HIBRIDISASI *GENETIC-TABU SEARCH ALGORITHM* UNTUK PENJADWALAN *JOB* TERHADAP BEBERAPA RESOURCE DI DALAM KOMPUTASI GRID

Irfan Darmawan
Teknik Elektro dan Informatika, Universitas Siliwangi

ABSTRAK
Permasalahan penjadwalan job terhadap beberapa mesin (scheduling jobs on multiple machines / SJMM) merupakan salah satu permasalahan penjadwalan klasik yang dapat ditemui pada proses komputasi terlebih jika komputasi dilakukan secara terdistribusi. Beberapa metode penyelesaian permasalahan tersebut telah dikembangkan baik dengan pendekatan eksak maupun heuristik/metaheuristik. Tabu Search sebagai salah satu metode metaheuristik yang relatif baru dapat menjadi alternatif metode untuk mendapatkan pendekatan penyelesaian permasalahan tersebut. Metode ini sudah diaplikasikan pada permasalahan optimasi kombinatorial, optimasi multi ekstermal, serta rare event simulation, dengan hasil penyelesaian yang cukup optimal dengan waktu yang relative singkat. Penelitian ini mengimplementasikan metode Tabu Search yang digabungkan dengan algoritma genetika (Incorporation Genetic-Tabu Search Algorithm / IGTS) dalam permasalahan SJMM pada komputasi grid, serta membandingkan kelebihan dan kekurangan antara metode IGTS tersebut dengan metode lain pada permasalahan yang sama. Hasil yang diharapkan dari penelitian ini adalah pengembangan algoritma IGTS pada permasalahan SJMM, untuk mendapatkan hasil makespan yang lebih baik.

Kata Kunci : Tabu search, Genetic algorithms, computational grid, makespan, job scheduling

ABSTRACT
Job scheduling problem on multiple machines (scheduling jobs on multiple machines / SJMM) is one of the classic scheduling problem can be found in the computing process especially if done in distributed computing. Several methods of solving these problems has been developed both with exact and heuristic approaches / metaheuristik. Tabu Search as one metaheuristik a relatively new method can be an alternative method to get the problem solving approach. This method has been applied to combinatorial optimization problems, optimization of multi ekstermal, and rare event simulation, with results that are optimal solution with a relatively short time. This research implements Tabu Search method combined with genetic algorithm (Incorporation Genetic-Tabu Search Algorithm / IGTS) in SJMM problems on grid computing, and compare the advantages and disadvantages between these IGTS method with other methods on similar problems. The expected outcome of this research is the development of algorithms IGTS on SJMM problems, to get a better makespan.

Keywords: Tabu search, Genetic algorithms, computational grid, makespan, job scheduling

1. PENDAHULUAN

Permasalahan SJMM merupakan permasalahan yang tergolong *non-polynomial hard* (*NP-Hard*), terutama pada permasalahan banyak mesin (*m-machines)* (Pan, 2009). Pada permasalahan *job shop* secara umum, urutan pengerjaan operasi pada masing-masing *job* tidak sama, sehingga ketidak-tepatan penjadwalan dapat mengakibatkan bertambahnya waktu penyelesaian keseluruhan *job* (*makespan*). Beberapa penelitian untuk mendapatkan penyelesaian permasalahan SJMM tersebut telah dilakukan, misalnya dengan menggunakan *Genetic Algorithm-Simulated Annealing* (Schuster, 2003) dan *Tabu Search* Hibrida (Bożejko, 2009). Beberapa dari metode tersebut mendapatkan hasil *makespan* yang kurang optimal, namun ada pula yang mendapatkan hasil *makespan* yang optimal namun dengan waktu komputasi yang relatif lama.

Metode *tabu search* sebagai salah satu metode metaheuristik telah digunakan dalam beberapa permasalahan optimasi kombinatorial, optimasi kontinu, optimasi *noisy*, dan *rare event simulation* (Rubinstein, 2004). Pada beberapa permasalahan, metode ini dapat menghasilkan penyelesaian yang cukup optimal dengan perbandingan waktu penghitungan yang relatif lebih singkat, misalnya pada permasalahan *Support Vector Machine* (Widyarini, 2009). Penelitian ini akan melakukan suatu pengembangan algoritma dengan pendekatan *tabu search* yang digabungkan dengan algoritma genetika (untuk selanjutnya dinamakan *Incorporation tabu search-genetic algorithm* atau IGTS) untuk menyelesaikan permasalahan SJMM. Penggunaan pendekatan *tabu search-genetic algorithm* untuk mengembangkan algoritma dalam menyelesaikan permasalahan SJMM diharapkan

menjadi alternatif untuk menghasilkan jadwal dengan makespan yang optimal.

2. MODEL PERMASALAHAN

Merujuk pada definisi Graham dkk. pada penelitian Schuster (2003), permasalahan SJMM secara umum adalah sebagai berikut:

Terdapat seperangkat mesin M = {1, 2, …, m} yang akan digunakan untuk mengerjakan seperangkat job J = {1, 2, …, n}. Untuk setiap job ke-i ∈ J, diberikan urutan-urutan pengerjaan sebanyak k operasi O = {O(i,1), O(i,2), …, O(i,j)} yang merupakan detail dari pengerjaan semua job ke-i. Masing-masing operasi tersebut memiliki (m(i,j), w(i,j)) ∈ M× ℕ, yang menunjukkan bahwa operasi o(i,j) dikerjakan pada mesin m(i,j) dengan waktu operasi w(i,j).

Sedangkan merujuk pada model yang diusulkan Brizuela pada penelitian Pan (2009), permasalahan SJMM dengan minimasi makespan dapat dimodelkan dalam bentuk pemrograman integer sebagai berikut:

Model Minimize C_{max}

$$\sum_{k=1}^{m} r_{(i,i',k)}(s_i^k + w_i^k) = \sum_{k=1}^{m} r_{(i,j+1,k)}\, s_i^k \qquad (1)$$

$$s_i^k - s_{i'}^k \geq w_i^k - M(1 - Z_{(i,i',k)}) \qquad (2)$$

$$s_i^k - s_{i'}^k \geq w_{i'}^k - MZ_{(i,i',k)} \qquad (3)$$

$$\sum_{k=1}^{m} r_{(i,N_i,k)}(s_i^k + w_i^k) \leq C_{max} \qquad (4)$$

$$C_{max} \geq 0;\ \ s_i^k \geq 0 \qquad (5)$$

dengan

$i \in \{1,2,\dots,n); j \in \{1,2,\dots,N_i - 1\}$
$k \in \{1,2,\dots,m\}; Z_{(i,i',k)} \in \{0,1\};$
$dan\ 1 \leq i \leq i' \leq n$

Variabel yang digunakan

J_i	Job ke-i
M_k	Mesin ke-k
O_i^k	Operasi job ke-i yang dijalankan di mesin ke-k
$O_{(i,j)}$	Operasi ke-j pada job ke-i
N_i	Banyaknya operasi pada job ke-i
M	Bilangan positif sembarang yang sangat besar
n	Banyaknya job untuk diproses
m	Banyaknya mesin yang digunakan
w_i^k	Waktu proses job ke-i pada mesin ke-k
$r_{(i,j,k)}$	Biner pemroses, bernilai 1 jika O(i,j) diproses di Mk, bernilai 0 jika tidak
C_{max}	Waktu penyelesaian maksimal tercepat (makespan)
s_i^k	Waktu mulai tercepat job ke-i pada mesin ke-k
$Z_{(i,i',k)}$	Biner pendahulu, bernilai 1 jika job ke-i mendahului job ke-i' pada mesin ke-k, bernilai 0 jika tidak

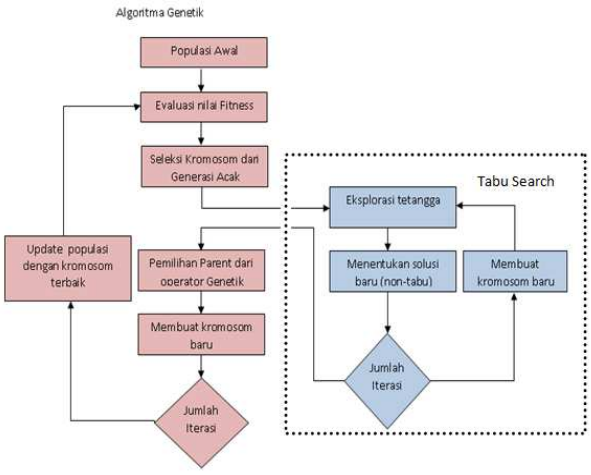
Persamaan (1) memberi batasan bahwa mesin ke-*k* akan memulai *O(i,j+1)* sesaat setelah *O(i,j)* sudah selesai (untuk meyakinkan bahwa batasan tanpa tunggu terpenuhi). Persamaan (2) dan (3) memberi batasan bahwa hanya ada satu *job* yang diproses dalam satu mesin pada satu waktu. Keduanya secara bersama-sama dipakai untuk meyakinkan bahwa hanya satu persamaan yang akan berdampak, terkait dengan nilai biner *Z(i,i',k)*. Persamaan (4) digunakan untuk meminimalkan *Cmax* terkait dengan fungsi tujuan. Sedangkan persamaan (5) digunakan untuk meyakinkan bahwa nilai *Cmax* dan *si k* tak negatif.

3. PERANCANGAN ALGORITMA

Dalam penelitian ini, metode yang diusulkan yakni metode *tabu search* yang digabungkan dengan algoritma genetika (untuk selanjutnya dinamakan *Incorporation Genetic-Tabu Search Algorithm / IGTS*). Kaidah *tabu search* digunakan sebagai kaidah dasar dalam algoritma, sedangkan kaidah algoritma genetika hanya terbatas penggunaannya pada proses pembangkitan sampel.

Tabu Search merupakan salah satu metode pemecahan permasalahan optimasi kombinatorial yang tergabung ke dalam *local search methods.* Metode ini bertujuan untuk mengefektifkan proses pencarian solusi terbaik dari suatu permasalahan optimasi kombinatorial yang berskala besar (bersifat *np-hard*), contohnya permasalahan penjadwalan *job shop,* dengan waktu komputasi yang relatif lebih kecil, namun tanpa ada jaminan akan tercapainya solusi yang optimal.

Secara umum alur algoritma *IGTS* yang digunakan sebagai metode penyelesaian permasalahan *SJMM* ini adalah sebagai berikut:



Gambar 1. Flowchart penggabungan algoritma genetic dan tabu search (IGTS)

Penjelasan dari alur di atas adalah sebagai berikut:

3.1. Pendefinisian *Input* dan *Output*

Pada langkah ini ditentukan *input* apa saja yang akan diproses pada algoritma dan *output* apa saja yang akan ditampilkan sebagai hasil dari proses. Adapun pada penelitian ini, mengingat permasalahan yang diteliti adalah permasalahan *SJMM* dan metode yang digunakan adalah *tabu search* yang digabungkan dengan algoritma genetika, maka *input* dan *output*-nya adalah sebagai berikut:

- *Input*:
  - Matriks rute mesin (***R(j,k)***; dengan *j* menyatakan nomor job dan *k* menyatakan nomor urutan operasi)
  - Matriks waktu proses (***W(j,k)***)
  - Jumlah sampel pembangkitan (***N)***
  - Parameter kejarangan (***ρ***)
  - Koefisien penghalusan (***α***)
  - Parameter pindah silang/*crossover rate* (***Pps***) inisial
  - Toleransi pemberhentian (***β***)

- *Output:*
  - *Timetable* jadwal optimal (waktu mulai dan waktu selesai masing-masing operasi)
  - Nilai makespan jadwal optimal (**Cmax**)
  - Waktu komputasi (***T***)
  - Jumlah iterasi (***iterasi***)

3.2. Penentuan Nilai Parameter Inisial

Parameter-parameter yang dimasukkan sebagai *input*, yakni ***N***, ***ρ*** , ***α, Pps*** inisial, dan ***β***, ditentukan sebagai tolok ukur performansi *output* yang dihasilkan. Dengan nilai parameter yang berbeda pastinya akan didapatkan hasil yang berbeda.

Adapun aturan untuk pemberian parameter tersebut di atas adalah sebagai berikut:

- Untuk jumlah sampel pembangkitan ***N***, tidak ada aturan khusus terkait dengan penentuan jumlah sampel pada optimasi kombinatorial, namun semakin besarjumlah job pastinya semakin besar sampel yang harusnya diambil. Pada algoritma digunakan jumlah sampel *default* sama dengan jumlah *job* pangkat tiga (n^3).
- Untuk parameter kejarangan ρ, biasanya digunakan nilai 1% - 10% (Kroese, 2009). Pada algoritma digunakan nilai *default* 2%
- Untuk koefisien penghalusan ***α***, nilainya berada antara 0 – 1, namun secara empiris nilai 0,4 - 0,9 merupakan nilai yang paling optimal (de Boer, 2003). Pada algoritma digunakan nilai *default* 0,8
- Untuk parameter pindah silang (***Pps***) inisial digunakan nilai 1 sebagai nilai *default*
- Untuk toleransi pemberhentian ***β***, nilainya mendekati 0 ($lim_{\beta \to 0} \beta$). Pada algoritma digunakan nilai *default* 0,001

3.3. Pembangkitan Sampel

Sampel yang dibangkitkan di sini adalah merupakan urutan prioritas job yang terlebih dahulu dijadwalkan. Pembangkitan sampel inisial (iterasi=1) dilakukan dengan menggunakan teknik acak penuh (*fully randomized*), sedangkan pada pembangkitan
sampel setelahnya (iterasi>1) dilakukan dengan menggunakan kaidah algoritma genetika, yakni pindah silang dan mutasi, dengan alur sebagai berikut:

a) Pembobotan sampel
Pembobotan ini diperlukan untuk tahap pemilihan induk pertama, di mana induk pertama akan dipilih dari sampel elite dengan mempertimbangkan bobot dari masing-masing urutan pada sampel elite. Adapun aturan pembobotan yang digunakan yakni, jika *makespan* yang dihasilkan oleh urutan lebih baik daripada *makespan* terbaik yang pernah dikunjungi pada iterasi sebelumnya, maka bobot yang diberikan adalah senilai banyaknya sampel elite, selebihnya hanya diberi bobot 1.

b) Penghitungan *linear fitness ranking*
Linear fitness ranking (*LFR*) dihitung dari perbandingan nilai kesesuaian (*fitness*) semua sampel yang telah dibangkitkan pada iterasi sebelumnya. Nilai *LFR* adalah sebesar $LFR(I(N-i+1))=Fmax-(Fmax-Fmin)((i-1)/(N-1))$ Dengan *i* merupakan nilai yang berkisar antara 1 sampai dengan jumlah sampel (*N*), serta *I* menunjukkan indeks job pada matriks sampel.

c) Pemilihan induk
Pemilihan induk dilakukan dengan menggunakan mekanisme roda *roulette* (*roulette wheel selection*), di mana bobot yang lebih besar akan mendapatkan peluang yang lebih tinggi untuk terpilih sebagai induk. Induk pertama dipilih dari sampel elite dengan pembobotan yang sudah ditentukan sebelumnya, sedangkan induk kedua dipilih dari sampel pada iterasi sebelumnya dengan pembobotan dari nilai *linear fitness ranking*.

d) Pindah silang
Pindah silang dilakukan dengan menggunakan teknik *order crossover* dua titik, dimana pemilihan dua titik tersebut dilakukan secara acak pada kedua induk. Hasil anakan yang didapat mempunyai segmen yang sama di antara dua titik tersebut sebagaimana induknya, namun segmen yang lain dijaga urutannya sesuai dengan induk yang lain dari masing-masing anakan.

e) Mutasi
Mutasi dilakukan dengan menggunakan metode *swapping mutation*, di mana mutasi dilakukan

dengan mempertukarkan *job* yang termutasi dengan *job* lain pada anakan yang sama.

3.4. Penghitungan Makespan

Penghitungan nilai *makespan* ini dilakukan dengan menggunakan metode *timetabling* geser sederhana (*simple shift timetabling*) yang diadaptasi dari metode geser dari Zhu dkk. (2009)). Langkah-langkahnya sebagai berikut:

a) Jadwalkan *job* urutan priotitas pertama mulai t = 0
b) Jadwalkan *job* urutan prioritas selanjutnya mulai t=0, kemudian cek apakah terjadi *overload* pada mesin. Jika ya, maka geser *job* ke kanan sampai tidak terdapat *overload*
c) Ulangi langkah b) hingga seluruh job terjadwalkan

3.5. Pemilihan Sampel Elite

Sampel elite dipilih sebanyak $\lceil \rho N \rceil$ dari jumlah sampel yang dibangkitkan berdasarkan nilai *makespan*-nya. Sebelum langkah ini dilakukan, terlebih dahulu *makespan* yang didapatkan dari semua sampel diurutkan dari yang terkecil hingga yang terbesar.

3.6. Pembaharuan Parameter Pindah Silang dan Parameter Mutasi

Pembaharuan parameter dilakukan dengan menggunakan perbandingan antara rata-rata *makespan* pada sampel dengan 2 kali nilai *makespan* terbaik pada tiap iterasi. Nilai perbandingan tersebut didefinisikan sebagai u. Parameter pindah silang kemudian diperbaharui dengan menggunakan rumus $Pi=(1-\alpha)u+Pi\text{-}1$ Sedangkan parameter mutasi nilainya didefinisikan sebagai ½ dari nilai parameter pindah silang.

3.7. Pengecekan terhadap Syarat Pemberhentian

Syarat pemberhentian pada penelitian ini adalah bahwa selisih antara parameter pindah silang hasil pembaharuan dengan parameter pindah silang sebelumnya lebih kecil dari β.

Jika syarat pemberhentian ini terpenuhi, maka hentikan iterasi dan lanjutkan ke langkah berikutnya. Jika tidak, maka ulangi kembali iterasi mulai langkah keempat.

3.8. Penampilan Hasil

Hasil yang ditampilkan dari algoritma ini adalah berupa *output-output* yang sudah didefinisikan sebelumnya.

3. PENGUJIAN ALGORITMA

Pengujian algoritma dilakukan dengan membuat kode program algoritma pada *software* Matlab). Kode ini kemudian dijalankan sebanyak 30 kali pengulangan, dan menggunakan *software* Matlab.

Asumsi terdapat 5 pekerjaan dan 5 mesin dengan urutan operasi dan waktu proses untuk setiap operasi yang ditentukan dalam Table1 dan Tabel 2. Selanjutnya menjalankan program untuk lima kali ukuran populasi = 10, jumlah iterasi untuk mutasi adalah 100 dan crossover adalah 200. Algoritma berakhir setelah 200 generasi. Dari hasil di gambar2, dapat ditunjukkan bahwa kombinasi blok kritis, jarak dan algoritma genetika dapat memberikan hasil sebagus metode lainnya. Dari gambar2 kita dapat melihat bahwa pekerjaan terakhir diproses adalah tugas 5 pada mesin 4. Jadi, kita nilai makespan untuk masalah ini adalah 34. Hasil penelitian ini juga memberi kita urutan pekerjaan bagi setiap mesin untuk pengolahan, waktu mulai dan waktu akhir untuk setiap operasi. Misalnya, pada mesin 1, kita mulai dengan proses job3 pada waktu 0 dan berakhir pada waktu 7. Kemudian kita proses job1, diikuti oleh job4, job5 dan job2.

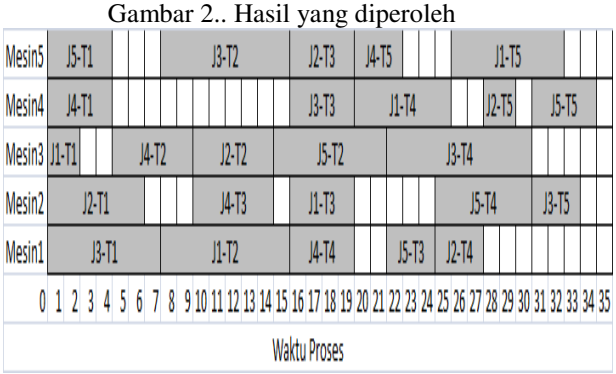
Tabel 1. Operasi Task

	Task1	Task2	Task3	Task4	Task5
Job1	Mesin3	Mesin1	Mesin2	Mesin4	Mesin5
Job2	Mesin2	Mesin3	Mesin5	Mesin1	Mesin4
Job3	Mesin1	Mesin5	Mesin4	Mesin3	Mesin2
Job4	Mesin4	Mesin3	Mesin2	Mesin1	Mesin5
Job5	Mesin5	Mesin3	Mesin1	Mesin2	Mesin4

Table 2. Waktu Proses Setiap Task

	Mesin1	Mesin2	Mesin3	Mesin4	Mesin5
Job1	8	4	2	6	7
Job2	3	6	5	2	4
Job3	7	3	9	4	8
Job4	4	5	5	4	3
Job5	3	6	7	4	5

Gambar 2.. Hasil yang diperoleh



Dari percobaan selama lima kali, kita dapat menentukan nilai optimum sebelum generasi ke 100 tercapai dengan nilai makespan 34.

4. KESIMPULAN

Dari hasil analisis dan interpretasi hasil pengujian serta perbandingan hasil eksperimen dengan metode lain, dapat ditarik kesimpulan bahwa metode *IGTS* dapat digunakan sebagai metode alternatif penyelesaian permasalahan *SJMM*, dan dapat

diaplikasikan kegunaannya pada aplikasi yang diproses secara terdistribus dengan karakteristik *SJMM*. Ini mengingat performansi yang didapatkan dari algoritma ini relatif baik, terutama pada kasus *SJMM* berukuran kecil, walaupun performansinya semakin menurun seiring dengan bertambahnya ukuran kasus.

Diharapkan pada penelitian-penelitian selanjutnya, ada upaya untuk melakukan modifikasi lanjutan bagi algoritma *IGTS* bagi permasalahan *SJMM* agar dapat meningkatkan performansinya untuk ukuran populasi yang besar. Selain itu, penggunaan algoritma ini pada permasalahan yang lain juga dianjurkan.

DAFTAR PUSTAKA

[1]. Baker, Kenneth R. 1974. *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons, Inc.

[2]. Beasley, J. E., 1990. *OR-Library: Distributing Test Problems by Electronic Mail*. Journal of the Operational Research Society 41: 1069 – 1072.

[3]. Boubezoul, Abderrahmane; Paris, Sébastien; dan Ouladsine, Mustapha. 2008. *Application of the Cross Entropy Method to the GLVQ Algorithm*. Pattern Recognition 41: 3173– 3178

[4]. Bożejko, Wojciech dan Makuchowski, Mariusz. 2009. *A Fast Hybrid Tabu Search Algorithm for the No-Wait Job Shop Problem*. Computers & Industrial Engineering 56:1502–1509

[5]. Caserta, M.; Rico, E. Quiñonez; dan Uribe, A. Márquez. 2008. *A Cross Entropy Algorithm for the Knapsack Problem with Setups*. Computers & Operations Research 35: 241 –252

[6]. Dewi, Dian Retno Sari. *Pengembangan Algoritma Penjadualan Produksi Job Shop untuk Meminimumkan Total Biaya Earliness dan Tardiness.* Jurnal Ilmiah Teknik Industri 4 no. 2: 57 – 65

[7]. Ganduri, Chandrasekhar V. 2004. *Rule Driven Job-Shop Scheduling Derived from Neural Networks Through Extraction*. Department of Industrial and Manufacturing

[8]. Systems Engineering, Ohio University Kroese, Dirk P. 2009. *Cross-Entropy Method*. Brisbane: Mathematics Department, University of Queensland

[9]. Liaw, Ching-Fang. 2008. *An Efficient Simple Metaheuristic for Minimizing The Makespan in Two-Machine No-Wait Job Shops*. Computer & Operations Research 35: 3276 –3283

[10]. Mascis, Alessandro, dan Pacciarelli, Dario. 2002. *Discrete Optimization: Job-Shop Scheduling with Blocking and No-Wait Constraints*. European Journal of Operational Research 143: 498 – 517

[11]. Pan, Jason Chao-Hsien, dan Huang, Han-Chiang. 2009. *A Hybrid Genetic Algorithm for No-Wait Job Shop Scheduling Problems*. Expert Systems with Application 36: 5800 – 5806

[12]. Rubinstein, Reuven Y., dan Kroese, Dirk P. 2004. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. New York: Springer Science+Business Media, Inc.

[13]. Schuster, Christoph J. dan Framinan, Jose M. 2003. *Approximative Procedures for No-Wait Job Shop Scheduling*. Operations Research Letters 31: 308 – 318

[14]. Widyarini, Tiananda. 2009. *Aplikasi Metode Cross Entropy untuk Support Vector Machines*. Surabaya: Institut Teknologi Sepuluh Nopember 11

[15]. Waterloo Manufacturing Software. *Job Shop Scheduling* http://www.waterloosoftware.com/planning-and-scheduling/job-shop-scheduling.html. Diakses pada 24 Februari 2010

[16]. Wikipedia. *Cross Entropy Method*. http://en.wikipedia.org/wiki/Cross_Entropy_Method. Diakses pada 19 Februari 2010

[17]. _____. *Crossover* (*Genetic Algorithm*). http://en.wikipedia.org/wiki/Crossover_(Genetic_Algorithm). Diakses 18 Juni 2010

[18]. _____. *Genetic Algorithm*. http://en.wikipedia.org/wiki/Genetic_Algorithm. Diakses pada 18 Juni 2010

[19]. _____. *Kullback-Leibler Divergence*. http://en.wikipedia.org/wiki/Kullback-Leibler_Divergence. Diakses pada 19 Februari 2010

[20]. _____. *Mutation* (*Genetic Algorithm)*. http://en.wikipedia.org/wiki/Mutation_(Genetic_Algorithm). Diakses pada 18 Juni 2010

[21]. _____. *Selection (Genetic Algorithm*). http://en.wikipedia.org/wiki/Selection_(Genetic_Algorithm). Diakses pada 18 Juni 2010

[22]. Zhu, Jie; Li, Xiaoping; dan Wang, Qian. 2009. *Complete Local Search with Limited MemoryAlgorithm for No-Wait Job Shops to Minimize Makespan*. European Journal of Operational Research 198: 378–386