ANALISIS ALGORITMA HYBRID ANT COLONY OPTIMIZATION (ACO) DAN LOCAL SEARCH UNTUK OPTIMASI PEMOTONGAN BAHAN BAKU

Warih Maharani
Fakultas Teknik Informatika, Institut Teknologi Telkom
Jl. Telekomunikasi No.1 Bandung 40286
Telp. (022) 7564108 ext. 2104, Faks. (022) 7565931
E-mail: wrh@ittelkom.ac.id

ABSTRAK

Pemilihan susunan pemotongan dalam proses pemotongan suatu objek/bahan baku sangat berpengaruh terhadap sisa hasil pemotongan. Susunan yang tidak optimal akan menyebabkan bahan yang terbuang tidak minimal. Masalah ini dapat dipecahkan menggunakan FFD (First Fit Decreasing) salah satu algoritma sederhana dan paling efektif. Algoritma ini tidak menjamin mendapatkan solusi yang optimal namun dengan metode ini dapat dihasilkan solusi yang dapat diterima dengan cepat. Penelitian ini menganalisis penggunaan algoritma hybrid ant colony untuk mengoptimasi susunan potongan objek, kemudian akan dilakukan perbandingan hasil solusi yang dihasilkan dengan hasil solusi yang menggunakan algoritma FFD. Penelitian ini menggunakan algoritma ant colony optimization digabungkan dengan algoritma local search. Pada awalnya algoritma ant colony optimization akan menghasikan suatu solusi awal setelah itu digunakan local search untuk mengoptimasi solusi awal tersebut. Hanya semut terbaik yang dapat mengupdate pheromone trail yang digunakan untuk iterasi selanjutnya. Hasil penelitian menunjukkan bahwa algoritma FFD jauh lebih cepat dibandingkan algoritma hybrid ant colony optimization, walaupun stock yang digunakan lebih besar. Parameter yang paling berpengaruh dalam algoritma hybrid ant colony adalah parameter nilai stock (k), dimana nilai k=2 dapat menghasilkan solusi yang optimal. Selain itu parameter jumlah semut dan iterasi juga mempengaruhi terhadap solusi yang dihasilkan.

Kata Kunci: ffd (first fit decreasing), ant colony, local search

1. PENDAHULUAN

1.1 Latar Belakang

Dalam dunia nyata masalah pemotongan dapat kita lihat dalam banyak kasus, misalnya pada industri baja dimana terdapat beberapa potongan baja besar dengan ukuran tertentu yang akan dipotong menjadi beberapa pola potongan tertentu yang diinginkan, kasus ini dapat juga kita temukan pada industri kertas. Dalam melakukan pemotongan kemungkinan terdapat sisa hasil pemotongan, sisa hasil pemotongan ini akan dibuang karena sudah tidak dapat digunakan lagi. Penentuan pola susunan pemotongan mempunyai pengaruh yang besar karena sisa hasil pemotongan yang banyak akan mengakibatkan semakin banyak bahan baku yang dibutuhkan. Pada kasus dimana pemakaian bahan baku yang terbatas dan mempunyai aspek bisnis masalah pemotongan akan menjadi penting.

Andaikan pada industri baja terdapat proses pemotongan kaca tanpa mempertimbangkan pola pemotongan sehingga bahan baku yang dibutuhkan bertambah banyak, hal ini akan menambah biaya yang harus dikeluarkan dalam proses produksi. Dengan besarnya biaya produksi akan berpengaruh terhadap harga barang tiap unitnya.

1.2 Permasalahan

Permasalahan yang akan diselesaikan pada penelitian ini adalah jika terdapat n jumlah objek kecil persegi dengan lebar w_i dan panjang l_i dengan bahan baku yang jumlahnya terbatas dengan ukuran lebar W dan panjang L, bagaimana menentukan pola pemotongan sehingga bahan baku yang dibutuhkan seminimal mungkin menggunakan pemotongan *non guillotine*.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah diatas, maka tujuan dari penelitian ini adalah :

1. Menerapkan algoritma *hybrid ant colony* dan *local search* untuk menyelesaikan masalah pengoptimasian pemotongan objek/bahan baku persegi panjang berbentuk dua dimensi.
2. Menguji dan menganalisis sistem berdasarkan parameter jumlah *stock* (k), alpha, jumlah semut dan globa best.
3. Membandingkan hasil dari perangkat lunak yang telah dibuat dengan hasil yang didapatkan dengan algoritma FFD (*First Fit Decreasing*).

Batasan masalah yang digunakan dalam penelitian ini adalah:

1. Objek/bahan baku berupa benda berbentuk persegi panjang dengan model dua dimensi menggunakan pemotongan *non guillotine*.
2. Total ukuran objek sebelum dan setelah pemotongan tetap.
3. Solusi dikatakan lebih baik jika bahan yang terbuang lebih sedikit.

4. Tidak di bahas algoritma lain dalam pengoptimasian pemotongan.

2. GAMBARAN SISTEM

2.1 Pemotongan Objek / Bahan Baku

Masalah pemotongan muncul dalam industri yang memproduksi benda secara massal dimana *stock* yang besar (*stock material*) dipotong menjadi bagian yang lebih kecil (*order list*), Biasanya *stock* yang besar tersebut berupa pulp dan kertas, baja, kaca, kayu, plastic dan textile (Dyckhoff, 1990).

Pemotongan terhadap suatu objek dapat diklasifikasikan menjadi (Dyckhoff, 1990):

1. Dimensionalitas
   N = Jumlah Dimensi.
2. Jenis penugasan
   B = Semua objek besar dan beberapa objek kecil yang dipakai.
   V = Beberapa objek besar dan semua objek kecil yang dipakai.
3. Macam objek besar
   O = Satu objek besar.
   I = Banyak objek besar yang sama.
   D = Objek besar yang berbeda.
4. Macam objek kecil
   F = Sedikit objek kecil dengan ukuran yang berbeda.
   M = Banyak objek kecil dengan banyak ukuran.
   R = Banyak objek kecil dengan relatif sedikit ukuran.
   C = Objek kecil yang sama.

Pada penelitian ini yang dibahas adalah pemotongan dengan tipe 2/V/I/R.

2.2 Order List

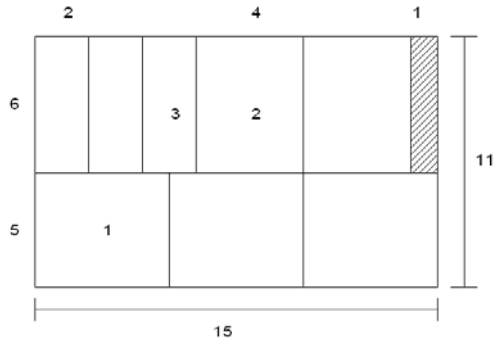
Order list merupakan sekumpulan objek kecil (item) yang harus dipenuhi yang berasal dari hasil pemotongan objek besar (*stock*).

Misalkan terdapat *order list* seperti tabel 1. Sedangkan, lay out pemotongannya dapat digambarkan seperti pada Gambar 1.

Pada lay out diatas data bahan yang digunakan P=15 satuan, L=11 satuan, sedangkan daerah yang diarsir merupakan sisa dengan luas 1 x 6 = 6 satuan. Luas bahan baku 15 x 11 = 165 satuan, maka persentase area yang terbuang adalah (6/165) x 100% = 3,636 %.

Tabel 1. Contoh *order list* dengan 3 item

NO	*UKURAN*	*JUMLAH*
1	5 x 5	3
2	2 x 6	3
3	4 x 6	2



Gambar 1. Lay out pemotongan 2 dimensi

2.3 Algoritma FFD (*First Fit Decreasing*)

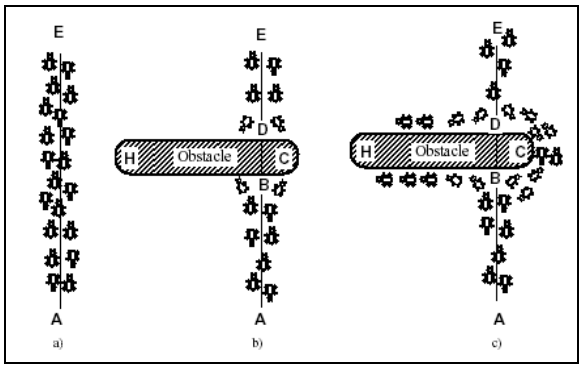
Algoritma ini melakukan pemotongan terhadap *stock* sesuai dengan item yang ada dalam *order list* selama *stock* tersebut masih dapat dipotong. Langkah-langkah menggunakan algoritma ini sebagai berikut :

a. Urutkan *order list* dari yang terbesar ke yang terkecil.
b. Potong *stock* sesuai dengan urutan order list yang telah diurutkan.
c. Jika tidak cukup potong *stock* baru.
d. Ulangi sampai semua order list terpenuhi

2.4 Algoritma *Ant Colony Optimization*

Ant Colony Optimization(ACO) merupakan teknik pencarian *multi agent* untuk menyelesaikan permasalahan kombinatorial dan persoalan yang lain yang di inspirasi tingkah laku semut dalam suatu koloni (Marco, 2004). Algoritma *ant colony* pertama kali diperkenalkan oleh Marco Dorigo pada tahun 1992 sebagai thesis PhD-nya yang kemudian di publikasikan dengan nama *Ant System (AS)* dan di aplikasikan pada Traveling Salesman Problem pada tahun 1996. Algoritma ini di inspirasi oleh tingkah laku koloni semut, bagaimana hewan yang hampir buta dengan kemampuan individu yang sederhana dapat menemukan jalan terpendek (sarang semut dengan sumber makanan) jika bersama dalam suatu koloni.

Pada awalnya semut akan mengelilingi daerah sekitar sarangnya dengan acak, begitu mengetahui ada makanan semut itu akan mengenali kualitas dan kuantitas makanan tersebut dan membawa beberapa bagian ke sarangnya. dalam perjalanannya mereka akan meninggalkan jejak zat kimia *pheromone* di dasarnya. *pheromone* ini akan membimbing semut lain untuk menemukan sumber makanan, banyaknya *pheromone* yang ditinggalkan oleh semut tergantung jumlah makanan yang ditemukan. Semakin banyak semut yang melewati semakin kuat jejak *pheromone* yang ditinggalkan, untuk lebih jelasnya perhatikan gambar 2 (Marco, 1991).



Gambar 2. Algoritma *Ant Colony*

Gambar di atas menunjukkan bagaimana semut dapat menemukan jalan terpendek.

a. Semut melewati jalan antara titik E dan A.
b. Terdapat benda yang menghalangi jalan, semut dapat memilih dua jalan yang berbeda dengan probabilitas yang sama.
c. *Pheromone* pada jalan terpendek lebih banyak.

Ant System yang ada dalam hal ini adalah ACO yang diterapkan pada TSP (*Traveling Salesman Problem*). Permasalahan pada TSP adalah bagaimana mengunjungi *n* kota dengan seminimal mungkin dimana setiap kota hanya boleh dikunjungi sekali. Langkah-langkahnya adalah :

a. setiap semut membangun solusi dari kota awal yang dipilih secara acak.
b. kota berikutnya dipilih berdasarkan probabilitas dengan formula tertentu.
c. *pheromone trail* τ (i,j) mendefinisikan kecenderungan mengunjungi kota *j* setelah mengunjungi kota *i*.
d. η(i,j) merupakan informasi *heuristic* yang digunakan, yaitu perbandingan terbalik jarak kota *i* dan kota *j*.
e. *pheromone trail* dirubah setelah setiap semut telah mengunjungi seluruh kota.
f. *Pheromone trail* akan menguap tergantung parameter ρ (parameter evaporation).

2.5 Algoritma *Local Search*

Local search merupakan metode pencarian solusi berdasarkan *neighborhood* dari solusi awal (Marco, 2001). Pada versi lain metode ini dikenal dengan nama *iterative improvement*. Algoritma ini mencari solusi disekitar solusi awal untuk memperbaiki solusi. Jika ditemukan solusi yang lebih baik maka solusi itu akan menggantikan solusi awal dan *local search* akan diteruskan, langkah ini akan terus dilakukan sampai tidak ada kemungkinan untuk memperbaiki solusi lagi. Pada saat itu algoritma akan berhenti dan kondisi saat itu akan mencapai *local optimum*.

Untuk memenuhi spesifikasi algoritma *Local Search* diperlukan pemeriksaan skema "*neighborhood*". Bagaimana skema pencarian "*neighborhood*" dan "*neighbor*" yang mana yang akan menggantikan solusi awal, aturan ini disebut *pivoting rule* (Manniezzo, 1999). Macam-macam *pivoting rule* antara lain :

a. *the best-improvement rule*
b. *the first-improvement rule*.

Jika *k-opt* "*neighborhood*" maka solusi tetangga yang berbeda paling banyak *k* busur.

2.6 Algoritma *Hybrid Ant Colony*

Pada algoritma *Hybrid Ant Colony* yang akan digunakan dalam menyelesaikan masalah ini secara garis besar hampir sama dengan algoritma *Ant Colony Optimization* namun solusi yang dihasilkan oleh setiap semut dilakukan optimasi *local search* sebelum mengupdate *pheromone*.

3. PEMBANGUNAN SISTEM

3.1 Pheromone Trail

Pheromone trail akan didefinisikan dengan τ (*i,j*) yaitu kecenderungan mempunyai item dengan ukuran *i* dan ukuran *j* dalam *stock* yang sama.

3.2 Pembangunan Solusi

Setiap semut akan mulai dengan sekumpulan item dan *stock* yang tersedia, kemudian *stock* dipotong berdasarkan item yang ada sampai *stock* tersebut tidak dapat dipotong lagi dengan item yang ada, maka *stock* yang baru akan dipakai. Semut *k* akan memilih item *i* sebagai item berikutnya untuk *stock r* di sebagian solusi x adalah (Marco, 2004):

$$Pk(x,i,r) = \frac{[\tau(i)]^{\alpha}.[\eta(i)]^{\beta}{}_{r}}{\sum_{g \in Ik(x,r)}[\tau_{r}(g)]^{\alpha}.[\eta(g)]^{\beta}} \quad jika\ i \in I_{k}(x,r) \quad (1)$$

Dimana $I_k(x,r)$ adalah himpunan item yang harus dipotong dan masih cukup untuk digunakan pada *stock* saat itu serta merupakan bagian item yang tersisa setelah bagian solusi x dibentuk. β merupakan parameter yang menunjukkan informasi heuristic. α parameter yang menunjukkan informasi pheromone. η(i) adalah ukuran dari item i, sedangkan nilai *pheromone* τr(i) untuk item i pada *stock* r diberikan pada persamaan dibawah:

$$\tau = \begin{cases} \frac{\sum_{i \in r}\tau(i,j)}{|r|} \ jika\ r \neq 1 \end{cases} \quad (2)$$

Persamaan di atas menyatakan penjumlahan semua *pheromone* antara item i dan item j yang dipotong dari *stock r* dibagi jumlah item yang telah dipotong dari *stock r*, jika *r* kosong maka τ r(i) = 1.

3.3 Mengupdate *Pheromone Trail*

Untuk mengupdate *pheromone trail* akan digunakan pendekatan yang digunakan dalam MMAS (MAX-MIN Ant System) dimana terdapat aturan hanya semut terbaik yang diperbolehkan meninggalkan *pheromone* di setiap iterasi, dengan aturan ini maka proses pencarian lebih agresif

[STU00], dimana τ(i,j) akan selalu ditingkatkan jika setiap i dan j dalam satu *stock*. Rumus yang digunakan untuk mengupdate *pheromone trail* adalah sebagai berikut:

$$\tau(i,j) = \rho.\tau(i,j) + m.\int(S^{best}) \quad (3)$$

Persamaan diatas terdiri dari dua bagian dimana bagian kiri terdapat parameter ρ adalah faktor Decay yang menyebabkan *pheromone* menguap. Sedangkan pada bagian kanan akan meningkatkan *pheromone* pada semua bagian yang dikunjungi semut. *m* menunjukkan berapa kali *i* dan *j* bersama dalam solusi terbaik $S^{best.}$. Dalam MMAS terdapat beberapa *feature* untuk menyeimbangkan antara eksplorasi dan eksploitasi yang pertama adalah pemilihan antara penggunaan *iteration best* dan *global best*. Dengan menggunakan *global best* akan memperbesar eksploitasi yang kuat sehingga akan digunakan alternatif lain yaitu menggunakan *iteration best*. Parameter γ akan mendefinisikan angka mengupdate sebelum kita menggunakan *global best* lagi. Yang kedua dengan cara mengeset nilai awal *pheromone* ke $\tau_{\max}$. Sedangkan *feature* lainnya yaitu dengan cara mendefinisikan batas atas dan batas bawah untuk nilai *pheromone* (oleh sebab itu disebut MAX-MIN)

Rumus untuk batas atas dan batas bawah masing-masing diberikan sebagai berikut:

$$\tau_{\min} = \frac{\frac{1}{1-\rho}.(1-\sqrt[n]{\rho best}}{(avg-1).\sqrt[n]{\rho best}} \quad (4)$$

$$\tau_{\max} = \frac{1}{1-\rho} \quad (5)$$

dimana:
n = jumlah total item
avg = jumlah rata-rata item yang dipilih dari setiap titik pemilihan ketika pembangunan solusi yang didefinisikan $\frac{n}{2}$.

3.4 Fungsi Fitness

Agar dapat mengetahui solusi yang bagus maka diperlukan suatu Fungsi Fitness yang dapat dirumus sebagai berikut:

$$\int(s) = \frac{\sum_{i=1}^{N}(Fi/C)^2}{N} \quad (6)$$

dimana :
N = total jumlah *stock*
Fi = luas yang digunakan (dipakai) pada *stock* i
C = luas *stock* i

3.5 Penambahan *Local Search*

Setiap satu semut akan membuat satu solusi dimana solusi itu nanti akan melalui fase optimasi local dengan cara *n stock* yang terpotong dibatalkan sehingga menjadi bebas, kemudian untuk setiap *stock* yang ada dicoba apakah item yang bebas dapat menggantikan item yang ada pada *stock* saat itu. Jika tidak ada perbaikan yang dapat dilakukan lagi maka akan digunakan pemotongan pada *stock* baru.

4. ANALISIS HASIL

Data uji yang akan dipakai berjumlah tiga kasus, yaitu :

a. Jumlah item = 600, ukuran stok = 120 x 1, jenis item = 36
b. Jumlah item = 60, ukuran stok = 25 x 1, jenis item = 8
c. Jumlah item = 200, ukuran stok = 86 x 1, jenis item = 18

Untuk mengetahui pengaruh tiap parameter akan dilakukan perubahan terhadap satu parameter sedangkan yang lainnya tetap, dan setiap pengujian dilakukan sebanyak 20 kali yang kemudian akan dilakukan perhitungan rata-rata penggunaan *stock* yang terpakai.

4.1 Analisis Terhadap nilai *stock* (k)

Pada penelitian ini digunakan algoritma 3-opt *local search* dimana maksimal 3 item saja yang boleh ditukar, sedangkan untuk jumlah *stock* yang boleh dirombak dapat kita inputkan. Misalkan k = 4 maka setiap satu solusi terbentuk maka 4 *stock* yang paling banyak terbuang akan dibuka kemudian akan dilakukan proses pengoptimasian.

Hasilnya seperti ditunjukkan pada Tabel 2.

Tabel 2. Pengaruh parameter k terhadap *stock*

k	*Rata-rata stock terpakai*		
	Kasus ke-1	*Kasus ke-2*	*Kasus ke-3*
0	228.75	20	86.1
1	218.45	20.1	86.3
2	217.7	19.05	79.95
3	217.95	19	79.95
4	217.95	19	80
5	217.75	19	79.85
7	217.75	19	79.85
10	217.55	20	79.85

Untuk nilai k terlalu kecil yaitu antara k=0 atau k=1 solusi yang dihasilkan mengalami penurunan kualitas solusi sedangkan mulai k=2 solusi yang dihasilkan mulai mengalami peningkatan.

4.2 Analisis Terhadap Nilai Alpha (α)

Alpha merupakan besaran yang menunjukan relatif pengaruh *trail* terhadap pembangunan solusi. Inputan harus $\alpha \geq 0$

Hasil terhadap perubahan alpha ditunjukkan pada Tabel 3. Berdasarkan tabel 3, perubahan nilai alpha tidak terlalu berpengaruh, seperti halnya pada kasus kedua dan ketiga untuk nilai alpha=0 solusi yang dihasilkan buruk namun pada kasus pertama nilai alpha=0 justru memberikan solusi yang cukup baik diantara yang lainnya.

Tabel 3. Pengaruh parameter alpha terhadap *stock*

Alpha	*Rata-rata stock terpakai*		
	Kasus ke-1	*Kasus ke-2*	*Kasus ke-3*
0	218	19.05	80.2
1	218	19	79.95
2	218	19	80.25
3	218.15	19	80.1
5	218	19	80.15
15	218	19	80.15
30	218	19	80.15

4.3 Analisis Terhadap Jumlah semut

Parameter ini menandakan jumlah semut yang akan digunakan selama algoritma dijalankan. Inputan harus ≥ 1.

Hasilnya ditunjukkan pada tabel 4.

Tabel 4. Pengaruh parameter jumlah semut terhadap *stock*

Jml semut	*Rata-rata stock terpakai*		
	Kasus ke-1	*Kasus ke-2*	*Kasus ke-3*
1	218.25	19	80.25
7	218	19	79.95
20	217.8	19	80.05
30	217.75	19	79.95
60	217.9	19	80.05

Dari tabel diatas dapat kita lihat bahwa jika jumlah semut yang digunakan terlalu sedikit, maka kualitas solusi yang dihasilkan jelek, semakin banyak jumlah semut hasilnya semakin baik namun pada titik tertentu solusi yang dihasilkan akan tetap tidak terjadi banyak perubahan.

4.4 Analisis Penggunaan Global Best (γ)

Salah satu aspek penting dalam *Ant Colony* adalah penyeimbangan antara eksploitasi dan eksplorasi. Dalam MMAS hanya satu semut yang dapat mengupdate yaitu semut yang terbaik untuk semut *global best* maupun semut *best* iterasi. Semut *global best* adalah semut terbaik yang ditemukan selama algoritma dijalankan sedangkan semut *best* iterasi merupakan semut terbaik pada suatu iterasi yang bisa juga menjadi *global best*. Mengupdate *pheromoene trail* menggunakan *global best* akan memberi penekanan terhadap eksploitasi pada solusi terbaik sehingga eksplorasi menjadi terbatas, hal ini memungkinkan algoritma terjebak pada solusi yang jelek. Sedangkan jika menggunakan *best iteration* yang kemungkinan berbeda tiap iterasi akan memberikan eksplorasi lebih banyak. Oleh karena itu akan digunakan perubahan secara bergantian dimana *best* iterasi akan digunakan sebagai *update default* namun setelah gamma iterasi akan digunakan *global best* iterasi. Inputan harus $\gamma \geq 0$

Hasilnya ditunjukkan pada tabel 5. Dari tabel 5, untuk nilai gamma yang kecil, gamma = 0, *stock* yang dibutuhkan lebih banyak dari pada kalau digunakan gamma >1 karena penggunaan *iteration best* tanpa *global best* memungkinkan algoritma terjebak ke local optimal.

Tabel 5. Pengaruh parameter gamma terhadap *stock*

Gamma	Rata-rata *stock* terpakai		
	Kasus ke-1	Kasus ke-2	Kasus ke-3
0	229.05	19	86.35
1	218.1	19	86.35
3	218.1	19	80.15
5	217.7	19	79.95
10	217.95	19	80.25
15	217.95	19	80.15

4.5 Analisis Perbandingan Terhadap FFD

Secara keseluruhan jumlah *stock* yang dibutuhkan oleh algoritma FFD lebih banyak dibandingkan HACO, hal ini terlihat jelas pada kasus ketiga dimana algoritma FFD membutuhkan 85 *stock*. Algoritma ini akan selalu menghasilkan suatu solusi yang tetap karena telah menggunakan aturan tertentu, sedangkan algoritma HACO untuk hasil terbaiknya hanya menggunakan 79 *stock*. Namun kalau kita melihat waktu yang dibutuhkan algoritma HACO untuk eksekusi program dibandingkan FFD terlihat jelas bahwa algoritma FFD sangat cepat dalam membentuk solusi. Hal ini karena FFD tidak memerlukan proses yang rumit karena tinggal memotong item yang terbesar sampai yang terkecil sedangkan pada HACO, proses untuk membentuk satu solusi harus melewati proses yang panjang.

5. PENUTUP

5.1 Kesimpulan

Berdasarkan analisis yang dilakukan, maka dapat ditarik kesimpulan sebagai berikut :

1. Dalam membentuk solusi, algoritma FFD jauh lebih cepat dibandingkan algoritma HACO, namun *stock* yang digunakan lebih besar.
2. Dari semua parameter yang ada, yang paling berpengaruh adalah parameter k, nilai k=2 dapat memberikan perbaikan cukup baik terhadap solusi yang dihasilkan. Pemakaian gamma>0 yang berarti mengupdate *pheromone trail* antara *global best* dengan *iteration best* lebih baik jika dibandingkan hanya menggunakan *global best*. Selain itu jumlah semut dan iterasi yang terlalu sedikit juga menghasilkan solusi yang buruk.

5.2 Saran

Saran untuk pengembangan selanjutnya adalah :

1. Sebaiknya percobaan dilakukan dengan parameter yang lebih bervariasi.
2. Untuk pengembangan program lebih lanjut, *master stock* dan *orderlist* dapat berbentuk lainnya selain persegipanjang, atau bisa juga menggunakan jenis pemotongan lainnya seperti *master stock* ukurannya berbeda-beda

PUSTAKA

Dorigo, Marco and Stuzle, Thomas. (2004). *Ant Colony Optimization*. Bradford Book.

Dorigo, Marco and Stuzle, Thomas. (2001). *The ant colony optimization metaheuristics: Algorihtms, applications and advances*. Glover and G.Kochenberger

Dorigo, Marco., Colorni, Alberto and Maniezzo, Vittorio. (1996). The ant system: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man and Cybernetics*–Part B 26, pp 29–41.

Dorigo, Marco., Colorni, Alberto., Maniezzo, Vittorio. (1991). Distributed optimization by Ant Colonies, *Proceedings of the first European Conference on Artificial Life*, Paris, ed. by FJ. Varela and P. Bourgine MIT/Press/Bradford Books, Cambridge, Massachusetts.

Ducatelle, Frederick and Levine, John. (2002). *Ant colony optimization for bin packing and cutting stock problems*.

Dyckhoff, H. (1990). A Typology of Cutting and Packing Problems, *European Journal of Operational Research*, Vol 44 pp. 145-159.

Karelahti, janne. (2002). *Solving the Cutting Stock Problem in the steel industry*. Accenture and AvestaPolarit Corporations.

Socha, Krzysztof. (2003). *The Influence of Run-Time Limits on Choosing Ant System Parameters*, Universite Libre de Bruxelles, Bruxelles, Belgium.

Stuzle, Thomas and Hoos, Holger. Max-Min ant system. (2000). *Future Generation Computer System*, 16(8):889.

Manniezzo, Z., and Carbonaro. (1999). *Ant Colony Optimization: an overview*. Knowledge and Data Engineering.