PERBANDINGAN KINERJA ALGORITMA GENETIKA DAN *SIMULATED ANNEALING* UNTUK MASALAH *MULTIPLE OBJECTIVE* PADA PENJADWALAN *FLOWSHOP*

I Gede Agus Widyadana
Dosen Fakultas Teknologi Industri, Jurusan Teknik Industri - Universitas Kristen Petra

Andree Pamungkas
Alumnus Fakultas Teknologi Industri, Jurusan Teknik Industri - Universitas Kristen Petra

ABSTRAK

Penelitian ini difokuskan pada pembandingan algoritma Genetika dan *Simulated Annealing* ditinjau dari aspek performa dan waktu proses. Tujuannya adalah untuk melihat kemampuan dua algoritma tersebut untuk menyelesaikan problem-problem penjadwalan *flow shop* dengan kriteria minimasi *makespan* dan *total flowtime*.

Kemampuan kedua algoritma tersebut dilihat dengan melakukan simulasi yang dilakukan pada kombinasi-kombinasi *job* dan mesin yang berbeda-beda. Hasil simulasi menunjukan algoritma *Simulated Annealing* lebih unggul dari algoritma Genetika hingga 90%, algoritma Genetika hanya unggul pada waktu proses saja, namun dengan tren waktu proses yang terbentuk, diyakini pada problem dengan kombinasi *job* dan mesin yang banyak, algoritma *Simulated Annealing* dapat lebih cepat daripada algoritma Genetika.

Kata kunci: Algoritma Genetika, *Simulated Annealing, flow shop, makespan, total flowtime*.

ABSTRACT

The research is focused on comparing Genetics algorithm and Simulated Annealing in the term of performa and processing time. The main purpose is to find out performance both of the algorithm to solve minimizing makespan and total flowtime in a particular flowshop system.

Performances of the algorithms are found by simulating problems with variation of jobs and machines combination. The result show the Simulated Annealing is much better than the Genetics up to 90%. The Genetics, however, only had score in processing time, but the trend that plotted suggest that in problems with lots of jobs and lots of machines, the Simulated Annealing will run much faster than the Genetics.

Keywords: *Genetics Algorithm, Simulated Annealing, flow shop, makespan, total flowtime.*

1. PENDAHULUAN

Penjadwalan bagi perusahaan manufaktur adalah aspek yang sangat penting, karena penjadwalan merupakan salah satu elemen perencanaan dan pengendalian produksi. Jadwal yang baik diperoleh dari algoritma yang baik pula, sehingga perlu untuk menentukan suatu algoritma yang tepat untuk suatu masalah penjadwalan.

Penelitian-penelitian sebelumnya dalam hal penjadwalan biasanya difokuskan pada satu kriteria saja sedangkan pada kenyataannya tidak selalu menggunakan satu kriteria,

oleh karena itu penelitian ini menggunakan dua criteria dalam penjadwalannya yaitu kriteria *Makespan* dan *Total flowtime.* Algoritma *Metaheuristic* telah menjadi satu obyek penelitian yang mendalam disebabkan kemampuan algoritma ini dalam mendapatkan solusi yang mendekati global optimal dengan waktu komputasi yang relatif singkat. Kesulitannya adalah sering tidak adanya acuan untuk melihat kemampuan algoritma ini. Demikian juga tidak adanya acuan perbandingan kemampuan algoritma *Genetic* dan *Simulated Annealing.* Oleh karena itu penelitian ini berusaha melihat kemampuan keduanya dalam menyelesaikan masalah penjadwalan *flow shop* dengan kriteria *makespan* dan *total flowtime.*

Ruang lingkup penelitian adalah sebagai berikut:

- Algoritma yang diperbandingkan kinerjanya adalah Algoritma Genetika dan *Simulated Annealing* sebagai dua algoritma metaheuristik yang banyak dianjurkan oleh jurnal-jurnal industri.
- Pembuatan desain dan program komputer yang dilakukan hanya menggunakan Algoritma Genetika dan *Simulated Annealing* saja.
- Sistem produksi adalah *flow shop*
- Tidak terdapat *pre-emption.*
- Setiap job memiliki *ready time* yang sama.

2. *SIMULATED ANNEALING*

Simulated Annealing dikembangkan berdasarkan ide dari mekanisme perilaku pendinginan dan proses kristalisasi (*annealing)* material panas. Algoritma ini melakukan peningkatan iteratif untuk memperbaiki solusi yang dihasilkan teknik-teknik penjadwalan heuristik, dalam hal ini adalah sebuah solusi awal yang dibuat dengan teknik heuristik ataupun random, diiterasi secara berulang dengan metode *annealing* dengan menggunakan perturbasi lokal hingga tidak ada peningkatan lagi atau hingga jumlah iterasi yang diinginkan sudah dicapai.

Struktur algoritma *simulated annealing* secara umum adalah sebagai berikut (Ponnambalan *et al.* 1999) :

1. Cari solusi awal *S* menggunakan parameter awal dan metode heuristik awal yang dapat ditentukan sendiri.
2. Tetapkan suatu nilai temperatur awal *T* yang cukup tinggi, dimana *T>0*
3. Pada keadaan tidak *frozen*, lakukan:
   a. Lakukan *L* kali :
      i. Cari solusi *neighbourhood S'* dari *S* menggunakan metode yang dapat ditetapkan sendiri.
      ii. Ä = Nilai objektif (*S'*) – Nilai objektif (*S*)
      iii. Jika Ä<0, maka tetapkan *S=S'*, jika tidak maka tetapkan *S=S'* dengan probabilitas *exp(-Ä/T)*
   b. *T = r x T*, dimana *r* adalah faktor reduksi suhu.
4. Dapatkan solusi optimal.

   Parameter awal yang diperlukan:
   - Temperatur Awal, merupakan penanda awal iterasi. Dimana nanti temperatur awal ini akan terus berkurang hingga mencapai temperatur akhir.

- Temperatur Akhir, merupakan batas akhir penanda iterasi sudah dapat dihentikan.
- Faktor Reduksi Suhu, merupakan angka yang digunakan untuk menurunkan suhu secara bertahap dan terkendali.
- Angka replikasi, merupakan angka yang menunjukkan berapa kali *loop* dalam harus dilakukan sebelum menurunkan suhu.

Langkah sistematis dari algoritma *SA* yang digunakan:

1. Solusi awal didapatkan dari penjadwalan heuristik *SPT* (*Shortest Processing Time)*, dihitung nilai objektifnya yaitu *Ms* dan *F*.
2. Menentukan parameter-parameter awal yang dibutuhkan.
3. Mengambil dua angka random r_1 dan r_2, antara 1 dan jumlah job, dimana r_1 tidak sama dengan r_2.
4. Menukarkan job pada posisi r_1 dengan job pada posisi r_2.
5. Menghitung nilai objektif jadwal baru ini, Ms_{new} dan F_{new}.
6. Menghitung dengan rumusan sebagai berikut:

$$e_1 = w_1\left(\frac{(Ms - \min(Ms; Ms_{new}))}{\min(Ms; Ms_{new})}\right) + w_2\left(\frac{(F - \min(F; F_{new}))}{\min(F; F_{new})}\right)$$

$$e_2 = w_1\left(\frac{(Ms_{new} - \min(Ms; Ms_{new}))}{\min(Ms; Ms_{new})}\right) + w_2\left(\frac{(F_{new} - \min(F; F_{new}))}{\min(F; F_{new})}\right)$$

$$\Delta = e_2 - e_1$$

7. Jika >*0*, maka hitung $p = \exp\left(-\frac{\Delta}{T_0}\right)$. Ambil satu bilangan random *pm* antara 0 dan 1, jika *pm*<*p* maka jadwal baru diterima, jika tidak jadwal lama tetap dipertahankan. Jika <*0* maka jadwal baru langsung diterima. Secara umum memiliki pengertian selisih keunggulan jadwal baru terhadap nilai minimum yang dihasilkan dengan jadwal lama terhadap nilai minimum yang dihasilkan.
8. Apabila kondisi replikasi lokal telah dipenuhi, maka lakukan reduksi terhadap suhu dengan rumusan $T_i = r \times T_{i-1}$.
9. Kembali ke langkah tiga sampai kriteria penghentian iterasi terpenuhi.

3. ALGORITMA GENETIKA

Algoritma Genetika adalah algoritma pencarian solusi tetangga berdasarkan mekanisme seleksi alam dan Genetika alam. Algoritma Genetika dapat mengacu pada semua metode pencarian solusi tetangga dengan mensimulasikan proses evolusi alam. Pada tiap generasi, solusi terbaik (individu) diperbolehkan menghasilkan solusi baru (anak) dengan mengambil fitur terbaik dari *parent* dan mencampurkan dengan fitur lainnya (atau dengan mutasi). Individu terburuk mati untuk menjaga kestabilan populasi.

Algoritma ini berbeda dari algortima tradisional pada beberapa hal (Goldberg, 1989) yaitu:

1. GA berkeja dalam bentuk kode sekumpulan parameter, bukan parameter itu sendiri.
2. GA mencari solusi dari sekumpulan populasi, bukan dari satu titik solusi.
3. GA langsung menggunakan informasi berupa fungsi tujuan dan bukan turunannya.
4. GA menggunakan aturan probabilitas, bukan aturan deterministik.

Berikut adalah skema umum dari algoritma Genetika (Li Dawei *et al.*, 1999):

- Menginitialisasi populasi awal P_0 (biasanya didapatkan dari algoritma heuristik yang lebih sederhana ataupun dengan random)
- Mengevaluasi populasi P_0
- Menginitialisasi generasi = 1
- Mengerjakan langkah-langkah berikut sampai kondisi tertentu:
  - Reproduksi $P_{generasi}$
  - *Crossover* $P_{generasi}$
  - Mutasi $P_{generasi}$
  - Evaluasi $P_{generasi}$
  - Generasi = generasi + 1

Parameter-parameter yang digunakan dalam skema algoritma Genetika ini adalah sebagai berikut :

- Banyak individu dalam 1 generasi, ini dapat ditentukan sendiri tanpa ada aturan tertentu. Namun pada jumlah job yang banyak sebaiknya banyak individu dalam 1 generasi juga banyak agar pencarian solusi akan lebih baik.
- Probabilitas mutasi, mutasi dalam kehidupan nyata jarang sekali terjadi dan kemungkinan terjadinya cukup kecil bila dibandingkan dengan kehidupan yang normal. Hal ini berlaku pula untuk algoritma Genetika, karena itu perlu ditentukan besar kemungkinan terjadinya mutasi pada satu individu. Biasanya digunakan probabilitas 0,01.
- Jumlah generasi, sebenarnya menunjukkan berapa kali iterasi yang dilakukan. Misalkan hendak dilakukan 10 kali iterasi, maka iterasi dihentikan apabila telah didapatkan generasi baru dari generasi ke-10.
- Bobot untuk masing-masing kepentingan *Makespan* dan *Total flowtime*. Bila tidak ditentukan, program dapat menghitungnya secara otomatis dimana akan dianggap keduanya memiliki bobot berimbang (tidak ada yang dipentingkan antara *Makespan* dan *Total flowtime*).

Langkah spesifik algoritma Genetika yang digunakan:

1. Tentukan semua parameter awal yang dibutuhkan.
2. Solusi awal didapatkan dari *SPT* (P_1), jadwal-jadwal berikutnya dalam generasi awal ini dibangun dengan metode random.
3. Untuk operator reproduksi, semua parent dalam satu generasi melakukan perkawinan dengan setiap parent lainnya. Jadi seandainya ada 8 parent akan terjadi 28 kali perkawinan yang menghasilkan 56 anak.
4. Dihitung semua fungsi objektif anak yang dihasilkan, kemudian diurutkan dari yang terbaik hingga terburuk. Sebanyak g anak dibiarkan hidup untuk menjadi generasi penerus berikutnya, sisanya mati. Besar g sesuai dengan banyak parent pada generasi awal yang telah ditentukan sebelumnya pada penentuan parameter awal. Seandainya telah ditetapkan dalam 1 populasi terdapat 8 individu, maka yang diluluskan menjadi generasi selanjutnya adalah 8 terbaik dari keseluruhan populasi baru.
5. Untuk pengawinannya, operator *crossover* yang diberlakukan adalah metode PMX (*partially matched crossover*). Metode ini dimulai dengan mengambil dua buah angka random antara 1 dan banyak job dalam *string*. Misalkan *crossover* diberlakukan pada dua jadwal A dan B berikut ini, angka random yang muncul adalah 4 dan 7, maka:

A = 9 8 4 | 5 6 7 | 1 3 2 10
B = 8 7 1 | 2 3 10 | 9 5 4 6

Dengan metode PMX ini, maka job-job dalam rentangan angka random itu akan bertukar tempat secara bersesuaian posisinya. Jadi job 5 pada jadwal *A* bertukar tempat dengan job 2, demikian pula pada jadwal *B*, job 2 bertukar tempat dengan job 5. Job 6 bertukar dengan job 3 dan job 7 bertukar dengan job 10. Sehingga hasil *crossover* akan menjadi :

A' = 9 8 4 | 2 3 10 | 1 6 5 7
B' = 8 10 1 | 5 6 7 | 9 2 4 3

A' dan *B'* ini adalah anak dari perkawinan metode crossover *A* dan *B*.

6. Untuk perhitungan nilai objektif, karena ada lebih dari dua nilai yang diperbandingkan tentu tidak bisa menggunakan operator Ä seperti pada *simulated annealing.* Maka dicari cara lain untuk melakukannya, sebagai berikut :
   - Menghitung semua *Ms* dan *F* dari anak yang dihasilkan
   - Menghitung *Msr* dan *Fr* yang merupakan nilai rata-rata dari *Ms* dan *F* untuk seluruh anak.
   - Menghitung $w_1 = \frac{Msr}{Fr}$ dan $w_2 = 1 - w_1$
   - Apabila bobot *w1* dan *w2* telah ditentukan sendiri, maka tiga langkah diatas tidak perlu dilakukan.
   - Berikutnya menghitung nilai objektif $Y_i = w_2 Ms_i + w_1 F_i$. Konsep rumusan ini adalah untuk menyeimbangkan bobot *Ms* dan *F*.
7. Setelah didapat semua nilai *Y* untuk semua anak barulah diurutkan dari yang terbaik ke yang terburuk dan diambil sebanyak jumlah individu per generasi yang ditentukan.
8. Operator mutasi diberlakukan setelah didapatkan generasi baru ini. Caranya adalah dengan mengambil satu angka random antara 0 dan 1, apabila angka ini lebih kecil atau sama dengan probabilitas mutasi yang ditentukan dari awal tadi, maka individu itu dikenai operator mutasi. Jika tidak maka individu itu lolos tanpa perubahan apapun.
9. Bila terpilih terkena operator mutasi, langkah berikutnya adalah mengambil dua angka random misal d_1 dan d_2. Kemudian job d_1 dan job d_2 pada individu tersebut bertukar posisi. Misal jadwal *A* = 3 1 2 4 5, d_1 = 3 dan d_2 = 4. Maka job 3 bertukar tempat dengan job 4 menjadi *A'* = 4 1 2 3 5. Ulangi lagi langkah 3 hingga syarat penghentian iterasi dipenuhi.
10. Jadwal terbaik adalah jadwal nomor 1 pada generasi terakhir.

4. PERBANDINGAN GENETIK DAN *SIMULATED ANNEALING* (SA)

Untuk membandingkan hasil akhir Genetika dan algoritma SA, cukup dilakukan pembandingan pada satu jadwal yaitu jadwal terbaik dari masing-masing algoritma. Pembandingannya dilakukan sebagai berikut:

$$e_1 = w_1\left(\frac{\left(Ms_{GA} - \min(\,Ms_{GA}; Ms_{SA})\right)}{\min(\,Ms_{GA}; Ms_{SA})}\right) + w_2\left(\frac{\left(F_{GA} - \min(\,F_{GA}; F_{SA})\right)}{\min(\,F_{GA}; F_{SA})}\right)$$

$$e_2 = w_1\left(\frac{\left(Ms_{SA} - \min(\,Ms_{SA}; Ms_{GA})\right)}{\min(\,Ms_{SA}; Ms_{GA})}\right) + w_2\left(\frac{\left(F_{SA} - \min(\,F_{SA}; F_{GA})\right)}{\min(\,F_{SA}; F_{GA})}\right)$$

$$\Delta = e_2 - e_1$$

Jika 0, maka SA lebih baik dari GA dan jika 0 maka GA lebih baik dari SA.

5. PERHITUNGAN PERFORMA

Performa adalah ukuran seberapa baik suatu jadwal mengungguli jadwal lainnya, cara pencarian performa adalah dari dua jadwal terbaik yang didapatkan, hitung ΣMs , Msr (Makespan rata-rata), ΣF , Fr (Total Flowtime rata-rata). Dengan menetapkan:

$$w_1 = \frac{Msr}{Fr} \qquad w_2 = 1 - w_2$$

Dengan rumus $Y_i = w_2 Ms_i + w_1 F_i$, maka dapat dihitung nilai Y untuk masing-masing algoritma. Jika GA unggul, maka

$$x = \left|\frac{Y_{GA} - Y_{SA}}{Y_{GA}}\right| \times 100\%$$

Jika SA unggul, maka:

$$x = \left|\frac{Y_{SA} - Y_{GA}}{Y_{SA}}\right| \times 100\%$$

6. SIMULASI

Simulasi dilakukan dengan aturan sebagai berikut:

- Jumlah job bervariasi dari 10 hingga 50 job dengan kenaikan 10 job per tahap. (Rajendran, 1993)
- Jumlah mesin bervariasi dari 5 hingga 30 dengan kenaikan 5 mesin per tahap. (Rajendran, 1993)
- Jumlah problem tiap kombinasi adalah 20.
- Waktu proses untuk job dirandomkan oleh komputer pada rentangan 1 hingga 99 unit waktu. (Rajendran, 1993)
- Untuk algoritma Genetika akan dilakukan iterasi 12 parent untuk 16 generasi, sehingga total akan dilakukan $_{12}C_2 \times 15 = 66 \times 15 = 990$ iterasi yang berarti 990 kali pencarian solusi *neighbourhood*.
- Untuk algoritma *simulated annealing* ditetapkan parameter awalnya adalah sebagai berikut:

- Probabilitas penerimaan pertama diharapkan akan sebesar 0.95, diharapkan adalah sebesar 0.05 yang berarti jadwal baru ditoleransi boleh diterima walaupun kalah dengan jadwal awal jika kekalahannya itu ada dikisaran angka 0.05, yaitu

  $0.95 = exp\left(-\frac{\ddot{A}}{T}\right) = exp\left(-\frac{0.05}{T}\right)$ atau

  $\ln 0.95 = -\frac{0.05}{T}$, jadi didapatkan

  $T = -\frac{0.05}{\ln 0.95} = -\frac{0.05}{0.0513} = 0.975$

  maka T_0 ditetapkan sebesar 1
- Faktor reduksi suhu ditetapkan sebesar 0.9
- Replikasi lokal ditetapkan 3 kali.
- Agar sama dilakukan 990 iterasi dengan 3 kali replikasi lokal, maka diharapkan terjadi 330 kali penurunan suhu. Sehingga Ta dapat dihitung sebagai berikut: $T_a = 1 \times (0.9)^{330} = 7.9 \times 10^{-16}$

Dengan demikian, akan didapatkan simulasi dari 600 problem yang digenerate oleh komputer. Pembandingan dilakukan pada performa tiap algoritma dan waktu proses tiap algoritma.

Simulasi dilakukan pada *processor* AMD dengan kecepatan 400MHz, dengan RAM 256Mb pada BUS 133. Perlu diingat kecepatan komputer dan kapasitas serta sistem BUS *MotherBoard* tentu akan berpengaruh pada kecepatan kalkulasi untuk problem yang diberikan. Sehingga bila dilakukan pada sistem komputer dengan kecepatan yang berbeda, kecepatan prosesnya tentu akan berbeda pula.

Setelah dilakukan semua kombinasi simulasi yang direncanakan, didapatkan hasil seperti pada Lampiran Tabel 1.

7. ANALISA

Pertama dilakukan analisa pada kolom *score*. Kolom *score* merupakan angka keunggulan satu algoritma terhadap algoritma yang lain, dari kolom ini dapat dilihat bahwa algoritma SA cukup telak unggul terhadap algoritma GA dalam simulasi yang dijalankan. Total *score* untuk GA adalah 60 keunggulan terhadap SA, sedangkan total *score* SA adalah 540 keunggulan terhadap GA. Dengan demikian, secara keseluruhan GA hanya memiliki 10% saja dari seluruh problem simulasi dimana GA unggul terhadap SA, sebaliknya SA unggul 90% terhadap GA pada simulasi ini.

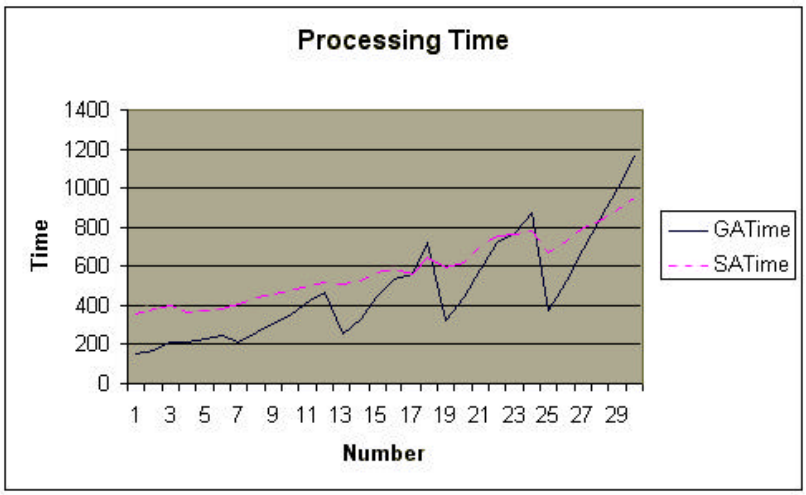
Analisa pada kolom performa, dilakukan perhitungan sebagai berikut:

Rata-rata performa GA pada simulasi $\frac{\acute{O}\overline{GA}}{30} = \frac{33.84595}{30} = 1.1282$

Rata-rata performa SA pada simulasi $\frac{\acute{O}\overline{SA}}{30} = \frac{185.3231}{30} = 6.1774$

Perhitungan diatas menunjukkan performa SA dalam simulasi ini jauh lebih baik dari performa GA. Hasil-hasil dimana SA unggul terhadap GA jauh lebih baik daripada hasil-hasil yang didapat bila GA yang unggul terhadap SA. Dari tabel pun dapat dilihat bahwa performa GA tidak pernah melebihi performa SA.

Analisa terhadap kolom waktu proses, GA memang lebih cepat dalam melakukan perhitungannya, namun terjadi peningkatan yang cukup besar seiring dengan bertambahnya *job* atau mesin dalam problem yang diberikan. Hal berbeda ditampakkan oleh SA, waktu proses SA tidak mengalami peningkatan sebesar GA walaupun waktu proses awalnya sudah lebih besar. Hal ini menyebabkan pada kombinasi *job* dan mesin yang besar, waktu proses SA bisa lebih cepat dibandingkan waktu proses GA. Melihat tren ini, dapat diperkirakan apabila kombinasi 10 *job* pada simulasi ini diteruskan, akan ada titik kombinasi dimana waktu SA bisa lebih cepat daripada waktu GA.



Gambar 1. Grafik Waktu Proses

Pada Gambar 1 di atas, dapat dilihat tren yang timbul. Waktu proses SA memiliki tren dengan gradasi yang lebih kecil, sedangkan waktu proses GA memiliki tren dengan gradasi yang lebih besar.

Grafik diatas menunjukkan bahwa waktu proses GA sangat dipengaruhi oleh jumlah mesin yang dikalkulasi, semakin banyak jumlah mesin yang dikalkulasi, waktu prosesnya menjadi semakin besar. Pada titik nomor 7, 13, 19, dan 25, dimana disini keadaan problem adalah pada jumlah mesin sedikit, kenaikannya tidak curam. Berbeda dengan pada titik nomor 6, 12, 18, dan 24, dimana disini keadaan problemnya adalah pada jumlah mesin yang banyak, kenaikan waktu proses sangat besar.

Sedangkan waktu proses SA cenderung memiliki kenaikan yang konstan yang berarti pengaruh jumlah job dan jumlah mesin yang dikalkulasi hampir sama besar dan ditinjau dari kelandaiannya, pengaruh keduanya tidak begitu besar.

8. KESIMPULAN DAN SARAN

Melihat pada hasil simulasi dan analisanya, dapat ditarik kesimpulan-kesimpulan sebagai berikut:

- Algoritma SA lebih unggul dibandingkan algoritma GA pada simulasi yang diberikan. Terbukti pada persentase keunggulan SA terhadap GA adalah 90% berbanding 10%.
- Walaupun waktu proses SA terlihat lebih lama daripada waktu proses GA, namun tren waktu proses SA lebih landai dibandingkan tren waktu proses GA. Sehingga dapat diperkirakan pada kasus dengan jumlah job dan mesin yang banyak, SA akan lebih cepat menyelesaikan iterasinya daripada GA. Namun mengingat keunggulan SA yang begitu besar terhadap GA dengan selisih waktu yang relatif kecil, SA tetap akan lebih baik dalam menyelesaikan problem-problem yang memiliki kombinasi mesin dan *job* kecil. Dapat dilihat dari Tabel 1, untuk kombinasi 10 *job* saja, SA masih memiliki keunggulan 69% terhadap GA.
- Performa SA saat unggul terhadap GA juga jauh lebih baik dibandingkan performa GA saat unggul terhadap SA. Hal ini semakin memperkuat rekomendasi SA sebagai metode penjadwalan meta-heuristik yang handal dalam mencari nilai objektif.
- Algoritma GA ternyata memakan waktu proses yang lebih lama pada kasus dengan job dan mesin yang banyak. Hal ini dapat disebabkan oleh perhitungan GA yang memang lebih banyak dari SA yang sangat tergantung pada jumlah *job* dan mesin yang diiterasikan, misal saja perhitungan bobot pencarian nilai Y, *crossover* yang akan menukarkan lebih banyak job dan kemungkinan mutasi.

Saran-saran yang dapat diberikan dari hasil penelitian ini, adalah:

- Algoritma GA yang diusulkan tampak baik pada perhitungan kasus dengan kombinasi job dan mesin yang sedikit, namun jika dilakukan untuk banyak kasus yang sama, ternyata SA masih lebih baik. Mungkin diperlukan modifikasi pada algoritma GA agar bisa lebih baik performanya. Kemungkinan terbesar yang dapat diperkirakan penulis adalah pada metode crossovernya.
- Program simulasi ini dirancang untuk dapat menyelesaikan kasus-kasus nyata yang terjadi di perusahaan-perusahaan manufaktur, namun karena tidak dicoba ke arah sana maka tingkat keberhasilannya belum bisa ditetapkan.
- Program dapat dikembangkan lagi dengan menambahkan algoritma *Branch and Bound* sehingga ukuran performa dapat dibandingkan langsung terhadap algoritma tersebut yang merupakan algoritma paling optimal, tidak perlu membandingkan secara langsung. Hal lain adalah penambahan *tool* untuk perhitungan secara kumulatif dan grafik tren waktu proses secara kumulatif.

DAFTAR PUSTAKA

Goldberg, David E., 1989. *Genetic Algorithms, in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co. Inc.

Ponnambalan, S.G., Jawahar, N., and Aravindan, P., 1999. "A Simulated Annealing Algorithm for Job Shop Scheduling", *Production Planning and Control*, Vol. 10, No.8, 767-777.

Dawei, L., Li, W., Mengguang, W., 1999. "Genetic Algorithm for Production Lot Planning of Steel Pipe", *Production Planning and Control*, Vol. 19, No.1, 54-57.

Rajendran, C., 1993. "Heurisctic for Schedulling in Flowshop with Multiple Objectives", *European Journal of Operational Research 82*.

Lampiran Tabel 1.

Tabel 1. Kumulatif Hasil Simulasi

No.	Kombinasi		Score		Performa		Waktu (detik)	
	Job	Mesin	GA	SA	$\overline{GA}$	$\overline{SA}$	$\overline{GA}$	$\overline{SA}$
1	10	5	6	**14**	2.342	**6.448**	**152.5**	356.5
2	10	10	4	**16**	1.9035	**4.36235**	**168.5**	382.5
3	10	15	7	**13**	2.04	**5.96**	**211**	395.5
4	10	20	8	**12**	2.4435	**4.5486**	**212.5**	363
5	10	25	4	**16**	1.20805	**3.6733**	**227.5**	370
6	10	30	9	**11**	1.7737	**4.9096**	**246**	384.5
7	20	5	2	**18**	2.9795	**7.5396**	**211.5**	408
8	20	10	2	**18**	2.0555	**8.19729**	**263.5**	444
9	20	15	2	**18**	1.6085	**6.4238**	**306**	459.5
10	20	20	0	**20**	0	**5.3029**	**352**	473
11	20	25	0	**20**	0	**5.5575**	**425**	500.5
12	20	30	1	**19**	2.671	**4.7354**	**463**	519.5
13	30	5	1	**19**	0.8203	**6.9955**	**252.5**	506.5
14	30	10	2	**18**	3.3465	**7.4435**	**332.5**	526.5
15	30	15	0	**20**	0	**6.595**	**437.5**	566
16	30	20	0	**20**	0	**5.6777**	**531**	584
17	30	25	0	**20**	0	**6.3852**	**560**	563
18	30	30	0	**20**	0	**5.955**	717.5	**646**
19	40	5	1	**19**	1.553	**6.1475**	**321.5**	596
20	40	10	2	**18**	1.5215	**5.63**	**438**	619.5
21	40	15	0	**20**	0	**6.7987**	**589**	701
22	40	20	0	**20**	0	**7.594**	**722**	752.5
23	40	25	0	**20**	0	**7.635**	775.5	**761**
24	40	30	0	**20**	0	**5.8978**	875.5	**781**
25	50	5	5	**15**	1.5684	**5.2587**	**372.5**	673
26	50	10	1	**19**	2.964	**7.96**	**529.5**	732
27	50	15	3	**17**	1.047	**6.681**	**691**	794
28	50	20	0	**20**	0	**7.22535**	839	**833.5**
29	50	25	0	**20**	0	**6.4657**	998	**887**
30	50	30	0	**20**	0	**5.31915**	1169.5	**951.5**