Sudi dan Implementasi Algoritma Optimasi Pemotongan Bar Steel

Odit Ekwardo - 13504079 [1)]

1) Program Studi Teknik Informatika ITB, Bandung 40132, email: if14079@students.if.itb.ac.id

Abstract *– Pada tugas akhir ini dibahas mengenai permasalahan pemotongan bar steel, dimana ukuran bar steel yang dihasilkan pabrik dalam bentuk rol tidak selalu sesuai dengan ukuran yang dibutuhkan dalam konstruksi sehingga rol tersebut harus dipotong-potong sesuai kebutuhan. Penyusunan pola pemotongan sangat penting dalam mendapatkan sisa minimal bar steel karena perbedaan pola pemotongan dapat menghasilkan sisa bar steel terbuang yang berbeda pula. Dalam kajian informatika dikenal beberapa algoritma optimasi yang dapat menyelesaikan permasalahan pola pemotongan. Oleh karena itu perlu dibuat sebuah perangkat lunak yang mengimplementasikan algoritma optimasi tersebut. Perangkat lunak dibuat menggunakan bahasa C# Visual Studio 2005. Perangkat lunak ini mengimplementasikan algoritma optimasi brute force, greedy, dan program dinamis untuk mencari solusi. Setelah diuji, perangkat lunak ini menunjukkan bahwa algoritma brute force selalu menghasilkan solusi paling optimal, tetapi paling lambat dalam proses pencarian solusi. Algoritma greedy selalu paling cepat dalam proses pencarian solusi, dan algoritma program dinamis adalah algoritma terbaik secara keseluruhan karena algoritma ini menghasilkan solusi yang mendekati optimal dengan kecepatan proses yang cepat untuk tipe persoalan yang mirip dengan kenyataan. Metode brute force tidak muncul sebagai algoritma paling optimal karena kecepatan prosesnya yang bisa mencapai lebih dari 2 hari untuk persoalan kompleks yang biasa dihadapi di kenyataan.*

Kata Kunci*: bar steel*, algoritma optimasi, *brute force*, *greedy*, program dinamis, *rol*, pola pemotongan.

1. PENDAHULUAN

Sebagai negara berkembang, Indonesia banyak sekali melakukan pembangunan, baik yang dilakukan oleh pemerintah maupun pihak swasta. Pembangunan yang dimaksud antara lain pembangunan jalan, pusat perbelanjaan, jembatan, jalan layang, gedung perkantoran, dan lain-lain.

Seiring dengan pesatnya pembangunan di Indonesia, industri jasa konstruksi pun semakin marak berkembang. Banyaknya perusahaan jasa konstruksi yang berdiri di Indonesia membuat persaingan untuk merebut hati pemilik proyek semakin tinggi. Tiap perusahaan konstruksi tersebut ikut dalam persaingan yang dinamakan *tender*. hal yang paling mempengaruhi keputusan pemilik proyek untuk menentukan perusahaan mana yang akan memenangi tender adalah anggaran yang diajukan oleh perusahaan konstruksi tersebut untuk dapat menyelesaikan proyek hingga tuntas. Anggaran merupakan dana perencanaan yang diusulkan oleh perusahaan konstruksi kepada pemilik proyek. Perusahaan konstruksi yang memenangkan tender akan mengoptimalkan penggunaan anggaran yang diterimanya untuk memperoleh keuntungan yang sebesar-besarnya. Optimasi yang dilakukan adalah dengan meminimalisasi pemakaian bar steel.

Bar steel merupakan salah satu bahan baku utama pembentuk bangunan. Bar steel digunakan sebagai penopang berdirinya suatu bangunan. Bar steel yang digunakan untuk mendirikan bangunan terdiri dari berbagai macam panjang, sedangkan bar steel yang disediakan di pasaran dijual dengan panjang standar. Oleh karena itu, bar steel yang telah dibeli harus dipotong sesuai dengan kebutuhan konstruksi bangunan. Pemotongan bar steel pasti menyisakan sisa yang tidak dapat digunakan untuk konstruksi bangunan karena panjang yang tidak sesuai. Namun, perusahaan konstruksi ingin mengoptimalkan pemotongan bar steel agar sisa besi yang tidak digunakan seminimal mungkin.

Dalam kajian informatika, terdapat beberapa algoritma yang digunakan dalam melakukan optimasi, misalnya algoritma greedy. Algoritma greedy memiliki kelebihan dan kekurangan jika dibandingkan dengan algoritma lain. Keuntungannya adalah algoritma ini dapat menyelesaikan masalah relatif lebih cepat dari algoritma lain dan hasilnya pun cukup akurat. Namun jika dibandingkan dengan teknik brute force algoritma ini tidak menghasilkan pola optimasi terbaik karena tidak memeriksa semua kemungkinan yang ada. Untuk lebih jelasnya dapat dilihat pada [MUN06].

Oleh karena itu, pada tugas akhir ini akan dikaji algoritma *brute force*, *greedy*, dan program dinamis untuk mencari pola pemotongan *bar steel* yang paling optimal, kemudian dibuat perangkat lunak untuk analisis lebih lanjut.

2. LANDASAN TEORI

2.1. Bar Steel

Bar steel merupakan salah satu bahan dasar bangunan

yang memiliki peranan penting dalam satu kesatuan bangunan. *Bar steel* menjadi fondasi dari berdirinya bangunan karena dipakai hampir diseluruh bagian bangunan. Pada tahap awal konstruksi, bentuk bangunan dibuat dari bermacam-macam *bar steel*, sesuai dengan kebutuhan bangunan. *Bar steel* dibuat oleh pabrik dengan panjang standar. Satu batang *bar steel* yang dihasilkan oleh pabrik dengan panjang standar tersebut biasanya dihitung sebagai satu *rol bar steel*. Oleh karena itu, *bar steel* dipotong-potong sesuai dengan kebutuhan bentuk bangunan. Saat perancangan, arsitek yang merancang bangunan telah memilki perhitungan mengenai panjang-panjang *bar steel* yang digunakan, sehingga dapat ditentukan kebutuhan panjang *bar steel* untuk konstruksi.

2.2. Algoritma Optimasi

Jika diberikan suatu permasalahan yang solusinya dapat diselesaikan dengan menggunakan fungsi *f* dan solusi optimal yang diinginkan adalah *f(x)*, maka algoritma optimasi adalah metode yang digunakan untuk menemukan nilai *x*, misalnya menemukan kemungkinan yang terbesar (atau terkecil) dari suatu fungsi *f* dengan *constraint* yang diberikan oleh variabel *x*. Dalam hal ini, *x*, dapat berupa nilai skalar atau vektor dari nilai yang kontinu atau nilai diskrit [WIK07-a].

2.2.1 Algoritma Brute Force

Brute force adalah sebuah pendekatan yang lempang (*straightforward*) untuk memecahkan masalah, biasanya didasarkan pula pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan. Algoritma *brute force* memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (*obvious way*) [MUN06].

2.2.2 Algoritma Greedy

Algoritma *greedy* membentuk solusi langkah per langkah (*step by step*). Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak bisa diubah lagi pada langkah selanjutnya. Sebagai contoh, jika menggunakan algoritma *greedy* untuk menempatkan komponen diatas sirkuit, sekali sebuah komponen telah ditetapkan posisinya, komponen tersebut tidak dapat dipindahkan lagi. Pendekatan yang digunakan di dalam algoritma *greedy* adalah membuat pilihan yang "tampaknya" memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (*local optimum*) pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global (*global optimum*) [MUN06].

2.2.3 Algoritma Program Dinamis

Program dinamis adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Pada penyelesaian persoalan dengan metode ini terdapat sejumlah berhingga pilihan yang mungkin, solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya, dan kita menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap [MUN06].

3. ANALISIS

3.1. Latar Belakang Masalah

Adanya keterbatasan mesin penghasil *bar steel* dan bermacam-macamnya ukuran *bar steel* yang diinginkan oleh konsumen menimbulkan beberapa permasalahan dalam menangani keterbatasan mesin produksi *bar steel* pada pabrik produsen *bar steel*. Masalah pertama adalah *assortment problem*, yaitu permasalahan untuk menentukan ukuran-ukuran *rol bar steel* yang harus diproduksi yang paling mendekati pemenuhan terhadap kebutuhan konsumen sehingga stok yang terpakai minimal. Masalah yang kedua adalah permasalahan untuk menemukan pola terhadap *rol bar steel* yang dihasilkan pabrik menjadi potongan-potongan yang lebih kecil umtuk memenuhi kebutuhan konsumen. Kedua masalah ini jika diteliti dapat diselesaikan dengan algoritma-algoritma yang telah ada, namun lebih baik lagi jika diteliti lagi untuk mencari algoritma lain yang lebih baik yang dibuat dari hasil modifikasi algoritma yang telah ada.

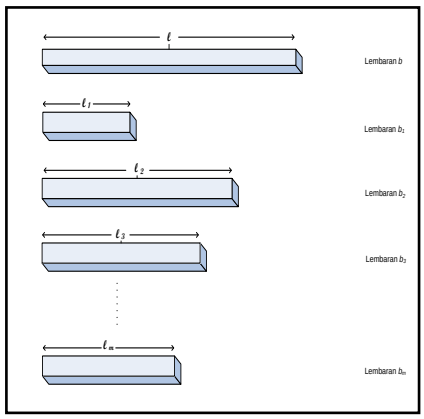
3.2 Pola Pemotongan

Pola adalah bentuk atau model (atau, lebih abstrak, suatu *set* peraturan) yang bisa dipakai untuk membuat atau untuk menghasilkan suatu atau bagian dari sesuatu, khususnya jika sesuatu yang ditimbulkan cukup mempunyai suatu yang sejenis untuk pola dasar yang dapat ditunjukkan atau terlihat, yang mana sesuatu itu dikatakan memamerkan pola [WIK07-b].

Dalam pola pemotongan, yang dimaksud pengelolaan objek adalah bagaimana cara memotong-motong sebuah objek yang besar menjadi sebuah objek yang lebih kecil. Objek yang besar tersebut dikelola dengan dipotong-potong menjadi potongan yang lebih kecil dengan tujuan agar objek tersebut menjadi lebih berguna. Biasanya pemotongan terhadap suatu objek dilakukan karena adanya kebutuhan terhadap objek yang lebih kecil, sedangkan yang objek yang tersedia tidak bisa memenuhi kebutuhan tersebut. Perbedaan cara yang ditempuh dalam melakukan pemotongan terhadap suatu objek dapat menghasilkan sesuatu yang jauh berbeda satu sama lain. Untuk lebih jelasnya, definisi pola pemotongan dapat diperjelas melalui ilustrasi berikut ini:

> Seorang pengusaha lembaran baja memproduksi *rol b* dengan panjang standar *l*. *Order* besar sedang dikerjakan sehubungan dengan permintaan

pelanggan yang memerlukan lembaran dengan panjang yang bermacam-macam. Secara khusus, lembaran b_i dengan panjang l_i untuk i= 1, 2, ..., m akan di*order* (lebih jelasnya dapat dilihat pada Gambar 1). Pengusaha berkeinginan untuk memotong *rol* standar sedemikian sehingga memenuhi order dan meminimalkan sisanya. Karena potongan sisa tidak berguna bagi pengusaha, tujuannya adalah untuk meminimalkan jumlah *rol* yang diperlukan untuk memenuhi *order*. Diketahui lembaran standar dengan panjang *l*, ada banyak cara memotongnya. Cara yang demikian disebut dengan pola pemotongan.



Gambar 1 Deskripsi Pola Pemotongan

3.3 Metode Penyelesaian

Permasalahan pada perusahaan konstruksi dalam memilih pola yang cocok untuk diterapkan pada *bar steel* agar hanya menyisakan potongan *bar steel* yang tak terpakai seminimal mungkin. Seperti yang telah dijelaskan pada Bab II, permasalahan pencarian pola pemotongan paling optimal dapat diselesaikan dengan berbagai macam algoritma optimasi. Khusus untuk tugas akhir ini, algoritma yang diperbandingkan adalah algoritma *greedy*, program dinamis, dan *brute force*. Ketiga algoritma ini terpilih karena ketiga algoritma ini paling lazim digunakan berbagai program optimasi pola pemotongan (kecuali algoritma *brute force*, yang hanya akan dijadikan pembanding hasil kedua algoritma lainnya).

3.3.1 *Brute Force*

Metode penyelesaian dengan *brute force* ini pasti menghasilkan solusi yang paling baik karena metode ini menelusuri dan membandingkan seluruh kemungkinan yang ada, sehingga solusi yang muncul pasti yang terbaik dari semua metode. Namun, metode ini memiliki kompleksitas yang sangat besar (akan dijelaskan selanjutnya), sehingga metode ini tidak akan digunakan sebagai algoritma optimasi terbaik, melainkan sebagai metode pembanding algoritma lain.

Untuk lebih jelasnya, langkah-langkah penyelesaian persoalan diatas menggunakan metode *brute force* adalah sebagai berikut:

1. Buat *list list_solusi* dan *solution*, kemudian diinisialisasi kosong.
2. Buat variabel *sisa* untuk menghitung jumlah sisa yang telah dihasilkan oleh pola pemotongan. Buat juga variabel *sisaMin* untuk membandingkan pola mana yang menghasilkan pola minimal. Inisisalisasi *sisa* dengan 0 dan *sisaMin* dengan 999.
3. Inisialisasi *i*=0.
4. Masukkan *pattern*[*i*] kedalam *list_solusi*. Kurangi *problem* dengan *pattern*[*i*] Tambah *sisa* dengan sisa*pattern*[*i*].
5. Lakukan tahap 6 hingga terdapat minimal satu nilai negatif pada *problem*. Nilai *i* bertambah 1.
6. Kembali ke tahap 6 hingga semua nilai pada *problem* adalah 0. Hitung nilai *sisa* kemudian bandingkan dengan *sisaMin*, jika sisa lebih kecil maka nilai *sisaMin* diganti dengan *sisa*, kosongkan *solution* kemudian *copy list_solusi* ke *solution*.
7. Hapus *pattern*[*i*] dari *list_solusi* dan tambahkan nilai sisa dengan nilai *sisapattern*[*i*]. Nilai *i* bertambah 1. Kembali ke tahap 8.
8. Lakukan tahap 9 hingga nilai *i* sama dengan *numberofpattern* hingga akhirnya seluruh tahap pernah ditempati oleh semua *pattern*.
9. Tampilkan *list solution* yang berisi langkah-langkah pemilihan pola pemotongan ke layar.

Penyelesaian persoalan menggunakan algoritma *brute force* memiliki kompleksitas $T(n,k) = n(1+(k-1)(2+(k^2-k)/2))$ dengan n adalah jumlah *pattern* yang digunakan dari sebuah permasalahan pemotongan dan k adalah jumlah tahap pencarian hingga selesai.

3.3.2 *Greedy*

Penyelesaian dengan algoritma persoalan diatas dengan menggunakan algoritma *greedy* adalah dengan memilih langkah yang terlihat paling mendekati solusi, yaitu dengan memilih pola yang memiliki sisa terkecil terlebih dahulu. Pada implementasi algoritma *greedy*, yang dilakukan hanya memilih *pattern* yang memiliki sisa yang paling kecil dan menghasilkan *bar steel* paling panjang dari kumpulan *pattern* yang masih mungkin diterapkan. Kompleksitas yang dimiliki algoritma *greedy* dalam menyelesaikan masalah ini adalah $T(n,k) = n+(k-1)$ dengan n adalah jumlah *pattern* yang dapat terbentuk dari sebuah permasalahan pemotongan dan k adalah jumlah tahap

pencarian hingga selesai.

3.3.3 Program Dinamis

Penyelesaian dengan algoritma persoalan diatas dengan menggunakan algoritma program dinamis adalah dengan memilih langkah yang memenuhi prinsip optimalitas. Dengan prinsip optimalitas ini dijamin bahwa setiap keputusan yang diambil pada suatu tahap adalah keputusan yang benar untuk tahap-tahap selanjutnya juga, tidak hanya benar untuk tahap itu saja.

Secara matematis, penyelesaian masalah ini dengan program dinamis dapat dituliskan sebagai berikut:

$$f_1(s) = c_{x1s} \quad (basis)$$

$$f_k(s) = \min\{c_{xks} + f_{k-1}(x_k)\}, \; k=2,3,4, \ldots, n \quad (rekurens)$$

dengan k adalah tahap (*level*) pencaraian, s adalah status yang berhubungan dengan masing-masing tahap (*level*) yang merupakan simpul-simpul dalam *graf* (x_1, x_2, x_3, …, x_n), c_{xks} menyatakan bobot sisi dari x_k ke s, $f_k(x_k,s)$ menyatakan total bobot lintasan dari x_k ke s, $f_k(s)$ adalah nilai minimal dari $f_k(x_k,s)$, dan n adalah jumlah tahap pencarian yang dilakukan.

Implementasi algoritma program dinamis akan menggunakan tabel-tabel pembandingan pada setiap tahap pencarian. Kompleksitas yang dimiliki algoritma program dinamis dalam menyelesaikan masalah optimasi pemotongan adalah O(k) dengan k adalah jumlah *level* yang dijalankan oleh algoritma untuk menyelesaikan sebuah permasalahan pemotongan.

3.4 Analisis Metode Penyelesaian

Jika terdapat persoalan seperti berikut ini:

> Misalkan kita ingin mendapatkan *bar steel* dengan panjang 5 meter sebanyak 3 buah, 7 meter sebanyak 2 buah, dan 8 meter sebanyak 2 buah dengan memotong *bar steel* yang panjangnya 10 meter dan 12 meter.

Perbandingan hasil dan performansi 3 algoritma dapat dilihar pada tabel 1.

Algortima	Kompleksitas (T(n,k))	Sisa *Bar steel* minimal yang tidak terpakai
Brute force	$n(1+(k-1)(2+(k^2-k)/2))$	7 meter
Greedy	$n+k-1$	9 meter
Program Dinamis	k	7 meter

Keterangan:
n: jumlah pola yang terdefinisi
k: jumlah tahap yang dilalui hingga selesai

3.5 Analisis Metode Penyelesaian Menggunakan Perangkat Lunak

Berdasarkan data keluaran dan dari berbagai model data masukan yang telah dilakukan pada saat pengujian perangkat lunak, dapat dilihat bahwa penyelesaian menggunakan metode *brute force* selalu menghasilkan sususan pola pemotongan yang menghasilkan sisa paling sedikit. Hal itu terjadi karena algoritma *brute force* membandingkan seluruh kemungkinan solusi. Namun, ada konsekuensi dari hal metode lempang tersebut, yaitu waktu proses pencarian solusi yang sangat lama. Hal ini dapat dilihat dari catatan waktu proses yang dihasilkan *brute force* meningkat tajam seiring bertambah kompleksnya data masukan.

Buruknya catatan waktu yang dihasilkan algoritma *brute force* dalam menyelesaikan masalah, membuat pencarian algoritma lain yang lebih efisien menjadi penting. Untuk persoalan yang sederhana seperti persoalan 1,2, dan 3, *brute force* menjadi yang terbaik, tetapi untuk persoalan yang agak kompleks, perlu dicermati kedua algoritma lainnya karena waktu pencarian *brute force* dengan persoalan yang kompleks mencapai lebih dari 2 hari. Jika dilihat dari hasil data keluaran pada pengujian diatas, 2 algoritma lain, *greedy* dan program dinamis, berimbang dalam hal sisa yang dihasilkan. Pada beberapa kasus *greedy* lebih baik dibanding program dinamis, pada kasus lain terjadi sebaliknya. Namun, catatan penting yang harus diperhatikan adalah kecepatan performansi yang dibuat oleh algoritma *greedy*. Hampir pada semua kasus, baik yang kompleks ataupun sederhana, waktu yang diperlukan oleh *greedy* untuk menyelesaikan masalah tidak lebih dari 1 milidetik. Hal ini menjadi poin tersendiri bagi algoritma *greedy*. Pada kasus dimana program dinamis lebih baik dibanding *greedy*, data masukan lebih masuk akal dan lebih mirip dengan kehidupan nyata. Data yang digunakan merupakan data yang biasa digunakan perusahaan jasa konstruktor. Ciri-ciri data tersebut adalah ukuran panjang yang diinginkan tidak berurut, tetapi dengan berselisih lebih dari 1. Untuk persoalan dengan data ukuran *bar steel* yang dibutuhkan dimana antar ukuran hanya berselisih 1 meter, *greedy* mengungguli program dinamis walaupun selisihnya tidak terlalu besar.

4. KESIMPULAN

Dari keseluruhan isi makalah ini, dapat diambil kesimpulan sebagai berikut:

1. Algoritma *brute force* selalu menghasilkan solusi paling optimal (sisa *bar steel* yang tidak terpakai paling minimal) untuk permasalahan pemotongan *bar steel*. Hal ini dapat dilihat dari hasil pengujian menggunakan beberapa tipe persoalan, dimana algoritma *brute force* selalu menghasilkan solusi paling optimal. Namun jika dilihat dari hasil data pengujian, algoritma *brute force* sangat lambat dalam menyelesaikan persoalan dengan data yang

sangat kompleks.

2. Algoritma *greedy* adalah algoritma paling efisien dalam meyelesaikan persoalan pemotongan *bar steel* jika dibandingkan dengan algoritma *brute force* dan program dinamis.
3. Algoritma program dinamis merupakan algoritma yang terbaik untuk menyelesaikan tipe persoalan pemotongan *bar steel* yang biasa dihadapi pengusaha konstruksi di kehidupan nyata.

DAFTAR PUSTAKA

[CEN07] *http://www.centralsteel.com*

[DIM97] Dimyati T.T., & Dimyati A., *Operation Research*: Model-Model Pengambilan Keputusan, Sinar Baru, Bandung, 1992.

[HOR78] Horowitz, Ellis, & Sartaj Sahni, *Fundanental of Computer Algorithms*, *Pitman Publishing Limited*, 1978.

[MUN06] Munir, Rinaldi, Strategi Algoritmik, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2006.

[NEA96] Neapolitan, Richrad R., *Foundations of Algorithms*, *D.C. Heath and Company*, 1996.

[WIK07-a] *http://en.wikipedia.org/wiki/ Category:Optimization_algorithms*

[WIK07-b] *http://id.wikipedia.org/wiki/Pola*