Meminimalkan Sisa Pemotongan Besi Beton dalam Proyek Konstruksi

Bambang Santoso[1], Sofyan Mufti Prasetiyo[2], Agung Wijoyo[3]

[1,2,3] Program Studi Teknik Informatika, Universitas Pamulang
Jln. Surya Kencana No. 1 Pamulang, Tangerang Selatan, Banten, Indonesia
e-mail: {[1]dosen01692, [2]dosen01809, [3]dosen01671}@unpam.ac,id

Abstract

Construction industries are a must in developing countries. Constructions take part in almost every corner of the country. In construction projects, reinforced concrete plays big part in any construction. A long bar of steel are cut into be several lengths to satisfy the requirements of the buildings to be build. In this process, waste cannot be avoided. And the waste of steel bar is very destructive to environment. This research is to find a way to minimize the waste in the process of cutting steel bars. The algorithm used is Greedy Search. Greedy Search is where the optimized solution in every step is taken, in the hope that the overall solution will be optimized for the problem. Optimized solution here is the minimum waste by the steel bar cutting. This will minimize the cost by the construction companies and at the same time preserve the environment.

Keywords: minimize waste, greedy search, optimized solution, construction project, cutting stock problems.

1. Pendahuluan

Di negara maju maupun negara berkembang, pembangunan tidak bisa terlepas dari proyek konstruksi baik berupa gedung, jalan, jembatan atau lainnya. Dalam banyak hal, bangunan ditopang oleh beton bertulang. Pembuatan beton bertulang membutuhkan besi beton dengan panjang yang beragam. Besi beton dibeli dalam panjang yang sama. Kemudian dipotong-potong sesuai kebutuhan yang diperlukan konstruksi.

Dalam memotong besi beton, banyak terjadi sisa potongan yang terlalu pendek untuk digunakan. Sisa ini akan dibuang dan menjadi sampah. (Abuhassan & Nasereddin, 2011) Sampah besi beton ini akan mencemari lingkungan dan sangat lama untuk dihancurkan oleh alam. Di samping itu, sisa yang banyak akan menambah ongkos produksi.

Penelitian ini mencoba menjawab masalah tersebut dengan memakai algoritma *Greedy Search.* Pemotongan besi beton yang efektif akan meminimalkan sampah besi. Ini pada gilirannya akan membantu memperbaiki lingkungan, di samping mengurangi ongkos produksi.

2. Metodologi

Kebutuhan Besi Beton Dalam Konstruksi

Contoh kebutuhan untuk konstruksi adalah sebagai berikut.

Tabel 1 Kebutuhan besi konstruksi

Diameter	Item	Panjang	Banyaknya
10 mm	1	1.9 m	200
	2	2.2 m	150
	3	2.7 m	100
	4	3.1 m	200
13 mm	1	2,1 m	220
	2	3,5 m	180
	3	5,4 m	170
	4	6,1 m	140
	5	6,8 m	130

Kebutuhan satu tipe besi tidak dapat digantikan dengan tipe besi beton lain. Misal tipe 10mm tidak dapat digantikan dengan tipe 13mm. Maka untuk penelitian, dapat diambil hanya satu tipe saja. Area tipe 1 dan tipe 2 akan selalu terpisah satu sama lain.

Panjang besi beton standar adalah 12 meter. Dengan keperluan konstruksi beragam, besi ini harus dipotong-potong agar sesuai kebutuhan.

Untuk ini, perencanaan diperlukan untuk pemotongan besi beton. Perencanaan ini harus menjawab dua pertanyaan:

1. Bagaimana pola pemotongan yang harus dilakukan untuk setiap batang besi beton?
2. Berapa batang besi beton yang harus dibeli untuk memenuhi seluruh keperluan konstruksi?

Pemilihan Metode

Cara terbaik dalam pencarian cara memotong besi adalah dengan metode *Brute Force*. Dengan *Brute Force*, seluruh kemungkinan diperiksa satu per satu dan dicari solusi optimal. Cara ini terbaik dan dijamin menghasilkan solusi optimal, tapi membutuhkan waktu yang lama. Waktu yang diperlukan meningkat secara eksponesial mengikuti banyaknya variabel yang dipakai. (Feo & Resende, 1995) (Wilt, Thayer, & Ruml, 2010) Karenanya *Brute Force* sering ditinggalkan kecuali jika medan yang dipakai dipastikan sempit dan terbatas. *Greedy Search* adalah pendekatan cara lain yang akan dipakai di penelitian ini.

Algoritma Greedy Search

Algoritma didefinisikan sebagai prosedur dengan langkah-langkah berbatas untuk mencapai tujuan yang diharapkan. (Subhadra, 2016). *Greedy Search* (Pencarian Rakus) adalah algoritma dengan mengikuti penyelesaian masalah di mana dipilih nilai optimal lokal di setiap tahapan dengan harapan nanti akan mendapatkan hasil optimal secara global. Dalam beberapa kasus, strategi Pencarian Rakus tidak selalu mendapatkan hasil global optimal. Tapi secara umum, algoritma Pencarian Rakus dapat menghasilkan solusi optimal lokal yang mendekati hasil optimal global dengan waktu relatif singkat dan dengan banyak langkah yang bisa diprediksi.

Algoritma Pencarian Rakus mengandung dua sifat. (Malik, Sharma, & Saroha, 2013)

1. Sifat pemilihan rakus: solusi global optimal dapat dicapai dengan membuat solusi lokal optimal. Dengan kata lain, solusi optimal dapat diperoleh dengan membuat pilihan rakus.
2. Sub struktur optimal: solusi optimal mengandung sub solusi optimal. Atau dengan kata lain, solusi optimal untuk suatu masalah adalah solusi optimal untuk sub masalah.

Mencari Kombinasi

Untuk mencari kombinasi pemotongan, dicari terlebih dahulu jumlah panjang maksimum kombinasi. Langkah yang dipakai adalah:

- Mencari panjang minimum kebutuhan.
- Membagi panjang besi (12 meter) dengan panjang minimum tersebut, hasil dibulatkan ke bawan

Misal untuk tabel 1 di atas, panjang minimum adalah 1,9 meter. Maka panjang kombinasi adalah 12/1,9 = 6 (dibulatkan ke bawah). Ini disebut *MaxCountItem*.

Kemudian, dicari semua kombinasi yaitu sejumlah C(p,m) di mana p=n+*MaxCountItem*-1 dan m=n-1. Ini adalah kombinasi dengan pengulangan (Brualdi, 2010). Jumlah kombinasi C(p,m) adalah

```
C(p,m) = p!/(m!*(p-m)!)
```

dengan p! = 1 * 2 * 3 * ... * p

Dalam contoh tabel 1 di atas, kombinasi adalah C(9,3). Jumlah item yang diinginkan n=4. Dengan *MaxCountItem*=6 maka p=9 dan m=3. Jumlah kombinasi C(9,3) adalah 9!/(3!*6!) = 84.

Tabel 1 Daftar Semua Kombinasi

No	P1	P2	P3	P4	P5	P6	Tot	Ket.
1	1	1	1	1	1	1	11,4	Valid
2	1	1	1	1	1	2	11,7	Valid
3	1	1	1	1	1	3	12,2	Tidak Valid
4	1	1	1	1	1	4	12,6	Tidak Valid
5	1	1	1	1	2	2	12	Valid
6	1	1	1	1	2	3	12,5	Tidak Valid
7	1	1	1	1	2	4	12,9	Tidak Valid
8	1	1	1	1	3	3	13	Tidak Valid
9	1	1	1	1	3	4	13,4	Tidak Valid
10	1	1	1	1	4	4	13,8	Tidak Valid
11	1	1	1	2	2	2	12,3	Tidak Valid
12	1	1	1	2	2	3	12,8	Tidak Valid
13	1	1	1	2	2	4	13,2	Tidak Valid
14	1	1	1	2	3	3	13,3	Tidak Valid
15	1	1	1	2	3	4	13,7	Tidak Valid
16	1	1	1	2	4	4	14,1	Tidak Valid
17	1	1	1	3	3	3	13,8	Tidak Valid
18	1	1	1	3	3	4	14,2	Tidak Valid
19	1	1	1	3	4	4	14,6	Tidak Valid
20	1	1	1	4	4	4	15	Tidak Valid
21	1	1	2	2	2	2	12,6	Tidak Valid
22	1	1	2	2	2	3	13,1	Tidak Valid
23	1	1	2	2	2	4	13,5	Tidak Valid
24	1	1	2	2	3	3	13,6	Tidak Valid
25	1	1	2	2	3	4	14	Tidak Valid
26	1	1	2	2	4	4	14,4	Tidak Valid
27	1	1	2	3	3	3	14,1	Tidak Valid
28	1	1	2	3	3	4	14,5	Tidak Valid
29	1	1	2	3	4	4	14,9	Tidak Valid
30	1	1	2	4	4	4	15,3	Tidak Valid
31	1	1	3	3	3	3	14,6	Tidak Valid
32	1	1	3	3	3	4	15	Tidak Valid
33	1	1	3	3	4	4	15,4	Tidak Valid
34	1	1	3	4	4	4	15,8	Tidak Valid
35	1	1	4	4	4	4	16,2	Tidak Valid
36	1	2	2	2	2	2	12,9	Tidak Valid
37	1	2	2	2	2	3	13,4	Tidak Valid
38	1	2	2	2	2	4	13,8	Tidak Valid
39	1	2	2	2	3	3	13,9	Tidak Valid
40	1	2	2	2	3	4	14,3	Tidak Valid

No	P1	P2	P3	P4	P5	P6	Tot	Ket.
41	1	2	2	2	4	4	14,7	Tidak Valid
42	1	2	2	3	3	3	14,4	Tidak Valid
43	1	2	2	3	3	4	14,8	Tidak Valid
44	1	2	2	3	4	4	15,2	Tidak Valid
45	1	2	2	4	4	4	15,6	Tidak Valid
46	1	2	3	3	3	3	14,9	Tidak Valid
47	1	2	3	3	3	4	15,3	Tidak Valid
48	1	2	3	3	4	4	15,7	Tidak Valid
49	1	2	3	4	4	4	16,1	Tidak Valid
50	1	2	4	4	4	4	16,5	Tidak Valid
51	1	3	3	3	3	3	15,4	Tidak Valid
52	1	3	3	3	3	4	15,8	Tidak Valid
53	1	3	3	3	4	4	16,2	Tidak Valid
54	1	3	3	4	4	4	16,6	Tidak Valid
55	1	3	4	4	4	4	17	Tidak Valid
56	1	4	4	4	4	4	17,4	Tidak Valid
57	2	2	2	2	2	2	13,2	Tidak Valid
58	2	2	2	2	2	3	13,7	Tidak Valid
59	2	2	2	2	2	4	14,1	Tidak Valid
60	2	2	2	2	3	3	14,2	Tidak Valid
61	2	2	2	2	3	4	14,6	Tidak Valid
62	2	2	2	2	4	4	15	Tidak Valid
63	2	2	2	3	3	3	14,7	Tidak Valid
64	2	2	2	3	3	4	15,1	Tidak Valid
65	2	2	2	3	4	4	15,5	Tidak Valid
66	2	2	2	4	4	4	15,9	Tidak Valid
67	2	2	3	3	3	3	15,2	Tidak Valid
68	2	2	3	3	3	4	15,6	Tidak Valid
69	2	2	3	3	4	4	16	Tidak Valid
70	2	2	3	4	4	4	16,4	Tidak Valid
71	2	2	4	4	4	4	16,8	Tidak Valid
72	2	3	3	3	3	3	15,7	Tidak Valid
73	2	3	3	3	3	4	16,1	Tidak Valid
74	2	3	3	3	4	4	16,5	Tidak Valid
75	2	3	3	4	4	4	16,9	Tidak Valid
76	2	3	4	4	4	4	17,3	Tidak Valid
77	2	4	4	4	4	4	17,7	Tidak Valid
78	3	3	3	3	3	3	16,2	Tidak Valid
79	3	3	3	3	3	4	16,6	Tidak Valid
80	3	3	3	3	4	4	17	Tidak Valid
81	3	3	3	4	4	4	17,4	Tidak Valid
82	3	3	4	4	4	4	17,8	Tidak Valid
83	3	4	4	4	4	4	18,2	Tidak Valid
84	4	4	4	4	4	4	18,6	Tidak Valid

Keterangan:
P1=potongan 1, dan seterusnya.
Tot=Total panjang dari semua potongan.
Ket.: Jika lebih dari 12 meter, maka potongan dianggap tidak valid. Karena melebihi panjang besi utuh.

Untuk Potongan (P1 sampai P6), 1 adalah 1,9 meter, 2 adalah 2,2 meter, 3 adalah 2,7 meter, dan 4 adalah 3,1 meter.

Mencari Pola

Setelah mempunyai daftar semua kombinasi, maka dipilih kombinasi yang dapat dipakai. Ini kita sebut pola. Tidak semua kombinasi menjadi pola. Pemilihan adalah dengan cara eliminasi. (Guichard, 2018) Tiga cara eliminasi yang dipakai adalah sebagai berikut.

1. Panjang total harus sama atau lebih kecil dari panjang besi utuh.
   Karena panjang besi utuh adalah 12 meter, maka total yang lebih dari 12 meter dianggap tidak valid. Potongan akan dieliminasi dari kanan. Misal kombinasi nomor 65 adalah 2-2-2-3-4-4. Eliminasi dari kanan akan menghasilkan 2-2-2-3 dengan total panjang 9,3 meter.
2. Sisa seharusnya kurang dari kebutuhan potongan terpendek.
   Jika setelah dieliminasi dari kanan sisa ternyata lebih panjang dari kebutuhan terpendek, maka sisa tersebut dipotong lagi sesuai kebutuhan. Misal kombinasi nomor 65 setelah dieliminasi dari kanan menjadi 2-2-2-3 dengan total panjang 9,3 meter. Dari besi utuh 12 meter, ini menimbulkan sisa 2.7 meter. Maka sisa ini dapat dipotong lagi dengan kebutuhan 1 yaitu 1.9 meter, atau kebutuhan 2 yaitu 2.2 meter, atau kebutuhan 3 yaitu 2.7 meter. Maka kombinasi yang didapat adalah 1-2-2-2-3 atau 2-2-2-2-3 atau 2-2-2-3-3.
3. Tidak ada duplikasi.
   Jika setelah langkah 1 dan 2 di atas ternyata kombinasi sudah ada, maka duplikasi akan dihilangkan.

Setelah menerapkan tiga cara eliminasi, dari contoh 84 kombinasi di atas ternyata didapat 38 pola yang akan dipakai, seperti daftar di bawah.

Tabel 2 Pola yang dipakai

No	P1	P2	P3	P4	P5	P6	Tot	Sisa
1	1	1	1	1	1	1	11,4	0,6
2	1	1	1	1	1	2	11,7	0,3
3	1	1	1	1	2	2	12,0	0
4	1	1	1	1	3	0	10,3	1,7
5	1	1	1	1	4	0	10,7	1,3
6	1	1	1	2	3	0	10,6	1,4
7	1	1	1	2	4	0	11,0	1
8	1	1	1	3	3	0	11,1	0,9
9	1	1	1	3	4	0	11,5	0,5
10	1	1	1	4	4	0	11,9	0,1
11	1	1	2	2	2	0	10,4	1,6
12	1	1	2	2	3	0	10,9	1,1
13	1	1	2	2	4	0	11,3	0,7
14	1	1	2	3	3	0	11,4	0,6
15	1	1	2	3	4	0	11,8	0,2
16	1	1	3	3	3	0	11,9	0,1
17	1	2	2	2	2	0	10,7	1,3
18	1	2	2	2	3	0	11,2	0,8
19	1	2	2	2	4	0	11,6	0,4
20	1	2	2	3	3	0	11,7	0,3
21	1	2	4	4	0	0	10,3	1,7
22	1	3	3	4	0	0	10,4	1,6
23	1	3	4	4	0	0	10,8	1,2
24	1	4	4	4	0	0	11,2	0,8
25	2	2	2	2	2	0	11,0	1
26	2	2	2	2	3	0	11,5	0,5
27	2	2	2	2	4	0	11,9	0,1
28	2	2	2	3	3	0	12,0	0
29	2	2	3	4	0	0	10,2	1,8
30	2	2	4	4	0	0	10,6	1,4
31	2	3	3	3	0	0	10,3	1,7
32	2	3	3	4	0	0	10,7	1,3
33	2	3	4	4	0	0	11,1	0,9
34	2	4	4	4	0	0	11,5	0,5
35	3	3	3	3	0	0	10,8	1,2
36	3	3	3	4	0	0	11,2	0,8
37	3	3	4	4	0	0	11,6	0,4
38	3	4	4	4	0	0	12,0	0

Menghitung Sisa Pemotongan

Ada dua macam sisa pemotongan.

Tipe 1: Setelah dipotong ternyata masih ada sisa yang terlalu pendek untuk dipotong lagi. Misal di pola nomor 1 di atas, 0,6 meter tidak dapat lagi dipotong karena lebih pendek dari kebutuhan terpendek (1.9 meter).

Tipe 2: Ketika satu kebutuhan sudah dipenuhi semua, maka pola yang mempunyai kebutuhan itu dianggap sisa. Misal kebutuhan 1,9 meter sudah dipenuhi sebanyak 200 buah. Maka ketika satu pola dipakai yang mengandung panjang kebutuhan 1,9 meter, ini dianggap sisa karena sudah tidak lagi dibutuhkan.

Sisa yang dihitung adalah tipe 1 ditambah tipe 2. Untuk optimasi, dicari sisa yang minimum, yaitu jumlah sisa yang paling kecil.

Langkah Pengulangan

Untuk tiap langkah pengulangan, diperiksa sisa terkecil dari semua pola. Kemudian pola dengan sisa minimum akan dipakai. Jumlah kebutuhan yang sudah dipotong bertambah sesuai pola yang dipakai.

Contoh tabel pola di atas, pola nomor 3 dengan sisa 0 adalah pola dengan sisa terkecil. Pola ini yang dipakai di pengulangan pertama. Kemudian larik pemenuhan kebutuhan akan diubah sesuai pola nomor 3.

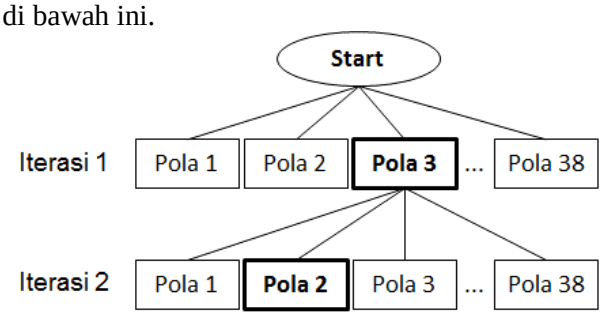
Tabel 3 Larik Pemenuhan Kebutuhan

Item	Panjang	Banyaknya	Sudah dipotong
1	1.9 m	200	4
2	2.2 m	150	2
3	2.7 m	100	0
4	3.1 m	200	0

Larik pemenuhan kebutuhan ini akan dilihat setiap kali melakukan pengulangan untuk melihat kebutuhan mana yang sudah lengkap terpenuhi.

Langkah selanjutnya adalah mengambil besi berikutnya, dan dicari pola yang memberikan sisa terkecil dari semua pola yang ada. Pengulangan akan dihentikan jika semua kebutuhan sudah dipenuhi.

Langkah iterasi dapat digambarkan seperti di bawah ini.



Gambar 1 Langkah pengulangan

Pada contoh di atas, pada pengulangan pertama Pola 3 adalah pola dengan sisa terkecil. Ini diambil sebagai pola pemotongan besi 1.

Pada langkah kedua, Pola 2 ternyata memberi sisa terkecil. Maka pola 2 dipakai untuk pemotongan besi kedua. Demikian seterusnya.

Pada Pencarian Rakus, hanya pola yang sudah dipilih di langkah pertama akan diikuti.

Kemungkinan pola ini tidak optimal tidak didiskusikan di Pencarian Rakus.

Solusi Optimal

Dalam tiap langkah pengulangan, kemungkinan ada lebih dari satu pola yang optimal. Dalam 38 pola di atas, kita dapat melihat ada tiga pola yang menghasilkan sisa 0 meter, yaitu Pola 3, Pola 28 dan Pola 38. Ketiganya adalah pola optimal. Tapi dalam Pencarian Rakus kita hanya akan memakai satu pola saja. Pola yang lain diabaikan meski pun sama-sama optimal.

Pola yang diabaikan ini kemungkinan bisa mendapatkan solusi global yang lebih optimal, karena adanya sisa tipe 2 yang berbeda untuk langkah iterasi selanjutnya.

Demikian juga pola yang kurang optimal di langkah pertama, kemungkinan bisa mendapatkan solusi global yang lebih optimal di langkah-langkah selanjutnya.

3. Implementasi Pencarian Rakus

Pemrograman dilakukan dengan bahasa C++ di lingkungan desktop Microsoft Windows. (Kirch-prinz & Prinz, 2002) Compiler yang dipakai adalah Dev-C++.

Hasil yang didapat dengan beberapa kebutuhan adalah sebagai berikut.

Kebutuhan 1

```
Item  Panjang     Banyaknya
1     1,9 m       150
2     2,2 m       120
3     2,7 m       200
4     3,1 m       100

Besi terpakai : 120 batang
Sisa terbuang : 41 meter
Persen        : 2.8%
```

Kebutuhan 2

```
Item  Panjang     Banyaknya
1     1,9 m       1300
2     2,2 m       1250
3     2,7 m       1320
4     3,1 m       1270

Besi terpakai : 1073 batang
Sisa terbuang : 155 meter
Persen        : 1.2%
```

Kebutuhan 3

```
Item  Panjang     Banyaknya
1     1,5 m       150
2     1,8 m       75
3     2,3 m       100
4     2,8 m       210
5     3,1 m       110

Besi terpakai : 121 batang
Sisa terbuang : 54 meter
Persen        : 3.4%
```

Di atas adalah hasil dari beberapa contoh kebutuhan besi yang dipakai dalam konstruksi. Dalam percobaan yang dilakukan dengan 10 data kebutuhan, sisa yang dihasilkan bervariasi antara 0,5% sampai 6,9%.

4. Kesimpulan

Algoritma Pencarian Rakus tidak menjamin mendapatkan solusi global optimal. Algoritma ini akan mendapatkan solusi lokal optimal. Ini bisa jadi adalah solusi global optimal, bisa juga bukan. Untuk memastikan mendapatkan solusi global optimal, seluruh pola harus terus dilacak sampai akhir kemudian dicari mana yang paling optimal. Ini adalah metode *Brute Force*. Cara ini akan memakan banyak resource karena banyaknya kemungkinan akan meningkat secara eksponensial seiring dengan banyaknya variabel.

Algoritma Pencarian Rakus tetap banyak dipakai untuk mencari solusi yang mendekati optimal karena algoritma ini efektif dan hasilnya cukup baik.

Referensi

Abuhassan, I. A. O., & Nasereddin, H. H. O. (2011). *Cutting Stock Problem : Solution Behaviors*. *6*(4), 429–433. Retrieved from https://www.researchgate.net/publication/281120697_CUTTING_STOCK_PROBLEM_SOLUTION_BEHAVIORS

Brualdi, R. A. (2010). Introductory Combinatorics. In *Pearson Education, Inc*. https://doi.org/10.2307/2320280

Feo, T. A., & Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, *6*(2), 109–134. https://doi.org/10.1007/BF01096763

Guichard, D. (2018). *An Introduction to Combinatorics and Graph*. Creative Commons.

Kirch-prinz, U., & Prinz, P. (2002). *A Complete Guide to Programming in C++* (1st ed.). Sudbury, MA: Jones And Bartlett Publishers.

Malik, A., Sharma, A., & Saroha, V. (2013). Greedy Algorithm. *International Journal of Scientific and Research Publications*, *3*(8), 1–4. Retrieved from http://xlinux.nist.gov/dads//HTML/greedyalgo.html

Subhadra, A. (2016). Greedy Algorithms:

Analysis, Design & Applications. *International Journal of Informative & Futuristic Research*, *3*(5), 1749–1764.

Wilt, C., Thayer, J., & Ruml, W. (2010). A comparison of greedy search algorithms. *Proceedings of the 3rd Annual Symposium on Combinatorial Search, SoCS 2010*, (July 2010), 129–136.