PEMAKAIAN ALGORITMA GENETIK UNTUK PENJADWALAN JOB SHOP DINAMIS NON DETERMINISTIK

Nico Saputro, Yento
Jurusan Ilmu Komputer – FMIPA, Universitas Katolik Parahyangan
E-mail: nico@home.unpar.ac.id

ABSTRAK

Penjadwalan job shop dinamis non deterministik merupakan persoalan pengurutan sejumlah operasi yang diproses pada mesin-mesin tertentu dengan urutan pengerjaan berbeda untuk setiap job yang berbeda, dimana kedatangan job tersebut bervariasi dan tidak diketahui sebelumnya. Penjadwalan ulang diperlukan bila telah disusun suatu jadwal dan kemudian tiba suatu pekerjaan baru. Algoritma genetik dapat dipergunakan untuk menyusun jadwal maupun untuk menyisipkan jadwal saat ada penambahan pekerjaan tanpa mengubah jadwal yang telah dikerjakan sebelumnya.

Kata kunci: algoritma genetik, penjadwalan dan penjadwalan ulang job shop dinamis non deterministik.

ABSTRACT

Dynamic Job shops non deterministic scheduling problem is concerned with ordering some operations that processed by certain machines with variable and unknown arrival time of jobs. When new jobs arrive, it requires modifications in the existing schedule. We use Genetic Algorithm approach to the dynamic job shop non deterministic scheduling and rescheduling problem.

Keywords: *genetic algorithm, dynamic job shop non-deterministic scheduling and rescheduling.*

1. PENDAHULUAN

Algoritma Genetik merupakan algoritma pencarian yang bekerja berdasarkan mekanisme seleksi alam dan genetika. Pada genetika, kromosom tersusun dari gen-gen. Tiap gen mempunyai sifat tertentu (*allele*), dan posisi tertentu (*locus*). Satu atau lebih kromosom bergabung membentuk paket genetik yang disebut *genotif*. Interaksi *genotif* dengan lingkungannya disebut *fenotif*. Pada algoritma genetik, kromosom berpadanan dengan string dan gen dengan karakter. Setiap karakter mempunyai posisi (*locus*) dan arti tertentu (*allele*). Satu atau lebih string bergabung membentuk struktur (*genotif*), dan bila struktur tersebut di-*decode*-kan akan diperoleh salah satu alternatif solusi (*fenotif*).

Kedatangan job seringkali tidak dapat diramalkan (non deterministik). Apabila ada job baru datang, terkadang perlu disusun jadwal baru. Jika ternyata job sebelumnya sudah mulai di proses, perlu dilakukan penjadwalan ulang agar sesuai dengan kondisi yang baru. Penjadwalan ulang harus dilakukan dengan cepat supaya tidak menunda pemrosesan job baru.

Ada 3 pendekatan penjadwalan ulang yaitu: dengan menjadwalkan kembali dari awal, melakukan perbaikan terhadap jadwal saat perubahan terjadi, atau menghilangkan beberapa operasi yang telah dikerjakan pada jadwal lama dan melakukan penjadwalan ulang terhadap sisa operasi yang belum dikerjakan. Pendekatan pertama dilakukan dengan membuang jadwal lama dan membuat jadwal baru dari awal dengan data yang baru. Pendekatan kedua dilakukan secara iteratif dengan memodifikasi jadwal lengkap sehingga didapat jadwal yang memuaskan. Sedangkan pendekatan ketiga tetap mempertahankan sebagian jadwal yang telah disusun sebelumnya dan menyusun jadwal baru dari sisa job-job yang belum selesai dikerjakan (Fang, 1994). Pada makalah ini, pendekatan ketiga akan diterapkan dengan memakai algoritma genetik.

2. ALGORITMA GENETIK

Populasi merupakan kumpulan string, setiap string mempunyai nilai *fitness* dan mewakili satu solusi dalam domain solusi (Saputro, 2003). Nilai *fitness* dapat diperoleh dari fungsi obyektif dari permasalahan yang dihadapi. Umumnya, string ber-*fitness* tinggi akan bertahan dan berlanjut ke generasi berikutnya. Pencarian solusi secara iteratif terhadap populasi untuk menghasilkan populasi baru. Dalam satu siklus iterasi (disebut generasi), terjadi proses seleksi dan rekombinasi.

Proses seleksi dilakukan dengan mengevaluasi setiap string berdasarkan nilai *fitness* untuk mendapatkan peringkat string. Selanjutnya dipilih secara acak, string-string yang akan mengalami proses rekombinasi. Umumnya string ber- fitness tinggi berpeluang lebih besar untuk terpilih. Proses rekombinasi memakai operator genetik untuk menghasilkan string-string baru yang berbeda dari string-string induknya. Ada 3 operator dasar yaitu Reproduksi, *Crossover*, dan mutasi. Operator lainnya merupakan hasil modifikasi dari operator dasar.

3. PENJADWALAN JOB SHOP

Masalah penjadwalan *job shop* merupakan persoalan pengurutan sejumlah operasi yang diproses pada mesin-mesin tertentu (Dimyati, 1999). Masalah penjadwalan *job shop* adalah bagaimana menyusun semua operasi dari semua *job* pada tiap mesin dalam rangka meminimasi fungsi obyektif. Fungsi obyektif yang dimaksud dapat berupa waktu pengerjaan total, rata-rata waktu pengerjaan, rata-rata waktu keterlambatan penyelesaian job, atau lainnya (Husbands).

Berdasarkan waktu kedatangan *job*, penjadwalan *job shop* dapat dikelompokkan sebagai *static job shop scheduling* dan *dynamic job shop scheduling.* Pada *static job shop scheduling*, semua *job* diterima pada saat yang sama. Pada *dynamic job shop scheduling*, waktu kedatangan job bervariasi tetapi sudah diketahui sebelumnya (*deterministic*) atau waktu kedatangan job bervariasi dan tidak dapat diketahui sebelumnya (*non deterministic / stochastic*) (Lin, 1997).

4. PEMODELAN

Penjadwalan *job shop* dinamis non deterministik yang akan diimplementasikan memakai algoritma genetik dibatasi sebagai berikut : tiap *job* diproses oleh sebuah mesin maksimal satu kali, tidak memiliki tenggat waktu penyelesaian, waktu perpindahan antar mesin dan waktu *setup* dapat diabaikan, dan penjadwalan bersifat *non-preemptive*.

Ada empat hal dasar yang perlu diperhatikan, yaitu : pemilihan representasi masalah ke bentuk string, operator genetik, fungsi *fitness*, dan parameter genetik.

4.1 Representasi ke bentuk string

Representasi string bergantung pada masalah yang akan diselesaikan. Tipe pengkodean yang umumnya dipakai antara lain *binary encoding*, *permutation encoding*, dan *value encoding* (Koza, 2001). Pada penjadwalan *job shop* akan dipergunakan *permutation encoding*. Satu karakter mewakili suatu operasi, nilai karakter (*allele*) mewakili nomor *job* dan nomor urut operasi pada *job* tersebut. *Allele* akan dipetakan oleh fungsi yang memasangkan nomor *job* dan nomor urut operasi menjadi nomor mesin dan lama proses. Jika ada n *job* dan tiap *job* maksimal terdiri dari m operasi, panjang string maksimal n x m karakter. Sebagai contoh, misalkan ada 3 *job* yang diproses pada 3 buah mesin dengan urutan dan waktu operasi seperti tabel 1. m_i adalah nomor mesin pada operasi ke i dan p_i lamanya penggunaan mesin untuk operasi ke i.

Tabel 1. Contoh Job Shop

	m_1, p_1	M_2, p_2	m_3, p_3
Job 1	3,4	1,3	2,6
Job 2	1,7	3,5	2,5
Job 3	3,6	2,4	1,5

Pada tabel 1, job 1 diproses secara berturutan pada mesin 3, 1, dan 2 dengan lama pengerjaan ditiap mesin 4, 3, dan 6 satuan waktu. Secara keseluruhan terdapat sembilan buah operasi, sehingga string terdiri dari 9 karakter.

3-1	1-1	2-2	3-2	2-1	2-3	1-3	1-2	3-3

Gambar 1. Contoh Representasi String untuk Tabel 1.

Pada gambar 1, *allele* 3-1 berarti *job* ke 3 operasi ke 1, bila dipetakan ke tabel terlihat dikerjakan dimesin 3 selama 6 satuan waktu. *Allele* 1-1 berarti *job* ke 1 operasi ke 1 dimesin 3 selama 4 satuan waktu, dst. Bila sebuah *job* tidak diproses di semua mesin, operasi akhir dari *job* tersebut akan dikosongkan.

4.2 Pemilihan Operator Genetik

Operator genetik yang dipakai adalah operator reproduksi gabungan dari *elitism* dan *roulette wheel selection*, operator *Precedence Preservative Crossover* (PPX) (Bierwirth, 1999) dan operator mutasi *remove and insert* (Manderick, 1991). Metode *elitism* membuat sejumlah string terbaik tiap generasi akan otomatis diturunkan ke generasi

berikutnya Koza, 2001). Metode *roulette wheel* untuk memilih string-string yang akan dilakukan proses rekombinasi (Saputro, 2003).

Pada PPX, string baru disusun secara acak dari *allele* string-string induk. Angka acak 1 atau 2 dipakai untuk memilih induk. Jika 1 diturunkan *allele* paling kiri dari induk pertama, jika 2 diturunkan *allele* paling kiri dari induk kedua. Selanjutnya *allele* yang terpilih tadi dihapus dari kedua induk. Proses dilakukan sampai karakter di kedua induk habis. Sebagai contoh 2 induk ABCDEF dan CABFDE, dan angka acak 1 2 1 1 2 2, akan menghasilkan string baru ACBDFE.

Pada mutasi, satu *locus* dipilih secara acak dan karakter diposisi tersebut di hapus. Satu *locus* baru dipilih, dan karakter yang telah dihapus tadi disisipkan. Gambar 2 menunjukkan proses mutasi pada *locus* ketiga dan ketujuh.

A	**3-1**	**1-1**	**2-2**	**3-2**	**2-1**	**2-3**	**1-3**	**1-2**	**3-3**

Hasil	**3-1**	**1-1**	**3-2**	**2-1**	**2-3**	**1-3**	**2-2**	**1-2**	**3-3**

Gambar 2. String A dan Hasil Mutasi

4.3 Fungsi *Fitness*

Pada dynamic scheduling dengan kedatangan job tidak dapat diperkirakan sebelumnya, minimalisasi makespan dirasakan kurang berarti (Lin, 1997). Oleh karena itu, rata-rata waktu penyelesaian sebuah *job* (*average flow time*) pada satu periode penjadwalan digunakan sebagai fungsi fitness. Waktu selesainya sebuah job dapat dihitung dari selisih waktu tiba (r_i) dengan waktu selesainya operasi terakhir dari job (C_i). Satu periode penjadwalan melibatkan operasi-operasi yang belum mulai diproses mesin. Tujuan penjadwalan adalah minimasi fungsi fitness, sedangkan pada algoritma genetik, sesuai proses di alam, prosesnya adalah maksimasi. Oleh karena itu, fungsi fitness penjadwalan diubah menjadi 1/f.

4.4 Parameter Genetik

Parameter Genetik berguna dalam pengendalian operator-operator genetik. Parameter yang digunakan adalah : ukuran populasi, jumlah generasi, Probabilitas *Crossover* (P_c), dan Probabilitas Mutasi (P_m). Tidak ada aturan pasti tentang berapa nilai setiap parameter ini (Koza, 2001). Ukuran populasi kecil berarti hanya tersedia sedikit pilihan untuk crossover dan sebagian kecil dari domain solusi saja yang dieksplorasi untuk setiap generasinya. Sedangkan bila terlalu besar, kinerja algoritma genetik menurun. Penelitian menunjukkan ukuran populasi besar tidak mempercepat proses pencarian solusi. Disarankan ukuran populasi berkisar antara 20-30, probabilitas *crossover* berkisar 80%-95%, dan probabilitas mutasi kecil berkisar 0.5%-1%. Jumlah generasi besar berarti semakin banyak iterasi yang dilakukan, dan semakin besar domain solusi yang akan dieksplorasi.

4.5 Pemodelan penjadwalan

Pemodelan penjadwalan yang dimaksud adalah proses penyusunan jadwal dari string. Permutasi operasi-operasi yang direpresentasikan oleh string akan di-*decode* untuk

menghasilkan jadwal. Ada tiga karakteristik jadwal yang dapat dihasilkan, yaitu *semi-active*, *active*, dan *non delay* (Bierwirth, 1999). Makalah ini memakai jadwal *hybrid* yaitu gabungan antara jadwal *active* dan *non delay*. Notasi-notasi yang dipakai pada jadwal hybrid maupun proses kerja penjadwalan job shop dengan algoritma genetik dapat dilihat pada Tabel 2.

Tabel 2. Notasi

m	Jumlah mesin
n	Jumlah job
m_i	Jumlah operasi dari job ke-i, $1 \le i \le n$, $m_i \le m$
$\mu_i(k)$	Urutan mesin yang harus dilalui oleh job ke-i operasi ke-k.
o_{ik}	Operasi ke-k dari job ke-i, $1 \le k \le m_i$.
o_{i1}	Operasi ke-1 dari job ke-i
p_{ik}	lama proses operasi ke-k dari job ke-i
t_{ik}	waktu mulai (*starting time*) operasi ke-k dari job ke-i
r_i	waktu tiba job ke i
δ	parameter yang menunjukkan batas lamanya suatu mesin dapat mengganggur, $\delta \in [0,1]$, $\delta = 0$ untuk jadwal *non-delay*, $\delta = 1$ untuk jadwal *active*

Langkah-langkah prosedur *hybrid* (Bierwirth, 1999) :

1. Buat himpunan operasi yang mengawali pekerjaan: $A=\{o_{i1} \mid 1 \le i \le n\}$
2. i. Pilih o^1, operasi dengan waktu selesai tercepat, $t^1 + p^1 \le t_{ik} + p_{ik}$ untuk semua $o_{ik} \in A$. Jika lebih dari satu operasi, pilih operasi yang terletak paling kiri di string.
   ii. Jika M^1 adalah mesin yang dipakai oleh o^1, buat himpunan B yang berisi semua operasi dari *A* yang diproses di M^1, $B:=\{o_{ik} \in A \mid m_i(k)=M^1\}$
   iii. Pilih o^{11}, operasi dengan waktu mulai paling awal di *B*, $t^{11} < t_{ik}$ untuk semua $o_{ik} \in B$. Jika lebih dari satu operasi, pilih operasi yang terletak paling kiri di string.
   iv. Hapus operasi di *B* menurut parameter *d*, sehingga himpunan *B* sekarang $B:=\{o_{ik} \in B \mid t_{ik} \le t^{11} + d((t^1+p^1)-t^{11})\}$
   v. Pilih operasi di *B* yang terletak paling kiri di string dan hapus dari *A*. Operasi yang dipilih tersebut adalah o^*_{ik}.
3. Masukkan operasi o^*_{ik} pada jadwal, dan hitung waktu mulai: $t^*_{ik} = max(t^*_{i,k1} + p^*_{i,k-1}, t_{hl} + p_{hl})$, t^*_{ik} waktu mulai operasi o^*_{ik}, $t^*_{ik-1} + p^*_{ik-1}$ waktu selesai operasi ke-*(k-1)*, yaitu operasi sebelumnya dari job ke-*i* dan o_{hl}=operasi ke-*l* dari job ke-*h* yang mendahului o^*_{ik} pada mesin yang sama.
4. Jika terdapat suksesor dari o^*_{ik}, yaitu $o^*_{i,k+1}$, tambahkan ke *A*.
5. Ulangi langkah 2 sampai isi *A* habis.

4.6 Langkah-langkah penjadwalan job shop dengan algoritma genetik

a Tetapkan peluang crossover, peluang mutasi, *d, nElit* (banyaknya elitism) dan *MaxGen* (jumlah generasi maksimal).
b Bentuk populasi awal secara acak sebanyak Nmax string, susun jadwal tiap string dengan prosedur hybrid dan hitung nilai fitness tiap string.
c Bentuk populasi berikutnya :

c.1 Masukkan *nElit* buah string terbaik ke populasi berikut (elitism)
c.2 Sisa (*Nmax-nElit*) string di dapat dari proses genetika (crossover dan mutasi)
c.3 Susun jadwal tiap string baru dengan prosedur hybrid dan hitung nilai fitness-nya.

d Ulangi langkah c sampai *MaxGen* tercapai.
e Tampilkan jadwal akhir dari string terbaik.
f Bila ada tambahan job baru pada saat t_1

f.1 Simpan jadwal dari operasi o_{ik} yang dimulai sebelum t_1. Jadwal ini tidak perlu di ubah lagi saat penjadwalan ulang.

f.2 Hapus semua operasi o_{ik} yang dimulai sebelum t_1 ($t_{ik}<t_1$) dari kumpulan job lama. Suatu job telah selesai dikerjakan seluruhnya bila $m_i = 0$. Bila ada job ke-i yang belum selesai, hitung waktu tiba yang baru :

$$r_i = \max_{1 \le k \le m_i} \{t_{ik} + p_{ik} | t_{ik} < t_1\}$$

f.3 Jika ada operasi o_{ik} yang masih diproses pada t_1 dan belum selesai ($t_{ik}<t_1< t_{ik}+p_{ik}$), mesin tidak dapat dipakai sampai operasi tersebut selesai. Perlu dihitung initial setup time s_j, yaitu waktu dimana mesin ke-j baru dapat mulai digunakan lagi.

$$s_j = \max\left(\max_{1 \le i \le n} \{t_{ik} + p_{ik} | t_{ik} < t_1\}, t_1 \right)$$

Waktu setup ini akan dipergunakan saat penjadwalan ulang dan dipakai saat menghitung waktu mulai (starting time) dari operasi yang memakai mesin tersebut pertama kali. Waktu mulai untuk mesin ke-j saat dipergunakan pertama kali oleh operasi o_{ik} dapat dihitung: $t_{ik} = \max(t_{i,k-1} + p_{i,k-1}, s_j)$

g Bentuk string baru dari sisa operasi yang belum dikerjakan pada jadwal lama, dan dari operasi-operasi dari job-job yang tiba pada saat t_1.
h Bentuk populasi awal yang baru secara acak sebanyak Nmax string, susun jadwal tiap string dengan prosedur hybrid dan hitung nilai fitness tiap string.
i Ulangi langkah c sampai d.
j Tampilkan jadwal akhir dari string terbaik.

5. PENGUJIAN DAN PEMBAHASAN

Pengujian dilakukan dengan penambahan job terhadap suatu jadwal yang telah dibuat. Spesifikasi job saat t=0 dapat dilihat pada tabel 3 dan hasil penjadwalan pada gambar 3.. Selanjutnya saat muncul *job* baru pada t=5, dengan spesifikasi job pada tabel 4, job baru tersebut ditambahkan pada jadwal yang sudah ada. Terlihat job 6, 7, dan 8 dapat ditambahkan dengan benar ke dalam jadwal yang sudah ada seperti terlihat pada gambar 4.

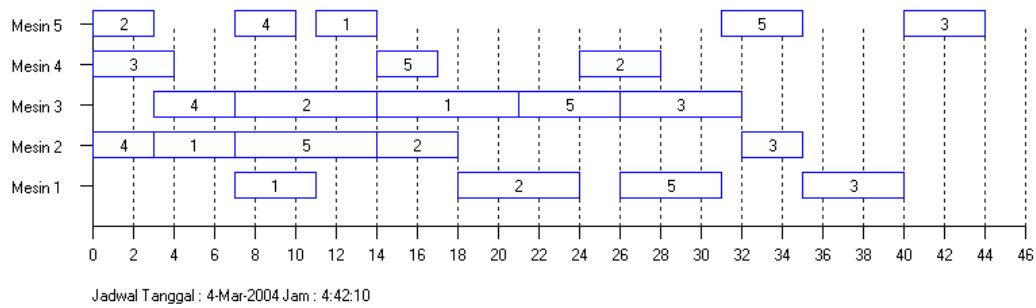
Parameter genetik yang digunakan untuk mendapatkan jadwal pada gambar 3 dan gambar 4 adalah : ukuran populasi = 30, jumlah generasi=20, jumlah *elitism* = 5, Pc = 0.8, Pm = 0.2, δ=1.0.

Tabel 3. Spesifikasi Job Awal

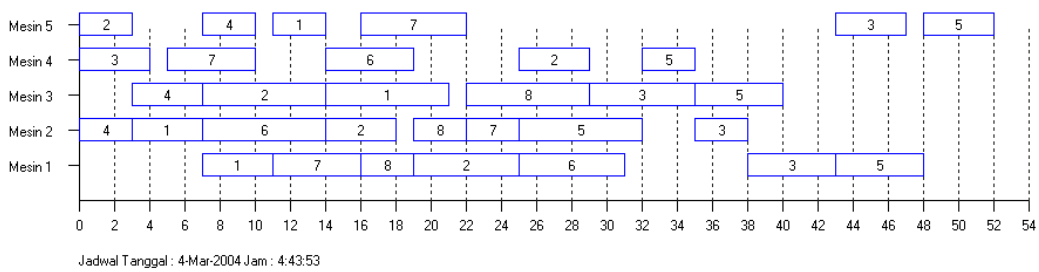
	m_1, p_1	m_2, p_2	M_3, p_3	M_4, p_4	M_5, p_5
Job 1	2,4	1,4	5,3	3,7	
Job 2	5,3	3,7	2,4	1,6	4,4
Job 3	4,4	3,6	2,3	1,5	5,4
Job 4	2,3	3,4	5,3		
Job 5	2,7	4,3	3,5	1,5	5,4

Tabel 4. Spesifikasi Job Tambahan, saat t=5

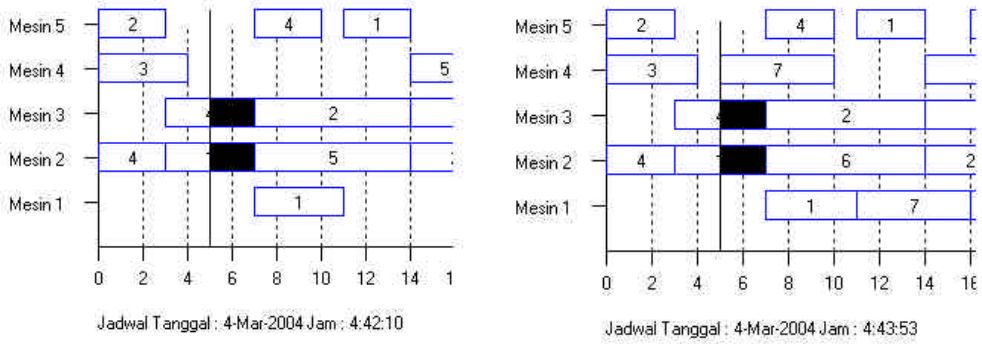
	M_1, p_1	m_2, p_2	M_3, p_3	M_4, p_4	m_5, p_5
Job 6	2,7	4,5	1,6		
Job 7	4,5	1,5	5,6	2,3	
Job 8	1,3	2,3	3,7		



Gambar 3. Hasil Penjadwalan dari Tabel 3, d=1.0



Gambar 4. Hasil Penjadwalan setelah Penambahan Job, d=1.0

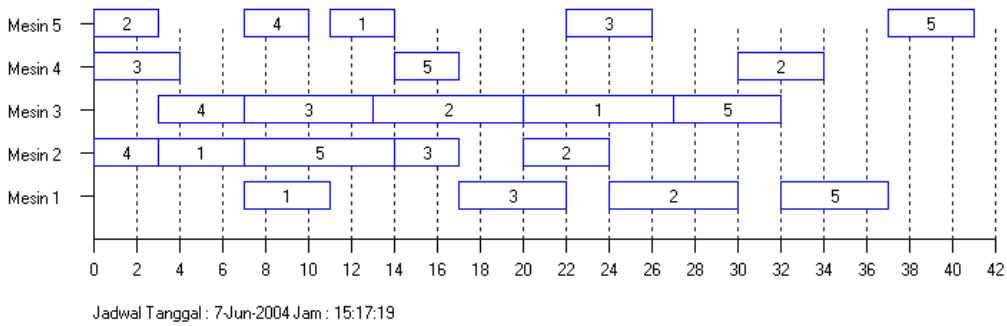


(a) Sebelum penambahan job (b) Setelah penambahan job

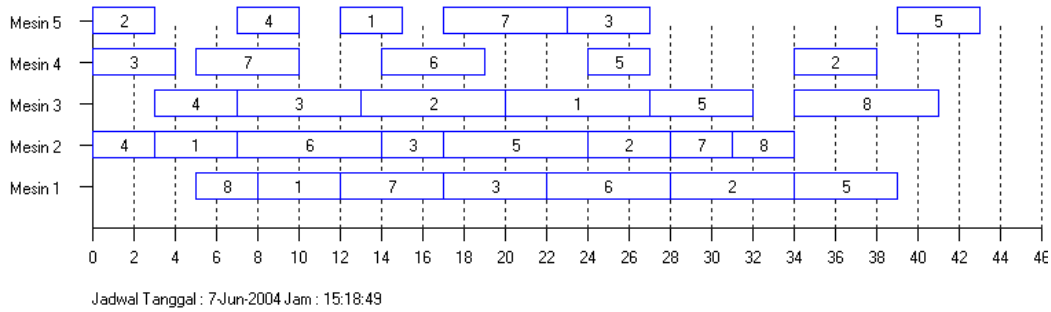
Gambar 5. Waktu Setup

Saat t=5, mesin 2 dan mesin 3 tidak dapat dipergunakan sampai t=7 karena masih dipakai oleh job 4 dan job 1. Oleh algoritma yang dipakai, selang waktu tersebut dianggap sebagai waktu setup, waktu dimana mesin baru dapat mulai digunakan lagi. Bila dibandingkan pada gambar 5, perubahan penggunaan mesin 2 dari job 5 oleh job 6 baru dapat dilakukan saat t=7 sedangkan mesin 4 karena tidak ada waktu setup dapat langsung dipakai oleh job 7 saat t=5.

Pengujian berikutnya adalah melihat hasil penjadwalan terhadap perubahan parameter δ. Parameter genetik yang digunakan sama yaitu : ukuran populasi = 30, jumlah generasi=20, jumlah *elitism* = 5, Pc = 0.8, Pm = 0.2. Gambar 6 dan gambar 7 menunjukkan jadwal sebelum dan setelah penambahan job untuk δ=0.0. Terlihat jadwal yang dihasilkan lebih ketat dibandingkan dengan δ=1.0. Mesin 1 dan mesin 2 pada δ=0.0 setelah adanya penambahan job terlihat tidak perlu menunggu sama sekali (bandingkan gambar 4 dan gambar 6).

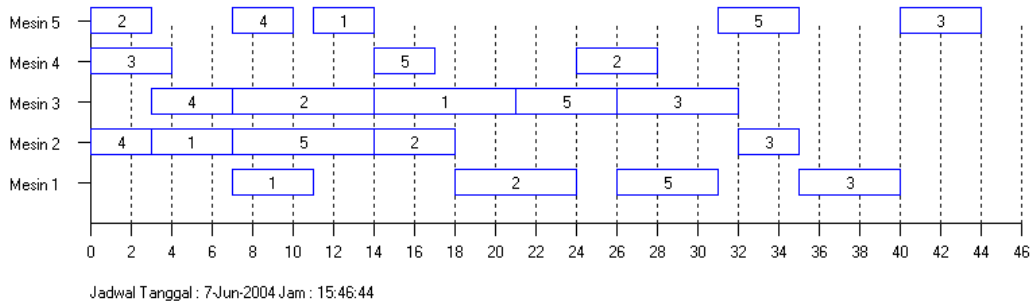


Gambar 6. Hasil penjadwalan dari tabel 3, d=0.0

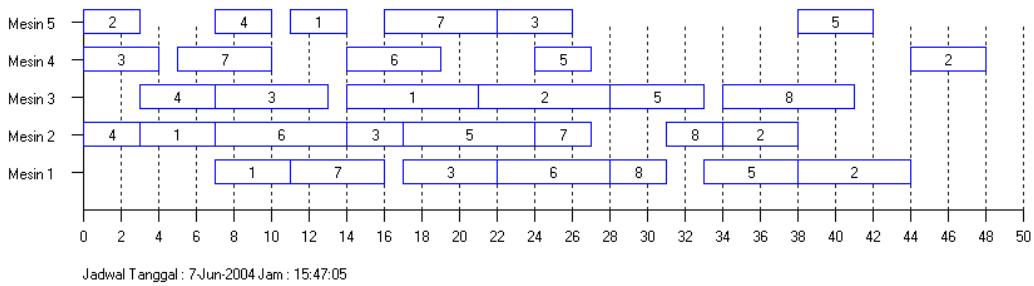


Gambar 7. Hasil Penjadwalan Setelah Penambahan Job, d=0.0

Gambar 8 dan gambar 9 menunjukkan jadwal sebelum dan setelah penambahan job untuk δ=0.8. Terlihat, jadwal awal pada gambar 8 sama seperti jadwal awal untuk δ=1.0. (gambar 3). Tetapi setelah ada penambahan job, jadwal yang dihasilkan terlihat lebih ketat dibandingkan jadwal untuk δ=1.0 (gambar 4), meskipun tidak seketat jadwal untuk δ=0.0.



Gambar 8. Hasil Penjadwalan dari Tabel 3, d=0.8



Gambar 7. Hasil Penjadwalan Setelah Penambahan Job, d=0.8

6. KESIMPULAN

Makalah ini membahas tentang penerapan algoritma genetik pada penjadwalan job shop dinamis non deterministik. Pengujian yang dilakukan menunjukkan bahwa algoritma genetik dapat dipergunakan untuk menyusun jadwal job shop dan dapat menambahkan beberapa job pada jadwal yang sudah ada. Penggunaan parameter δ membuat penjadwalan yang dihasilkan dapat lebih bervariasi. Parameter δ ini merupakan batasan lamanya waktu mesin dapat menganggur (*idle time*). Semakin kecil δ berarti semakin singkat mesin boleh menganggur, dan akibatnya semakin ketat jadwal yang dihasilkan.

DAFTAR PUSTAKA

Bierwirth, C., D.C. Mattfield, 1999. "Production Scheduling and Rescheduling with Genetic Algorithm", *Evolutionary Computation*, 7 (1), 1999, 1-17.

Dimyati, T.T, dkk, 1999. "Model Optimasi untuk Integrasi Alokasi Produksi dengan Penjadwalan Operasi Job Shop dan Perencanaan Kapasitas", *Jurnal Teknik dan Manajemen Industri*, 19 (1), April 1999, 17-28.

Fang, Hsiao-Lan, 1994. "Genetic Algorithms in Timetabling and Scheduling", *Ph.D. Dissertation*, Department of Artificial Intelligence, University of Edinburgh.

Husbands, P., *Genetic Algorithms for Scheduling*, School of Cognitive and Computing Sciences, University of Sussex.

Koza, J., 2001. *Genetic Algorithm*, http://cs.felk.cvut.cz/~xobitko/ga/ intro.html [8 November].

Lin, S., Goodman, E., Punch, W., 1997. *A Genetic algorithm approach to dynamic job shop scheduling problems*, dalam Back, T., editor, Proceedings of the Seventh International Conference on Genetic Algorithms, 481 – 489, Morgan Kaufmann.

Manderick, B., 1991. "Selectionism as a Basis of Categorization and Adaptive Behavior", *PhD. Dissertation*, Faculty of Sciences, Vrije Universiteit Brussel.

Saputro, N., Oktober 2003. "Pengenalan Huruf dengan memakai Algoritma Genetik", *Jurnal Integral*, Volume 8, no. 2.