PERBANDINGAN KINERJA ALGORITMA GENETIKA DAN ALGORITMA HEURISTIK RAJENDRAN UNTUK PENJADUALAN PRODUKSI JENIS *FLOW SHOP*

Didik Wahyudi
Dosen Fakultas Teknik, Jurusan Teknik Mesin – Universitas Kristen Petra

Tessa Vanina Soetanto
Dosen Fakultas Teknik, Jurusan Teknik Industri – Universitas Kristen Petra

Ervin Medianti
Alumnus Fakultas Teknik, Jurusan Teknik Industri – Universitas Kristen Petra

ABSTRAK

Masalah penjadualan *flow shop* adalah menjadualkan proses produksi dari masing-masing ***n*** *job* yang mempunyai urutan proses produksi dan melalui ***m*** mesin yang sama. Kebanyakan penelitian hanya mengacu pada satu tujuan saja yaitu meminimumkan *makespan*. Tujuan yang lain, seperti meminimumkan *total flow time* atau *multiple objectives* yang meminimumkan *makespan*, *total flow time* dan *machine idle time* akan lebih efektif dalam mengurangi biaya penjadualan, sebagaimana dikatakan oleh French (1982).

Algoritma Rajendran (1995) yang menyelesaikan masalah *flow shop* dengan *multiple objectives* akan dipergunakan untuk mengevaluasi algoritma usulan: Algoritma Genetika, yang dikembangkan oleh Sridhar & Rajendran (1996) pada suatu masalah yang ditemui di suatu perusahaan sepatu.

Kata kunci: *flow shop*, algoritma genetika, *multiple objectives*

ABSTRACT

Flow shop scheduling problem is to schedule a production process of ***n*** *jobs that go through the same process sequence and the same* ***m*** *machines. Most researches are don to accomplish only one objective, i.e. minimizing makespan. The other objective, such as total flow time, or multiple objectives that is minimizing makespan, total flow time and machine idle time, will be more effective in reducing scheduling cost, as written in French (1982).*

Rajendran algorithm (1995) that solves flow shop problem with multiple objectives will be used to evaluate the proposed algorithm: Genetic Algorithm, developed by Sridhar & Rajendran (1996) *on a problem that existed in a shoe factory.*

Keywords: flow shop, genetic algorithm, multiple objectives

1. PENDAHULUAN

Masalah penjadualan *flow shop* adalah menjadualkan proses produksi dari masing-masing ***n*** *job* yang mempunyai urutan proses produksi dan melalui ***m*** mesin yang sama. Beberapa penelitian yang sudah dilakukan untuk meminimumkan *makespan* di antaranya adalah **Campbell et al**. (1970), **Dannenbring** (1977), **Nawaz et al** (1983), serta **Widmer & Hertz** (1989). Namun tujuan yang lain seperti meminimumkan *total flow time* (lamanya waktu yang dihabiskan seluruh *job* pada lantai produksi) atau *multiple objectives* yang meminimumkan *makespan*, *total flow time* dan *machine idle time* (waktu menganggur mesin) akan lebih efektif dalam mengurangi biaya penjadualan,

sebagaimana dikatakan oleh **French** (1982). Oleh karena itu **Rajendran** (1995) memberikan algoritma usulan untuk menyelesaikan masalah *flow shop* dengan *multiple objectives*. Algoritma ini akan dipergunakan untuk mengevaluasi algoritma usulan: *Algoritma Genetika*, yang dikembangkan oleh **Sridhar & Rajendran** (1996). Solusi awal untuk algoritma heuristic Rajendran diberikan berdasarkan algoritma heuristic **CDS** (1970). Sedangkan untuk algoritma genetika diberikan berdasarkan algoritma heuristic **NEH** (1983)/**CDS** (1970) dan algoritma heuristic **RC** (1992).

Beberapa asumsi yang dipakai dalam penjadualan *flow shop* ini adalah :
- proses produksi dari *job-job* sudah diketahui secara jelas
- tidak terdapat *pre-emption* (interupsi untuk mengerjakan produk lain di tengah-tengah pengerjaan suatu produk)
- setiap *job* memerlukan ***m*** mesin dan setiap proses memerlukan mesin yang berbeda
- waktu ***set-up*** bersifat *independent* dan termasuk dalam waktu proses
- semua *job* mempunyai *ready time* yang sama

2. LANGKAH-LANGKAH PENELITIAN

1. Mempelajari aliran proses produksi dan mengumpulkan data-data
2. Melakukan perhitungan waktu baku
3. Penjadualan dengan Algoritma Heuristik CDS
4. Penjadualan dengan Algoritma Heuristik Rajendran
5. Penjadualan dengan Algoritma Heuristik NEH
6. Penjadualan dengan Algoritma Heuristik RC
7. Penjadualan dengan Algoritma Genetika dengan bantuan program
8. Membandingkan hasil penjadualan Algoritma Heuristik Rajendran dengan hasil penjadualan Algoritma Genetika
9. Mengevaluasi dan menarik kesimpulan dari hasil-hasil penjadualan

3. FORMULASI *TIME TABLE FLOW SHOP*

Umumnya pada sistem produksi yang bersifat *flow shop*, terdiri dari beberapa mesin (***m***) dan mempunyai sejumlah *job* yang harus dikerjakan (***n***) serta waktu proses per unit *job i* pada mesin *j*, $\mathbf{t_{ij}}$ (untuk i =1,…….n; j = 1,…….m) maka :

$$T_{ij} = t_{ij} \times d_i \qquad (1)$$

dengan $\boldsymbol{d_i}$ adalah jumlah permintaan untuk *job i* dan $\boldsymbol{T_{ij}}$ adalah total waktu proses (sesuai *demand*) *job i* pada mesin *j* serta saat dimulainya *job i* pada mesin *j* ($\boldsymbol{S_{ij}}$) dan saat selesainya *job i* pada mesin *j* ($\mathbf{E_{ij}}$). Sedangkan $\mathbf{t_{[i]j}}$ adalah waktu proses per unit *job* posisi ke-*i* pada mesin *j* dan $\boldsymbol{T_{[i]j}}$ merupakan total waktu proses *job* posisi ke-*i* (dalam *sequence*) pada mesin *j* maka saat dimulainya *job* urutan ke-*i* pada mesin *j* ($\boldsymbol{S_{[i]j}}$) dan saat selesainya *job* urutan ke-*i* pada mesin *j* ($\boldsymbol{E_{[i]j}}$) dapat dirumuskan :

- untuk j = 1
  - $S_{[1]1} = 0 \qquad E_{[1]1} = T_{[1]1}$ (2)
  - $S_{[i]1} = E_{[i-1]1} \qquad E_{[i]1} = S_{[i]1} + T_{[i]1}$ (3)
    ([i] = 2, 3,….., n)

- untuk j = 2
  - $S_{[1]2}=E_{[1]1\ 1]2}$ $E_{[1]2}=S_{[1]2}+T_{[1]2}$ (4)
    Jika $E_{[i-1]2} \le E_{[i]1}$ maka
    $S_{[i]2} = E_{[i]1}$ $E_{[i]2} = S_{[i]2} + T_{[i]2}$ (5)
    Jika $E_{[i-1]2} > E_{[i]1}$ maka
    $S_{[i]2}=E_{[i-1]2}$ $E_{[i]2} = S_{[i]2} + T_{[i]2}$ (6)
    ([i] = 2,3,….., n)
- formulasi untuk j = 3, 4,….., m adalah sama dengan formulasi j = 2.

4. ALGORITMA HEURISTIK RAJENDRAN

Rajendran, C. (1995) mencoba untuk memenuhi ketiga kriteria *performance* penjadualan di atas dengan cara meminimumkan *gap* yang ada di antara *succesive operations*, sehingga didapatkan solusi yang berkualitas. Jika didapatkan pasangan *job-job* dengan *gap* yang semuanya negatif, maka pasangan *job-job* ini adalah merupakan penjadualan yang terakhir dan bila didapatkan pasangan *job-job* dengan *gap* yang semuanya positif, ini merupakan awal dari proses penjadualan. Keuntungan yang lebih baik akan diperoleh apabila mengganti pasangan *job-job* yang positif dengan *gap-gap* yang negatif.

Urutan atau langkah-langkah untuk melakukan perancangan dengan menggunakan Algoritma Heuristik Rajendran ini adalah sebagai berikut :

(a) Mendapatkan urutan awal dengan menggunakan Algoritma CDS, menukar *job* posisi ke-***r*** dan ke-***r+1*** (***r*** = 1, 2, 3,…., ***n***) dari urutan awal tersebut, sehingga didapatkan ***n*** urutan berbeda (***n*** = jumlah *job*). Dari ***n*** urutan dicari *sequence* dengan *makespan* terminimum.

(b) *Sequence* (***s***) dengan *makespan* terminimum, dihitung $D_{[r]}$ -nya, dari *job* pada posisi ke-***r*** (1≤ r ≤n-1)

$$D_{[r]} = \sum_{j=1}^{m} t_{[r]j} - \sum_{j=1}^{m} t_{[r+1]j} \quad (7)$$

$$D'_{[r]} = \sum_{j=1}^{m} \left\{ (m-j+1) t_{[r]j} \right\} - \sum_{j=1}^{m} \left\{ (m-j+1) t_{[r+1]j} \right\} \quad (8)$$

(c) Memilih *job* yang memiliki $D_{[r]} \ge 0$

(d) Jika tidak ada $D_{[r]} \ge 0$, dilanjutkan ke langkah 10, tetapi bila ada *job* yang memiliki $D_{[r]} \ge 0$ dilanjutkan ke langkah 5

(e) Mengurutkan $D_{[r]} \ge 0$ dari yang paling besar terlebih dahulu hingga yang paling kecil, jika didapatkan $D_{[r]}$ yang sama besar, hitunglah $D'_{[r]}$

(f) Menentukan *job* yang didapatkan dari langkah ke-5 (*job* yang memiliki $D_{[r]} \ge 0$ dari yang paling besar) yaitu *job* pada posisi ***q***. Menukar *job* pada posisi ke-***q*** dengan *job* pada posisi ke-***q+1*** pada *sequence* ***s***, dan *sequence* dari hasil penukaran tersebut disebut ***s'*** dengan *makespan*, *total flow time* dan *machine idle time*, ***M'***, ***F'*** dan ***I'***

(g) Menghitung $\boldsymbol{R}_{S'}$ dan $\boldsymbol{R}_{S}$, dimana:

$$R_{S'} = \frac{M' - \min(M', M)}{\min(M', M)} + \frac{F' - \min(F', F)}{\min(F', F)} + \frac{I' - \min(I', I)}{\min(I', I)} \quad (9)$$

$$R_S = \frac{M - \min(M', M)}{\min(M', M)} + \frac{F - \min(F', F)}{\min(F', F)} + \frac{I - \min(I', I)}{\min(I', I)} \quad (10)$$

M = *makespan sequence* s, M' = *makespan sequence* s'
F = *total flow time sequence* s, F' = *total flow time sequence* s'
I = *machine idle time sequence s*, I' = *machine idle time sequence s'*

(h) Jika $\boldsymbol{R_{S'} < R_S}$, maka *sequence* ***s'*** menjadi urutan jadual yang baru dan kembali ke langkah 2, sehingga untuk selanjutnya $\boldsymbol{s = s'}$, $\boldsymbol{M = M'}$, $\boldsymbol{F = F'}$, dan $\boldsymbol{I = I'}$, tetapi jika tidak ($\boldsymbol{R_{S'} > R_S}$) dilanjutkan ke langkah 9.

(i) Memilih *job* pada posisi $\boldsymbol{q}$ yang masih ada dari langkah 5, yaitu *job* yang memiliki $\boldsymbol{D_{[r]} \geq 0}$ paling besar berikutnya dan kembali melakukan proses pada langkah 6, tetapi bila sudah tidak ada dilanjutkan ke langkah 10

(j) *Sequence* yang terakhir yang didapatkan merupakan urutan jadual yang terbaik.

5. ALGORITMA HEURISTIK CDS

Penjadualan dengan metode Campbell, Dudek, and Smith (CDS) ini melangkah dengan *stage*, jumlah *stage* tersebut yaitu s = 1, 2, …., m-1
Adapun perhitungan tersebut adalah sebagai berikut :

$$Xs = \sum_{j=1}^{s} t_{i,j} \qquad Ys = \sum_{j=m-(s-1)}^{m} t_{i,j}$$

$t_{i,j}$ = waktu proses job i pada mesin j
m = mesin yang terakhir
Xs = stage langkah pertama
Ys = stage langkah kedua
Selanjutnya disusun penjadualan dengan Algoritma Johnson.

6. ALGORITMA HEURISTIK RAJENDRAN

Pada umumnya penjadualan heuristik jenis *flowshop* hanya bertujuan untuk meminimumkan *makespan.* Padahal ada tujuan lain pada penjadualan *flowshop* yang tidak kalah pentingnya yaitu meminimumkan *total flowtime*. Masing-masing tujuan ini memiliki manfaat yang berbeda, tetapi sama-sama menguntungkan. Jika *makespan* dapat diminimumkan maka dapat meminimumkan *production run*, sedangkan apabila *total flowtime* dapat diminimumkan maka penggunaan *resources* dapat lebih teratur, mempercepat perputaran antar *job*, dan dapat meminimumkan *in-process inventory* (Baker and French).

Oleh karena itu Algoritma Heuristik Rajendran ini mencoba untuk memenuhi ketiga kriteria yang ada, yaitu *minimizing makespan, total flowtime* dan *machine idletime* secara bersama-sama, dan juga telah diketahui bahwa penjadualan dengan *multiple objective* lebih efektif untuk mengurangi total biaya penjadualan (Chandrasekharan Rajendran).

Pada dasarnya prinsip dari Algoritma Heuristik Rajendran ini adalah meminimumkan *gap* yang ada diantara *succesive operations*, sehingga didapatkan solusi yang berkualitas. Jika didapatkan pasangan *job-job* dengan *gap* yang semuanya negatif, maka pasangan

job-job ini adalah merupakan penjadualan yang terakhir. Tetapi apabila didapatkan pasangan *job-job* dengan *gap* yang semuanya positif, maka pasangan job-job ini merupakan awal dari proses penjadualan. Akan memberikan keuntungan yang lebih baik apabila mengganti pasangan *job-job* yang positif dengan *gap-gap* yang negatif.

Pertama kali yang harus dilakukan untuk menyelesaikan penjadualan Algoritma Heuristik Rajendran adalah mendapatkan urutan awal dengan menggunakan Algoritma Heuristik CDS, dilanjutkan dengan dengan langkah-langkah sebagai berikut :

(a) Menukar *job* ke-r dan *job* ke-r+1, hingga didapatkan 35 urutan jadual yang berbeda, kemudian dipilih yang memiliki *makespan* terminimum.
(b) *Sequence* (s) dengan *makespan* terminimum tersebut dihitung $D_{[r]}$.
(c) Memilih dan mengurutkan $D_{[r]}$ yang memiliki $D_{[r]} \geq 0$
(d) Menentukan $D_{r]} \geq 0$ yang paling besar terlebih dahulu, kemudian menukar *job* tersebut dengan *job* berikutnya dari urutan jadual s, dan jadual yang telah ditukar ini disebut s'
(e) Menghitung $R_{S'}$ dan R_S
(f) Karena $R_{S'} < R_S$, maka *sequence* s' menjadi urutan jadual yang baru dan kembali ke langkah 2 yaitu menghitung kembali $D_{r]}$ dengan berdasarkan urutan jadual yang baru yaitu *sequence* s, kemudian dilanjutkan dengan langkah-langkah berikutnya.
Perhitungan seperti yang telah dijelaskan di atas, diulang terus menerus, hingga semua $D_{[r]} \geq 0$ telah diproses.

7. ALGORITMA HEURISTIK NEH

Dalam menyelesaikan penjadualan pada sistem produksi bersifat *flowshop*, Nawaz, Enscore, and Ham (1983) mengusulkan algoritma heuristik yaitu *job* yang memiliki total waktu proses lebih besar dari *job* lain dengan total waktu proses yang lebih kecil, seharusnya diberi bobot yang lebih tinggi, sehingga dapat meminimumkan *makespan*.
Dengan :

s = job yang sudah dijadualkan dari **n** *job* yang ada (*partial schedule*)
π = kumpulan *job-job* yang belum dijadualkan
$q(s,j)$ = waktu penyelesaian dari *partial schedule* s pada mesin **j** (waktu penyelesaian pada mesin **j** setelah memproses beberapa *job* dalam *partial schedule* $\boldsymbol{s}$)
a = salah satu *job* dalam **p**
$q(s_a,j)$ = waktu penyelesaian dari *job* **a** di mesin **j**, saat *job* **a** ditambahkan pada *partial schedule* $\boldsymbol{s}$

Sehingga secara umum didapatkan rumusan : $q(s_a,j) = \max [q(s,j) ; q(s_a,j-1)] + T_{aj}$
Sedangkan *makespan* dari partial schedule s_a: $Ms_a = q(s_a,m)$

(a) Langkah pertama yang dilakukan untuk membuat penjadualan menggunakan Algoritma Heuristik NEH adalah menghitung $\boldsymbol{T_i}$ untuk setiap *job* yang ada.
(b) Langkah selanjutnya adalah membuat suatu urutan *job* berdasarkan $\boldsymbol{T_i}$, dari $\boldsymbol{T_i}$ yang terkecil sampai yang terbesar.
(c) Dari urutan *job* tersebut, diambil 2 *job* dari urutan yang pertama.
(d) Berdasarkan formulasi *time table* yang telah dijelaskan, dipilih jadual dengan *makespan* terkecil antara kedua jadual tersebut.

(e) Dari jadual yang dipilih tersebut ditambah satu *job* yang diambil dari urutan berikutnya.

(f) Prosedur tersebut diulang-ulang beberapa kali hingga semua *job* dalam urutan *job* yang berasal dari perhitungan T_i tersebut sudah dijadualkan.

8. ALGORITMA HEURISTIK RC

Berdasarkan pada kriteria yaitu meminimumkan *total flowtime* dapat mengurangi biaya penjadualan secara signifikan, maka Rajendran and Chaudhuri (1991) mengusulkan suatu algoritma heuristik yaitu melakukan penjadualan job yang mempunyai waktu proses lebih pendek terlebih dahulu daripada job yang mempunyai waktu proses lebih panjang sehingga waktu tunggu (*waiting time*), waktu idle mesin (*machine idletime*), dan waktu penyelesaian (*completion time*) menjadi minimum yang pada akhirnya menghasilkan *total flowtime* yang minimum pula.

Parameter yang dipakai sama dengan Algoritma NEH dengan beberapa parameter tambahan sebagai berikut :

$\boldsymbol{s}_c$: *partial schedule* $\boldsymbol{s}$ yang diikuti dengan *job* **c**

F_s : *total flowtime job-job* dalam $\boldsymbol{s}$

b : *job* terakhir dalam $\boldsymbol{s}$

W_{ba} : jumlah waktu tunggu *job* **a** pada berbagai mesin bila *job* **a** mengikuti *job* **b** dalam *partial schedule* $\boldsymbol{s}$

Dimana $q(s_a,0) = 0$ dan $q(\text{Æ},j) = 0$, j = 1,2,…,m ; Æ adalah *initial null schedule* maka : $F_s = F_s + q(s_a,m)$

Untuk menyelesaikan penjadualan dengan menggunakan Algoritma Heuristik RC ini terbagi dalam 3 kriteria heuristik. Kriteria heuritik pertama adalah memilih jadual dengan waktu menganggur mesin (*machine idletime*) yang terkecil, kriteria kedua adalah memilih jadual yang memiliki jumlah antara waktu menganggur mesin (*machine idletime*) dan waktu tunggu *job* (*waiting time*) yang paling kecil, dan kriteria heuristik yang ketiga adalah menentukan jadual yang mempunyai jumlah antara waktu menganggur mesin (*machine idletime*), waktu tunggu *job* (*waiting time*) dan waktu penyelesaian (*completion time*) yang terkecil.

Persamaan ketiga kriteria heuristik adalah sebagai berikut :

(a) Meminimumkan *machine idletime* :

$$\min \sum_{j=2}^{m} \max[q(s_a,j-1) - q(s,j),0]$$

(b) Meminimumkan jumlah *machine idletime* dan *waiting time* :

$$\min \sum_{j=2}^{m} abs[q(s_a,j-1) - q(s,j)]$$

(c) Meminimumkan jumlah *completion time*, *machine idletime*, dan *waiting time*:

$$\min \sum_{j=2}^{m} abs[q(s_a,j-1) - q(s,j)] + \sum_{j=1}^{m} q(s_a,j)$$

9. PENJADUALAN DENGAN ALGORITMA GENETIKA

Di dalam menyelesaikan masalah penjadualan *flow shop* dengan *multiple objectives*, meminimumkan *makespan, total flow time* dan *machine idle time* maka **Sridhar and Rajendran** (1996) menggunakan Algoritma Genetika.

Bentuk generik dari Algoritma Genetika adalah sebagai berikut :
- Menginitialisasi populasi P(0)
- Mengevaluasi populasi P(0)
- Menginitialisasi generasi = 1
- Mengerjakan langkah-langkah berikut sampai kondisi tertentu :
  - Memilih P(generasi) dari P(generasi-1)
  - Crossover P(generasi)
  - Mutasi P(generasi)
  - Mengevaluasi P(generasi)
  - Generasi = generasi + 1

Jika Algoritma Genetika diaplikasikan pada problem penjadualan, maka *chromosome* menggambarkan urutan *job*, *gen-gen* melambangkan posisi-posisi *job* tersebut, sedangkan *allele* menggambarkan *job-job* yang memiliki posisi-posisi tersebut. Untuk menginitialisasi dua sub populasi digunakan *makespan minimizing* Algoritma Heuristik NEH (1983) dan *total flow time minimizing* Algoritma Heuristik RC (1992).

Crossover operator yang digunakan adalah *Partially Matched Crossover* (PMX) yang ditemukan oleh Goldberg ang Lingle (1985). Operator ini memberikan pemisah antara posisi *job-job* dengan garis vertikal seperti contoh berikut ini:
Misalnya bilangan random yang dihasilkan adalah 1 dan 3 maka garis vertikal akan diletakan pada posisi ke-1 dan posisi ke-3 *chromosome parent* :

P_1 :	1	4	5	2	3	C_1 :	3	1	5	2	4
P_2 :	3	1	2	5	4	C_2 :	1	4	2	5	3

Untuk membangun *Chid 1* (***C1***), maka dilihat pemisahan pada *Parent 2* (***P2***) untuk mencari *job* mana yang akan ditukar dengan *job* pada *Parent 1*(***P1***). Ternyata *job* 1 dan *job* 3, *job* 4 dan *job* 1 ditukar dengan *Parent 1* untuk menghasilkan *Child 1* dan demikian pula untuk menghasilkan *Child 2* (***C2***).

Sedangkan cara untuk mengevaluasi *chromosome* yang tidak sesuai dengan fungsi tujuan dan menggantikannya dengan *chromosome* yang potensial menggunakan operator Delta. Apabila terdapat dua *chromosome* (urutan/*sequence*) yaitu ***S1*** dan ***S2*** dengan *makespan* $\boldsymbol{MS_1}$ dan $\boldsymbol{MS_2}$, *total flow time* $\boldsymbol{FT_1}$ dan $\boldsymbol{FT_2}$, serta *machine idle time* $\boldsymbol{IT_1}$ dan $\boldsymbol{IT_2}$, serta bobot kepentingannya dari *makespan, total flow time* dan *machine idle time* adalah $\boldsymbol{w_1}$, $\boldsymbol{w_2}$, dan $\boldsymbol{w_3}$ ($\boldsymbol{w_1}+\boldsymbol{w_2}+\boldsymbol{w_3}=1$).

$$e_1 = w_1\left\{\frac{MS_1 - \min[MS_1; MS_2]}{\min[MS_1; MS_2]}\right\} + w_2\left\{\frac{FT_1 - \min[FT_1; FT_2]}{\min[FT_1; FT_2]}\right\} + w_3\left\{\frac{IT_1 - \min[IT_1; IT_2]}{\min[IT_1; IT_2]}\right\} \quad (11)$$

$$e_2 = w_1\left\{\frac{MS_2 - \min[MS_1; MS_2]}{\min[MS_1; MS_2]}\right\} + w_2\left\{\frac{FT_2 - \min[FT_1; FT_2]}{\min[FT_1; FT_2]}\right\} + w_3\left\{\frac{IT_2 - \min[IT_1; IT_2]}{\min[IT_1; IT_2]}\right\} \quad (12)$$

Delta = $e_1 - e_2$

Jika Delta ≤ 0 maka S_1 (*sequence* 1) mempunyai *makespan, total flow time* dan *machine idle time* yang lebih baik dibandingkan S_2, sedangkan jika Delta > 0 maka S_2 yang lebih baik dari S_1. Bobot *makespan* (w_1), bobot *total flow time* (w_2), dan bobot *machine idle time* (w_3) pada penelitian ini mempunyai nilai yang sama yaitu 0.333 karena tujuan yang ingin dicapai adalah meminimumkan *makespan, total flow time* dan *machine idle time* secara bersama-sama, tidak lebih mementingkan salah satu dari tiga kriteria tersebut.Operator Delta sebagai *replacement policy* digunakan untuk menentukan apakah *chromosome* keturunan (*child*) menggantikan *chromosome parent* pada generasi berikutnya. Hal ini penting karena *replacement policy* ini akan membuang *chromosome* yang bermutu rendah dalam proses selanjutnya. Selain itu *chromosome* terbaik (***G***) akan diidentifikasikan pada kedua subpopulasi setiap akhir generasi seperti yang dilakukan pada generasi yang pertama. Hal tersebut dilakukan dengan mengambil 2 *chromosome* setiap kali dan menggunakan operator Delta. Prosedur pengerjaan Algoritma Genetika adalah sebagai berikut :

Step 0. Mengumpulkan data-data : n *job*, m mesin, serta T_{ij}

Step 1. Set generation number = NGEN = 0

Step 2. Membentuk subpopulasi 1 dan 2 :

(1) Menggunakan hasil algoritma heuristic NEH dan melakukan penukaran antara *job* ke-k dan ke-k+1 sehingga didapatkan n urutan berbeda. Ke-n jadual tersebut diurutkan berdasarkan *makespan* terkecil.

(2) Menggunakan hasil algoritma heuristic RC dan melakukan penukaran antara *job* ke-k dan ke-k+1 sehingga didapatkan n urutan berbeda. Ke-n jadual tersebut diurutkan berdasarkan total *flow time* terkecil.

Step 3. Mencari urutan (*chromosome*) terbaik dari Sub-populasi 1 dan 2 : G, dengan pengambilan 2 *chromosome* setiap kali dan dievaluasi menggunakan operator Delta.

Step 4. Set NGEN = NGEN+1, jika NGEN > 11 langsung ke step 7, lainnya ke step 5

Step 5. Untuk i=1,2,…..,n maka lakukan

(1) Mengambil *chromosome* posisi ke-*i* pada Subpopulasi 1 & 2 sebagai *chromosome Parent 1 & Parent 2* yang akan dikenai *Partially Matched Crossover* (PMX) sehingga didapatkan *chromosome Child 1 & Child 2*

(2) Dilakukan evaluasi antara *chromosome Parent 1* dan *Child 1* juga antara *chromosome Parent 2* dan *Child 2* menggunakan operator Delta. Chromosome Child yang lebih unggul akan menggantikan *chromosome Parent*-nya.

Step 6. Jika NGEN kelipatan 5 maka dilakukan operator mutasi terhadap semua *chromosome* pada subpopulasi 1 & 2. Lainnya, kembali ke Step 3.

Step 7. Ambil chromosome G dan bentuklah (n-1) urutan dengan menukar *job* ke-k dan ke-k+1 pada G (k=1,2,…..,(n-1)) sehingga total didapatkan n urutan. Ambil 2 urutan (*chromosome*) setiap kali untuk dievaluasi menggunakan operator Delta dan ambil hasil yang terbaik, ini merupakan urutan (*chromosome*) terakhir
STOP

10. APLIKASI

Penulis telah mencoba menerapkan beberapa algoritma yang telah dijelaskan sebelumnya pada penjadualan produksi di suatu perusahaan sepatu. Tujuan penjadualan

ini adalah untuk mempersingkat *makespan*, meminimumkan *total flow time* dan meminimumkan *machine idle time*.

Pada proses produksi yang diamati terdapat 35 *job* yang diproses pada 12 mesin dan *demand* yang diambil adalah selama 1 bulan. Langkah pertama yang dikerjakan adalah mendapatkan solusi awal untuk algoritma heuristic Rajendran dan algoritma Genetika dengan menggunakan algoritma heuristic CDS, NEH dan RC. Jadual yang didapatkan dengan menggunakan algoritma heuristic CDS memberikan *makespan* sebesar 159787.02 detik sedangkan algoritma heuristic NEH yang bertujuan sama, yaitu meminimumkan *makespan* dapat memberikan hasil yang lebih minimum sebesar 159787.02 detik. Algoritma heuristic RC yang bertujuan meminimumkan *total flow time* menghasilkan jadual dengan *total flow time* sebesar 2443119 detik.

Dalam melakukan penjadualan dengan algoritma Genetika, peneliti dibantu oleh suatu program Pascal, karena setiap kali dilakukan proses perhitungan terjadi perbedaan pada bilangan *random* yang dibangkitkan sehingga jadual yang didapat juga berbeda-beda, maka diambil 10 jadual untuk mendapatkan nilai rata-rata *makespan*, *total flow time*, dan *machine idle time*.

Tabel 1. Perbandingan *Performance* Jadual Hasil Algoritma Rajendran dan Genetika

Performance	Rajendran	Genetika
Makespan (detik)	173071.99	167817.92
Total Flow time(detik)	2440935.5	2477039.75
Machine Idle time(detik)	473445.89	414155.062

Dari perbandingan *performance*, jadual hasil Algoritma Heuristik Rajendran dan Algoritma Genetika, terlihat bahwa algoritma heuristik Rajendran memiliki hasil *total flow time* yang lebih baik, sedangkan untuk *makespan* dan *machine idle time*, Algoritma Genetika memiliki hasil yang lebih baik. Dilihat secara keseluruhan maka yang menghasilkan jadual yang terbaik adalah Algoritma Genetika. Hal ini dapat diketahui dengan menggunakan operator Delta. Apabila hasil jadual Algoritma Heuristik Rajendran sebagai S_1 dan hasil jadual Algoritma Genetika sebagai S_2, maka didapatkan operator Delta sebesar 0.053173 yang berarti algoritma Genetika menghasilkan jadual yang lebih baik bila dibandingkan dengan jadual hasil algoritma heuristik Rajendran.

Penulis juga mencoba melakukan penjadualan dengan algoritma Genetika menggunakan solusi awal hasil dari algoritma heuristik CDS dan algoritma heuristik RC, diperoleh *makespan* sebesar 170966.99 detik, *total flow time* sebesar 2460714.15 detik dan *machine idle time* sebesar 419043.365 detik. Jika hasil algoritma heuristik Rajendran sebagai S_1 dan hasil algoritma Genetika dengan menggunakan algoritma heuristik CDS dan algoritma heuristik RC sebagai populasi awalnya disebut S_2, maka didapatkan operator Delta sebesar 0.044634 yang berarti jadual Algoritma Genetika dengan menggunakan algoritma heuristic CDS dan algoritma heuristic RC sebagai populasi awal, lebih baik daripada jadual algoritma heuristik Rajendran.

11. KESIMPULAN

Algoritma Genetika yang diterapkan pada masalah penjadualan dalam mencapai kriteria *multiple objectives* yaitu *makespan*, *total flowtime* dan *machine idletime* yang

minimum secara bersama-sama telah menunjukkan keunggulannya baik menggunakan algoritma heuristic CDS dan algoritma heuristic RC maupun menggunakan algoritma heuristic NEH dan algoritma heuristic RC sebagai populasi awalnya, tetap menghasilkan jadual yang lebih baik bila dibandingkan dengan jadual hasil algoritma heuristik Rajendran. Hal ini disebabkan karena algoritma Genetika memproses sekumpulan solusi yang potensial secara terus menerus dan membuang solusi lain yang kurang potensial dalam pencapaian kriteria tujuan yang diinginkan.. hal ini dapat terlihat dari solusi yang dihasilkan pada setiap generasi menunjukkan kecenderungan *performance* yang semakin meningkat.

DAFTAR PUSTAKA

French, Simon, 1982. *Sequencing and Scheduling: An Introduction to the Mathematics of Job Shop*., Chichester: Ellis Horwood**.**

Rajendran, Chandrasekharan,. 1995. "Heuristics for Scheduling in Flow Shop with Multiple Objectives". *European Journal of Operational Research* , No. 82, 540-555.

Sridhar, J., and Rajendran, C., 1996. "Scheduling in Flow Shop and Cellular Manufacturing Systems with Multiple Objectives-A Genetic Algorithmic Approach",. *Productian Planning and Control,* Vol 7, No 4, 374-382.

Campbell, H.G., Dudek, R.A., and Smith, M.L.,. 1970. "A Heuristic Algorithm for the n-job, m-machine Sequencing Problem", *Management Science*, No. 16, B630-B637.

Dannenbring, D. G.,. 1977. "An Evaluation of Flow Shop Sequencing Heuristics", *Management Science*, No. 23, 1174-1182.

Nawaz, M., Enscore, E. E., and Ham, I.,. 1983. "A Heuristic Algorithm for the m-machine, n-job Flow Shop Sequencing Problem", *Omega*, No. 11, 91-95

Widmer, M. and Hertz, A.,. 1989. "A New Heuristic Method for the Flow Shop Sequencing Problem",. *European Journal of Operation Research*, No. 41, 189-193.

Rajendran, C., and Chaudhuri, C., 1992. "An Efficient Heuristic Approach to the Scheduling of Jobs in a Flow Shops", *European Journal of Operational Research*, No. 61, 318-325.