PENJADUALAN *FLOWSHOP* DENGAN ALGORITMA GENETIKA

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

Danny Prabowo Soetanto
Dosen Fakultas Teknik, Jurusan Teknik Industri – Universitas Kristen Petra

ABSTRAK

Makalah ini menyajikan penjelasan dan contoh aplikasi dari algoritma genetika untuk penyelesaian masalah penjadualan *flowshop* di PT "X" dengan tujuan meminimumkan *makespan, total flowtime*, dan *machine idletime* secara bersama-sama. Selain itu untuk mengetahui keunggulan dari algoritma genetika juga dilakukan perbandingan terhadap algoritma*heuristic* HC (Ho and Chang, 1991).

Kata kunci: *flowshop, multiple objectives*, algoritma genetika

ABSTRACT

This paper explains and gives an application of genetic algorithm for flowshop scheduling at "X" company to reach minimum multiple objectives of makespan, total flowtime and machine idletime. Comparison to HC heuristic algorithm (Ho and Chang, 1991) is done to show the differences of genetic algorithm.

Keywords: flowshop, multiple objectives, genetic algorithm

1. PENDAHULUAN

Masalah penjadualan *flowshop* adalah menjadualkan proses produksi dari masing-masing *n job* yang mempunyai urutan proses produksi dan melalui *m* mesin yang sama. Sudah banyak penelitian yang dilakukan untuk mencari penyelesaian yang optimal terhadap masalah ini dan kebanyakan hanya mengacu pada satu tujuan saja yaitu meminimumkan *makespan* **(**waktu penyelesaian *job* terakhir yang meninggalkan sistem). Beberapa diantaranya adalah **Campbell et al**. (1970), **Dannenbring** (1977), **Nawaz et al** (1983), **Ogbu & Smith** (1990) serta **Widmer & Hertz** (1989). Namun tujuan yang lain seperti meminimumkan *total flowtime* (lamanya waktu yang dihabiskan seluruh *job* pada lantai produksi) atau *multiple objectives* yang meminimumkan *makespan, total flowtime* dan *machine idletime* (waktu *idle* mesin) akan lebih efektif dalam mengurangi biaya penjadualan, hal ini dikatakan oleh **French** (1982).

Oleh karena itu **Ho & Chang** (1991) memberikan algoritma usulan untuk menyelesaikan masalah *flowshop* dengan *multiple objectives* dan dipergunakan untuk mengevaluasi algoritma usulan: genetika, yang dikembangkan oleh **Sridhar & Rajendran** (1996). Sedangkan solusi awal untuk algoritma genetika diberikan berdasarkan algoritma heuristic **NEH** (1983) dan algoritma heuristic **RC** (1991)
Beberapa asumsi yang dipakai dalam penjadualan *flowshop* ini adalah:

- proses produksi dari *job-job* sudah diketahui secara jelas

- tidak terdapat pre-e*mption* (interupsi untuk mengerjakan produk lain ditengah-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. FORMULASI *TIME TABLE* PADA *FLOWSHOP*

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

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

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

- untuk j = 1
  - $S_{[1]1} = 0 \qquad E_{[1]1} = T_{[1]1}$ (2)
  - $S_{[i]1} = E_{[i-1]1} \qquad E_{[i]1} = S_{[i]1} + T_{[i]1}$, ([i] = 2,3,…..n) (3)
- untuk j = 2
  - $S_{[1]2} = E_{[1]1} \qquad E_{[1]2} = S_{[1]2} + T_{[1]2}$ (4)
    Jika $E_{[i-1]2} \le E_{[i]1}$ maka
  - $S_{[i]2} = E_{[i]1} \quad E_{[i]2} = S_{[i]2} + T_{[i]2}$ (5)
    Jika $E_{[i-1]2} > E_{[i]1}$ maka
  - $S_{[i]2} = E_{[i-1]2} \quad E_{[i]2} = S_{[i]2} + T_{[i]2}$, ([i] = 2,3,…..n) (6)
- formulasi untuk j = 3,4,…..m adalah sama dengan formulasi j = 2

Dengan melihat kondisi lantai produksi yang ada, yaitu setiap unit produk yang selesai dikerjakan pada suatu mesin dapat langsung berpindah pada mesin selanjutnya melalui *conveyor* tanpa harus menunggu penyelesaian seluruh *demand* produk tersebut maka rumusan $S_{[i]j}$ dan $E_{[i]j}$ mengalami perubahan pada saat j = 2,3,…..m sebagai berikut :

- untuk j = 1, tidak mengalami perubahan
- untuk j = 2
  - $S_{[1]2} = E_{[1]2} + t_{[1]2} - T_{[1]2} \qquad E_{[1]2} = E_{[1]1} + t_{[1]2}$ (7)
    Jika $S_{[1]2} < 0$ maka
  - $S_{[1]2} = S_{[1]1} + t_{[1]1} \qquad E_{[1]2} = S_{[1]2} + T_{[1]2}$ (8)
    $S_{[i]2} = E_{[i]1} + t_{[i]2} - T_{[i]2} \qquad E_{[i]2} = E_{[i]1} + t_{[i]2}$
    Jika ($S_{[i]2} \le S_{[i]1}$) atau ($S_{[i]2} < E_{[i-1]2}$) maka
    $S_{[i]2} = S_{[i]1} + t_{[i]1} \qquad E_{[i]2} = S_{[i]2} + T_{[i]2}$ (9)
    Jika ($E_{[i]2} \le E_{[i]1}$) atau ($S_{[i]2} \le S_{[i]1}$) atau ($S_{[i]2} < E_{[i-1]2}$) maka
    $S_{[i]2} = E_{[i-1]2} \qquad E_{[i]2} = S_{[i]2} + T_{[i]2}$ (10)
    (untuk [i] = 2,3,…..n)
- formulasi untuk j = 3,4,…..m adalah sama dengan formulasi j = 2

3. ALGORITMA HEURISTIC NEH

Dalam menyelesaikan masalah penjadualan pada sistem produksi bersifat *flowshop;* **Nawaz, Enscore, and Ham** (1983) mengusulkan algoritma heuristic yaitu *job* dengan total waktu proses pada semua mesin lebih besar seharusnya diberi bobot yang lebih tinggi daripada job lain dengan total waktu proses yang lebih kecil, sehingga dapat diraih *makespan* yang minimum.

Parameter – parameternya adalah sebagai berikut :

σ = *job* yang sudah dijadualkan dari *n job* yang ada/partial *schedule*

π = kumpulan *job-job* yang belum dijadualkan

$q(\sigma,j)$ = waktu penyelesaian pada mesin j setelah memproses beberapa *job* dalam *partial schedule* σ (*completion time partial schedule* σ pada mesin j)

a = salah satu job dalam π

$q(\sigma a,j)$= *completion time job* a di mesin j, saat *job* a ditambahkan pada σ

Sehingga secara umum, didapatkan rumusan :

$$q(\sigma a,j) = \max [q(\sigma,j) ; q(\sigma a,j-1)] + T_{aj} \quad (11)$$

Sedangkan *makespan* dari partial *schedule* σa :

$$M_{\sigma a} = q(\sigma a,m) \quad (12)$$

Namun dengan adanya perubahan pada formulasi *time table* yang disesuaikan dengan kondisi lantai produksi yang ada maka

$$q(\sigma a,j) = E_{aj} \quad (13)$$

$$M_{\sigma a} = q(\sigma a,m) = E_{am} \quad (14)$$

4. ALGORITMA HEURISTIC RC

Berpijak pada kriteria yaitu meminimumkan *total flowtime* dapat mengurangi biaya penjadualan secara *significant*, maka **Rajendran and Chaudhuri** (199) mengusulkan suatu algoritma heuristic yaitu dengan menjadualkan *job* yang mempunyai waktu proses yang lebih pendek terlebih dahulu daripada *job* dengan waktu proses yang 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.

Ada beberapa parameter tambahan selain yang sudah didefinisikan sebelumnya sebagai berikut :

σc = partial *schedule* σ yang diikuti dengan *job* c

F_σ = *total flowtime job-job* dalam σ

b = *job* terakhir dalam σ

W_{ba} = jumlah waktu tunggu *job* a pada berbagai mesin bila *job* a mengikuti *job* b dalam partial *schedule*

Dimana $q(\sigma a,0) = 0$ dan $q(\phi,j) = 0$, j = 1,2,…..,m ; ϕ adalah *initial null schedule* maka

$$F_{\sigma =} F_\sigma + q(\sigma a,m) \quad (15)$$

Jika *job* a tidak mempunyai waktu tunggu sebelum memasuki mesin manapun maka *completion time job* a pada mesin ke-m bila *job* a mengikuti *job* b (*job* terakhir dalam partial *schedule* σ) : $q(\sigma a,m)$ =

$$q(\sigma,1) + \sum_{j=1}^{m} t_{aj} \quad (16)$$

Tetapi jika *job* a menunggu sebelum memasuki mesin-mesin tertentu, maka *completion time* job a menjadi :

$$q(\sigma a,m) = q(\sigma,1) + W_{ba} + \sum_{j=1}^{m} t_{aj} \quad (17)$$

$$W_{ba} = \sum_{j=2}^{m} \max[\, q(s, j) - q(sa, j-1), 0] \quad (18)$$

Oleh karena itu, Algoritma RC mengusulkan 3 kriteria heuristic sebagai berikut :

1) meminimumkan *machine idletime* :

$$\min \sum_{j=2}^{m} \max[\, q(sa, j-1) - q(s, j), 0] \quad (19)$$

2) meminimumkan jumlahan *machine idletime* dan *waiting time* :

$$\min \sum_{j=2}^{m} abs[q(sa, j-1) - q(s, j)] \quad (20)$$

3) meminimumkan jumlahan *completion time, machine idletime* dan *waiting time*:

$$\min \sum_{j=2}^{m} abs[q(sa, j-1) - q(s, j)] + \sum_{j=1}^{m} q(sa, j) \quad (21)$$

Dengan adanya perubahan pada formulasi *time table* yang disesuaikan dengan kondisi lantai produksi yang ada maka (misal *job* a pada urutan ke-k) terdapat beberapa rumusan yang berubah yaitu :

$$F_{\sigma a} = \sum_{i=1}^{k} E_{[i]m} \quad (22)$$

$$\text{Jumlahan } \textit{completion time}_{\sigma a} = \sum_{j=1}^{m} q(sa, j) = \sum_{j=1}^{m} E_{aj} \quad (23)$$

$$\textit{Machine Idletime}_{\sigma a} = \sum_{i=1}^{k-1} \sum_{j=2}^{m} \max[(S_{[i+1],j} - E_{[i],j}), 0] + \sum_{j=2}^{m} E_{[1]j} \quad (24)$$

$$\textit{Waiting Time}_{\sigma a} = \sum_{i=1}^{k} S_{[i]1} \quad (25)$$

5. ALGORITMA GENETIKA

Di dalam menyelesaikan masalah penjadualan *flowshop* dengan *multiple objectives*, meminimumkan *makespan, total flowtime* dan *machine idletime* maka **Sridhar and Rajendran** (1996) menggunakan Algoritma Genetika.

Algoritma Genetika adalah suatu algoritma/prosedur penelusuran yang berdasarkan pada mekanisme dari *natural selection* dan *natural genetics* yang dapat digunakan untuk memecahkan *combinatorial optimization problems* yang sulit. Algoritma ini

dikembangkan oleh **John Holland**, rekan kerja dan muridnya dari Michigan University (1975); yang mengkombinasikan keberhasilan suatu struktur untuk bertahan hidup *(survival of the fittest)* dengan pengubahan informasi dari struktur tersebut secara random untuk membentuk suatu mekanisme penelusuran seperti yang dilakukan dalam proses pembentukan manusia atau mahkluk hidup (gen/sifat yang diturunkan).

Dalam Algoritma Genetika dikenal adanya *chromosome* yaitu kandidat penyelesaian yang digambarkan dengan urutan *binary digits* atau *integers* sesuai kondisi yang dikehendaki. *Chromosome* terdiri dari *gen-gen* yang melambangkan ciri-ciri unik dari *chromosome* tersebut sedangkan nilai yang berkaitan dengan ciri-ciri yang dilambangkan oleh *gen* tertentu disebut *allele*. Populasi adalah sekumpulan *chromosome* yang akan diperbaharui pada setiap generasi.

Dikenal pula adanya dua operator yang memegang peranan penting dalam Algoritma Genetika yaitu *crossover operator* dan *mutation operator*. *Crossover operator* adalah operator yang menggabungkan dua *chromosome* sehingga menghasilkan *child chromosome* yang mewarisi ciri-ciri dasar dari *parent chromosome* sedangkan *mutation operator* dipergunakan untuk memperkenalkan transformasi (informasi) baru pada populasi yang melibatkan perubahan *chromosome* secara random. 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 Heuristic NEH (**Nawaz et al,** 1983) dan *total flowtime minimizing* Algoritma Heuristic RC (**Rajendran and Chaudhuri**, 1992).

Crossover operator yang digunakan adalah *Partially Matched Crossover* (PMX) yang ditemukan oleh **Goldberg and Lingle** (1985), operator ini memberikan pemisah antara posisi *job-job* dengan garis vertikal seperti contoh berikut ini , misalnya bilangan random yang dihasilkan adalah 2 dan 4 maka garis vertikal akan diletakkan pada posisi ke-2 dan posisi ke-4 *chromosome parent.*

P_1 : 1 | 4 5 | 2 3 $\quad$ C_1 : 4 | 1 2 | 5 3
P_2 : 3 | 1 2 | 5 4 $\quad$ C_2 : 3 | 4 5 | 2 1

Untuk membangun *Child 1* (C_1), maka dilihat pemisahan pada *Parent 2* (P_2) untuk mencari *job* mana yang akan ditukar dengan *job* pada *Parent 1* (P_1). Ternyata *job* 4 dan *job* 1, *job* 5 dan *job* 2 ditukar dengan *Parent 1* untuk menghasilkan *Child 1* dan demikian pula untuk menghasilkan *Child 2* (C_2). Sedangkan cara mengevaluasi *chromosome* yang tidak sesuai dengan fungsi tujuan dan menggantikannya dengan *chromosome* yang potensial menggunakan operator Delta. Apabila terdapat dua *chromosome* (urutan/ *sequence*) yaitu S_1 dan S_2 dengan *makespan* MS_1 dan MS_2, *total flowtime* FT_1 dan FT_2 serta *machine idletime* IT_1 dan IT_2; serta bobot kepentingannya atau bobot biaya relatif dari *makespan, total flowtime*, *machine idletime* adalah w_1, w_2, dan w_3 ($w_1+w_2+w_3 = 1$).

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

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

$$Delta = e_1 - e_2 \qquad (26)$$

Jika Delta ≤ 0 maka S_1 (*Sequence* 1) mempunyai *makespan, total flowtime* dan *machine idletime* yang lebih baik dibandingkan S_2, sedangkan jika Delta > 0 maka S_2 (*Sequence* 2) yang lebih baik daripada S_1. Bobot *makespan* (w_1), bobot *total flowtime* (w_2), dan bobot *machine idletime* (w_3) pada tugas akhir ini mempunyai nilai yang sama yaitu 0,333 karena tujuan yang ingin dicapai adalah meminimumkan *makespan, total flowtime* dan *machine idletime* secara bersama-sama (tidak mementingkan salah satu kriteria).

Operator Delta sebagai *replacement policy* digunakan untuk menentukan apakah suatu *chromosome* keturunan (*child*) menggantikan *chromosome parent* pada generasi berikutnya, hal ini penting karena *replacement policy* ini akan membuang *choromosome* yang bermutu rendah dalam proses selanjutnya. *Chromosome* terbaik (G) akan diidentifikasi pada kedua sub populasi setiap akhir generasi seperti yang dilakukan pada generasi yang pertama, hal tersebut dilakukan dengan mengambil 2 *chromosome* setiap kali dan menggunakan operator Delta. Namun ada hal yang harus dicatat bahwa tidak ada suatu jadual (*schedule*) yang akan mendominasi jadual-jadual lainnya dalam hal *makespan, total flowtime,* dan *machine idletime*. Selain itu tidak ada suatu jadual yang akan menghasilkan *makespan* yang optimal, *total flowtime* yang optimal dan *machine idletime* yang optimal. Prosedur pengerjaan Algoritma Genetika adalah sebagai berikut :

Step 0. Mengumpulkan data-data : n job, m mesin, serta Tij

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 flowtime* terkecil.

Step 3. Mencari urutan (*chromosome*) terbaik dari Subpopulasi 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

6. ALGORITMA HEURISTIC HC

Algoritma heuristic baru yang diusulkan oleh **Ho and Chang** (1991) adalah pada setiap penelitian yang dilakukan selalu didapati bahwa pada mesin 1 (M_1) tidak ditemukan adanya *idletime* diantara *job-job* yang telah dijadualkan. Dalam problem 3 mesin dengan *k* sebagai *job* terakhir dalam jadual maka *gap* 1 didefinisikan sebagai waktu antara berakhirnya *job k* pada M_1 dan dimulainya *job k* pada mesin 2 (M_2). Demikian pula *gap* 2 didefinisikan sebagai waktu berakhirnya *job* k pada M_2 dan dimulainya *job* k pada mesin 3 (M_3) sehingga *makespan* untuk n *job* 3 mesin adalah:

$$\text{Makespan}=\sum_{i=1}^{n} t_{i1} + t_{k2} + t_{k3} + gap1 + gap2 \qquad (27)$$

Oleh karena itu Algoritma HC meminimumkan besarnya *gap* 1 dan *gap* 2 sehingga tidak hanya menghasilkan *makespan* yang minimum tetapi juga *total flowtime* dan *machine idletime* yang minimum pula. Besarnya gap-gap yang ditimbulkan oleh suatu jadual dapat dihitung dengan menggunakan rumus:

$$d_{ij}^{k} = t_{i,k+1} - t_{j,k} \quad \text{untuk i, j} = 1,2.....,\text{n ; k} = 1,2,.....,\text{m - 1 dan i} \neq \text{j} \qquad (28)$$

Jika *job* i diikuti *job* j pada jadual, maka nilai d_{ij}^{k} positif mempunyai arti bahwa *job* j harus menunggu pada mesin k+1 setidak-tidaknya selama d_{ij}^{k} sampai *job* i selesai. Sedangkan nilai d_{ij}^{k} negatif mempunyai arti bahwa terdapat *idletime* antara *job* i dan *job* j pada mesin k+1. Untuk meminimumkan gap-gap maka *job-job* dengan gap paling negatif diletakkan pada akhir jadual karena bila diletakkan pada awal jadual maka kemungkinan besar akan terbuang percuma (misal: menjadi *idletime*) sehingga *job-job* dengan *gap* paling positif diletakkan pada awal jadual. Suatu faktor diperlukan untuk mengurangi *(discount)* nilai *gap* negatif yang disebut dengan faktor (k). Dibawah ini merupakan rumusan faktor (k):

$$\text{faktor(k)} = ((1.0\text{-}0.1)/(\text{m-}2)) \text{ x } (\text{m-k-}1) + (0.1) \qquad (29)$$

Dengan k = 1,2,…..,m-1, dan m adalah jumlah mesin, bobot yang lebih tinggi (factor (k) mendekati 1) diberikan pada mesin-mesin awal atau k yang kecil sedangkan bobot yang lebih rendah (faktor (k) mendekati 0) pada mesin-mesin terakhir atau k yang besar. Sebagai contoh, untuk k = 1 maka faktor(k) = 1.0; untuk k = m-1 maka faktor (k) = 0.1 dan untuk k = 2,…..,m-2 maka faktor (k) merupakan interpolasi linear antara 1.0 dan 0.1 sehingga fungsi *discount* dapat didefinisikan sebagai berikut:

$$\delta_{ij}^{k} = \begin{cases} \text{faktor(k)} & \text{jika } d_{ij}^{k} < 0 \\ 1 & \text{lainnya} \end{cases}$$

i,j = 1,2,…..,n ; k = 1,2,…,m-1

Dengan menggabungkan semua d_{ij}^{k} dan *factor discount* maka didapatkan *overall revised gaps* dinotasikan sebagai d_{ij}^{R}.

$$d_{ij}^{R} = \sum_{k=1}^{m-1} d_{ij}^{k} \boldsymbol{d}_{ij}^{k} \quad \text{untuk i, j} = 1,2,.....,\text{n} \qquad (30)$$

7. CONTOH APLIKASI

Pabrik "X" mempunyai 8 jenis produk yang melalui 5 jenis mesin yang sama dan total waktu proses *job* i pada pada mesin j teringkas dalam tabel ini.

Tabel 1. Total waktu proses *job* i pada mesin per operator : T_{ij} (detik)

Job	M_1	M_2	M_3	M_4	M_5
1	25454	27682	24186	34105	27490
2	27251	32743	30462	28787	30064
3	78915	83514	80166	73606	80318
4	86231	93211	70121	58079	92064
5	29889	31401	32523	32169	32131
6	22997	25701	26020	26612	26334
7	78612	39886	43124	41589	81240
8	80325	43389	43322	42702	85246

Langkah pertama yang harus dilakukan adalah mencari populasi awal dengan menggunakan agoritma heuristic NEH dan RC.

Tabel 2. Hasil dari algoritma heuristic NEH & RC

Algoritma	Jadual	*Performance*
NEH	6-5-4-2-1-3-7-8	MS = 488670
RC	6-5-2-1-3-8-4-7	FT =1699354

Algoritma Genetika

Dari hasil pengerjaan penjadualan berdasarkan algoritma heuristic NEH dan RC didapatkan populasi awal seperti dalam tabel 3.

Tabel 3. Populasi awal (NGEN = 0)

No.	Subpopulasi 1	Subpopulasi 2
1	6-5-4-1-2-3-7-8	6-5-2-1-3-8-7-4
2	6-5-4-2-1-3-7-8	6-5-2-1-3-8-4-7
3	6-5-4-2-1-7-3-8	6-5-1-2-3-8-4-7
4	5-6-4-2-1-3-7-8	6-2-5-1-3-8-4-7
5	6-5-4-2-3-1-7-8	5-6-2-1-3-8-4-7
6	6-5-4-2-1-3-8-7	6-5-2-1-8-3-4-7
7	6-4-5-2-1-3-7-8	6-5-2-1-3-4-8-7
8	6-5-2-4-1-3-7-8	6-5-2-3-1-8-4-7

Dari populasi awal ini dicari *chromosome* terbaik dengan menggunakan operator Delta, didapatkan *chromosome* terbaik (G) adalah 6-5-2-1-3-8-4-7 dengan MS = 480699.64, FT = 1699350, dan IT = 256226. NGEN bertambah menjadi 1 dan populasi awal akan diperbaharui menggunakan *crossover operator* yang disebut *Partially Matched Crossover* (PMX) sehingga didapatkan populasi baru seperti dalam Tabel 4.

Tabel 4. Populasi 1 (NGEN = 1)

No.	Subpopulasi 1	Subpopulasi 2
1	6-5-4-1-2-3-7-8	6-5-2-1-3-8-7-4
2	6-5-4-2-1-3-7-8	6-5-2-1-3-8-4-7
3	6-5-1-2-4-7-3-8	6-5-1-2-3-8-4-7
4	5-6-4-2-1-3-7-8	6-2-5-1-3-8-4-7
5	6-5-4-2-3-1-7-8	5-6-2-1-3-8-4-7
6	6-5-8-2-1-3-4-7	6-5-2-1-8-3-4-7
7	6-5-2-4-1-3-7-8	6-5-2-1-3-4-8-7
8	6-5-2-3-1-4-7-8	6-5-2-3-1-8-4-7

NGEN bertambah lagi menjadi 2 dan populasi 1 akan diperbaharui menggunakan *crossover operator* sehingga didapatkan populasi 2, juga akan dicari *chromosome* terbaik (G) dan demikian seterusnya. Proses ini akan diulang hingga NGEN = 10 dan untuk mendapatkan populasi 10, bukan *crossover operator* lagi yang bekerja melainkan operator lainnya yaitu *mutation operator.* Misalnya untuk *chromosome* pertama dalam subpopulasi 1 yaitu 6-5-2-1-4-3-7-8 dengan random number 8 dan 7 maka job pada posisi ke-8 dan ke-7 akan ditukar, yang akan menghasilkan *chromosome* baru : 6-5-2-1-4-3-8-7. Hal ini dilakukan pada seluruh *chromosome* yang ada populasi 9 sehingga terbentuklah populasi 10.

Langkah selanjutnya adalah membentuk populasi 11 (NGEN = 11) dengan menggunakan *crossover operator* kembali dan mendapatkan *chromosome* terbaik (G). Pada saat NGEN = 12 maka pembentukan populasi baru dengan menggunakan salah satu operator genetik dihentikan. Dari *chromosome* terbaik (G) yang terakhir, didapatkan 8 *chromosome* baru dengan menukarkan *gen* ke-k dan *gen* ke-k+1 yang dapat dilihat pada Tabel 4. Diantara kedelapan *chromosome* baru tersebut dipilih *chromosome* terbaik dengan menggunakan operator Delta dan akan menjadi hasil akhir dari Algoritma Genetika ini. *Chromosome* terbaik sebagai hasil akhir dari Algoritma Genetika adalah 6-5-2-1-3-7-4-8 dengan *makespan* sebesar 483334 detik, *total flowtime* sebesar 1699242 detik dan *machine idletime* sebesar 252390 detik.

Tabel 5. Populasi akhir

No.	Chromosome
1	6-5-2-1-3-7-4-8
2	5-6-2-1-3-7-4-8
3	6-2-5-1-3-7-4-8
4	6-5-1-2-3-7-4-8
5	6-5-2-3-1-7-4-8
6	6-5-2-1-7-3-4-8
7	6-5-2-1-3-4-7-8
8	6-5-2-1-3-7-8-4

Algoritma Pembanding

Dalam menyusun penjadualan menggunakan algoritma heuristic HC, maka yang harus dilakukan dahulu adalah mendapatkan *gap* yang dinotasikan dengan d_{ij}^{k} (untuk i,j = 1,2,…..,8; k = 1,2,3,4 dan i ≠ j). Pencarian d_{ij}^{k} dengan cara:

- Mencari *idle time* jadual i-j pada mesin ke-k+1 sebagai nilai d_{ij}^{k} yang negatif (untuk i,j = 1,2,…..8; k = 1,2,3,4 dan i ≠ j), jikalau ada.
- Mencari *waiting time* jadual i-j pada mesin ke-k+1 sebagai nilai d_{ij}^{k} yang positif (untuk i,j = 1,2,…..8; k = 1,2,3,4 dan i ≠ j), jikalau ada.

Untuk mencari *idle time* job i yang diikuti job j pada mesin ke-k+1 menggunakan rumus:

$$d_{ij}^{k} = -(S_{j,(k+1)} - E_{i,(k+1)}) \text{ untuk } k = 1,2,3,4 \quad (31)$$

sebagai contoh

$$d_{12}^{1} = -(S_{2,2} - E_{1,2}) = 27687.06 - 27687.06 = 0$$

Sedangkan untuk *waiting time*, pada kasus penjadualan ini tidak ditemukan sehingga yang ada hanyalah d_{ij}^{k} yang negatif.

Langkah selanjutnya adalah menghitung faktor (k) (untuk k= 1,2,3,4):

- factor(1) = 1 factor(2) = 0.7
- factor(3) = 0.4 factor(4) = 0.1

Kemudian didapatkan *overall revised gaps,* d_{ij}^{R} dengan menggunakan persamaan (31). Sebagai contoh, untuk i = 1 & j = 2:

$$d_{12}^{R} = \sum_{k=1}^{4} d_{12}^{k} d_{12}^{k}$$
$$= (0 \times 1) + (-2280.56 \times 0.7) + (0 \times 1) + (-3.55 \times 0.1)$$
$$= -1596.75$$

Solusi awal yang digunakan ada dua, disesuaikan dengan Algoritma Genetika yaitu jadual yang merupakan hasil algoritma heuristic NEH dan algoritma heuristic RC. Dari masing-masing solusi awal ini akan didapatkan solusi akhir, dipilih yang terbaik dengan menggunakan operator delta.

Tabel 6. Hasil Algoritma RC

Solusi Awal	Hasil	*Makespan*	*Flowtime*	*Idletime*
Heuristic NEH	6-5-1-4-2-3-7-8	488858	1849020	257915
Heuristic RC	6-5-2-1-3-8-4-7	480700	1699350	256226

8. KESIMPULAN

1. Penjadualan *flowshop* menggunakan Algoritma Genetika yang bertujuan meminimumkan *makespan, total flowtime* dan *machine idletime* secara bersama-sama maka didapatkan penjadualan job : 6-5-2-1-3-7-4-8 dengan *makespan* 483334 detik, *total flowtime* 1699242 detik dan *machine idletime* sebesar 252390 detik.
2. Penjadualan menggunakan Algoritma Heuristic HC yang mempunyai tujuan yang sama dengan Algoritma Genetika maka didapatkan penjadualan job : 6-5-2-1-3-8-4-7 dengan *makespan* 480700 detik, *total flowtime* 1699350 detik dan *machine idletime* sebesar 256226 detik.

3. Dari kedua hasil yang dikemukakan diatas, jelaslah bahwa hasil penjadualan yang didapatkan oleh Algoritma Genetika tersebut lebih unggul daripada hasil penjadualan oleh Algoritma Heuristic HC apabila dipandang dari pencapaian ketiga kriteria secara bersama-sama.

DAFTAR PUSTAKA

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

Ho, J. C., and Chang, Y. L., 1991. "A new heuristic for the n-job, m-machine flowshop problem", *European Journal of Operational Research*, No. 52, 194-202.

Michalewicz, Zbigniew, 1992. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag.

Nawaz, M., Enscore, E. E., and Ham, I., 1983. "A heuristic algorithm for the m-machine, n-job flowshop sequencing problem", *OMEGA*, No. 11, 91-95.

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

Rajendran, C., and Chaudhuri, D., 1992. "An efficient heuristic approach to the scheduling of jobs in a flowshops", *European Journal of Operational Research*, No. 61, 318-325.

Sridhar, J., and Rajendran, C., 1996. "Scheduling in flowshop and cellular manufacturing systems with multiple objectives – a genetic algorithmic approach", *Production Planning & Control*, Vol. 7, No. 4, 374-382.