PENYELESAIAN MASALAH JOB SHOP MENGGUNAKAN ALGORITMA GENETIKA

Satriyo Adhy dan Kushartantya

Teknik Informatika Universitas Diponegoro Semarang
satriyo@undip.ac.id , kushartantya@undip.ac.id

ABSTRACT

Job Shop Scheduling Problem (JSP) is a problem of work scheduling that can be describe by one or several job that have to be done by one or several source, each job contain of several operation that has to be done without interuption in several time and spesific source. Many kind algorithm can be use to solve JSP, one of them is genetic algorithm. Genetic Algorithm use in this final project to solve JSP. Genetic algorithm step begin with individual representation, initial population, reproduction and selection process, crossover gen, mutation gen and evaluation. Solving JSP with genetic algorithm can give a schedule of job in source with optimal time process, then Delphi 6 software use to make the solving program.

Key Word : *Job Shop Scheduling Problem* (JSP), *Genetic Algorithm*, Delphi 6

1. Pendahuluan

JSP merupakan permasalahan penjadwalan dari sejumlah *n* pekerjaan pada sejumlah *m* mesin [6]. Contoh dari JSP adalah permasalahan membaca surat kabar, terdapat sejumlah orang yang hendak membaca sejumlah buah surat kabar. Tiap orang akan membaca semua surat kabar yang ada, sebagai gambaran jika terdapat 4 orang dan 4 surat kabar, maka akan terdapat 331776 kemungkinan jadwal atau $(n!)^m$ kemungkinan untuk permasalahan yang lain[5]. Permasalahan yang menjadikan JSP layak diteliti adalah menjadwalkan seluruh *n* pekerjaan dengan suatu jadwal yang dapat mengoptimalkan nilai tertentu dalam hal ini adalah waktu proses pada tingkat ukuran tertentu.

Algoritma Genetika pertama kali diperkenalkan oleh John Holland dari Universitas Michigan pada awal 1970 dengan tulisannya berjudul *Adapted in Natural and Artificial System* yang cara kerjanya berdasarkan pada seleksi dan genetika alam. Sedangkan aplikasi pertamanya pada manufacturing control dikemukakan oleh L Davids dalam *Proceedings of an International Conference on Genetic Algorithm and their Application* Hillsdale 1985[1]. GA bekerja dari satu populasi bukan dari satu titik dan mencari nilai optimum secara keseluruhan[3].

2. Permasalahan dan Batasan

2.1 Permasalahan

Permasalahan yang dibahas adalah bagaimana menyelesaikan JSP dengan menggunakan algoritma genetika, sehingga didapatkan urutan jadwal pengalokasian operasi-operasi dalam pekerjaan ke dalam mesin-mesin yang tersedia sedemikian sehingga jadwal tersebut diupayakan mempunyai waktu yang optimal dalam *n* iterasi atau pada nilai error tertentu. Selanjutnya mengimplementasikan algoritma genetika ke dalam suatu program dengan menggunakan perangkat lunak Delphi 6.

2.2 Batasan

Beberapa batasan yag dipergunakan dalam tulisan ini :

1. Metode presentasi yang digunakan *operation-based representation* yaitu metode presentasi yang berbasis pada operasi dalam pekerjaan.
2. Jumlah mesin *m*, jumlah job *n* serta operasi-operasi yang terdapat dalam pekerjaan tersebut, dalam tulisan ini batas maksimal pekerjaan = 10 dan mesin = 10.

3. Sebuah pekerjaan tepat mengunjungi satu mesin sekali.
4. Tiap mesin hanya dapat memproses sebuah operasi pada satu waktu dan selalu siap setiap waktu tanpa adanya gangguan kerusakan atau dalam perbaikan.
5. Operasi tidak dapat diinterupsi, dengan kata lain setelah operasi berlangsung, operasi itu harus diselesaikan sebelum operasi yang lain diproses pada mesin yang sama.
6. Terdapat batasan untuk lebih mendahulukan suatu operasi sebelum operasi berikutnya berlangsung atau berurutan pada pekerjaan yang sama, tetapi tidak berlaku pada pekerjaan yang berbeda.
7. Tiap tipe mesin jumlahnya hanya satu dan spesifik.
8. Setelah sebuah operasi selesai diproses pada suatu mesin, akan langsung ditransfer ke mesin berikutnya secepatnya dan waktu transfer diabaikan.
9. Setiap operasi mempunyai jenis kegiatan tertentu, dikerjakan mesin tertentu dan waktu proses yang tertentu pula.
10. Nilai default yang diberikan untuk maksimum generasi = 500, nilai error = 3, nilai laju crossover = 0.8, dan nilai laju mutasi = 0.3. Nilai default ini diberikan berdasarkan percobaan-percobaan/penelitian-penelitian telah dilakukan.

3. Dasar Teori

3.1 *Job Shop Scheduling Problem (JSP)*

Job Shop secara umum adalah terdapat *m* buah mesin dan *n* buah *jobs* atau pekerjaan yang harus diproses. Tiap pekerjaan terdiri dari *l* operasi dan tiap operasi harus diproses tanpa ada interupsi dalam jangka waktu dan mesin tertentu [6]. Operasi-operasi di dalam pekerjaan merupakan jenis kegiatan yang harus diselesaikan di dalam suatu mesin, tiap operasi memiliki karakteristik pada kebutuhan mesin dan lamanya waktu terproses. Pemrosesan operasi merupakan kegiatan penyediaan mesin untuk sebuah operasi dalam rangka penyelesaian atau pengeksekusian operasi tersebut dalam waktu tertentu dengan rute atau jadwal tertentu.

Jadwal merupakan suatu urutan antrian tertentu operasi-operasi untuk dieksekusi, sebuah penjadwalan adalah sebuah pengalokasian operasi dalam mendapatkan mesin dengan urutan tertentu[2]. Mesin bertugas untuk menyelesaikan operasi-operasi di dalam pekerjaan sebagai alat, tiap mesin dapat memproses hanya satu operasi dalam satu waktu. Permasalahan utama dalam JSP adalah untuk menentukan jadwal untuk tiap pekerjaan pada tiap mesin, jadwal tersebut mengatur pekerjaan-pekerjaan yang harus diselesaikan oleh satu mesin atau lebih. Banyaknya jadwal yang mungkin digambarkan oleh $(n!)^m$ untuk *n* pekerjaan dan *m* mesin[5]. Total waktu proses dipengaruhi oleh waktu setup atau waktu dimulainya pemrosesan tiap pekerjaan. Tiap jadwal dapat dievaluasi dengan berbagai macam pengukuran, tingkat keberhasilan pengukuran berhubungan dengan waktu yang dihabiskan satu pekerjaan dalam jadwal atau pengoptimalan waktu mesin atau ke duanya. Beberapa tingkat keberhasilan pengukurannya yaitu [7]:

1. *Flowtimes* adalah jumlah waktu yang dihabiskan tiap pekerjaan dalam jadwal, ini hampir mirip dengan pekerjaan dalam proses inventori.
2. *Makespan* adalah total waktu untuk seluruh pekerjaan dalam prosesnya.
3. *Lateness and earliness* adalah pengukuran nilai deviasi dari pertama dan ke dua.
4. *Machine and labor utilization* adalah pengoptimalan penggunaan mesin dan pekerja.

Pemilihan jadwal berdasarkan pada sasaran untuk meminimalisasi *makespan* atau meminimalisasi rata-rata *flowtime*, atau lainnya.

Beberapa *priority rule* atau aturan prioritas sederhana dapat digunakan terutama untuk kasus-kasus sejumlah kecil dari mesin atau sumber. Ini dikembangkan untuk menghasilkan jadwal yang baik untuk beberapa objek yang berbeda dan pada kondisi yang berbeda.

Terdapat beberapa batasan pada pekerjaan dan mesin, yaitu [2]:

1. Sebuah pekerjaan tepat mengunjungi satu mesin sekali.

2. Tiap mesin hanya dapat memproses sebuah operasi pada satu waktu dan selalu siap setiap waktu tanpa adanya gangguan kerusakan atau dalam perbaikan.
3. Operasi tidak dapat diinterupsi, dengan kata lain setelah operasi berlangsung, operasi itu harus diselesaikan sebelum operasi yang lain diproses pada mesin yang sama.
4. Terdapat batasan untuk lebih mendahulukan suatu operasi sebelum operasi berikutnya berlangsung atau berurutan pada pekerjaan yang sama, tetapi tidak berlaku pada pekerjaan yang berbeda
5. Tiap tipe mesin jumlahnya hanya satu.
6. Setelah sebuah operasi selesai diproses pada suatu mesin, akan langsung ditransfer ke mesin berikutnya secepatnya dan waktu transfer diabaikan.

Setiap operasi mempunyai jenis kegiatan tertentu, dikerjakan mesin tertentu dan waktu proses yang tertentu pula.

3.2 Algoritma Genetika

Algoritma Genetika adalah algoritma kecerdasan buatan tentang teknik pencarian dan optimasi yang berdasarkan pada mekanisme seleksi atau evolusi yang terjadi di alam [1]. Konsep yang dipergunakan oleh algoritma genetika adalah melakukan apa yang dilakukan oleh alam [1].

Secara alami semua organisme terdiri dari sel, di dalam setiap sel terdiri dari sekumpulan kromosom. Kromosom terbentuk dari sekumpulan *gen*, membuat satu kesatuan yang tersusun dalam rangkaian linier. Setiap gen mempunyai letak tersendiri di dalam kromosom, disebut lokus. *Gen* tersusun dari DNA yang membawa sifat-sifat keturunan. Setiap *gen* menyandikan protein tertentu suatu sifat contoh : *gen* warna mata binatang dengan posisi *lokus* 10. Bagian tertentu dari *gen* di dalam *genome* disebut *genotip*. Beberapa sifat individu yang menunjukkan perbedaan *gen* dan berada pada bagian yang berbeda disebut *alel*.

Perbandingan istilah alam dengan Algoritma Genetika [1]:

Alam	Algoritma Genetika
*chromosom*e	string
locus	posisi string
gene	karakter
allele	nilai karakter
genotype	struktur
phenotype	kode struktur

Algoritma Genetika merepresentasikan individu sebagai sebuah kromosom. Algoritma ini menyelesaikan permasalahan dalam pencarian kromosom yang terbaik dengan memanipulasi isi di dalam kromosom tanpa tahu permasalahan yang sedang diselesaikan seperti yang terjadi di alam. Informasi yang diberikan hanya evaluasi dari tiap kromosom yang dihasilkan, dan digunakan untuk membelokkan seleksi dari kromosom sehingga kromosom yang terbaik yang terpelihara untuk dikembangkan lebih banyak.

Algoritma genetika merupakan algoritma yang berdasarkan pada mekanisme dan seleksi alam dan mempunyai 5 komponen yaitu :

1. Representasi genetika untuk solusi potensial permasalahan.
2. Metode untuk membuat populasi awal dari solusi potensial.
3. Nilai untuk parameter yang bervariasi : jumlah kromosom, banyaknya gen dalam kromosom, laju mutasi, dan laju *crossover.*
4. Operator-operator genetika : mutasi dan perkawinan silang (*crossover*).
5. Evaluasi.

3.2.1 Struktur Algoritma Genetika [4]

```
Procedure Genetic_Algorithm
Begin
    Pt = 0
    Generate initial population Pt
    Evaluate population Pt
    While stopping criteria not
    satisfied Repeat
    Begin
        Pt = Pt+l
        Select elements from Pt-1 to
        copy into Pt
        Crossover elements of Pt
        Mutation elements of Pt
```

Evaluate new population ***P***t
End;
End;

Penjelasan procedure Genetic Algorithm :

1. **Begin**
2. ***P***t = 0
   Set generasi awal =0.
3. **Generate** initial population ***P***t
   Pembentukan populasi awal dari n individu (kromosom) secara random atau dipilih menurut metode tertentu. Setiap individu merepresentasikan suatu kandidat solusi bagi suatu masalah.
4. **Evaluate** population ***P***t
   Mengevaluasi individu-individu yang terbentuk pada populasi awal berdasarkan nilai fitness.
5. **While** stopping criteria not satisfied **Repeat**
   Jika belum memenuhi kondisi pemberhentian, maka algoritma genetika masuk ke dalam suatu loop proses genetika.
6. **Begin**
7. **P**t = ***P***t+l
8. **Select** elements from ***P***t-1 to copy into ***P***t
   Memilih kromosom dari populasi menurut nilai fitness, semakin besar fitness semakin besar kemungkinan diseleksi. Populasi akan dibentuk melalui suatu pemilihan dari alternative solusi pada P(t) secara random.
9. **Crossover** elements of ***P***t
   Proses crossover dua kromosom bergabung menghasilkan dua keturunan, dengan menentukan titik crossover secara random.
10. **Mutation** elements of ***P***t
    Mutasi merupakan proses pengubahan satu atau lebih gen.
11. **Evaluate** new population ***P***t
    Memeriksa kromosom-kromosom yang terbaik.
12. **End**
13. **End**

Algoritma Genetika memiliki karakteristik yang khas yaitu [4]:

1. Mengkodekan satu set parameter, bukan per parameter.
2. Melakukan pencarian dari sekumpulan titik bukan titik tungggal. Pencarian dilakukan dari generasi ke generasi sampai diperoleh solusi potensial.
3. Menggunakan fungsi fitness, bukan ilmu genetika turunan. Nilai fungsi fitness kembali pada pencarian secara langsung.

3.2.2 Parameter-parameter Algoritma Genetika

Parameter-parameter tersebut adalah [4]:

1. Ukuran Populasi (*pop_size)*
   Populasi adalah kumpulan beberapa individu yang sejenis yang hidup dan saling berinteraksi bersama pada suatu tempat. Jumlah individu dinyatakan sebagai ukuran dari populasi tersebut.
2. Laju *crossover*
   Pada saat proses genetika berlangsung, nilai dari laju *crossover* digunakan untuk menentukan individu-individu yang akan mengalami *crossover*.
3. Laju mutasi
   Nilai dari laju mutasi digunakan untuk menentukan individu yang akan mengalami mutasi, terjadi setelah proses *crossover* dilakukan.
4. Banyaknya gen dalam kromosom
5. Satu individu direpresentasikan sebagai sebuah kromosom yang terdiri dari sejumlah gen yang membentuk satu kesatuan.

3.2.3 Representasi Individu

Individu dalam algoritma genetika direpresentasikan sebagai sebuah kromosom *haploid* (n) yang terdiri dari sekumpulan gen[2]. Macam-macam representasi individu dalam algoritma genetika [2]:

1. Kromosom biner
   Kromosom biner yaitu kromosom yang disusun dari gen-gen yang bernilai 0 dan 1. Setiap gen dinyatakan dalam string biner
   Kromosom : 1 0 1 0 0 1 1 0 1 0
   Kromosom : 0 1 0 0 0 1 1 1 0 1
2. Kromosom bilangan bulat
   Kromosom bilangan bulat yaitu kromosom yang disusun dari gen-gen bilangan bulat.
   Kromosom : 1 2 3 4 5 6 7 8 9 0
   Kromosom : 3 4 5 6 2 1 8 7 0 9
   Untuk pembahasan selanjutnya digunakan kromosom bilangan bulat.

3.2.4 Nilai Fitnes

Nilai fitness adalah nilai yang menyatakan baik tidaknya suatu individu [1]. Semakin besar nilai fitness yang dimiliki suatu individu, maka semakin besar pula kesempatan individu tersebut untuk tetap bertahan atau berkembangbiak. Algoritma genetika bertujuan mencari individu dengan nilai fitness yang paling tinggi.

3.2.5 Pembentukan Populasi Awal

Membangkitkan populasi awal adalah proses membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Syarat-syarat yang harus dipenuhi untuk menunjukkan suatu solusi harus benar-benar diperhatikan dalam pembangkitan setiap individunya.

3.2.6 Reproduksi dan Seleksi

Reproduksi adalah tahap pengkopian string-string kromosom berdasarkan suatu kriteria [1]. Seleksi merupakan proses memilih kromosom induk dari populasi.

Dalam tahap reproduksi dan seleksi menggunakan nilai fitness untuk memilih kromosom yang akan dijadikan kromosom induk. Semakin besar nilai fitness kromosom maka akan semakin besar pula kemungkinan diseleksi.

Macam-macam seleksi :

1. Mesin roulette
2. Turnamen.

Seleksi mesin roullete untuk memilih induk dilakukan dengan menggunakan prosentasi fitness setiap individu, di mana setiap individu mendapatkan luas bagian sesuai dengan prosentase nilai fitnessnya. Sedangkan seleksi turnamen untuk memilih individu dilakukan dengan kompetisi dari tiap individu, individu yang terpilih merupakan individu pemenang. Seleksi mesin roullete lebih memberikan variansinya dibandingkan seleksi turnamen.

3.2.7 Perkawinan Silang (*Crossover*)

Perkawinan Silang atau *Crossover* merupakan proses mengkombinasikan dua individu untuk memperoleh individu-individu baru yang diharapkan mempunyai fitness lebih baik [1]. *Crossover* menukar bagian dari 2 kromosom induk yang berbeda, bekerja menggabungkan 2 kromosom induk dengan cara penyilangan untuk menghasilkan 2 individu baru. Tidak semua pasangan induk mengalami proses *crossover*, banyaknya pasangan induk yang mengalami *crossover* ditentukan dengan nilai laju *crossover* dan dilakukan secara acak. Sehingga pemilihan laju *crossover* yang tepat akan semakin mempercepat pencarian suatu hasil yang diinginkan begitu juga sebaliknya.

3.2.8 Mutasi

Mutasi adalah penggantian satu atau lebih gen dalam suatu kromosom yang terjadi tanpa melibatkan kromosom yang lain [1]. Pada kromosom biner, mutasi dilakukan dengan mengubah gen biner 0 menjadi 1 dan 1 menjadi 0. Pada kromosom float ada dua macam mutasi yang banyak dilakukan yaitu *random mutation* dan *shift mutation*.

1. Random mutation adalah mengganti gen yang termutasi dengan nilai acak.
2. Shift mutation adalah menggeser nilai gen termutasi sebesar ε, di mana ε adalah bilangan kecil yang ditentukan.

Probabilitas mutasi yang baik berada pada kisaran 0 sampai dengan 0.3 [8]. Probabilitas mutasi yang terlalu kecil menyebabkan terjebak dalam optimum lokal, dan probabilitas mutasi yang terlalu besar menyebabkan konvergensi sulit didapatkan [8].

3.2.9 Evaluasi

Penyeleksian kromosom-kromosom yang terpilih dilakukan dalam tahap ini. Setiap kromosom hasil *crossover* dan mutasi akan dilihat apakah lebih baik dari kromosom sebelumnya ataukah sebaliknya.

3.2.10 Terminasi (batasan iterasi generasi)

Dalam proses algoritma genetika, untuk menghentikan proses regenerasi yang berlangsung diadakan *Stopping criteria* yang mempunyai definisi sesuai dengan kebutuhan [4]. Misal proses regenerasi akan berhenti jika generasi berikutnya menghasilkan individu yang sama dari individu sebelumnya.

4. Penyelesaian JSP dengan Algoritma Genetika

JSP yang tergambarkan oleh *n* pekerjaan dan *m* mesin akan memiliki $(n!)^m$ kemungkinan jadwal yang dapat dibentuk. Tingkat keberhasilan pengukuran untuk sebuah jadwal menggunakan *makespan* yaitu total waktu untuk seluruh pekerjaan dalam terproses dan aturan prioritasnya menggunakan *First Come First Serve* (FCFS) yaitu memilih pekerjaan berdasarkan waktu kedatangan, waktu kedatangan yang dimaksud adalah urutan *gen* dalam kromosom. Penyelesaian JSP menggunakan algoritma genetika berlangsung dalam beberapa tahap, yaitu :

1. Pemodelan individu.
2. Pembentukan populasi awal.
3. Fungsi fitnes.
4. Evaluasi, reproduksi, dan seleksi.
5. Perkawinan silang atau *crossover*.
6. Mutasi gen.
7. *Terminasi* atau batasan iterasi generasi.

4.1 Pemodelan individu

Individu dalam algoritma genetika dimodelkan sebagai sebuah kromosom tunggal atau *haploid*, kemudian kromosom tersebut terdiri dari sejumlah gen yang terbentuk dari permasalahan banyaknya pekerjaan dan banyaknya mesin yang digunakan dalam *job shop scheduling problem*.

Kromosom direpresentasikan dalam kromosom bilangan bulat dengan aturan *operation-based representation* atau aturan yang berbasis pada operasi dalam pekerjaan.

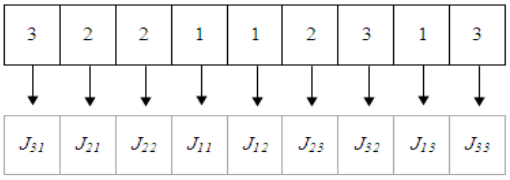
Metode presentasi berbasis pada operasi dalam pekerjaan adalah metode mengkodekan sebuah jadwal sebagai sebuah rangkaian dari operasi. Sebuah kromosom terdiri dari *n* pekerjaan dan *m* mesin akan mempunyai ukuran kromosom *n* x *m*.

Contoh

Dalam kasus terdapat 3 pekerjaan dan 3 mesin (gambar 1):

Kromosom : 3 2 2 1 1 2 3 1 3

Gen yang pertama ”3” menyatakan operasi 1 dari pekerjaan 3, gen ke dua “2”) menyatakan operasi 1 dari pekerjaan 2, gen yang ke tiga “2” menyatakan operasi 2 dari pekerjaan 2 dan seterusnya.



Gambar 1. Contoh kromosom

Tingkat keberhasilan tiap jadwal dihitung berdasarkan *makespan* atau waktu seluruh pekerjaan terproses. Penghitungan waktu contoh kromosom diatas berdasarkan pada dua buah matrik yaitu matrik proses order dan matrik waktu. Matrik proses order menyatakan pada mesin mana operasi tersebut akan diproses. Matrik waktu menyatakan lamanya operasi tersebut diproses oleh suatu mesin. Waktu *makespan* didapat pada waktu mesin yang paling lama beroperasi setelah semua operasi terproses dengan mengikuti batasan-batasan dan aturan yang telah dikemukakan. Contoh penghitungan waktu sebuah kromosom (gambar 2):

Job	Proses Order		
	1	2	3
J_1	1	2	3
J_2	1	3	2
J_3	2	1	3

Job	Waktu Proses		
	1	2	3
J_1	3	3	2
J_2	1	5	3
J_3	3	2	3

Gambar 2. Matrik *Order* dan Waktu

Dengan Kromosom : 3 2 2 1 1 2 3 1 3 .

Gen 1 = 3

Merupakan operasi pertama pekerjaan no 3.

Berdasarkan matrik order J_{31} = 2 maka mesin no 2 akan mulai bekerja menyelesaikan operasi pertama dari pekerjaan no 3.

Berdasarkan matrik waktu maka mesin 2 akan bekerja selama 3 satuan waktu.

Gen 2 = 2

Merupakan operasi pertama pekerjaan no 2.

Berdasarkan matrik order J_{21} = 1 maka mesin no 1 akan mulai bekerja menyelesaikan operasi pertama dari pekerjaan no 2.

Berdasarkan matrik waktu maka mesin 1 akan bekerja selama 1 satuan waktu.

Gen 3 = 2

Merupakan operasi ke dua pekerjaan no 2.

Berdasarkan matrik order J_{22} = 3 maka mesin no 3 akan mulai bekerja menyelesaikan operasi ke dua dari pekerjaan no 2.

Berdasarkan matrik waktu, mesin 3 akan bekerja selama 5 satuan waktu.

Mesin no 3 tidak dapat langsung bekerja menyelesaikan operasi ke dua dari pekerjaan no 2 karena operasi pertama dari pekerjaan no 2 masih dikerjakan pada mesin no 1, sehingga mesin no 3 bekerja setelah operasi pertama pekerjaan no 2 selesai dikerjakan mesin no 1, waktu awal mesin no 3 = 1 dan bekerja selama 5 satuan waktu.

Gen 4 = 1

Merupakan operasi pertama pekerjaan no 1.

Berdasarkan matrik order J_{11} = 1, mesin no 1 akan mulai bekerja menyelesaikan operasi pertama dari pekerjaan no 1.

Berdasarkan matrik waktu, mesin 1 akan bekerja selama 3 satuan waktu.

Mesin no 1 bekerja pada operasi pertama pekerjaan no 1 setelah menyelesaikan operasi pertama pekerjaan no 2 dan bekerja selama 3 satuan waktu.

Gen 5 = 1

Merupakan operasi ke dua pekerjaan no 1.

Berdasarkan matrik order J_{12} = 2, mesin no 2 akan mulai bekerja menyelesaikan operasi ke dua dari pekerjaan no 1.

Berdasarkan matrik waktu, mesin 2 akan bekerja selama 3 satuan waktu.

Mesin no 2 tidak dapat langsung bekerja menyelesaikan operasi ke dua dari pekerjaan no 1 karena operasi pertama dari pekerjaan no 1 masih dikerjakan pada mesin no 1, sehingga mesin no 2 bekerja setelah operasi pertama pekerjaan no 1 selesai dikerjakan mesin no 1, mesin no 2 mulai bekerja pada waktu = 4 dan bekerja selama 3 satuan waktu.

Gen 6 = 2

Merupakan operasi ke tiga pekerjaan no 2.

Berdasarkan matrik order J_{23} = 2, mesin no 2 akan mulai bekerja menyelesaikan operasi ke tiga dari pekerjaan no 2.

Berdasarkan matrik waktu, mesin 2 akan bekerja selama 3 satuan waktu.

Mesin no 2 bekerja pada operasi ke tiga pekerjaan no 2 setelah menyelesaikan operasi ke dua pekerjaan no 1 dan bekerja selama 3 satuan waktu.

Gen 7 = 3

Merupakan operasi ke dua pekerjaan no 3.

Berdasarkan matrik order J_{32} = 1, mesin no 1 akan mulai bekerja menyelesaikan operasi ke dua pekerjaan no 3.

Berdasarkan matrik waktu , mesin 1 akan bekerja selama 2 satuan waktu.

Mesin no 1 bekerja pada operasi ke dua pekerjaan no 3 setelah menyelesaikan operasi pertama pekerjaan no 1 dan bekerja selama 2 satuan waktu.

Gen 8 = 1

Merupakan operasi ke tiga pekerjaan no 1.

Berdasarkan matrik order J_{13} = 3 , mesin no 3 akan mulai bekerja menyelesaikan operasi ke tiga pekerjaan no 1.

Berdasarkan matrik waktu, mesin 3 akan bekerja selama 2 satuan waktu.

Mesin no 3 tidak dapat langsung bekerja menyelesaikan operasi ke tiga dari pekerjaan no 1 karena operasi ke dua dari pekerjaan no 1 masih dikerjakan pada mesin no 2, sehingga mesin no 3 bekerja setelah operasi ke dua pekerjaan no 1 selesai dikerjakan mesion no 2, mesin no 3 mulai bekerja pada waktu = 7 dan bekerja selama 2 satuan waktu.

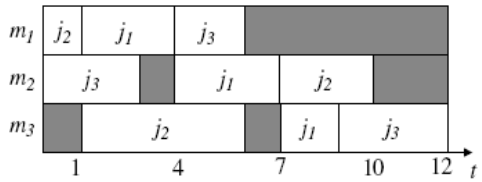
Gen 9 = 3

Merupakan operasi ke tiga pekerjaan no 3.

Berdasarkan matrik order J_{33} = 3, mesin no 3 akan mulai bekerja menyelesaikan operasi ke tiga pekerjaan no 3.

Berdasarkan matrik waktu, mesin 3 akan bekerja selama 3 satuan waktu.

Mesin no 3 bekerja pada operasi ke tiga pekerjaan no 3 setelah menyelesaikan operasi ke tiga pekerjaan no 1 dan bekerja selama 3 satuan waktu.



Gambar 3. Representasi hasil perhitungan *makespan*

Pada mesin 1 waktu berakhirnya = 6 satuan waktu, pada mesin 2 waktu berakhirnya = 10 satuan waktu, dan pada mesin 3 waktu berakhirnya = 12 satuan waktu. Sehingga waktu terproses semua operasi pada semua mesin atau *makespan* = 12 satuan waktu (Gambar 3).

4.2 Pembentukan populasi awal

Pembentukan populasi awal dilakukan dengan cara membangkitkan bilangan random sebanyak ukuran kromosom. Kromosom direpresentasikan dengan metode representasi berbasis operasi dalam pekerjaan. Ukuran kromosom berdasarkan banyaknya pekerjaan digandakan sebanyak banyaknya mesin (*n* x *m*). Penentuan ukuran populasi berdasarkan banyaknya pekerjaan digandakan sebanyak banyaknya mesin (*n* x *m*).

Individu / kromosom dalam populasi awal dibentuk dengan cara merandomkan banyaknya pekerjaan dengan batas munculnya bilangan yang sama sebanyak jumlah mesin. Sehingga terbentuk kromosom-kromosom baru dalam satu populasi.

4.3 Fungsi Fitnes

Penentuan nilai fitness dilakukan dengan ketentuan:

$$fitness(i) = \frac{total\ \ waktu}{waktu(i)}$$

$$prob\ \ fitness(i) = \frac{fitness(i)}{total\ \ fitness}$$

$$prob\ \ kumulatif\ (i) = \sum_{j=1}^{i} prob\ \ fitness(j)$$

dengan i = 1,2,3,…,n

4.4 Reproduksi dan Seleksi

Pemilihan induk menggunakan teknik mesin roulette yaitu berdasarkan angka random dan nilai *fitness* yang dimiliki tiap kromosom.
Langkah-langkah reproduksi dan seleksi :

1. Mencari waktu yang digunakan tiap jadwal (kromosom).
2. Menghitung total waktu keseluruhan jadwal (kromosom).
3. Menghitung nilai *fitness* dari tiap kromosom.
4. Menghitung total *fitness* keseluruhan kromosom.
5. Menghitung probabilitas *fitness* tiap kromosom.
6. Menghitung probabilitas *fitness* kumulatif tiap kromosom.

4.5 Perkawinan Silang (*crossover*)

Kromosom yang bercrossover terpilih berdasarkan nilai laju *crossover* yang ditentukan pengguna, penentuan nilai yang tepat akan semakin mempercepat pencapaian hasil yang diinginkan, begitu juga sebaliknya, defaultnya : 0.8

Proses *crossover* dapat dilakukan dengan beberapa cara, pada tulisan ini digunakan *Crossover* dua titik (*two-point crossover*). Kromosom-kromosom hasil *crossover* seringkali tidak dapat menjadi kromosom yang utuh, dengan kata lain rusak, sehingga diperlukan cara untuk dapat memperbaiki kromosom tersebut dengan tetap memprioritaskan proses *crossover*.

4.6 Mutasi

Mutasi terjadi selama evolusi sesuai dengan laju mutasi yang telah ditentukan. Penentuan laju mutasi yang terlalu tinggi dapat menyebabkan penelusuran keturunan kembali pada pola penelusuran acak primitif, tetapi jika terlalu kecil, maka akan memperlambat proses generasi untuk mencapai hasil yang didinginkan. Dalam tulisan ini digunakan default laju mutasi : 0.3 . Proses mutasi dapat dilakukan dengan beberapa cara, pada tulisan ini digunakan mutasi tingkat bit.

4.7 Terminasi (batasan iterasi generasi)

Proses terminasi dalam algoritma genetik dimaksudkan untuk mengakhiri siklus genetik yang dilakukan oleh program komputer lewat batasan yang diberikan. Batasan yang biasa dilakukan dapat berupa jumlah generasi maksimum yang harus dilewati oleh proses komputasi. Mekanisme terminasi lain yang akan digunakan dalam tugas akhir ini adalah, siklus genetik akan berakhir pada saat *error* yang dihasilkan telah mencapai nilai tertentu yang telah ditetapkan. Nilai *error* merupakan nilai yang ditemukan dengan waktu minimal yang

sama pada generasi berikutnya dan jika ditemukan waktu yang lebih kecil daripada waktu minimal, maka nilai *error* akan kembali ke nol. Dalam tulisan ini batasan generasi defaultnya : 500 generasi dan *error* defaultnya : 3.

5. Hasil

Contoh permasalahan :

	1	2	3	4
Adhy	SM, 60	K, 30	RS, 2	M, 5
Budi	K, 75	RS, 3	SM, 25	M, 10
Cris	RS, 5	K, 15	SM, 10	M, 30
Doni	M, 90	SM, 1	K, 1	RS, 1

Terdefinisi matrik order dan matrik waktu :

Job	Proses Order			
	1	2	3	4
J_1	1	2	3	4
J_2	2	3	1	4
J_3	3	2	1	4
J_4	4	1	2	3

Job	Waktu Proses			
	1	2	3	4
J_1	60	30	2	5
J_2	75	3	25	10
J_3	5	15	10	30
J_4	90	1	1	1

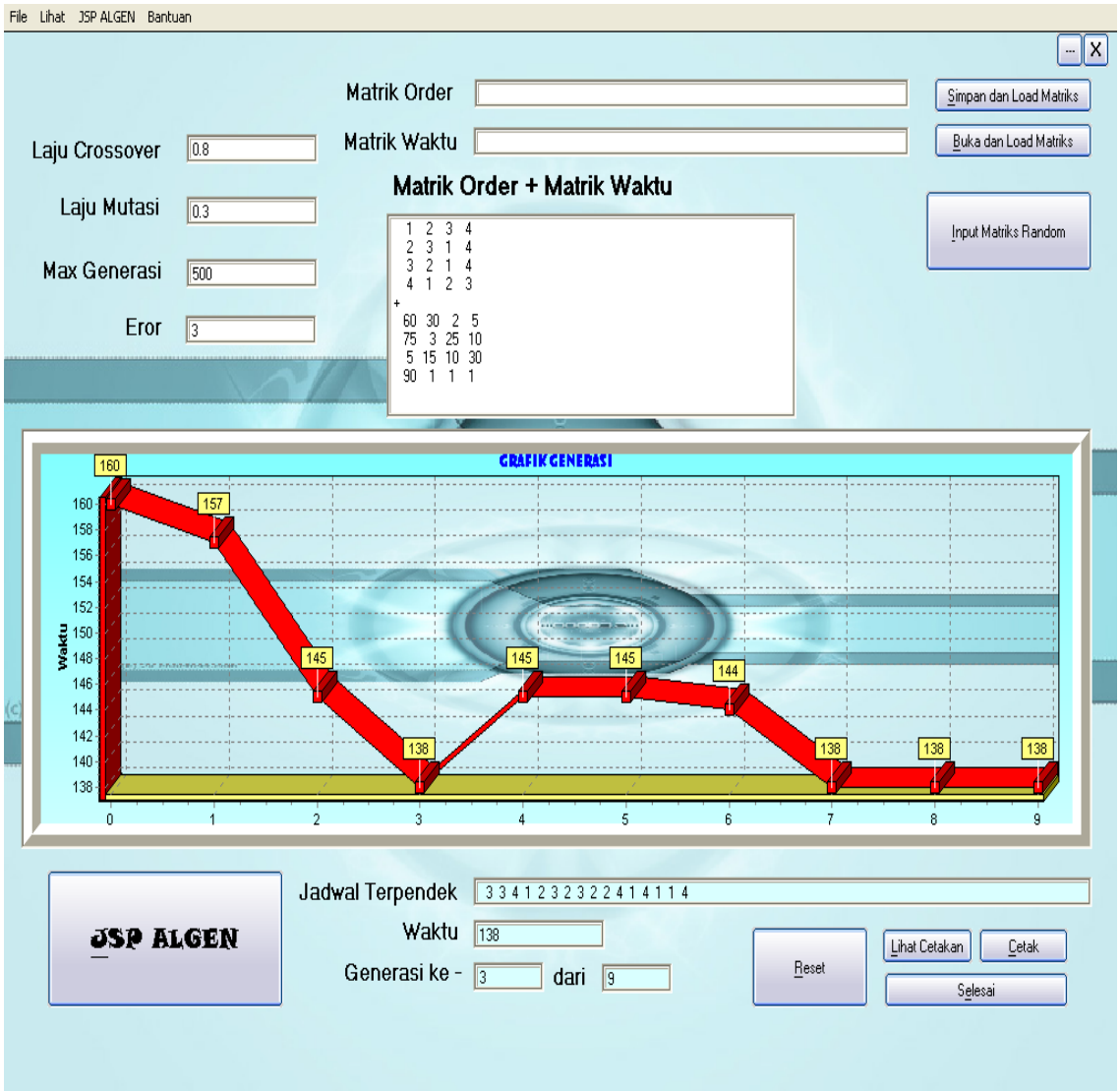
Populasi awal dan perhitungan nilai fitnes ;

Jadwal	waktu	fitness	Prob.fit	Prob.Kum
4 3 1 3 2 1 3 4 1 3 4 4 1 2 2 2	166	22.271084	0.082127	0.082127
4 3 3 4 1 3 2 4 3 1 4 1 2 2 1 2	221	16.728507	0.061688	0.143814
3 3 1 1 2 4 2 3 2 2 3 1 4 1 4 4	238	15.533613	0.057282	0.201096
4 4 2 3 2 3 4 1 4 2 2 1 3 3 1 1	221	16.728507	0.061688	0.262784
1 3 3 1 4 2 1 2 2 4 3 4 3 4 2 1	249	14.847390	0.054751	0.317535
3 4 2 3 4 4 4 2 3 2 2 1 3 1 1 1	223	16.578475	0.061135	0.378669
2 2 4 1 1 2 3 2 3 1 4 4 3 1 4 3	160	23.106250	0.085206	0.463875
3 4 1 2 4 4 2 3 1 3 3 2 2 1 1 4	162	22.820988	0.084154	0.548030
2 3 3 2 3 3 2 4 4 1 1 4 1 1 4 2	328	11.271341	0.041564	0.589594
1 3 1 2 2 3 3 3 2 4 2 4 1 4 4 1	325	11.375385	0.041948	0.631541
3 4 2 1 3 1 1 1 3 3 2 4 4 2 2 4	167	22.137725	0.081635	0.713176
1 4 4 3 4 2 2 3 2 2 4 3 1 1 1 3	249	14.847390	0.054751	0.767927
2 4 1 1 3 1 2 4 3 4 3 4 2 1 2 3	195	18.958974	0.069913	0.837840
2 1 4 4 4 1 1 4 1 3 3 3 2 2 3 2	195	18.958974	0.069913	0.907753
4 1 4 4 2 3 2 1 4 1 3 3 3 2 2 1	267	13.846442	0.051060	0.958813
2 3 2 2 2 4 4 1 4 3 4 1 1 1 3 3	331	11.169184	0.041187	1.000000
Total	3697	271.180229		

Nilai parameter algoritma genetika yang digunakan adalah :

1. Laju Crossover : 0.8
2. Laju Mutasi : 0.3
3. Max Generasi : 500
4. Error : 3

Dari pengembangan perangkat lunak yang dilakukan dan masukan permasalahan yang diangkat, dihasilkan generasi berhenti pada generasi ke 9 dan diperoleh waktu terpendek 138 menit, dengan jadwal : 3 3 4 1 2 3 2 3 2 2 4 1 4 1 1 4. Grafik terlihat pada hasil output perangkat lunak gambar 4.



Gambar 4. Output perangkat lunak yang dikembangkan

6. Kesimpulan

1. Algoritma genetika dapat digunakan dalam pencarian jadwal dalam permasalahan JSP dan dapat memberikan beberapa variasi solusi.
2. Dihasilkan program penyelesaian JSP dengan algoritma genetika menggunakan software Delphi 6.
3. Solusi yang didapatkan dalam penyelesaian JSP dengan algoritma genetika belum tentu merupakan hasil yang paling optimal. Hal

ini dikarenakan algoritma genetika menggunakan bilangan random yang berperan dalam pencarian sehingga dengan nilai parameter yang sama dapat menghasilkan solusi yang berbeda pada waktu yang berbeda.

7. Daftar Pustaka

[1] Davis, Lawrence, *Handbook of Genetic Algorithms*, Von Nostrand Reinhold, New York, 1991

[2] Garen, J, *Multiobjective Job-Shop Scheduling With Genetic Algorithms Using a New Representation and Standard Uniform Crossover*, Department of Econimics University of Osnabruck, Germany

[3] Goldberg, David E, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison Wesley Publishing Company, MA, 1989

[4] Michalewicz, Zbigniew, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, New York, 1992

[5] Oey, Kasin, Scott J. Manson, *Sceduling Batch Processing Machines in Complex Job Shops*, Depatment of Industrial Engineering University of Arkansas USA 2001

[6] Ponnambalam, S.G, P. Aravindan, P. Sreenivasa Rao, *Comparative Evaluation of Genetic Algorithm for Job-Shop Sceduling*, Taylor & Francis Ltd, 2001

[7] Taillard, E, *Benchmarks Basic Scheduling Problems*, ORWP89/21 Dec, 1989

[8] Tsujimura, Yasuhiro, Yuichiro Mafune, Mitsuo Gen, *Effects of Symbiotic Evolution in Genetic Algorithms for Job-Shop Scheduling*, Department of Industrial and Information Systems Engineering Ashikaga Institute of Technology, Japan, 2001