FOURIER Oktober 2012, Vol. 1, No. 2, 115 – 131

Penyelesaian Masalah Penjadwalan Job-Majemuk dengan Pemakaian Sumberdaya-Majemuk Menggunakan Algoritma Genetika

Taufiq Aji[1]

[1]*Program Studi Teknik Industri, Fakultas Sains dan Teknologi UIN Sunan Kalijaga*
Jl. Marsda Adisucipto No. 1 Yogyakarta 55281
ajiq15@yahoo.com

Abstrak

Scheduling problems with regard to the problem of determining the order to carry out a number of tasks. This issue covers a wide range of areas such as manufacturing, installation project, production planning, hospital management and reservation system. This problem can be seen as an optimization problem of dealing with a number of constraints. An increase in the complexity of the problem requires the existence of an efficient and effective techniques.

This study addresses the issue of scheduling multiple job-where there are several different types of resources that are working on an operation or activity simultaneously. Genetic algorithms are developed to solve these problems.

Genetic algorithm testing performed against a number of hipotetik example. The output agoritma of genetics compared against optimal technique of the output and the output algorithm based on Lagrange relaxation on the same issue. The results of the comparison with optimal techniques and algorithms based on Lagrange relaxation indicates a significant improvement of computing efficiency, but nevertheless occur a little decrease in effectiveness.

Keywords*: Simultaneous, multiple-job, resources-compounds, genetic algorithm, an algorithm based on Lagrange relaxation (ABRL).*

1. Pendahuluan

Permasalahan penjadwalan merupakan permasalahan yang sangat umum dijumpai. Permasalahan ini muncul, manakala terdapat suatu pilihan untuk membuat urutan sejumlah tugas yang akan dilaksanakan (Conway, 1967)(Baker, 1974, 2001). Permasalahan penjadwalan mencakup berbagai bidang seperti manufaktur, proyek instalasi, perencanaan produksi, manajemen rumah sakit, dan sistem reservasi. Banyak masalah dalam sistem produksi muncul dari masalah penjadwalan, seperti: tidak adanya peralatan yang tersedia pada saat dibutuhkan, penggunaan persediaan yang berlebihan dan fleksibilitas yang rendah dalam menghadapi perubahan, sehingga memberikan dampak ekonomi yang cukup signifikan pada perusahaan (Hoitomt et al.,1993).

Pada sistem manufaktur atau proyek, adakalanya terdapat penggunaaan beberapa jenis sumberdaya secara simultan untuk melayani suatu operasi atau aktivitas. Sebagai contoh pada

manufaktur yang melibatkan operasi penekukan dan pemotongan logam, maka mesin, operator, dan peralatan tertentu digunakan secara bersama-sama untuk mengerjakan aktivitas tersebut. Masalah penjadwalan sumber-majemuk simultan dalam lingkungan manufaktur telah dibahas dalam beberapa literatur misalnya dalam Dobson dan Karmakar (1989), Dobson dan Khosla (1995), Sabuncuoglu dan Bayiz (1998), Suprayogi dan Mardiono (1997), Toha dan Halim (1999), Suprayogi dan Toha (2002), serta Suprayogi dan Partono (2005). Dalam situasi tertentu, adakalanya jumlah ketersediaan untuk tiap sumberdaya lebih dari satu unit. Jika tiap unit sumberdaya adalah identik, permasalahan penjadwalan umumnya dikatakan permasalahan sumberdaya paralel (Luh, et. al., 1990).

Ketiga model yang dibahas pada penelitian Suprayogi dan Mardiono (1997), Suprayogi dan Toha (2002), Suprayogi dan Partono (2005), mengasumsikan bahwa pemakaian sumberdaya untuk mengerjakan suatu operasi tertentu menggunakan satu unit sumberdaya untuk tiap jenis sumberdaya yang digunakan. Pada penelitian Suprayogi dan Partono (2005), masalah penjadwalan diselesaikan dengan menggunakan algoritma berbasis relaksasi Lagrange. Salah satu kelemahan dalam penerapan algoritma relaksasi Lagrange yang dikembangkan dalam Suprayogi dan Partono (2005) adalah panjang horison perencanaan harus ditentukan di awal. Panjang horison perencanaan yang ditetapkan terlalu pendek akan menyebabkan ketidaklayakan. Sementara itu, penentuan panjang horison perencanaan yang terlalu panjang akan memberikan efek pada waktu komputasi.

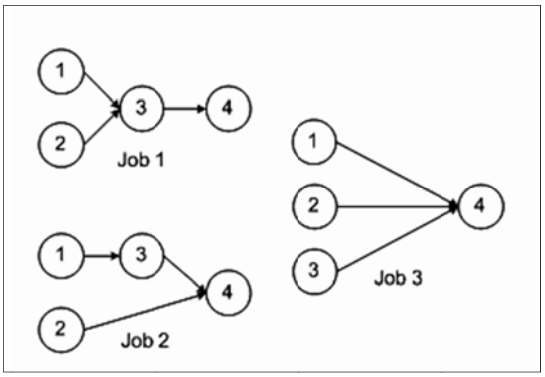
Kamarainen (1999) dan Norman (1998) menyatakan bahwa kebanyakan permasalahan penjadwalan adalah permasalahan kombinatorial yang kompleks dan biasanya tidak dapat diselesaikan secara optimal dengan lama waktu komputasi yang dapat diterima (*reasonable time*). Sedangkan teknik heuristik telah terbukti lebih efisien, meskipun mempunyai kelemahan dalam hal kualitas solusi yang dihasilkan. Menurut Norman (1999) algoritma genetika dapat menjadi alternatif heuristik sulitnya pemecahan masalah penjadwalan. Bahkan, algoritma genetika sebagai metode yang efektif dan robust merupakan alat bantu yang banyak dipakai untuk menyelesaikan permasalahan penjadwalan (Hartmann, 1997, 1999). Selain mampu mendapatkan solusi mendekati dan *robust*, beberapa simulasi empiris mendemonstrasikan bahwa algoritma genetika merupakan teknik pencarian yang efisien (Ingber dan Rosen 1992).

Penelitian ini akan menyajikan penyelesaian permasalahan penjadwalan job majemuk dengan pemakaian sumberdaya majemuk simultan dengan menggunakan algoritma genetika.

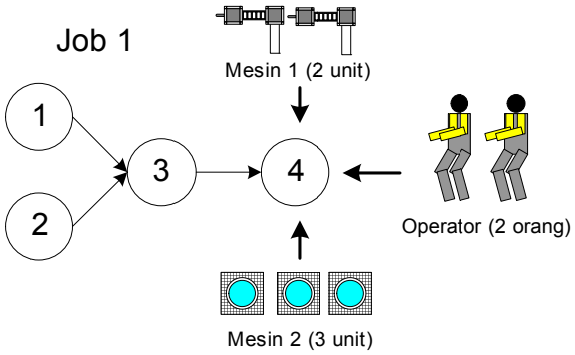
Model permasalahan mengacu pada model yang dikembangkan oleh Suprayogi dan Toha (2002). Untuk menguji performansi algoritma, hasil penyelesaian diuji terhadap hasil pada penelitian Suprayogi dan Partono (2005).

2. Permalasahan Penjadwalan Job Majemuk

Permasalahan penjadwalan job-majemuk, operasi-majemuk, sumberdaya-majemuk paralel simultan dengan pemakaian jumlah-jenis sumberdaya umum dijumpai pada sistem manufaktur modern. Secara konseptual permasalahan tersebut diilustrasikan gambar berikut:



Gambar 1. Contoh permasalahan job majemuk



Gambar 2. Ilustrasi penggunaan jumlah-jenis sumberdaya majemuk

Misalkan terdapat N job dan terdapat H jenis sumberdaya. Tiap job i terdiri atas O_i operasi. *Due date* untuk setiap job dinyatakan dengan D_i. Tiap operasi j untuk job i memerlukan waktu pengerjaan t_{ij}. Himpunan sumberdaya yang digunakan untuk mengerjakan operasi j untuk job i dinyatakan dengan H_{ij}. Horizon waktu didiskretisasi dalam K satuan waktu. Pada slot waktu k, tiap jenis sumberdaya h mempunyai ketersediaan M_{hk}. Bobot (kepentingan) tiap job i dinyatakan dengan w_i. Permasalahannya adalah menentukan jadwal

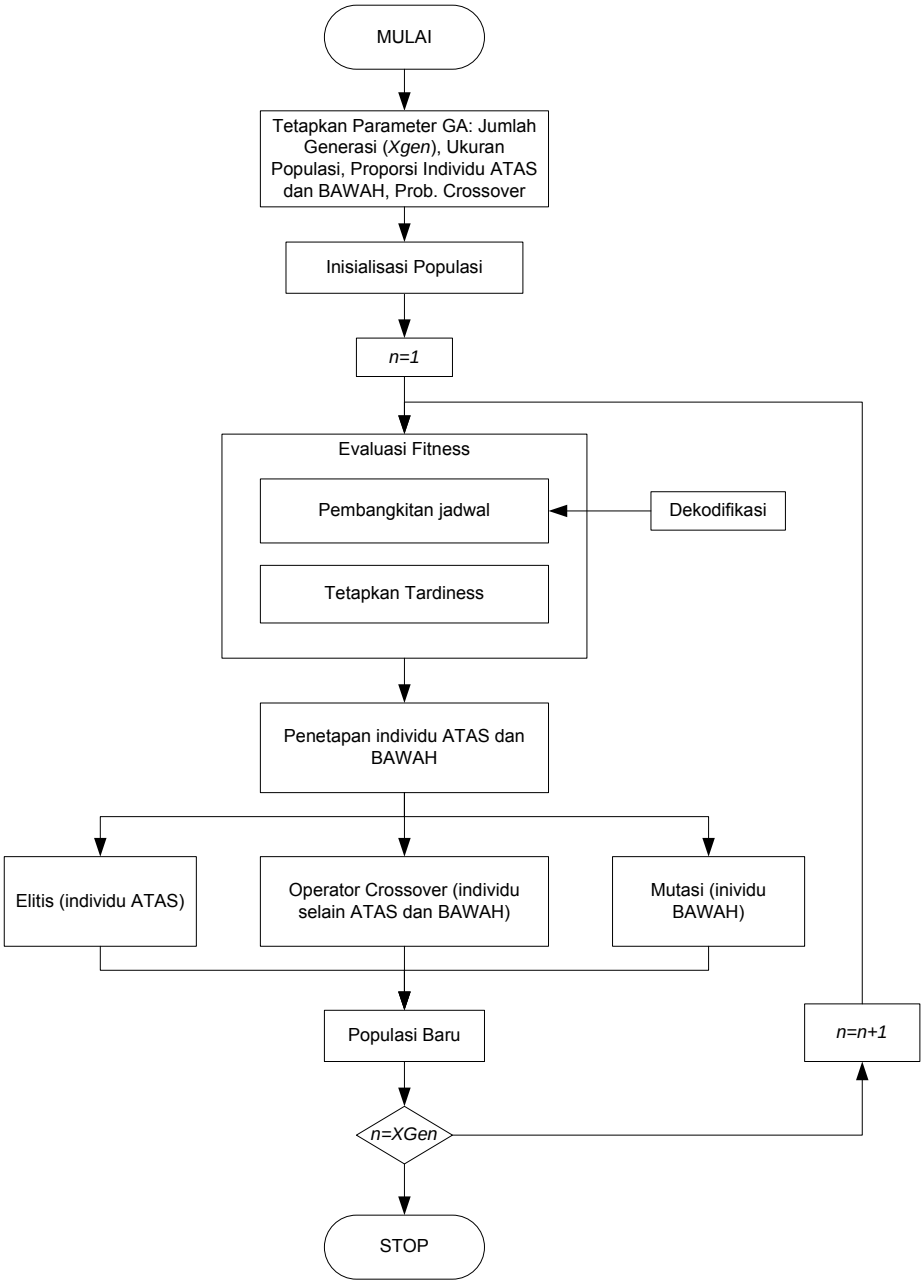
pengerjaan masing-masing operasi untuk setiap job agar diperoleh *tardiness* tertimbang total yang minimum.

Pengerjaan suatu operasi diasumsikan *nonpreemptive* (tanpa interupsi) sehingga suatu blok waktu dengan panjang t_{ij} dibutuhkan untuk pengerjaan operasi j pada job i. Job-job dianggap independen antara satu dengan yang lain. Hubungan ketergantungan dari operasi untuk suatu job dianggap membentuk suatu *directed acyclic graph*, dan tiap job diasumsikan berakhir dengan satu operasi tunggal. Dengan demikian, saat selesai tiap job C_i, sama dengan saat selesai operasi terakhir pada job i tersebut, yaitu $C_i = c_{iO_i}$. Lamanya waktu pengerjaan tiap operasi pada setiap sumberdaya digunakan adalah sama. Untuk setiap jenis sumberdaya, tiap unit dianggap identik. Semua job, operasi dan sumberdaya dianggap siap tersedia pada waktu $k = 1$. Horizon waktu K dianggap cukup panjang untuk menyelesaikan semua job, $C_i \leq K$ untuk semua i.

Suatu solusi layak penjadwalan adalah sehimpunan waktu t_{ij} dibutuhkan oleh job i, operasi j yang dimulai pada b_{ij} dan berakhir pada c_{ij} dan memenuhi relasi *precedence* serta kapasitas sumberdaya yang digunakan. Fungsi tujuan penjadwalan yang digunakan adalah meminimumkan *tardiness* tertimbang total.

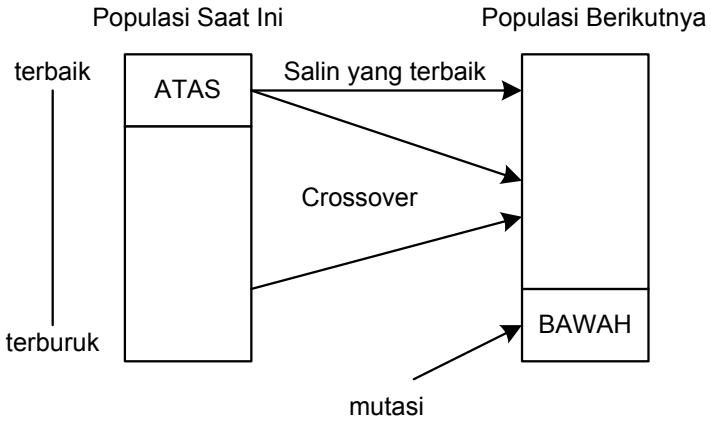
3. Teknik Pemecahan Masalah dengan Algoritma Genetika

Proses algoritma genetika secara umum dimulai dari proses inisialisasi populasi hingga proses penghentian iterasi, seperti terlihat dalam diagram alir pada Gambar 3 berikut:



Gambar 3. Diagram Algoritma Genetika Secara Umum

Pada proses tersebut terjadi perpindahan solusi antar generasi untuk mendapatkan solusi yang lebih baik melalui suatu strategi tertentu. Strategi yang digunakan pada penelitian ini merupakan modifikasi strategi evolusi pada Goncalvez et.al. (2006) yang digambarkan pada Gambar 4 berikut:



Gambar 4. Strategi Evolusi

Dengan skema strategi ini perlu ditetapkan beberapa parameter yang meliputi: besarnya jumlah iterasi generasi, besarnya ukuran populasi, besarnya persentase individu ATAS dan BAWAH, dan probabilitas *crossover*. Populasi pada generasi tertentu diranking berdasarkan nilai *fitness*-nya dari terbaik hingga terburuk. Setelah itu, ditentukan individu yang menjadi individu ATAS dan individu BAWAH sesuai proporsi yang telah ditentukan. Seluruh individu ATAS akan disalin ke dalam individu baru dan seluruh individu BAWAH akan mengalami mutasi. Seluruh individu selain individu ATAS dan BAWAH akan diganti dengan individu hasil *crossover*.

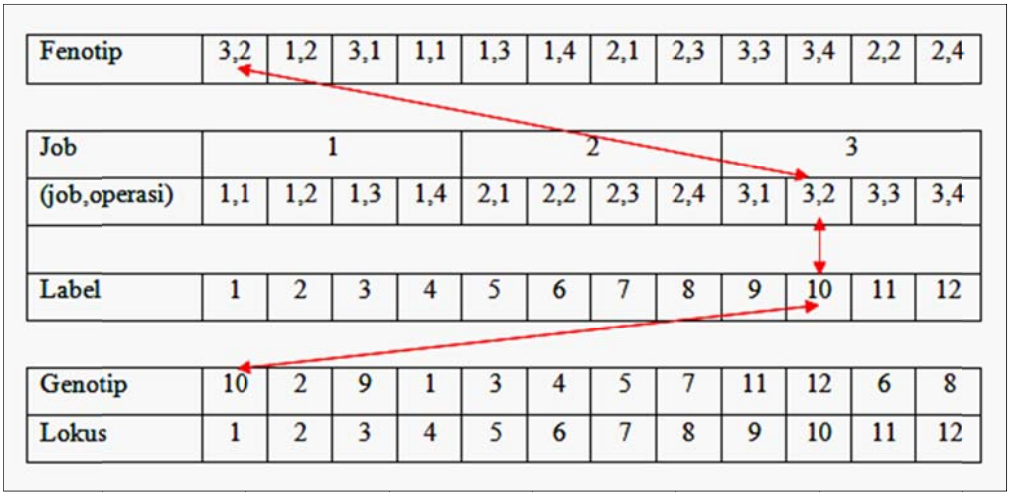
3.1. Kodifikasi dan Dekodifikasi

Kodifikasi yang digunakan pada penelitian ini adalah kodifikasi bilangan bulat. Kodifikasi ini cocok dan banyak dipakai untuk permasalahan penjadwalan (Norman, 1998)(Gen dan Cheng, 2000). Untuk melakukan kodifikasi, suatu solusi dalam bentuk fenotip dipetakan ke dalam bentuk genotip. Solusi yang dimaksud di sini bukanlah solusi jadwal, melainkan berupa urutan serial penempatan operasi-operasi pada jadwal (*sequence*) dengan memperhatikan keterkaitan relasi antar operasi pada satu job membentuk *job sesequence matrix* (Kobayashi et.al., 1995). Suatu nilai fenotip tertentu mempunyai nilai label (bilangan bulat) tertentu berdasarkan nama job-operasi (pada daftar pelabelan job-operasi) yang bersesuaian dengan nilai fenotip tersebut. Label (bilangan bulat) tersebut selanjutnya digunakan sebagai nilai genotip untuk job-operasi (fenotip) tersebut.

Tabel 1. Daftar pelabelan job-operasi pada contoh permasalahan Gambar 1

Job	1				2				3			
Operasi (job,operasi)	1,1	1,2	1,3	1,4	2,1	2,2	2,3	2,4	3,1	3,2	3,3	3,4
Label	1	2	3	4	5	6	7	8	9	10	11	12

Sedangkan proses dekodifikasi adalah proses yang berkebalikan dengan proses kodifikasi, yaitu memetakan suatu genotip ke dalam bentuk fenotipnya. Untuk lebih jelasnya, proses tersebut mengikuti gambar di bawah.



Gambar 5. Proses kodifikasi-dekodifikasi solusi

3.2. Inisialisasi Populasi

Proses inisialisasi populasi dilakukan untuk memulai algoritma genetika dengan membangkitkan sejumlah individu sesuai ukuran populasi, yang mencerminkan suatu solusi bagi permasalahan. Kromosom merepresentasikan urutan serial penempatan operasi dengan memperhatikan keterkaitan relasi antar operasi pada satu job. Untuk membangkitkan beragam urutan penempatan layak digunakan algoritma SGSMod yang merupakan modifikasi dari Skema Pembangkitan Jadwal Serial (*Serial Schedule Generation Schema*, SGS) (Kolisch dan Hartman, 1998). Algoritma SGS tersebut, merupakan algoritma yang selalu menghasilkan jadwal yang layak terhadap kendala sumberdaya dan relasi *precedence*.

Algoritma SGSMod hanya mempertimbangkan relasi *precedence* antar operasi dalam suatu job, sehingga dapat dipandang sebagai algoritma pembangkitan *sequence*. Hal tersebut sesuai dengan pengertian bahwa permasalahan *sequence* adalah permasalahan penjadwalan dengan sumberdaya tunggal, sehingga suatu jadwal hanya ditentukan oleh urutan job (Baker, 2001). Oleh karena menggunakan alur logika yang sama dengan algoritma SGS, maka algoritma SGSMod akan selalu menghasilkan solusi yang layak terhadap relasi *precedence*.
Algoritma SGSMod adalah sebagai sebagai berikut:

1. Tetapkan i=1, $i \in I$ dimana I adalah jumlah gen pada satu kromosom.
2. Tetapkan operasi-operasi tanpa *predecessor* dan operasi yang semua *predecessor*-nya telah ditempatkan pada *sequence*. Maukkan operasi-operasi tersebut ke dalam Daftar Kandidat Penempatan *Sequence*. Jika semua operasi tersebut telah dimasukkan, maka ke Langkah 3; sebaliknya ke Langkah 4.
3. Pilih secara acak satu operasi dari Daftar Kandidat Penempatan *Sequence* untuk ditempatkan pada *sequence* berikutnya, dan hilangkan dari Daftar Kandidat Penempatan *Sequence*. Kembali ke Langkah 2.

   Selesai

Hasil keluaran algoritma SGSMod merupakan bentuk fenotip dari solusi *sequence*, sehingga perlu dipetakan ke dalam bentuk genotip dengan proses kodifikasi. Oleh karena bentuk fenotip hasil keluaran algoritma SGSMod adalah layak terhadap relasi *precedence* dan pemetaan solusi-gen menggunakan pemetaan 1-1, maka kromosom hasil kodifikasi dari bentuk fenotipnya merupakan kromosom layak.

Tabel 2. Ilustrasi Algoritma SGSMod

Lokus	Kandidat Penempatan Operasi							Range Bil. Acak	Bil. Acak	Operasi Terpilih
	1	2	3	4	5	6	7			
1	1,1	1,2	2,1	2,2	3,1	3,2	3,3	[1,7]	2	1,2
2	1,1	2,1	2,2	3,1	3,2	3,3		[1,6]	6	3,3
3	1,1	2,1	2,2	3,1	3,2			[1,5]	3	2,2
4	1,1	2,1	2,2	3,1	3,2			[1,5]	2	2,1
5	1,1	2,2	2,3	3,1	3,2			[1,5]	1	1,1
6	1,3	2,3	3,1	3,2				[1,4]	1	1,3
7	1,4	2,3	3,1	3,2				[1,4]	4	3,2
8	1,4	2,3	3,1					[1,3]	2	2,3
9	1,4	2,4	3,1					[1,3]	3	3,1
10	1,4	2,4	3,4					[1,3]	2	2,4
11	1,4	3,4						[1,2]	2	3,4
12	1,4							[1,1]	1	1,4

3.4. Evaluasi Fungsi Fitness

Algoritma genetika yang dikembangkan ini mempergunakan ukuran evaluasi fungsi fitness *tardiness* tertimbang total. Apabila saat selesai suatu job melebihi *due-date*-nya, maka terdapat penalti sebesar keterlambatan yang dikalikan dengan suatu bobot.

Nilai *fitness* tersebut didapatkan dengan menyusun jadwal terlebih dahulu. Jadwal disusun berdasarkan kromosom dengan mempertimbangkan kelayakan terhadap relasi *precedence* dan ketersediaan sumberdaya. Suatu kromosom terlebih dahulu didekodifikasi menjadi bentuk fenotip job-operasi. Bentuk fenotip ini merupakan urutan penempatan setiap operasi pada jadwal. Selanjutnya, operasi-operasi ini dijadwalkan pada slot waktu paling awal (jadwal *non delay*) yang memenuhi kedua kendala di atas.

Saat paling awal dimulainya suatu operasi ditentukan oleh dua hal secara bersama-sama. Pertama adalah maksimal dari saat selesainya semua operasi *predecessor*, dan maksimal saat tersedianya unit sumberdaya yang mencukupi untuk semua jenis sumberdaya yang dipergunakan. Sebelum suatu operasi ditempatkan pada slot waktu tertentu, perlu dilakukan

pemindaian apakah tersedia cukup sumberdaya untuk semua titik pada slot waktu yang diinginkan. Proses ini harus dilakukan oleh karena adanya keragaman ketersediaan sumberdaya antar slot waktu, yang memungkinkan terjadinya pelanggaran terhadap pembatas sumberdaya pada suatu titik dalam slot waktu tersebut.

3.5. Operator Genetika

Elitis

Beberapa individu terbaik disalin dari populasi saat ini ke populasi baru. Strategi ini dinamakan elitis dan manfaat utamanya yaitu bahwa solusi terbaik dipertahankan secara monoton dari satu generasi ke generasi berikutnya. Pada penelitian ini, elitis diterapkan terhadap keseluruhan individu ATAS.

Crossover

Hal terpenting dalam melakukan crossover adalah individu keturunan menghasilkan solusi yang layak (Goncalves et.al., 2006). Pada kasus penjadwalan multi-job, mekanisme *crossover* perlu mempertimbangkan adanya operasi dari job-job lain dan memperhatikan relasi *precedence* pada setiap job. Untuk itu, dilakukan crossover dengan cara pertukaran gen dalam satu paket job sehingga dapat mempertahankan relasi *precedence* pada suatu job.

Dimisalkan dua individu dari populasi saat ini, dipilih untuk *crossover* sebagai individu *A* yang dipilih secara acak dari populasi ATAS dan individu *B* yang dipilih secara acak dari keseluruhan populasi. Dari individu *A* dan *B* ini akan dihasilkan satu individu keturunan untuk menggantikan salah satu individu pada populasi antara populasi ATAS dan BAWAH.

Pertama-tama dibangkitkan bilangan random q_1=[1,N_1] dan q_2=[1,N_2], dimana q_1 menentukan posisi awal lokus gen yang akan terlibat crossover dan q_2 sebagai lebar *crossover*. Lebar *crossover* didefinisikan sebagai jumlah lokus gen-gen yang akan dilibatkan dalam *crossover*, dimulai dari posisi q_1. Nilai N_1 merupakan jumlah gen dalam satu individu, sedangkan N_2 adalah bilangan yang dalam hal ini ditentukan N_2=5.

Tahap selanjutnya adalah menandai job-job yang terkait dengan gen-gen, pada posisi lokus [q_1,q_1+q_2-1] jika q_1+q_2-1<N_1 atau pada posisi [q_1,N_2] jika sebaliknya, pada kedua induk. Setelah itu, gen-gen individu *A* dan individu *B* yang terkait dengan job yang telah ditandai akan dipertukarkan. Dalam hal ini, individu *A* harus memiliki probabilitas yang lebih tinggi daripada individu *B* untuk mewariskan sifatnya pada individu keturunan, karena gen-gen pada individu *A* memiliki sifat yang lebih unggul. Untuk itu diterapkan probabilitas *crossover*

sebesar *Cprob* untuk memilih individu mana yang akan mewariskan lebih banyak gen pada individu baru. Apabila dibangkitkan bilangan acak x=[0,1] dimana $x \leq Cprob$, maka gen-gen yang terkait dengan job yang tidak ditandai pada individu *A* akan diwariskan pada individu baru, sedangkan sisanya akan diisi oleh gen-gen dari individu B yang terkait dengan q_1 dan q_2. Mekanisme *crossover* tersebut digambarkan sebagai berikut (kromosom pada gambar merupakan bentuk fenotip, untuk memudahkan ilustrasi):

	1	2	3	4	5	6	7	8	9	10	11	12
Genotip Ind. A	1	2	5	9	3	6	7	10	4	11	12	8
Fenotip Ind. A	1,1	1,2	2,1	3,1	1,3	2,2	2,3	3,2	1,4	3,3	3,4	2,4
Fenotip Ind. B	1,2	3,3	2,2	2,1	1,1	1,3	3,2	2,3	3,1	2,4	3,4	1,4
Genotip Ind. B	2	11	6	5	1	3	10	7	9	8	12	4

q_1=3, q_2=2

- Job yang ditandai adalah job 2 dan job 3
- Misal, *Cprob* = 0.75 dan x=0.61

Gambar 6. Penentuan posisi q1 dan q2 untuk menandai job

Lokus Ind. A	3	4	6	7	8	10	11	12
Fenotip Ind. A	2,1	3,1	2,2	2,3	3,2	3,3	3,4	2,4
	↓	↓	↓	↓	↓	↓	↓	↓
Lokus Ind. B	2	3	4	7	8	9	10	11
Fenotip Ind. B	3,3	2,2	2,1	3,2	2,3	3,1	2,4	3,4

Gambar 7. Mekanisme pertukaran gen pada crossover untuk job terkait

Sedangkan, individu keturunan yang dihasilkan dapat dilihat pada Gambar 8 berikut:

		1	2	3	4	5	6	7	8	9	10	11	12
Fenotip Keturunan	Ind.	1, 2	2, 1	3, 1	2, 2	1, 1	1, 3	2, 3	3, 2	3, 3	3, 4	2, 4	1, 4
Genotip Keturunan	Ind.	2	5	9	6	1	3	7	10	11	12	8	4

Gambar 8. Individu hasil crossover

Mutasi

Sesuai dengan strategi evolusi di atas, operator mutasi diterapkan pada keseluruhan individu BAWAH. Mutasi dilakukan dengan cara pertukaran lokus relatif antar dua job, dimana pemilihan job-job tersebut dilakukan secara acak.

Misalkan, job pertama adalah job dengan saat mulai operasi pertama lebih awal daripada job kedua. Maka, job kedua akan digeser ke depan sehingga saat mulai operasi pertama pada job kedua tersebut akan menempati saat mulai operasi pertama pada job pertama. Pergeseran job kedua dilakukan dengan mempertahankan urutan dan posisi relatif yang tetap. Sedangkan job pertama akan digeser ke belakang untuk menempati posisi-posisi yang kosong. Sementara itu, posisi dan urutan operasi-operasi pada job lain yang tidak terlibat dalam mutasi tetap dipertahankan. Dengan mekanisme tersebut, hasil susunan jadwal akan berubah dengan tetap mempertahankan kelayakan solusi terhadap relasi precedence.

Sebagai contoh, misalkan individu *B* pada contoh crossover diatas mengalami mutasi. Secara acak terpilih, job yang terlibat dalam mutasi adalah job 2 dan job 3. Operasi-operasi dari job yang terkait mutasi, digambarkan sebagai urutan yang didekatkan satu sama lain sebagai berikut:

Operasi dari job terkait:	3,3	2,2	2,1	3,2	2,3	3,1	2,4	3,4
Pertukaran posisi I:	2,2	2,1	3,3	2,3	3,2	2,4	3,1	3,4
Pertukaran posisi II:	3,3	2,2	3,2	2,1	3,1	3,4	2,3	2,4

Gambar 9. Pertukaran posisi gen pada proses mutasi

Individu *A* asal	1,2	3,3	2,2	2,1	1,1	1,3	3,2	2,3	3,1	2,4	3,4	1,4
Hasil mutasi I	1,2	2,2	2,1	3,3	1,1	1,3	2,3	3,2	2,4	3,1	3,4	1,4
Hasil mutasi II	1,2	3,3	2,2	3,2	1,1	1,3	2,1	3,1	3,4	2,3	2,4	1,4

Gambar 10. Hasil proses mutasi individu A

4. Percobaan Komputasi

4.1. Deskripsi Percobaan

Pada bagian ini disajikan hasil percobaan komputasi. Percobaan dilakukan dengan menggunakan komputer Pentium M 4 - 1.7 GHz, dengan RAM 256-*share* 32. Algoritma genetika pada penelitian ini dikodekan dalam bahasa pemrograman Visual Basic 6.0.

Algoritma genetika diujicoba dengan serangkaian contoh kasus hipotetik pada penelitian Partono (2004), sebanyak 25 contoh kasus dengan tingkat kompleksitas yang beragam. Untuk tiap contoh, dilakukan replikasi sebanyak lima puluh (50) kali untuk menjaga validitas hasil percobaan. Konfigurasi parameter algoritma genetika ditentukan sesuai dengan tabel berikut:

Tabel 3. Konfigurasi parameter Algoritma Genetika

Ukuran populasi	3 kali jumlah keseluruhan operasi
Individu ATAS, Elitis	30 % dari populasi
Probabilitas Crossover	0.7
Individu BAWAH, Mutasi	10 % dari populasi saat ini
Kriteria henti	2 kali jumlah keseluruhan operasi

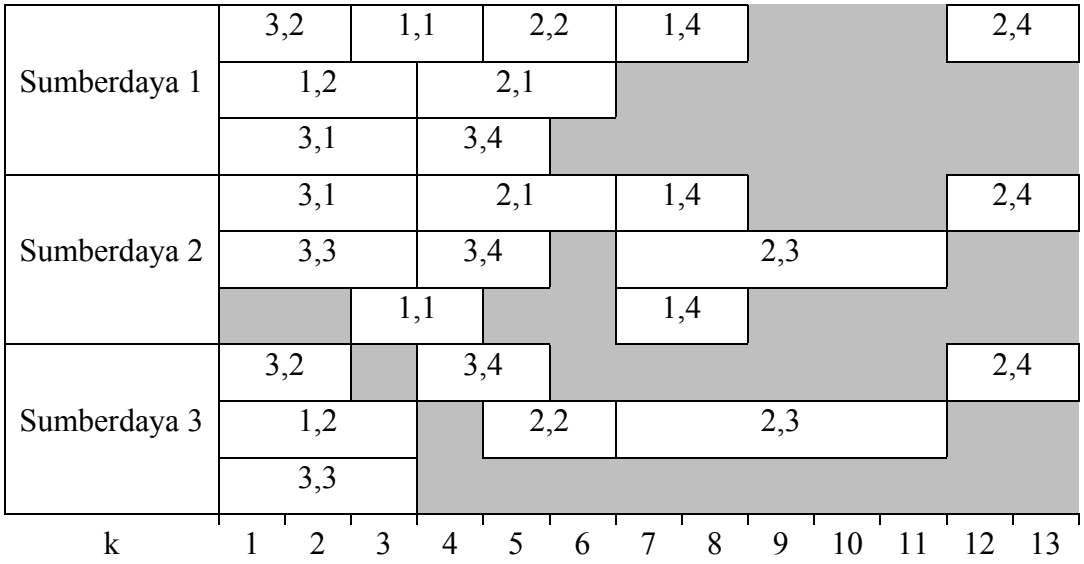
4.2. Ilustrasi Solusi Masalah Penjadwalan

Bagian ini menunjukkan sebuah ilustrasi mengenai solusi masalah penjadwalan terhadap percobaan yang dilakukan. Misal, diberikan permasalahan seperti digambarkan oleh Gambar 1 di atas dengan data seperti pada Tabel 4.

Tabel 4. Data detail contoh sesuai Gambar 1

Job	Operasi	Operasi Pendahulu	Pemakaian Sumberdaya			Waktu Pengerjaan	*Due Date*	Bobot
			1	2	3			
1	1		√	√	-	2	2	3
	2		√	-	√	3		
	3	1 dan 2	-	√	√	2		
	4	3	√	√	√	2		
2	1		√	√	-	1	1	1
	2		√	-	√	2		
	3	1	-	√	√	5		
	4	2 dan 3	√	√	√	2		
3	1		√	√	-	3	5	6
	2		√	-	√	2		
	3		-	√	√	3		
	4	1, 2, dan 3	√	√	√	2		

Algoritma genetika memberikan solusi sesuai Gambar 11 sebagai Gantt-Chart penjadwalan sumberdaya. Pada contoh ini, algoritma genetika mampu menemukan solusi optimal.



Gambar 11. Penjadwalan sumberdaya contoh Gambar 1

Algoritma genetika yang dibangun selanjutnya diujicobakan pada sepuluh permasalahan jenis yang dibangkitkan, dengan spesifikasi sebagai berikut:

Tabel 5. Spesifikasi umum data percobaan

Contoh	**Jumlah Job**	**Jumlah Operasi**	**Jumlah sumberdaya**	**Kapasitas Sumberdaya**	**Struktur Job**
Contoh 1	3	4,4,4	3	3,3,3	Pohon
Contoh 2	4	3,3,3,3	2	2,2	Lurus
Contoh 3	5	3,3,3,3,3	2	2,2	Lurus
Contoh 4	2	4,4	2	2,1	Pohon
Contoh 5	3	3,3,3	3	2,2,2	Pohon
Contoh 6	2	5,4	2	2,1	Pohon
Contoh 7	3	2,3,4	2	2,2	Pohon
Contoh 8	4	2,3,3,4	2	2,2	Pohon
Contoh 9	3	4,4,4	2	3,3	Pohon
Contoh 10	2	5,5	2	2,2	Pohon

Hasil percobaan terhadap 10 data uji dengan replikasi masing-masing sebanyak 50 kali percobaan dapat menemukan solusi optimal dengan rekapitulasi hasil sebagai berikut:

Tabel 6. Rekapitulasi hasil percobaan pada data uji

Contoh	Penyelesaian dengan Lingo 8.0		Pendekatan Algoritma Genetika yang Dikembangkan			
	Solusi Optimal	Waktu Komputasi (detik)	Range Fungsi Tujuan	Rata-rata Fungsi Tujuan	Rata-rata Waktu Komputasi (detik)	Deviasi Fungsi Tujuan
Contoh 1*	30	1,0	-	30,00	1,0	0,00 %
Contoh 2*	61	1,0	-	61,00	1,0	0,00 %
Contoh 3*	31	1,0	31 - 36	31,24	1,0	0,77 %
Contoh 4*	62	1,0	-	62,00	1,0	0,00 %
Contoh 5*	31	1,0	-	31,00	1,0	0,00 %
Contoh 6*	23	1,0	23 - 26	23,42	1,0	1,83 %
Contoh 7*	26	1,0	26 - 26	26,00	1,0	0,00 %
Contoh 8*	77	1,0	77 - 86	77,18	1,0	0,23 %
Contoh 9*	32	1,0	32 - 36	32,56	1,0	1,75 %
Contoh 10*	35	1,0	-	35,00	1,0	0,00 %

5. Kesimpulan

Penelitian ini memaparkan Algoritma Genetika untuk menyelesaikan permasalahan penjadwalan job-majemuk dengan penggunaan sumberdaya-majemuk menggunakan fungsi tujuan meminimumkan *tardiness* tertimbang total. Mekanisme inisialisasi populasi, mekanisme *crossover* dan mekanisme mutasi untuk job majemuk dan operasi majemuk dikembangkan untuk mewujudkan algoritma genetika yang dimaksud.

Berdasarkan percobaan dengan data uji Algoritma Genetika berhasil menemukan solusi optimal. Simpulan ini diperoleh dari hasil percobaan terbatas pada 10 contoh data uji. Untuk tujuan penarikan kesimpulan yang lebih akurat, maka algoritma genetika yang dikembangkan dapat diuji terhadap contoh-contoh dengan tingkat kompleksitas yang lebih tinggi daripada tingkat kompleksitas contoh-contoh permasalahan yang disajikan. Selain menguji dengan data

uji kompleksitas lebih tinggi, algoritma dapat dikembangkan untuk menyelesaikan permasalahan penjadwalan yang lebih kompleks.

6. Daftar Pustaka

[1] Baker, K.R. (1974), *Introduction to Sequencing dan Scheduling*, John Wiley & Sons, New York.

[2] Baker, K.R. (2001), *Elements of Sequencing and Scheduling*, John Wiley & Sons, New York.

[3] Conway, R.W. et.al. (1967), *Theory of Scheduling*, Addison-Wesley Publishing Company, Palo Alto.

[4] Dobson, G. dan Karmakar U.S. (1988), *Simultaneous Resource Scheduling to Minimize Weighted Flow Times*, *Operation Research*, 37(4), pp. 592-600.

[5] Dobson, G. dan Khosla I. (1995), *Simultaneous Resource Scheduling with Batching to Minimize Weighted Flow Times, IIE Transactions*, 27, 587-598.

[6] Goncalves, J.F. et.al. (2006), *A Genetic Algorithm for The Resource Constrained Multi-Project Scheduling Problem*, AT&T Labs Research Technical Report.

[7] Hartmann, S. (2002), *A Self-Adapting Genetic Algorithm for Project Scheduling under Resource Constraints*, *Naval Research Logistics* 49:433–448.

[8] Hoitomt, D.J., P.B. Luh, dan K.R. Pattipati (1993), *A Practical Approach to Job Shop Scheduling Problems*, IEEE Transactions on Robotics dan Automation, 9(1), pp. 1-13.

[9] Ingber, L. dan Rosen, B. (1992), Genetic Algorithms And Very Fast Simulated Reannealing: A Comparison**,** *Mathematical and Computer Modelling*, 16(11) 1992, 87-100.

[10] Kamarainen, O. et. al (1999), *A Generalized Resource Constrained Production Scheduling Problem: An Evolutionary Algorithm Approach*, Technical Report.

[11] Kolisch, R. and Harttman, S. (1998), *Heuristic Algorithms for Solving The Resource Constrained Project Scheduling Problem - Classification and Computational Analysis*, kontribusi pada "*Handbook on Recent Advances in Project Scheduling*".

[12] Luh, P.B. et.al. (1990), Schedule Generation and Reconfiguration for Parallel Machines, IEEE Transactions on Robotics and Automation, Vol. 6, No.6, pp. 687-696.

[13] Kobayashi, S., Ono, I. and Yamamura, M. (1995), *An efficient genetic algorithm for job shop scheduling problems*, *In Proceedings of the 6th ICGA*, pages 506-511.

[14] Morton, T.E. dan D.W. Pentico (1993), *Heuristic Scheduling Systems: with Applications to Production Systems and Project Management*, John Wiley and Sons, New York.

[15] Norman B.A. et al (1999), *A Genetic Algorithm Methodology for Complex Scheduling Problems*, *Naval Research Logistics,* Vol. 46.

[16] Partono, S. (2004), Algoritma Berbasis Relaksasi Lagrange untuk Pemecahan Masalah Penjadwalan Job-Majemuk, Operasi Majemuk, Sumber-Majemuk Paralel Simultan, Tugas Akhir Sarjana, Teknik Industri, Institut Teknologi Bandung.

[17] Pinedo, M. dan Chao, X. (1999), *Operations Scheduling with Apllications in Manufacturing and Services*, Irwin McGraw Hill.

[18] Sabuncuoglu, I. dan Bayiz M. (1998), *Job shop scheduling with beam search*. European Journal of Operational Research 118 (1999) 390-412

[19] Sakalauskas, L. dan Felinskas, G. (2006), *Optimization of Resource Constrained Project Schedules by Genetic Algorithm Based on The Job Priority List*, Information Technology And Control, 2006, Vol.35, No.4.

[20] Suprayogi, dan Mardiono N.M.T. (1997), Pengembangan Algoritma Bagi Masalah Penjadwalan Banyak Job Operasi Tunggal Banyak Sumber Paralel Simultan, *Jurnal Teknik dan Manajemen Industri*, 17, pp. 39-49.

[21] Suprayogi, dan Partono S. (2005), Pendekatan Pemecahan Berbasis Relaksasi Lagrange untuk Masalah Penjadwalan Job-Majemuk, Operasi Majemuk, Sumber Majemuk Paralel, Prosiding Seminar Sistem Produksi, VII.2005, ISSN, 0854-43IX.

[22] Suprayogi, dan Toha I.S. (2002), Model Pemrograman Linier Bilangan Bulat Untuk Masalah Penjadwalan Sumber-Majemuk Paralel Simultan, *Jurnal Teknik dan Manajemen Industri*, 22(1), pp. 27-38.

[23] Toha, I.S. dan A.H. Halim (1999), Algoritma Penjadwalan Produksi Berbasis Jaringan Untuk Penggunaan Sumber Tunggal dan Simultan, *Jurnal Teknik dan Manajemen Industri*, 19(2), pp. 10-20.