $\sum_{j=1}^{n} a_{ij} x_j \geq d_i$

$x_j \geq 0$

Langkah pertama dilakukan untuk menyelesaikan permasalahan ini adalah dengan menggunakan *solusi pola inisial*. Untuk menentukan sebuah pola baru (yang akan meminimumkan jumlah gelondongan kayu yang terpakai), variabel dual (y) dari permasalahan ini dipakai untuk menyelesaiakn submasalah yaitu permasalahan *knapsack* yang diselesaikan secara iterasi. Untuk melakukan satu iterasi ke iterasi berikutnya digunakan metoda simpleks yang direvisi (*revised simplex*).

Nilai x_B dihitung dengan menggunakan $B x_B = b$, dimana $x_B = B^{-1}b$. Komputasi pada simpleks yang direvisi banyak berkaitan dengan memperbarui (updating) B^{-1} .

Misalkan suatu persoalan Program Linier dengan m pembatas sedang diselesaikan, misalkan x_k akan masuk basis, maka uji rasio menunjukkan bahwa x_k menjadi basis pada baris r, dan begitu seterusnya.

2.4 Metode Simpleks yang Direvisi (*Revised simplex Method*)

Metode *revised simplex* adalah suatu prosedur yang sistematik untuk mengimplementasikan langkah-langkah dari metode simpleks biasa ke dalam bentuk yang lebih sederhana.

Tujuan utama dari metode *revised simplex* adalah penggunaan invers dari suatu basis B^{-1} untuk menyelesaikan perhitungan-perhitungan simpleks dalam menentukan variabel yang masuk dan keluar.

Misalkan pemasalahan

$$\textit{Minimumkan} \qquad Z = \sum_{j=1}^{n} c_j x_j$$

$$\textit{kendala} \quad : \sum_{j=1}^{n} a_{ij} x_j \geq d_i$$

$$x_j > 0$$

Langkah-langkah penyelesaian dengan metode *revised simplex*:

Misalkan diberikan solusi awal yang layak dengan basis B (dan inversnya B^{-1}) maka :

1. Solusi awal yang layak diberikan oleh $x_B^* = B^{-1} d = \bar{d}$ dan $x_N = 0$. Nilai objektif $Z = c_B B^{-1} d = c_B \bar{d}$, x_N :variabel non-basis.

2. Hitung $w_B = C_B B^{-1}$ dan $z_j - c_j = c_B B^{-1} a_j - c_j$. Misalkan $z_k - c_k = maksimum \quad z_j - c_j$. Jika $z_k - c_k \leq 0$, berhenti; solusi sudah optimal. Jika tidak, lanjutkan ke langkah 3.

3. Hitung $y_k = B^{-1} a_k$. Jika $y_k \leq 0$ berhenti; solusi optimal tak berbatas. Jika tidak hitung indeks variabel x_{B_r} seperti di bawah ini:

$$\frac{\bar{d}_r}{y_{rk}} = \underset{1 \leq i \leq m}{Minimum} \left\{ \frac{\bar{d}_i}{y_{ik}} : y_{ik} > 0 \right\}$$

Ganti B dengan mengganti a_{B_r} dengan a_k dan kembali ke langkah 1.

Permasalahan minimasi dengan metode *revised simplex* ini dapat dibuat dalam bentuk tabel seperti :

Invers basis	RHS	x_k
		$z_k - c_k$

w	$c_B\bar{d}$		y_{1k}
	$\bar{d}_1$		y_{2k}
	$\bar{d}_2$		$\vdots$
B^{-1}	$\vdots$		y_{rk}
	$\bar{d}_r$		$\vdots$
	$\vdots$		y_{mk}
	$\bar{d}_m$		

Atau

Invers Basis	RHS	x_k
w	$c_B\bar{d}$	$z_k - c_k$
B^{-1}	$\bar{d}$	y_k

Gambar 2.2 Metode simpleks yang direvisi

2.5 Permasalahan *Knapsack*

Knapsack atau karung digunakan untuk memuat sesuatu. Dan tentunya tidak semua objek dapat ditampung di dalam karung. Karung hanya dapat menyimpan beberapa objek dengan total ukurannya lebih kecil atau sama dengan ukuran kapasitas karung. Setiap objek itupun tidak harus dimasukkan seluruhnya tetapi bisa juga sebagian saja.

Permasalahan *knapsack* adalah permasalahan optimasi kombinatorial, dimana harus mencari solusi terbaik dari banyak kemungkinan yang dihasilkan. Sebuah *knapsack* memiliki kapasitas total panjang pesanan L, dimana terdapat m buah item pesanan berbeda yang dapat ditempatkan dalam *knapsack*. Item i memiliki bobot W_i dan benefitnya atau harga dualnya y_i. Sebagai tambahan,

terdapat sejumlah d_i dari item *i* yang tersedia, dimana d_i adalah bilangan bulat positif dalam kisaran $1 \le d_i \le \infty$. Jika z_i adalah jumlah ukuran panjang pesanan *i* yang akan dimasukkan dalam *knapsack*. Maka secara umum tujuan yang harus tercapai adalah:

$$\begin{aligned} &\textit{Maksimumkan} && Z = \sum_{i=1}^{m} y_i z_i \\ &\textit{Kendala} && \sum_{i=1}^{m} W_i z_i \le L \\ & && z_i \text{ integer} \end{aligned}$$

Jika nilai optimal $Z < 1$, permasalahan telah terselesaikan. Sebaliknya, pola baru ini akan ditambahkan ke dalam program linier yang pertama.

2.5.1 Menyelesaikan Permasalahan *Knapsack* dengan Metode *Branch and Bound*

Pandang kembali bentuk umum permasalahan *knapsack* sebagai berikut:

$$\begin{aligned} &\textit{Maksimumkan} && Z = \sum_{i=1}^{m} y_i z_i \\ &\textit{Kendala} && \sum_{i=1}^{m} W_i z_i \le L \\ & && z_i \text{ noninteger} \end{aligned}$$

Dalam aplikasinya, seperti masalah *cutting stock*, $W_1, W_2, ..., W_m$ dan L adalah integer positif dan dimisalkan bahwa $y_1, y_2, ..., y_m$ integer positif. Dimana W_i merupakan panjang pesanan, y_i menunjukkan nilai fungsi tujuan yang berhubungan. Sehingga rasio y_i / W_i menunjukkan nilai dari setiap panjang dari jenis ke-i. Atau dengan kata lain dimisalkan bahwa y_i / W_i sebagai *keefisiensian* dari variabel z_i.

Tanpa menghilangkan keumumannya, diasumsikan variabel dengan nilai efisiensi yang menurun:

$$y_1/W_1 \geq y_2/W_2 \geq ... \geq y_m/W_m$$

Akhirnya, setiap solusi optimal memenuhi:

$$L - \sum_{i=1}^{m} W_i z_i < W_m$$

Sebelum menyelesaikan permasalahan *knapsack* maka dibuat pohon polanya terlebih dahulu, dimana ketika dua atau lebih cabang yang berasal dari titik yang sama maka cabang dimulai dari nilai yang lebih tinggi dahulu. Pada umumnya solusi seperti dari nilai cabang yang paling tinggi diusahakan dimasukkan ke dalam formula dan jika nilai z_1 pasti, dibuat nilai z_2 yang besar dan begitu seterusnya. Umumnya dengan $\lfloor Z \rfloor$ menunjukkan bilangan integer yang diperoleh dengan pembulatan ke bawah *Z*, solusi ini didefinisikan dengan rekursif:

$$z_j = \left\lfloor \left(L - \sum_{i=1}^{j-1} W_i z_i \right) \Big/ W_j \right\rfloor \qquad (j = 1,2,...,m)$$

dan pada umumnya penyelesaian dimulai dengan menghitung nilai $z_1 = \lfloor L/W_1 \rfloor$.

Jika belum mendapatkan solusi yang baik, akan diteliti sampai ke ujung cabang, dan mencari nilai terbaik dari $z_1^*, z_2^*, ..., z_m^*$ dan akan menggantinya dengan suatu solusi yang lebih baik yang kapan saja bisa muncul.

Penyelesaian di atas tanpa disadari telah digunakan suatu diagram pohon. Dari setiap cabang yang sudah diuji proses kembali ke akar secara bertahap, dan tetap memperhatikan cabang yang belum diselidiki. Ketika satu cabang sudah ditemukan, maka pencabangan kembali dicabangkan ke arah cabang yang baru dari simpulnya masing-masing.

Diselesaikan kembali dari cabang yang bernilai lebih besar ke nilai yang kecil. Setelah menguji solusi yang layak $z_1, z_2,..., z_m$, ditetapkan *k* sama dengan $m-1$ dan jika perlu mengurangi atau mereduksi *k* sampai $z_k > 0$ dan menggantikan z_k dengan $z_k - 1$ dan mencari nilai dari $z_{k+1}, z_{k+2},..., z_m$ secara berulang dengan menggunakan:

$$z_j = \left\lfloor \left(L - \sum_{i=1}^{j-1} W_i z_i \right) \Big/ W_j \right\rfloor$$

Dengan cara yang cepat akan mencegah menghitung cabang yang sia-sia, artinya yang tidak memberikan nilai yang kurang baik dan dengan tepat memberi solusi terbaik sekarang $z_1^*, z_2^*,..., z_m^*$ dan menghasilkan

$$\sum_{i=1}^{m} y_i z_i^* = M$$

dan menarik kembali dari beberapa $z_1, z_2,..., z_m$ kearah akar, dan baru saja menemukan yang nilai paling besar *k* sehingga $k < m-1$ dan $z_k > 0$, sama seperti sebelumnya akan ditetapkan

$$\bar{z}_i = z \qquad \text{untuk semua } i = 1,2,...,k-1$$
$$\bar{z}_k = z_k - 1$$

Dan mulai untuk menguji berbagai pilihan $\bar{z}_{k+1}, \bar{z}_{k+2},..., \bar{z}_m$, dari nilai efisiensi yang semakin menurun seperti di atas dan masing-masing variabel $z_{k+1}, z_{k+2},..., z_m$, mempunyai suatu efisiensi paling banyak y_{k+1}/W_{k+1}, dengan demikian

$$\sum_{i=k+1}^{m} y_i \bar{z}_i \le \frac{y_{k+1}}{W_{k+1}} \sum_{i=k+1}^{m} W_i \bar{z}_i$$

Karenanya diasumsikan

$$\sum_{i=1}^{m} W_i \bar{z}_i \leq L$$

Akibatnya

$$\sum^{m} y_i \bar{z}_i \leq \sum^{k} y_i \bar{z}_i + \frac{y_{k+1}}{W_{k+1}} \left(L - \sum_{i=1}^{k} W_i \bar{z}_i \right)$$

Dalam bentuk pohon enumerasi, pertidaksamaan

$$\sum_{i=1}^{k} y_i \bar{z}_i + \frac{y_{k+1}}{W_{k+1}} \left(L - \sum_{i=1}^{k} W_i \bar{z}_i \right) \leq M$$

Akan dibuat suatu alur untuk suatu nilai yang sudah ditetapkan tidak bernilai dan untuk menyelidikinya lebih lanjut. Sesungguhnya jika semua koefisien $y_1, y_2, ..., y_m$ bilangan bulat positif, maka *M* adalah bilangan bulat, dan mungkin digantikan oleh pertidaksamaan

$$\sum_{i=1}^{k} y_i \bar{z}_i + \frac{y_{k+1}}{W_{k+1}} \left(L - \sum_{i=1}^{k} W_i \bar{z}_i \right) < M + 1$$

Dalam bentuk perhitungan yang memakai pohon enumerasi, tidak akan dicari lagi cabang yang tidak terlalu baik, sehingga dapat di potong (*pruned off*). Atau penyelesaian di atas merupakan penyelesaian dengan menggunakan ***metode Branch and bound.***

Prosedur atau tahapan dari metode *Branch and bound* ini dalam menyelesaikan permasalahan *knapsack* dijelaskan sebagai berikut:

Tahap 1. [awal] Misalkan $M = 0, k = 0.$

Tahap 2. [menentukan perpanjangan cabang yang menjajikan]. Untuk $j = k+1, k+2, \ldots, m$, tetapkan $z_j = \left\lfloor \left(L - \sum_{i=1}^{j-1} W_i z_i \right) \Big/ Wj \right\rfloor$, kemudian ganti *k* dengan *m*.

Tahap 3. [memperbaiki solusi] Jika $\sum_{i=1}^{m} y_i z_i > M$, kemudian ganti *M* dengan $\sum_{i=1}^{m} y_i z_i$ dan ganti $x_1^*, x_2^*, \ldots, x_m^*$ dengan $x_1, x_2, \ldots, x_m$.

Tahap 4. [tarik kembali ke cabang berikut]

a. jika $k = 1$, maka berhenti; jika tidak ganti *k* dengan *k-1*
b. jika $x_k = 0$, maka kembali ke a: jika tidak ganti x_k dengan $x_k - 1$.

Tahap 5. [pencarian cabang yang bernilai baik] jika $\sum_{i=1}^{k} y_i \bar{z}_i + \frac{y_{k+1}}{W_{k+1}} \left(L - \sum_{i=1}^{k} W_i \bar{z}_i \right) < M + 1$ dengan x_i adalah $\bar{x}_i$ gagal, maka kembali ke tahap 2; jika tidak maka kembali ke tahap 4.