**Model Penyelesaian *Job Shop* Scheduling *Problem***

**Menggunakan Metode *Local Search Algorithm* Dengan *Cross Over***

**Amiluddin Zahri1**

**Dosen Universtas Bina Darma**

**Jalan Jenderal Ahmad Yani No.3 Palembang**

**Sur –el:** [**amiluddin@binadarma.ac.id**](mailto:amiluddin@binadarma.ac.id1)

***Absract****: Job shop scheduling problem is one of the scheduling problem with sequence constraint processing tasks, and each task has to go through each machine exactly once. Many experiments have been done in solving job shop scheduling problem using genetic algorithms. In this case the researchers conducted experiments using a genetic algorithm to obtain a model of the completion of job shop scheduling problem using the local search algorithm with crossover using matlab 7.0.1 software. From experiments conducted at three case studies of randomly selected scheduling models obtained the best scheduling and the best processing time, the fitness value of 0.1429 for case 1, case 2 of 0.0625, and 0.0833 for the case 3.*

*Key words: Scheduling, Genetic Algorithms, software matlab 7.0.1*

**Abstrak:** *Job shop scheduling problem* merupakan salah satu masalah penjadwalan yang memiliki kendala urutan pemrosesan tugas, dan setiap tugas harus melalui setiap mesin tepat satu kali. Banyak percobaan yang telah dilakukan dalam menyelesaikan *job shop scheduling problem* dengan menggunakan metode algoritma genetik. Dalam hal ini peneliti melakukan eksperimen menggunakan algoritma genetika untuk mendapatkan suatu model penyelesaian *job shop* *scheduling problem* menggunakan metode *local search algorithm* dengan *crossover* menggunakan *software* matlab 7.0.1*.* Dari eksperimen yang dilakukan pada 3 studi kasus yang dipilih secara acak didapat model-model penjadwalan terbaik dan waktu proses terbaik, dengan nilai *fitness* untuk kasus 1 sebesar 0,1429, kasus 2 sebesar 0,0625, dan kasus 3 sebesar 0,0833.

Kata kunci : Penjadwalan, Algoritma genetik, *software* matlab 7.0.1

1. **PENDAHULUAN**

Penjadwalan produksi di dalam dunia industri manufaktur maupun agroindustri memiliki peranan penting sebagai bentuk pengambilan keputusan. Dalam suatu industri penjadwalan diperlukan dalam mengalokasikan tenaga operator, mesin dan peralatan produksi, urutan proses, jenis produk, pembelian material, dan sebagainya. Pada suatu lembaga pendidikan, penjadwalan diperlukan untuk mengalokasikan ruang kelas, peralatan mengajar, tenaga pengajar, staf administasi, pendaftaran penerimaan mahasiswa baru, dan sebagainya. Tujuan dari masalah penjadwalan antara lain: meminimumkan waktu penyelesaian semua tugas (*makespan*), meminimumkan keterlambatan pengerjaan, meminimumkan waktu tunggu pada mesin, meminimumkan biaya, dan lain-lain. (Santoso, 2010 : 1)

*Job shop scheduling problem* merupakan salah satu masalah penjadwalan yang memiliki kendala urutan pemrosesan tugas, dan setiap tugas harus melalui setiap mesin tepat satu kali. Terdapat dua jenis metode yang biasa digunakan untuk menyelesaikan masalah *job shop scheduling problem*. Metode eksak, seperti pemrograman linier dan pemrograman non-linier, dapat digunakan untuk ukuran *job shop scheduling problem* yang kecil. Sedangkan untuk ukuran masalah yang besar, digunakan suatu pendekatan secara aproksimasi, seperti *local search, simulated annealing, genetic algorithm*, *tabu search*, dan *ant colony optimization* (<http://gudangmakalah.blogspot.com>, 11 maret 2012 : 15.09).

Algoritma genetik ditemukan oleh John Holland pada tahun 1960. Algoritma ini menerapkan suatu proses evolusi biologi. Banyak percobaan dalam menyelesaikan *job shop scheduling problem* dengan menggunakan metode algoritma genetik, tetapi masih terdapat beberapa percobaan yang menghasilkan solusi yang tidak layak. Pada metode dalam penelitian ini, ketidak layakan dari solusi dapat dihindari dengan menggunakan suatu skema yang menjaga urutan pemrosesan tugas. Kualitas dari solusi akan diuji dengan menggunakan beberapa masalah uji yang biasa dipakai untuk menyelesaikan *job shop scheduling problem.*

Terdapat tiga operator dasar yang digunakan pada algoritma genetik, yaitu reproduksi, *crossover*, dan mutasi. Reproduksi digunakan untuk menyeleksi solusi-solusi yang akan diproses, *crossover* digunakan untuk memperoleh solusi-solusi melalui proses perkawinan, dan mutasi digunakan untuk mengubah kualitas dari suatu solusi. Pada penelitian ini, akan digunakan suatu operator *crossover* yang sederhana untuk mendapatkan solusi baru yang layak. Dari uraian diatas, maka perlu dilakukan penelitian eksperimen untuk mendapatkan suatu “ Model Penyelesaian *Job Shop* *Scheduling Problem* Menggunakan Metode *Local Search Algorithm* Dengan *Cross Over* “

1. METODOLOGI PENELITIAN

Baker dalam Satrio (2007) mendefinisikan penjadwalan sebagai proses pengalokasian sumber-sumber yang ada untuk menjalankan sekumpulan tugas dalam jangka waktu tertentu. Penjadwalan merupakan kegiatan yang penting dalam perencanaan dan pengendalian produksi karena penjadwalan yang baik akan memaksimumkan efektivitas pemanfaatan setiap sumber daya yang ada. Penjadwalan digambarkan sebagai alokasi sumber daya dari waktu ke waktu untuk melaksanakan suatu kumpulan pekerjaan. Penjadwalan merupakan tahap terakhir dari perencanaan sebelum dilaksanakannya proses produksi. Selain itu penjadwalan merupakan penjabaran kegiatan-kegiatan yang direncanakan yaitu yang berisikan kapan dimulainya kegiatan produksi sehingga perencanaan kebutuhan yang ditetapkan dapat dipenuhi pada waktunya (Gen dan Cheng di dalam Satrio, 2007 : 21)

* 1. **Lokasi dan Waktu**

Tempat penelitian dilakukan di Laboratorium Sistem Produksi Universitas Bina Darma alamat Jln A.Yani No. 3 Plaju kota Palembang. Sedangkan waktu penelitian dilaksanakan pada bulan Juli - November 2013.

* 1. **Pengumpulan Data**

Pengumpulan data dilakukan sebagai langkah awal dalam sebuah penelitian sebagai bahan yang mendukung dalam penelitian yang dilakukan. Data yang diperlukan dalam penelitian eksperimen ini yaitu pembangkitan bilangan random dan studi kasus.

Pembangkitan bilangan random dari suatu sample acak dari populaasi dengan menggunakan *Microsoft excel.* Mulai dengan memasukkan data pada kolom yang pertama, kemudian pilih *Tools* dan pilih *Data Analysis* dan pilih *Random Number Generation.* Dari hasil pembangkitan bilangan random di atas, maka *m job* yang digunakan pada penyelesaian *job shop scheduling problem* menggunakan metode *local search algorithm* dengan *cross over* dapat diketahui, yaitu dengan melakukan studi kasus pada 3 mesin.

* 1. **Metode Pengolahan Data**

Setelah melakukan pengumpulan data maka dilakukan pengolahan data sesuai dengan taksiran metode yang akan digunakan. Data yang didapat selanjutnya diselesaikan menggunakan algoritma *gen*etik menggunakan bantuan program matlab. Kasus yang akan diselesaikan menggunakan algoritma *gen*etika yaitu 3 mesin dan (1, 4, 2) *job* yang diselesaikan dalam beberapa tahap.

**2.4. Tinjauan Pustaka**

2.4.1 Penjadwalan Tipe *Job Shop*

Salah satu model penjadwalan yang cukup banyak ditemui adalah penjadwalan *job shop*. Pada model ini, *job-job* yang diproses biasanya memiliki rute dan waktu pemrosesan yang berbeda-beda satu sama lain. Setiap *job* terdiri dari operasi-operasi yang masing-masing diproses di satu mesin tertentu karena rute proses dari setiap *job* berbeda, maka kemungkinan jadwal yang dapat dihasilkan dan kendala yang muncul menjadi sangat banyak. Akibatnya model ini menimbulkan masalah penjadwalan yang kompleks dalam arti bahwa semua kendala dalam penjadwalan harus terpenuhi dan dalam arti bahwa jadwal yang bisa memenuhi fungsi tujuan penjadwalan secara optimal sangat sulit ditemukan di antara sekian banyak kemungkinan (Bagchi di dalam Satrio, 2007 : 24).

Penjadwalan pada proses produksi tipe *job shop* lebih sulit dibandingkan penjadwalan *flow shop.* Hal ini disebabkan oleh 3 alasan: (Nasution dan prasetyawan, 2008 : 358-359)

1. *Job shop* menangani variasi produk yang sangat banyak, dengan pola aliran yang berbeda-beda melalui pusat-pusat kerja.
2. Peralatan pada *job shop* digunakan secara bersama-sama oleh bermacam-macam *order* dalam prosesnya, sedangkan peralatan pada *flow shop* digunakan khusus hanya untuk satu jenis produksi.
3. *Job-job* yang berbeda mungkin ditentukan oleh prioritas yang berbeda pula. Hal ini mengakibatkan *order* tertentu yang dipilih harus diproses seketika pada saat order tersebut ditugaskan pada suatu pusat kerja. Sedangkan pada *flow shop* tidak terjadi permasalahan seperti diatas karena keseragaman output yang diproduksi untuk persediaan. Prioritas *order* pada *flow shop* dipengaruhi terutama pada pengirimannya dibandingkan tanggal pemrosesan.

Faktor-faktor tersebut menghasilkan sangat banyak kemungkinan kombinasi dari pembebanan (*loading*) dan urutan-urutan (*sequencing*). Perhitungan dari indentifikasi dan evaluasi jadwal-jadwal yang mungkin menjadi sulit sehingga banyak perhatian diarahkan pada riset penjadwalan *job shop*. Selain itu, persiapan suatu penjadwalan *job shop*, penyesuaian dan pembaharuannya membutuhkan investasi yang besar untuk fasilitas computer. (Rosnaini Ginting, 2009 : 53)

Penjadwalan *job shop* merupakan masalah penempatan sejumlah *n-job* pada sejumlah *m-mesin* dengan urutan proses. Setiap *job* dirancang sebagai satu *set* operasi dan mesin yang telah di-*set* untuk setiap operasinya.

Penjadwalan *job-shop* memiliki beberapa definisi yang meliputi : (Kusuma di dalam Satrio, 2007 : 25)

1. Jadwal *fisibel*, yaitu jadwal dimana seluruh operasi dari semua pekerjaan telah ditugaskan dan ketentuan rute operasi telah terpenuhi (tidak ada *overlap* antar operasi).
2. Jadwal semi-aktif, yaitu sekumpulan jadwal *fisibel* dimana tidak satu pun operasi dapat dijadwalkan lebih awal tanpa mengubah susunan operasi pada mesin.
3. Jadwal aktif yaitu sekumpulan jadwal *fisibel* dimana tidak satu pun operasi dapat dipindahkan lebih awal tanpa menunda operasi lain.
4. Jadwal *non-delay*, yaitu sekumpulan jadwal *fisibel* dimana tidak satu pun mesin dibiarkan menganggur jika pada saat yang bersamaan terdapat operasi yang memerlukan mesin tersebut.

Berdasarkan ada tidaknya kendala masalah penjadwalan *job-shop* dapat dibagi menjadi dua tipe yaitu *unconstrained* *job shop* (tanpa kendala) dan *constrained job shop* (dengan kendala). Pada tipe *unconstrained job shop*, tidak ada kendala-kendala yang dapat menghambat proses implementasi pekerjaan pada setiap mesin produksi sehingga semua alternatif jadwal bersifat legal.

Batas Penentuan *Job Shop* Mesin

Setiap operasi dapat dikarakterisasi dengan mesin tertentu dan waktu proses produksi yang telah ditentukan. Untuk setiap proses produksi dapat digunakan batasan dalam menentukan *job* dan mesin yang akan digunakan, yaitu : (Gen dan Cheng di dalam Satrio, 2007 : 24)

1. Satu pekerjaan tidak akan menggunakan mesin yang sama sebanyak lebih dari satu kali

2. Tidak adanya batasan yang lebih tinggi pada saat operasi dengan pekerjaan yang berbeda

3. Sebuah operasi tidak akan bisa disela dengan operasi yang lain

4. Sebuah mesin hanya akan melakukan sebuah pekerjaan dalam satu waktu

5. Tidak adanya waktu atau pekerjaan yang dispesifikasi

Karakteristik penjadwalan produksi tipe *job-shop* adalah penggunaan mesin oleh lebih dari satu pekerjaan sehingga ada keterbatasan waktu penggunaan. Berbeda dengan masalah penjadwalan seri dan parallel, karakteristik pekerjaan yang harus diselesaikan pada penjadwalan tipe *job-shop* harus melewati beberapa mesin (*routing*) dan tiap rute yang ditempuh masing-masing pekerjaan berlainan.

2.4.2 Bilangan Random

Dalam kehidupan sehari-hari, tidak ada seorangpun yang mampu memprediksi kejadian yang terjadi pada waktu yang akan datang. Segala hal yang berhubungan dengan masa depan, lebih mengedepankan kepada prediksi atas kejadian. Tidak bersifat benar tapi merupakan pembenaran atas suatu kemungkinan yang ada. Apakah besok akan sakit, mendapat hadiah, memperoleh jodoh dan lain sebagainya merupakan sifat prediktif. Tetapi manusia diberikan kemampuan untuk menganalisis kejadian-kejadian yang lalu untuk kemudian memberikan perkiraan-perkiraan dimasa depan. Perkiraan-perkiraan tersebut akan dipengaruhi oleh suatu ‘nilai’ untuk memberikan pola pada suatu kejadian. ‘Nilai’ ini didalam proses simulasi dinamakan bilangan acak atau bilangan random. (Arifin, 2009 : 72)

Bilangan acak atau bilangan random, pada dasarnya ingin meniru kejadian yang tak pasti dialami oleh manusia. Karena pada dasarnya pembangkitan bilangan random tidak ada yang bisa memprediksi berapa nilai yang akan keluar dimasa yang akan datang. Semakin besar *digit* random yang diminta, maka semakin tinggi tingkat prediksinya. (Arifin, 2009 : 72)

2.4.3 Algoritma Genetika

Algoritma genetika ditemukan oleh John H. Holland dari University of Michigan yang memulai penelitiannya pada awal tahun 1960. Penelitian pertamanya yang dipublikasikan adalah “*Adaptation in Natural and Artificial systems*” pada tahun 1975. Menurut Holland “Apabila evolusi dapat bekerja dengan sangat baik untuk organisme, mengapa tidak dapat digunakan untuk program komputer?”. Penelitian Holland menyimpulkan dua hal yaitu untuk menjelaskan dan mempelajari proses adaptasi sistem alami, dan untuk mendesain atau merancang sistem cerdas yang mempunyai persamaan atau mengandung mekanisme dengan sistem yang alami. Penerapan algoritma genetika terutama dikaitkan dengan metode adaptif untuk memecahkan masalah pencarian optimasi. Teori dasarnya ialah genetik bawaan dari populasi yang ada secara potensial memiliki solusi, atau solusi yang lebih baik, terhadap masalah yang akan dihadapi. Solusi ini belum aktif karena kombinasi genetik yang dialami terpecah dalam beberapa subjek. Algoritma genetika merupakan cabang dari algoritma evolusi yang merupakan metode adaptif yang biasa digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah optimasi. Algoritma genetika adalah teknik pencarian stokastik yang bekerja berdasarkan mekanisme seleksi alami dan genetika alami. Dan setiap individu dalam populasi disebut kromosom, yang direpresentasikan sebagai suatu solusi awal terhadap masalah yang sedang ditemui (mitsuo dalam Berlianty dan Arifin, 2010 : 99-100)

Pada algoritma genetika, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi disebut dengan istilah kromosom. Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut dengan generasi. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut dengan fungsi *ﬁtness*. Nilai *ﬁtness* dari suatu kromosom akan menunjukkan kualitas dari kromosom dalam populasi tersebut. Generasi berikutnya dikenal dengan istilah anak (*oﬀspring*) terbentuk dari gabungan dua kromosom generasi sekarang yang bertindak sebagai induk (*parent*) dengan menggunakan operator penyilangan (*crossover*). Selain operator penyilangan, suatu kromosom dapat juga dimodiﬁkasi dengan menggunakan operator mutasi. Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai *ﬁtness* dari kromosom induk (*parent*) dan nilai *ﬁtness* dari kromosom anak (*oﬀspring*), serta menolak kromosom-kromosom yang lainnya sehingga ukuran populasi (jumlah kromosom dalam suatu populasi) konstan. Setelah melalui beberapa generasi, maka algoritma ini akan *konvergen* ke kromosom terbaik (Sanjoyo, 2006 : 4-5).

Ada tiga keunggulan dari aplikasi Algoritma Genetika dalam proses optimasi, yaitu: (a) Algoritma Genetika tidak terlalu banyak memerlukan persyaratan matematika dalam penyelesaian proses optimasi. Algoritma Genetika dapat diaplikasikan pada beberapa jenis fungsi obyektif dengan beberapa fungsi pembatas baik berbentuk linier maupun non-linier: (b) Operasi evolusi dari Algoritma Genetika sangat efektif untuk mengobservasi posisi global secara acak: dan (c) Algoritma Genetika mempunyai ﬂeksibilitas untuk diimplementasikan secara eﬁsien pada problematika tertentu (Sanjoyo, 2006 : 5).

1. **PEMBAHASAN DAN HASIL**

**3.1 Pengumpulan data**

Data eksperimen yang diambil dalam penelitian ini yaitu pembangkitan bilangan random dan studi kasus. Pembangkitan bilangan random dari suatu sampel acak dari populasi dengan menggunakan *Microsoft excel.* Mulai dengan memasukkan data pada kolom yang pertama, kemudian pilih *Tools* dan pilih *Data Analysis* dan pilih *Random Number Generation.* Dari hasil pembangkitan bilangan random di atas, maka *m job* yang digunakan pada penyelesaian *job shop scheduling problem* menggunakan metode *local search algorithm* dengan *cross over* dapat diketahui, dengan melakukan studi kasus pada 3 mesin.

**3.2 Analisis Data**

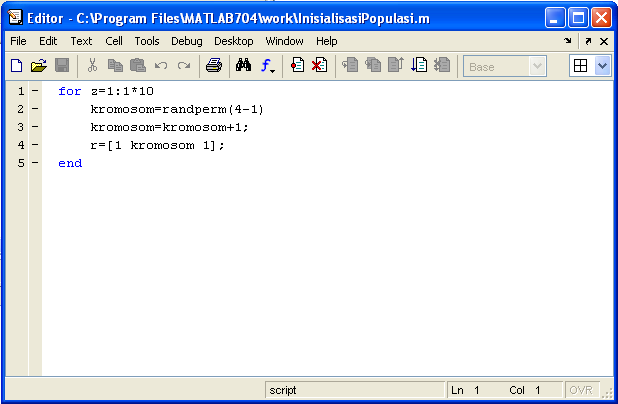
Setelah melakukan pengumpulan data maka dilakukan pengolahan data sesuai dengan taksiran metode yang akan digunakan. Data yang didapat selanjutnya diselesaikan menggunakan algoritma *gen*etik menggunakan bantuan program matlab. Kasus yang akan diselesaikan menggunakan algoritma *gen*etika yaitu 3 mesin dan (1, 4, 2) *job* yang diselesaikan dalam beberapa tahap.

**3.2.1. Kasus 1 (3 Mesin 1 *Job*)**

Dalam kasus ini akan dibangkitkan suatu populasi yang berisi sejumlah *kromosom* yang berisi sejumlah *gen* didalamnya untuk mendapatkan suatu model penyelesaian dengan nilai *fitness* terbaik yang dilakukan dengan tahap-tahap sebagai berikut:

**- Inisialisasi Populasi**

Inisialisasi populasi dilakukan untuk membangkitkan sejumlah  *kromosom*, yang berisi beberapa model pengerjaan *job* (*Kromosom*). Inisialisasi populasi dilakukan secara random dengan menggunakan MATLAB 7.0.1



Hasil inisialisasi populasi yang muncul pada *command window* adalah sebagai berikut:

*Kromosom* 1 = 2 3 1

*Kromosom* 2 = 1 3 2

*Kromosom* 3 = 2 1 3

*Kromosom* 4 = 1 2 3

*Kromosom* 5 = 3 2 1

*Kromosom* 6 = 1 3 2

*Kromosom* 7 = 3 1 2

*Kromosom* 8 = 3 1 2

*Kromosom* 9 = 1 3 2

*Kromosom* 10 = 3 2 1

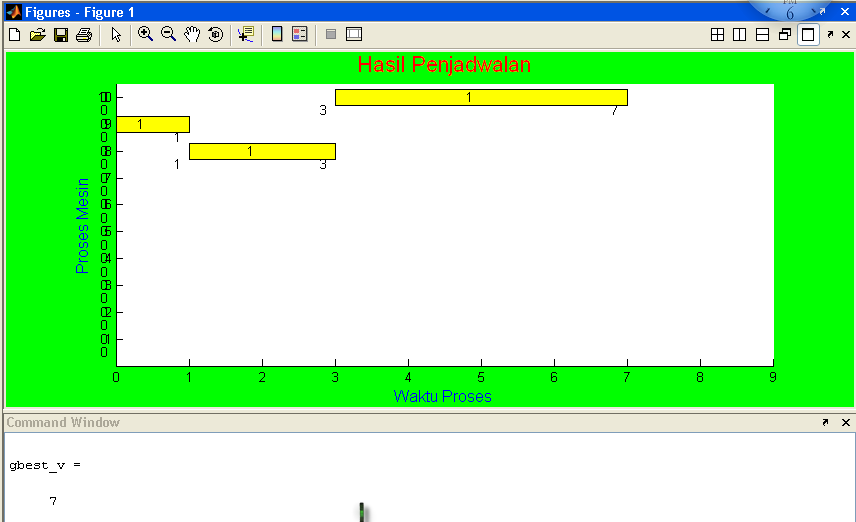
Ket : Kromosom = model penjadwalan

**3.2.2 Pemilihan Model Terbaik**

Pemilihan model terbaik dilakukan untuk memilih satu dari 10 *kromosom* acak yang dianggap paling optimal sebagai dasar pengoptimalan *makespan* dari model penyelesaian 3 mesin 1 *job*. Pemilihan model terbaik dari *makespan* terkecil dilakukan menggunakan MATLAB 7.0.1 dengan membuat beberapa kode yang dimasukkan dalam 1 *folder* agar dapat menjalankan *main* kode yang menghasilkan *output* model terbaik. Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Membuat kode yang digunakan untuk menentukan model terbaik.
2. Membuat kode yang digunakan untuk menentukan *makespan*.
3. Membuat kode yang digunakan untuk membuat *gant chart*.
4. Membuat *main* kode yang digunakan untuk menjalankan kode-kode yang telah dibuat

Maka didapat hasil berupa *gant chart* yang menunjukkan model terbaik dan *makespan* akan muncul pada tampilan command windows pada tampilan beikut :

Gambar 3.2 Hasil Penjadwalan Terbaik Untuk

Kasus 1

Dari urutan waktu proses pada gambar diatas maka di dapat model terbaik adalah:

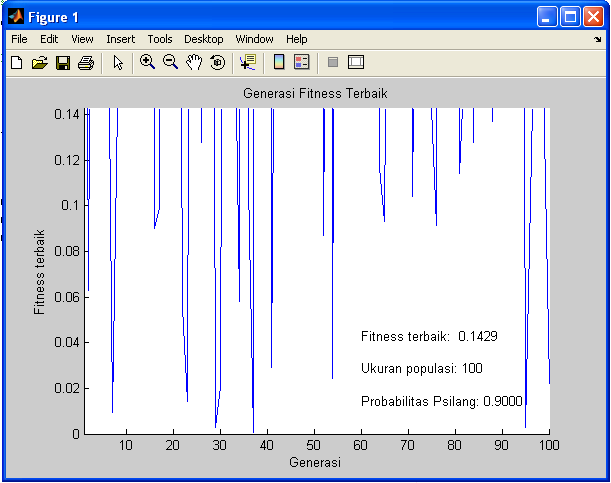
*Kromosom* Terbaik = 2 3 1

**3.2.3 Nilai *Fitness* Terbaik**

Setelah hasil pemilihan model terbaik didapat maka dilakukan pencarian nilai *fitness* terbaik dengan melakukan proses penyilangan (*crossover*) dengan peluang penyilangan (*Pc*) yang digunakan adalah 0,9. Nilai 0,9 berarti diharapkan 90% populasi yang terbentuk pada *gen*erasi selanjutnya adalah hasil penyilangan *gen*erasi sebelumnya. Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Membuat kode yang digunakan untuk memasukkan data-data yang diperlukan dalam pencarian nilai *fitness* terbaik.
2. Membuat kode yang digunakan membuat grafik nilai *fitness*.

Maka didapat hasil berupa grafik yang menunjukkan nilai *fitness* terbaik dalam 100 *gen*erasi seperti gambar berikut:

****

Gambar 3.3 Grafik Nilai Fitness Untuk Kasus 1

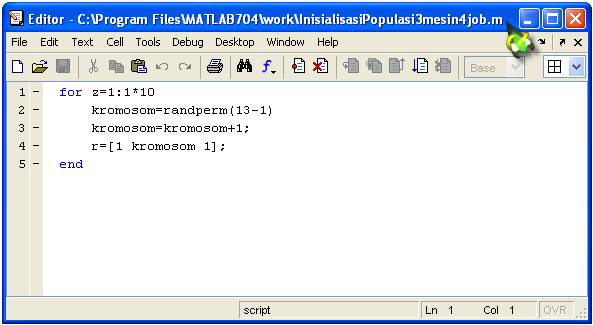
*Fitness* terbaik setelah 100 *gen*erasi yang muncul pada grafik adalah 0,1429

**3.3. Kasus 2 (3 Mesin 4 *Job*)**

Dalam kasus ini akan dibangkitkan suatu populasi yang berisi sejumlah *kromosom* yang berisi sejumlah *gen* didalamnya untuk mendapatkan suatu model penyelesaian dengan nilai *fitness* terbaik yang dilakukan dengan tahap-tahap sebagai berikut:

**3.3.1 Inisialisai Populasi**

Inisialisasi dilakukan untuk membangkitkan sejumlah *kromosom*, yang berisi beberapa model pengerjaan *job* (*Kromosom*). Inisialisasi populasi dilakukan secara random dengan menggunakan MATLAB 7.0.1



Gambar 3.4 Tampilan *Mfile* Kromosom Awal Untuk Kasus 2

Klik *Run* atau tekan *F5*, Maka akan muncul 10 *kromosom* secara acak dari kasus 3 mesin 4 *job* pada *comand window* .

Hasil inisialisasi populasi yang muncul pada *command window* adalah sebagai berikut:

*Kromosom* 1

= 3 6 1 4 5 2 12 7 8 10 9 11

*Kromosom* 2

= 6 4 2 5 3 1 12 7 10 11 9 8

*Kromosom* 3

= 2 4 6 3 1 5 7 12 9 10 8 11

*Kromosom* 4

= 4 5 1 3 2 6 8 10 11 9 7 12

*Kromosom* 5

= 4 6 1 5 2 3 11 10 8 7 12 9

*Kromosom* 6

= 4 2 3 6 5 1 11 10 9 12 8 7

*Kromosom* 7

= 5 6 4 3 2 1 12 11 8 9 10 7

*Kromosom* 8

= 5 2 6 3 1 4 11 9 7 10 12 8

*Kromosom* 9

= 4 5 6 2 3 1 10 11 9 8 7 12

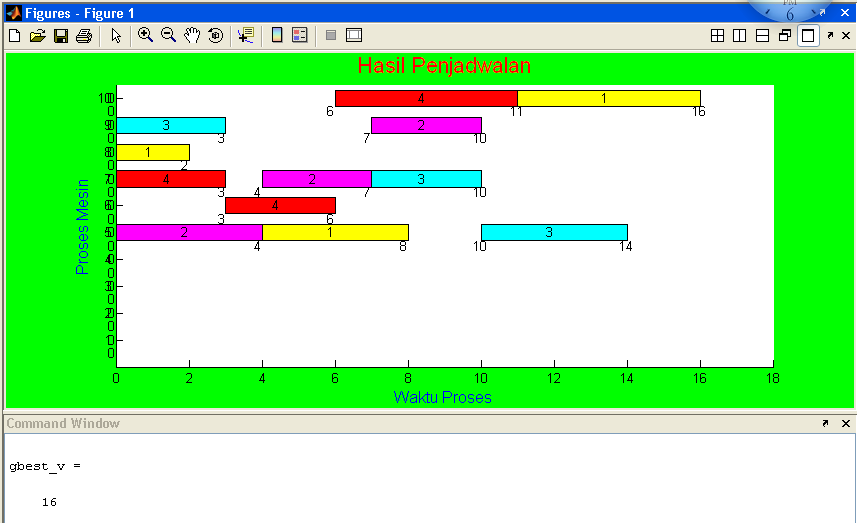
*Kromosom* 10

= 6 3 2 4 5 1 8 10 12 7 9 11

Ket : Kromosom = model penjadwalan

**3.3.2 Pemilihan Model Terbaik**

Pemilihan model terbaik dilakukan untuk memilih satu dari 10 *kromosom* acak yang dianggap paling optimal sebagai dasar pengoptimalan *makespan* dari model penyelesaian 3 mesin 4 *job*. Pemilihan model terbaik dari *makespan* terkecil dilakukan menggunakan MATLAB 7.0.1 dengan membuat beberapa kode yang dimasukkan dalam 1 *folder* agar dapat menjalankan *main* kode yang menghasilkan *output* model terbaik. Maka didapat hasil berupa *gant chart* yang menunjukkan model terbaik dan *makespan* akan muncul pada tampilan *command window* seperti gambar berikut:



Gambar 3.5

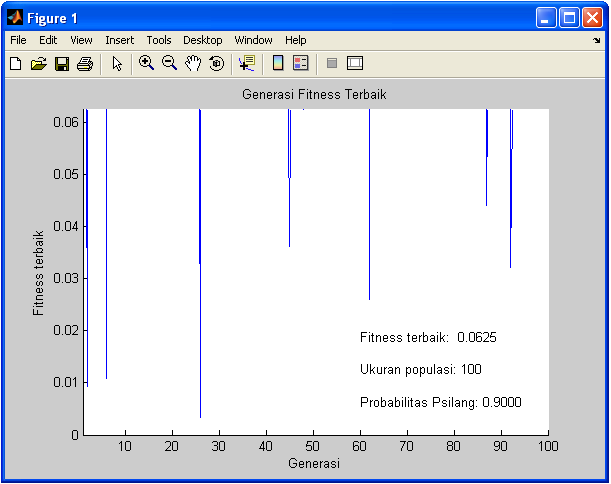
Hasil Penjadwalan Terbaik Untuk Kasus 2

Dari urutan waktu proses pada gambar diatas maka di dapat model terbaik adalah:

*Kromosom* Terbaik = 6 4 2 5 3 1 12 7 10 11 9 8

**3.3.3 Nilai *Fitness* Terbaik**

Setelah hasil pemilihan model terbaik didapat maka dilakukan pencarian nilai *fitness* terbaik dengan melakukan proses penyilangan (*crossover*) dengan peluang penyilangan (*Pc*) yang digunakan adalah 0,9. Nilai 0,9 berarti diharapkan 90% populasi yang terbentuk pada *gen*erasi selanjutnya adalah hasil penyilangan *gen*erasi sebelumnya. Maka didapat hasil berupa grafik yang menunjukkan nilai *fitness* terbaik dalam 100 *gen*erasi, seperti gambar berikut:



Gambar 3.6 Grafik Nilai Fitness Untuk Kasus 2

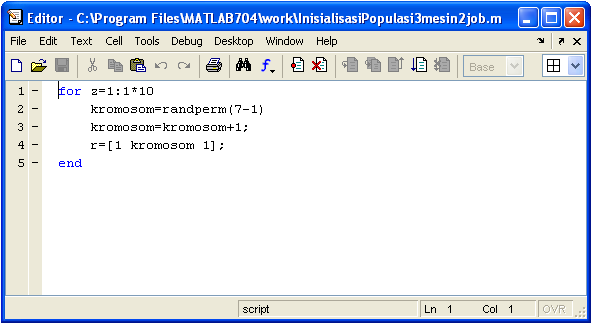
*Fitness* terbaik setelah 100 *gen*erasi yang muncul pada grafik adalah 0,0625

**3.4 Kasus 3 (3 Mesin 2 *Job*)**

Dalam kasus ini akan dibangkitkan suatu populasi yang berisi sejumlah *kromosom* yang berisi sejumlah *gen* didalamnya untuk mendapatkan suatu model penyelesaian dengan nilai *fitness* terbaik yang dilakukan dengan tahap-tahap sebagai berikut:

**3.4.1 Inisialisai Populasi**

Inisialisasi dilakukan untuk membangkitkan sejumlah  *kromosom*, yang berisi beberapa model pengerjaan *job* (*Kromosom*). Inisialisasi populasi dilakukan secara random dengan menggunakan MATLAB 7.0.1



Gambar 3.7 Tampilan *Mfile* Kromosom Awal Untuk Kasus 3

Klik *Run* atau tekan *F5*, Maka akan muncul 10 *kromosom* secara acak dari kasus 3 mesin 2 *job* pada *comand window* .

Hasil inisialisasi populasi yang muncul pada *command window* adalah sebagai berikut:

*Kromosom* 1 = 3 6 1 5 4 2

*Kromosom* 2 = 6 4 2 5 3 1

*Kromosom* 3 = 2 4 6 3 1 5

*Kromosom* 4 = 4 5 1 3 2 6

*Kromosom* 5 = 4 6 1 5 2 3

*Kromosom* 6 = 4 2 3 6 5 1

*Kromosom* 7 = 5 6 4 3 2 1

*Kromosom* 8 = 5 2 6 3 1 4

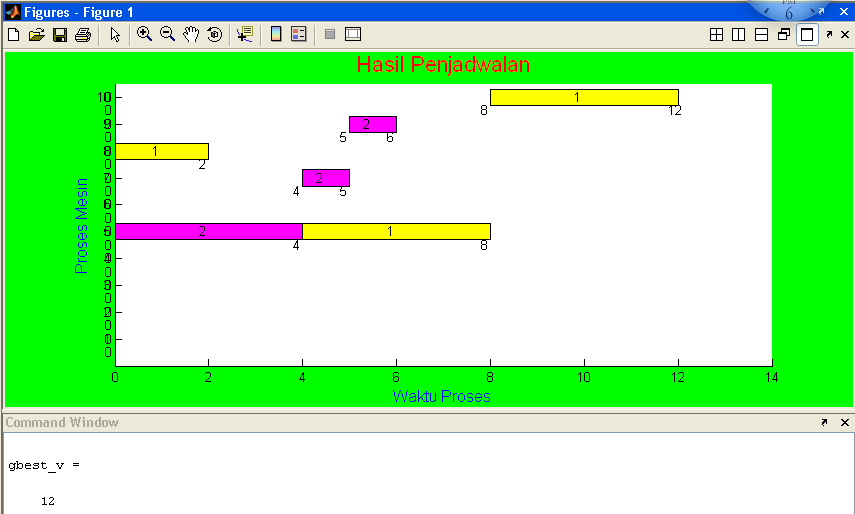
*Kromosom* 9 = 4 5 6 2 3 1

*Kromosom* 10 = 6 3 2 4 5 1

Ket : Kromosom = model penjadwalan

**3.4.2 Pemilihan Model Terbaik**

Pemilihan model terbaik dilakukan untuk memilih satu dari 10 *kromosom* acak yang dianggap paling optimal sebagai dasar pengoptimalan *makespan* dari model penyelesaian 3 mesin 2 *job*. Pemilihan model terbaik dari *makespan* terkecil dilakukan menggunakan MATLAB 7.0.1 dengan membuat beberapa kode yang dimasukkan dalam 1 *folder* agar dapat menjalankan *main* kode yang menghasilkan *output* model terbaik. Maka didapat hasil berupa *gant chart* yang menunjukkan model terbaik dan *makespan* akan muncul pada tampilan *command window* seperti gambar berikut:



Gambar 3.8

Hasil Penjadwalan Terbaik Untuk Kasus 3

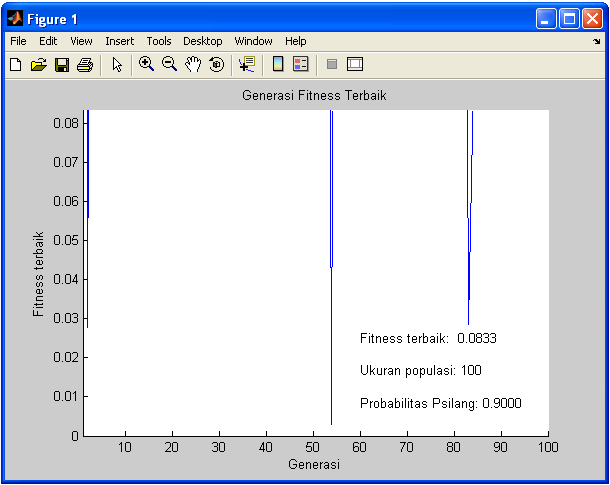
Dari urutan waktu proses pada gambar diatas maka di dapat model terbaik adalah *Kromosom* 1 = 6 4 2 5 3 1

**3.4.3 Nilai *Fitness* Terbaik**

Setelah hasil pemilihan model terbaik didapat maka dilakukan pencarian nilai *fitness* terbaik dengan melakukan proses penyilangan (*crossover*) dengan peluang penyilangan (*Pc*) yang digunakan adalah 0,9. Nilai 0,9 berarti diharapkan 90% populasi yang terbentuk pada *gen*erasi selanjutnya adalah hasil penyilangan *gen*erasi sebelumnya. Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Membuat kode yang digunakan untuk memasukkan data-data yang diperlukan dalam pencarian nilai *fitness* terbaik..
2. Membuat kode yang digunakan membuat grafik nilai *fitness*.

Maka didapat hasil berupa grafik yang menunjukkan nilai *fitness* terbaik dalam 100 *gen*erasi, seperti gambar berikut:



Gambar 3.9 Grafik Nilai Fitness Untuk Kasus 3

*Fitness* terbaik setelah 100 *gen*erasi yang muncul pada grafik adalah 0,0833

**4. SIMPULAN**

Berdasarkan hasil pengolahan data dari ketiga kasus yang telah dilakukan, dengan membangkitkan 10 *kromosom* secara acak, dengan probabilitas pindah silang (*crossover*) sebesar 90% dengan pencarian sebanyak 100 *gen*erassi, maka didapat Hasil penjadwalan mesin dan *job* serta nilai *fitness* terbaik dari ketiga kasus yang telah di olah melalui implementasi algoritma *gen*etika dalam matlab adalah sebagai berikut :

1. Model penjadwalan terbaik yang didapat dari hasil pengolahan data menggunakan matlab adalah:

Kasus 1 (3 mesin 1 *job*) = 2-3-1

Kasus 2 (3 mesin 4 *job*) = 6-4-2-5-3-1-12-7-10-11-9-8

Kasus 3 (3 mesin 2 *job*) = 6-4-2-5-3-1

1. Waktu proses terbaik didapat dari waktu proses model terbaik dari setiap kasus dengan menggunakan data waktu proses yang telah dibuat.

Kasus 1 (3 mesin 1 *job*) = 7 Menit

Kasus 2 (3 mesin 4 *job*) = 16 Menit

Kasus 3 (3 mesin 2 *job*) = 12 Menit

1. Nilai *fitness* terbaik didapat dari pengolahan data grafik pada matlab dengan probabilitas pindah silang (*crossover*) 90% dalam 100 generasi maka di dapat nilai *fitness* terbaik dari setiap kasus adalah :

Kasus 1 (3 mesin 1 *job*) = 0,1429

Kasus 2 (3 mesin 4 *job*) = 0,0625

Kasus 3 (3 mesin 2 *job*) = 0,0833

**DAFTAR RUJUKAN**

*Arifin, Miftahol. 2009. Simulasi Sistem Industri. Yogyakarta. Cetakan Pertama : Graha Ilmu.*

*Berlianty, Intan. dan Arifin, Miftahol. 2010. Teknik-Teknik Optimasi Heuristik. Yogyakarta. Cetakan Pertama : Graha Ilmu.*

*Ginting, Rosnani. 2009. Penjadwalan Mesin. Yogyakarta. Cetakan Pertama : Graha Ilmu*.

*Nasution, A. H. dan Prasetyawan, Y. 2008. Perencanaan Pengendalian Produksi. Yogyakarta : Graha Ilmu*.

*Sanjoyo. 2006. Aplikasi Algoritma Genetika.*

[*http://sanjoyo55.files.wordpress.com/2008/11/non-linier-gen-algol.pdf (di akses pada tanggal 11*](http://sanjoyo55.files.wordpress.com/2008/11/non-linier-gen-algol.pdf%20(11) *Maret 2012).*

*Santoso, Budi. 2010. Production And Planning Control (PPC). Palembang : Universitas Bina Darma*.

*Satrio, Ayip Bayu. 2007. Optimasi Masalah Penjadwalan Job-Shop Untuk Industri Peralatan Pengolahan Hasil Pertanian Dengan Menggunakan Algoritma Genetika. Bogor. Skripsi. Fakultas Teknologi Pertanian. Institut Pertanian Bogor.*

Suyanto. 2005. *Algoritma Genetika Dalam Matlab*. Yogyakarta : Andi offset.

Wignjosoebroto, Sritomo. 2003. *Pengantar Teknik Dan Manajemen Industri.* Surabaya. Edisi Pertama : Guna Widya.

[*http://gudangmakalah.blogspot.com*](http://gudangmakalah.blogspot.com)*. Makalah Local Search Algorithm. (di akses pada tanggal 11 Maret 2012).*