Jurnal Ilmiah KURSOR
Menuju Solusi Teknologi Informasi
Vol. 4, No. 2, 2008

ISSN 0216 - 0544

PENYELESAIAN MASALAH PENJADWALAN *FLOWSHOP* FLEKSIBEL 3 TAHAP UNTUK N-*JOB* DENGAN ALGORITMA *OUTER* DAN *INNER GAME*

Ahmad Saikhu
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Gedung Teknik Informatika, Kampus ITS, Jl. Raya ITS, Sukolilo, Surabaya – 60111
email : saikhu@its-sby.edu

Abstract

Flowshop scheduling is production process scheduling of n-job that has equal sequence of production process. There are many techniques to solve this problem, the one is outer and inner game algorithm. This Algorithm give new alternative to solve flowshop scheduling problem. In this research is developed an application system that able to schedule n-job 3-stage flexible flowshop. The flowshop are flexible in the sense that a job can be processed by any of the identical machines at each stage. All of scheduling process are integration of providing outer and inner game algorithm, where outer game consist of job that will be schedule at the system, and inner game consist of job will be reschedule at the system. The result of the test with various processing time show that minimum makespan is depend on a number of job, variation coefficient and range value of processing time. The resulf of scheduling will be optimize if the processing time is homogen, that is the variation coeficient lower than 1.

Keywords: *scheduling, flexible flowshop, outer and inner game algorithm*

1. Pendahuluan

Sekarang ini penjadwalan memegang peranan yang sangat penting dalam perencanaan sistem manufaktur. Sejalan dengan perkembangan ilmu dan teknologi, metode penjadwalan berkembang secara dinamis dan lebih kompleks. Pada sistem manufaktur, masalah penjadwalan akan semakin rumit ketika kejadian yang tidak dapat diramalkan terjadi pada sistem, karenanya penyelesaian masalah dengan algoritma *outer* dan *inner game* memberikan alternatif baru dalam menangani masalah kompleksitas tersebut.

Pada sistem yang masih tradisional, pemusatan rencana penjadwalan, serta mekanisme kontrol pada manufaktur tidak cukup fleksibel untuk menanggapi perubahan produksi serta mengikuti keinginan pasar yang selalu dinamis. Hal tersebut merupakan dampak kelemahan perencanaan serta adanya peningkatan respon yang tidak dapat diprediksi.

Pada penelitian ini, metodologi penjadwalan manufaktur dengan algoritma *outer* dan *inner game* digunakan untuk menyelesaikan permasalahan penjadwalan *flowshop* fleksibel 3 tahap untuk n-*job*. Permasalahan penjadwalan *flowshop* fleksibel 3 tahap untuk n-*job* pernah dipecahkan oleh Soewandi dan Elmaghraby (2001), tetapi solusi yang dikembangkan hanya digunakan untuk menyelesaikan permasalahan yang spesifik (*problem dependent*). Metodologi penjadwalan yang digunakan untuk penelitian ini lebih bersifat umum.

Kunci pokok permasalahan penjadwalan dengan algoritma *outer* dan *inner game* adalah kerjasama dan koordinasi antar *job*. Untuk koordinasi antar *job*, pengembangan *classical tool* pada teori *game* digunakan untuk memecahkan permasalahan, dengan *job* membentuk penggabungan untuk memaksimalkan hasil kerjasama tersebut. *Power index* dari tiap *job* dihitung dengan nilai Shapley (1953) yang menghubungkan *job* dengan waktu penyelesaian penjadwalan pada sistem yang dicapai oleh *job*. Secara keseluruhan proses penjadwalan merupakan integrasi dari penetapan *outer* serta *inner game*, dengan *outer game* berada di antara *job* yang memenuhi syarat untuk dijadual pada sistem, sedangkan *inner game* berada di antara *job* yang memenuhi syarat untuk dijadual ulang pada sistem.

Flowshop yang digunakan pada Penelitian ini merupakan *flowshop* fleksibel, dikatakan fleksibel dalam pengertian bahwa sebuah *job* dapat diproses pada mesin mana saja yang identik di tiap

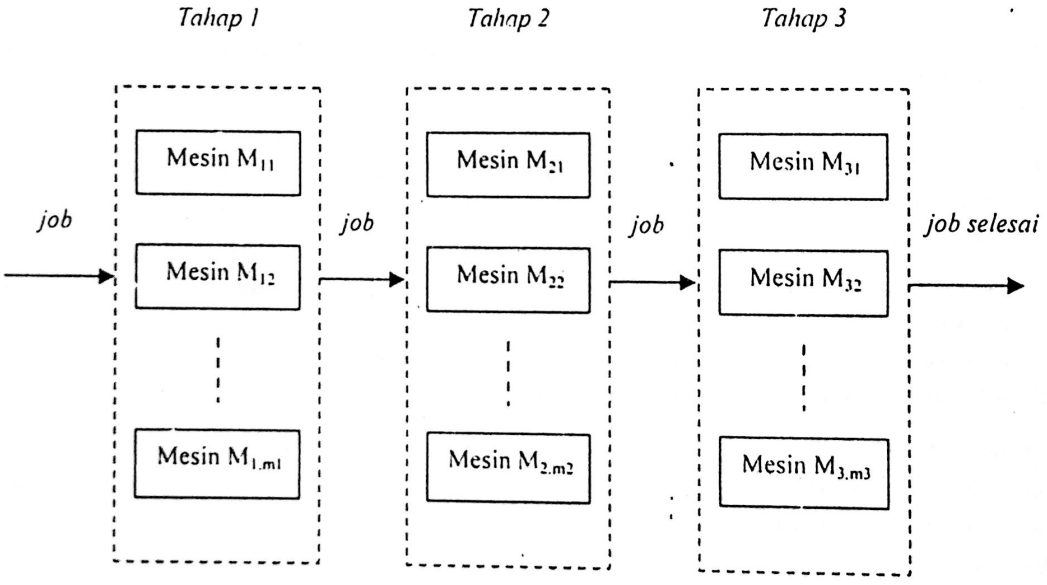
tahapnya. Tujuannya untuk menjadual kumpulan n-*job* untuk meminimalisasi *makespan* (C_{max}). *Makespan* atau *maximum completion time* merupakan waktu yang dibutuhkan untuk memproses seluruh *job*. *Minimum makespan* merupakan tujuan pada permasalahan optimasi di dalam *flowshop*. Optimalisasi *makespan* menjamin pengolahan sumber produksi secara efektif dan efisien.

2. *Flowshop* Fleksibel

Flowshop adalah proses penjadwalan produksi dari n-*job* yang mempunyai urutan proses produksi yang sama. Masalah yang dibahas dalam *flowshop* yaitu menjadualkan proses produksi masing-masing n-*job* yang mempunyai urutan proses produksi yang sama, urutan proses produksi akan selalu melewati tahap 1, lalu menuju tahap 2, dan terakhir ke tahap 3. Tiap tahap dalam *flowshop* menampilkan proses yang berbeda-beda sehingga mengakibatkan adanya perbedaan waktu produksi untuk masing-masing *job*. Pada kasus ini *flowshop* yang digunakan adalah *flowshop* fleksibel, yaitu tiap tahap pada *flowshop* memiliki sejumlah mesin yang identik, sehingga sebuah *job* dapat diproses pada mesin mana saja di tiap tahapnya.

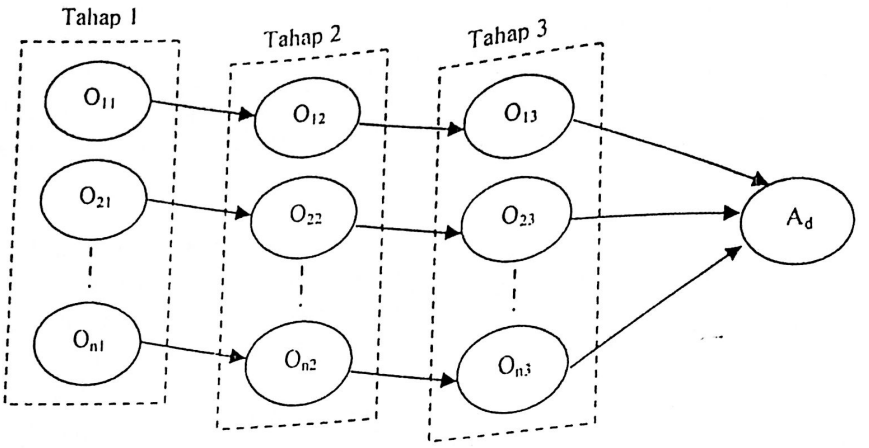
3. Masalah Penjadwalan

Pada masalah penjadwalan *flowshop* fleksibel 3 tahap untuk n-*job*, terdapat m_i mesin yang identik pada tahap i, dengan $i = 1,2,3$ dan n merupakan kumpulan *job* yang ada untuk diproses pada setiap tahap. *Job* pertama diproses pada tahap 1. Pada proses ini dilakukan operasi penyesuaian dan sesudahnya *job* akan dikirimkan pada tahap 2. Pada tahap ini dilakukan operasi penyelesaian, dan sesudahnya *job* dikirimkan pada tahap 3 untuk operasi final. Pada *flowshop* fleksibel, *job* diproses tanpa *pre-emption*, serta tersedianya kapasitas *buffer* yang tidak terbatas untuk menunggu *job* di antara tahapan produksi. Struktur sistem manufaktur 3 tahap, digambarkan pada Gambar 1.



Gambar 1. Struktur Sistem Manufaktur 3 Tahap

Apabila node melambangkan sebuah operasi dan panah melambangkan sebuah hubungan, maka *job* i dapat digambarkan dengan *digraph*, dengan $O_{ik}, i = 1,2,..., n; k = 1,2,3$, seperti tampak pada Gambar 2. Seluruh kumpulan n-*job* dapat digambarkan dengan *superimposed digraph*. *Superimposed digraph* dapat dibangkitkan dengan menghubungkan *final node* (operasi) *job* ke *dummy node* A_d dengan waktu proses sama dengan 0. Penggambaran seluruh tugas penjadwalan membantu menginisialisasi kumpulan *job* yang dapat dijadual. *job* yang tidak dapat dijadual, dan *job* yang telah dijadual pada tahap inisialisasi proses penjadwalan, serta untuk memperbarui kumpulan *job* pada tiap tahapnya.



Gambar 2. Representasi n-*Job* oleh *Digraph* Kompleks

4. Sistem *Flowshop* Fleksibel 3 Tahap

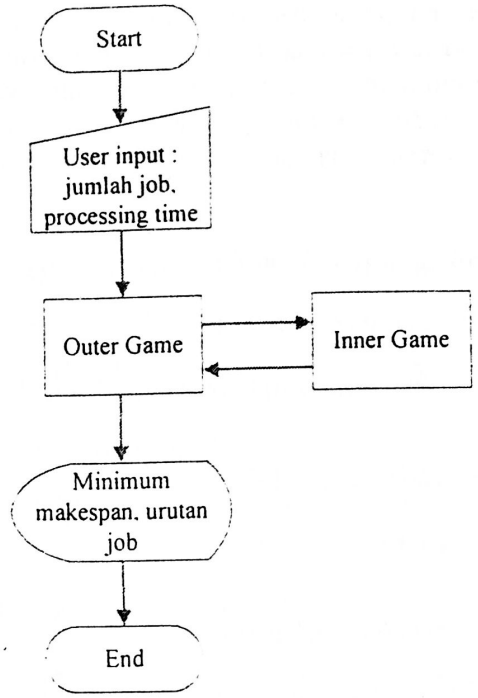
Flowshop fleksibel 3 tahap memiliki solusi penyelesaian masalah dengan algoritma *outer* dan *inner game*. Algoritma tersebut dikembangkan berdasarkan analisa teori *game*, yaitu terdapat *player*. *player* yang bekerja untuk mencapai hasil yang diinginkan. Pada permasalahan ini *player* adalah *job* yang akan dijadual.

Inputan berupa jumlah *job* dan *processing time* untuk masing-masing *job* pada tiap tahap, data-data itu kemudian akan diproses di dalam *outer game*. *Outer game* terdiri dari kumpulan *job* yang akan dijadual (SAJ), setelah masing-masing *job* diketahui nilainya melalui teknik *First In First Served*, maka langkah selanjutnya adalah nilai *power index* dari masing-masing *job* dihitung menggunakan teori Shapley, hal ini diperlukan untuk dapat mengetahui nilai persentase tiap *job* untuk kemudian dipilih yang terbesar. Persentase *power index* terbesar adalah *job* pemenang, yaitu kandidat *job* yang paling layak untuk masuk ke dalam *inner game*.

Di dalam *inner game* terdapat kumpulan *job* yang akan melakukan penjadwalan ulang. Masing-masing *job* melakukan penjadwalan ulang, yaitu *job* terpilih akan memperoleh prioritas paling rendah dalam penjadwalan gabungan dengan *job* lainnya. Hal ini dilakukan agar *job* terpilih tidak merubah pengerjaan yang telah dilakukan oleh *job* lain. Penjadwalan gabungan memiliki teknik *First In First Served*, teknik ini memberikan prioritas pengerjaan paling tinggi terhadap *job* yang paling awal memasuki sistem. Rangkaian urutan *job* yang paling awal memasuki sistem mengikuti keputusan *inner game* yang sudah ditetapkan sebelumnya.

Setelah seluruh *job* melakukan penjadwalan ulang, maka rangkaian urutan penjadwalan ulang yang memiliki *makespan* paling minimum akan ditetapkan sebagai keputusan *inner game*. Keputusan rangkaian urutan *job* ini disimpan, dan akan berubah setiap terdapat *job* baru yang masuk ke dalam *inner game*.

Apabila keputusan *inner game* telah ditetapkan, maka proses akan kembali menuju *outer game* untuk mencari *job* pemenang. Keadaan ini akan berlangsung secara iteratif sampai seluruh *job* masuk ke dalam *inner game*, yang artinya seluruh *job* sudah selesai dijadual. Pada akhir proses akan tampak minimum *makespan* yang dihasilkan serta rangkaian urutan *job* yang telah ditetapkan oleh *inner game*. Agar lebih jelas, seluruh proses dalam *flowshop* fleksibel digambarkan pada Gambar 3.



Gambar 3. Proses pada *Flowshop* Fleksibel

5. *Outer Game*

Outer Game terdiri dari *job-job* yang akan dijadual (SAJ), di sini masing-masing *job* menjadual individual *job* sehingga diperoleh *value* untuk kemudian dihitung menggunakan perhitungan Shapley. Perhitungan Shapley diperlukan untuk memperoleh nilai *power index* masing-masing *job*. Sebuah *job* dengan nilai persentase *power index* terbesar, dalam arti hal tersebut mengindikasikan *job* dengan persentase perubahan tertinggi pada penjadwalan saat itu ditetapkan sebagai *job* pemenang.

Pada tahap inisialisasi *outer game*, kumpulan *job* yang telah dijadual (SJ) masih kosong. Untuk memulai pemrosesan, pada tahap pertama penjadwalan, *job* yang menang adalah *job* yang 'paling penting' untuk dijadual, hal itu berdasar pada lama waktu sebuah *job* harus selesai dikerjakan. Setelah *job* pertama memasuki *inner game*, *job* dari kumpulan kandidat *job* yang mungkin, saling bersaing untuk membentuk sebuah koalisi dengan *job* yang berada pada *inner game*. Proses ini dinamakan proses estimasi. *Job* yang menang adalah *job* yang memiliki *power index* dengan persentase terbesar.

Power index untuk *job* j dihitung dengan Persamaan 1.

$$\phi_j(v) = \frac{(2-1)!(2-2)!}{2!}[v(S \cup \{j\}) - v(S)] + \frac{(1-1)!(2-1)!}{2!}[v(\{j\}) - v(\varnothing)]$$
$$= \frac{1}{2}[v(S \cup \{j\}) - v(S) + v(\{j\})] \quad (1)$$

Keterangan :

$\phi_j(v)$ = *power index* untuk *job j*
$v(S \cup \{j\})$ = nilai penjadwalan S dan *job j*
$v(S)$ = nilai S
$v(\{j\})$ = nilai *job j*

S merupakan kumpulan *job* yang sudah ditetapkan sebelumnya.

Persentase kontribusi dari *job* j pada penjadwalan, dihitung dengan rumus pada Persamaan 2. Yang menunjukkan rasio hasil individual, $\phi_j(v)$, dari hasil *grand* koalisi, $v(S \cup \{j\})$.

$$ShV_j = \frac{\phi_j(v)}{\phi_1(v) + \phi_2(v)} = \frac{\phi_j(v)}{v(S \cup \{j\})} \quad (2)$$

Keterangan :

ShV_j = persentase *power index*

Keterangan lainnya mengikuti rumus sebelumnya.

Job yang memperoleh persentase kontribusi terbesar pada tahap estimasi, memasuki *inner game*. Sekali sebuah *job* dari *outer game* memasuki *inner game*, tahap penjadwalan *outer game* ditambah 1, kumpulan kandidat *job* yang mungkin dijadual (SAJ), *job* yang tidak mungkin dijadual (NSJ), serta *job* yang telah dijadual (SJ), diperbarui. Lalu persaingan di antara *job* untuk memasuki *inner game* dimulai lagi. *Pseudo-code* algoritma pada *outer game* ditampilkan sebagai berikut.

Prosedur *Outer Game*

Langkah 1. *Input* : struktur sistem manufaktur dan struktur individual *job*.

Langkah 2. dekomposisi dan menginisialisasi $t = 0, SAJ_t, SJ_t, NSJ_t$.

Langkah 3. untuk $\forall i \in SAJ_t$, jadual individual *job* $(v(i) = C_{\max}, i)$ dengan teknik *First In First Served*.

Langkah 4. hitung $k = index\left(\max_{i \in SAJ_t} \{v(i)\}\right)$

Langkah 6. *update* $t = t + 1, SAJ_t, SJ_t, NSJ_t$.

Langkah 5. *job* a_k memasuki *inner game*.

Langkah 7. jika $(SAJ_t = \varnothing)$ maka lanjutkan ke Langkah.8 lainnya

untuk $\forall i \in SAJ_t$

hitung : $\phi_i(v) = \frac{1}{2}[v(SJ_t \cup \{i\}) - v(SJ_t) + v(\{i\})]$ dan

$$ShV_i = \frac{\phi_i(v)}{v(SJ_t \cup \{i\})}$$

$$k = index\left(\max_{i \in SAJ_t} \{ShV_i\}\right)$$

Job a_k memasuki *inner game*.

Update: $t = t + 1, SAJ_t, SJ_t, NSJ_t$.

Lanjutkan ke *inner game* $(S = SJ_t, v(S) = Schedule(SJ_t))$ Lanjutkan ke Langkah.7

Langkah 8. akhir dari *outer game*. *Output* keseluruhan penjadwalan.

Keterangan notasi mengikuti rumus sebelumnya.

6. *Inner Game*

Setelah sebuah *job* memasuki *inner game*, *job* tersebut menjadi pemain di dalam *inner game*. *Inner game* menangani masalah kerjasama untuk mencapai kemungkinan terbaik, yaitu *minimum makespan*. Tujuan kerjasama ini adalah untuk memperbaiki keseluruhan penjadwalan.

Setelah menentukan kebutuhan untuk kerjasama, proses diatur sebagai berikut. Tiap *job* pada *inner game* beranggapan melakukan penjadwalan ulang tanpa merubah tugas lain yang telah dilakukan oleh *job* lain pada *game*. Rangkaian urutan *job* yang menghasilkan *minimum makespan* merupakan keputusan yang dihasilkan oleh *inner game*, untuk kemudian digunakan pada proses perhitungan *inner game* selanjutnya. *Makespan* dengan rangkaian urutan *job* tersebut dihitung menggunakan teknik *First In First Served*.

Teknik ini memberikan prioritas besar kepada *job* yang paling awal memasuki sistem, dan prioritas paling kecil kepada *job* yang melakukan penjadwalan ulang. Rangkaian urutan *job* yang paling awal memasuki sistem mengikuti keputusan *inner game* yang sudah ditetapkan sebelumnya.

Setelah diperoleh rangkaian optimal, maka *inner game* mencapai tahap *stabilization*. Setelah tahap itu terjadi, *outer game* memulai proses lagi dan *job* baru memasuki *inner game*. *Pseudo-code* dari algoritma untuk mengimplementasikan *inner game* ditampilkan di bawah ini.

Prosedur *Inner Game*

Langkah 1. *input* : koalisi $S \subseteq N$

Langkah 2. untuk $\forall i \in S$

hitung : $v(S',) = v((S - \{i\}) \cup \{i^*\})$;

(i^* adalah stategi baru dari *job* a_i dengan menjadikannya sebagai prioritas terakhir)

Langkah 3. $k = index(\min_{i \in S} \{v(S'i)\})$.

Langkah 4. *job* a_k menjadual ulang *job* k.

Langkah 5. *inner game* telah mencapai tahap *stabilization*.

Keterangan :

koalisi $S \subseteq N$ = gabungan *job* yang masuk dalam SJ.

$v(S',)$ = nilai S yang baru.

Keterangan lainnya mengikuti rumus sebelumnya.

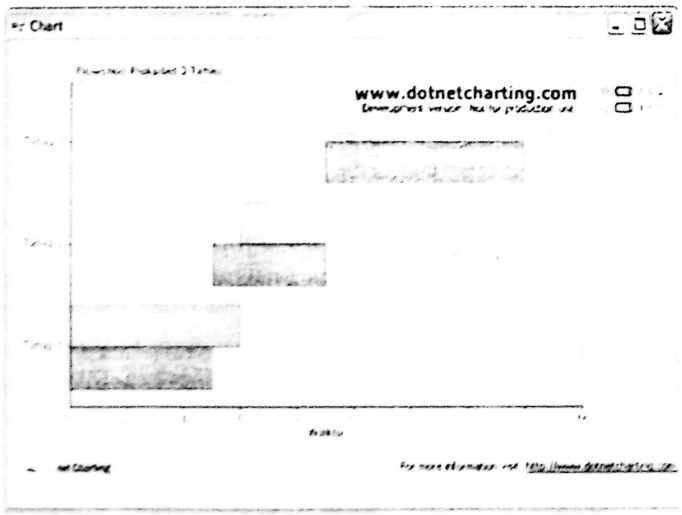
Saat *inner game* mencapai tahap *stabilization*, selanjutnya rangkaian urutan *job* disimpan untuk kemudian dilanjutkan pada proses *outer game*. Proses ini berjalan secara iteratif dan bersambung sampai *job* menemukan bahwa tidak diperlukan lagi kerjasama dan karena itu tidak diperlukan lagi penjadwalan ulang. Kondisi tersebut dicapai apabila seluruh *job* sudah masuk pada *scheduled jobs* (SJ). yaitu kumpulan *job* yang telah dijadual. Keseluruhan proses tersebut menghasilkan rangkaian akhir urutan *job, yang* pada rangkaian tersebut diperoleh *makespan* paling minimum.

7. Hasil Uji Coba

7.1. Uji Coba Skenario 1

Pengujian skenario 1 merupakan pengujian terhadap *flowshop* fleksibel 3 tahap dengan jumlah *job* sama dengan jumlah minimal mesin yang berada pada sistem. Pada sistem ini jumlah minimal mesin adalah 2 mesin. Karenanya jumlah *job* sama dengan 2, $n = 2$.

Minimum makespan yang dihasilkan adalah = 17 dengan urutan *job* 2 lalu 1. Hasil *chart* untuk skenario ini dapat dilihat pada Gambar 4.



Gambar 4. *Chart* Perhitungan Skenario 1

Tabel 1. *Processing Time* Skenario 1

Job 1	*Processing time* pada tahap 1	6
	Processing time pada tahap 2	3
	Processing time pada tahap 3	8
Job 2	*Processing time* pada tahap 1	5
	Processing time pada tahap 2	4
	Processing time pada tahap 3	7

7.2. Uji Coba Skenario 2

Untuk uji coba selanjutnya digunakan koefisien variasi, dan *range processing time* sebagai pengukur keoptimalan *minimum makespan* yang dihasilkan oleh *software* ini. Koefisien variasi merupakan standar deviasi dari processing time dibagi dengan rata-ratanya. Rumus standar deviasi tampak pada Persamaan 3.

$$\partial = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \chi)} \qquad (3)$$

Keterangan :

∂ = standar deviasi

N = jumlah data yang dihitung

x_i = data ke i

χ = rata-rata dari seluruh data

Tabel 2. Percobaan skenario 2

Jumlah *job*	Koefisien variasi	*Range*	*Minimum makespan*	*First In First Served*
2	0,75	0-5	12	12
2	0,51	0-100	142	142
3	1,07	0-50	61	61
3	0,67	0-100	171	195
4	1,25	0-30	47	47
4	0,94	0-100	142	173
5	1,01	0-30	49	44
5	1,06	0-30	51	49
6	1,03	0-30	48	73
6	1,05	0-30	40	39
7	1,01	0-30	74	77
7	1,13	0-30	40	39
8	0.75	0-10	17	19
8	1,04	0-25	52	51
9	1,01	0-35	56	76
9	1,11	0-25	52	51
10	0,54	0-10	37	41
10	1,18	0-25	52	51

Nilai dari koefisien variasi adalah standar deviasi dibagi dengan rata-rata *processing time* yang dihitung. Semakin besar nilai koefisien variasi dari suatu kasus, maka data *processing time* yang ada semakin beragam. Pada uji coba ini diberikan berbagai macam kasus dengan nilai koefisien variasi dan *range* yang berbeda-beda.

Sebagai pembanding keoptimalan *minimum makespan* yang dihasilkan, maka kumpulan *job-job* akan dijadual menggunakan teknik *First In First Served*, dengan *job* akan dijadual berdasarkan urutan secara *ascending*. Bila *minimum makespan* yang dihasilkan oleh *software* ini lebih kecil atau sama dengan *makespan* yang dihasilkan dengan teknik *First In First Served*, maka diputuskan bahwa sistem

bekerja optimal. Tetapi apabila *minimum makespan* yang dihasikan lebih besar dari teknik penjadwalan *First In First Served*, maka diputuskan bahwa sistem bekerja tidak optimal.

Pada percobaan berikutnya, telah dibangkitkan bilangan acak untuk *processing time* masing-masing *job* sehingga diperoleh hasil seperti pada Tabel 2.

8. Kesimpulan Dan Saran

Dalam penyelesaian masalah penjadwalan *flowshop* fleksibel 3 tahap dengan algoritma *outer* dan *inner game*, kunci pemecahan masalah ada pada kerjasama dan koordinasi tiap *job*. Untuk kerjasama dan koordinasi, diterapkan pengembangan teori *game* sebagai solusinya, dengan hasil akhirnya akan diperoleh *minimum makespan*.

Berikut adalah kesimpulan terhadap hasil dan analisis dalam membangun sistem aplikasi :

1. Modul *flowshop* fleksibel 3 tahap telah berhasil diimplementasikan seperti yang dijelaskan pada Bab III, yaitu penyelesaian masalah penjadwalan sekumpulan *job* yang memiliki *processing time* berbeda sehingga diperoleh minimum *makespan*.
2. *Minimum makespan* yang dihasilkan oleh sistem dipengaruhi oleh beberapa hal, antara lain : jumlah *job*, besar koefisien variasi *processing time*, dan lebar *range* data *processing time*.
3. Semakin banyak jumlah *job* atau semakin besar koefisien variasi dan *range processing time* seluruh *job*, maka akan menambah kerumitan perhitungan sehingga menghasilkan *makespan* yang kurang optimal.
4. Untuk penjadwalan dengan jumlah *job* 2, 3, dan 4 nilai minimum *makespan* yang dihasilkan akan selalu optimal.
5. Pada penjadwalan dengan jumlah *job* 5, 6, 7, 8, 9, dan 10 hasil minimum *makespan* akan optimal apabila koefisien variasi dari *processing time* lebih kecil dari 1 dan *range* tidak lebih besar dari 25.

Mengingat penjadwalan *flowshop* fleksibel 3 tahap untuk n-*job* $(F3\|C_{max})$ masuk ke dalam kelompok NP-hard, karenanya membangun algoritma yang menghasilkan solusi yang mendekati optimal menjadi persoalan yang banyak dibahas. Dengan adanya solusi ini dapat dijadikan langkah penting dalam pengembangan penyelesaian masalah penjadwalan di dalam sistem manufaktur. Saran untuk pengembangan selanjutnya antara lain :

1. Pengembangan algorima yang berbeda pada solusi penjadwalan, sehingga menghasilkan *makespan* yang lebih mendekati optimal.
2. Ukuran optimal dapat diperbaiki dengan menambahkan beberapa aspek perhitungan, seperti rata-rata operasi mesin terkecil (*shortest mean flow time*), maupun *minimum idle time*. Tentunya seluruh aspek tersebut harus sesuai dengan perhitungan riset yang sudah diteliti.

Daftar Pustaka

[1] Babayan, Astghik, David He, *Agent Based Agile Manufacturing System Scheduling*. Department of Mechanical and Industrial Engineering, The University of Illinois Chicago, 2004.

[2] Babayan, Astghik, David He, *Solving the n-Job 3-Stage Flexible Flowshop Scheduling Problem Using An Agent-Based Approach*. Int. J. Prod. Res., 2004, Vol. 42, No. 4, 777–799, Taylor & Francis Ltd, 2004.

[3] Babayan, *Scheduling Manufacturing Systems with Agents: A Game Theory Based Negotiation Approach*. Department of Mathematics and Engineering, Harold Washington College, 2005.

[4] Farber, Gerrit, Anna, *Overview on Sequencing in Mixed Model Flowshop Production Line with Static and Dynamic Context*, IOC-DT-P-2005-7, Universitat Politecnica de Catalunya (UPC), 2005,

[5] Jungwattanakita, J., Reodechaa, M., Chaovalitwongsea, P., and Wernerb, F., *An Evaluation of Sequencing Heuristics for Flexible Flowshop Scheduling Problems with Unrelated Parallel Machines and Dual Criteria*. Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University, Thailand, 2005.

[6] Lemaire, Jean. *Cooperative Game Theory and its Insurance Applications*. Department of Insurance and Risk Management, University of Pennsylvania, USA.

[7] Nugrahanto, Sony, *Penerapan Algoritma Genetik pada Penjadwalan Flowshop*. Institut Teknologi Sepuluh Nopember, Surabaya, 2005.