MULASI PENJADWALAN *JOB-SHOP* DINAMIS DENGAN MENGGUNAKAN ALGORITMA GENETIKA

TUGAS AKHIR

Disusun Oleh :

ADITYA EKA SAPUTRA

NRP. 5201 100 038

IULASI PENJADWALAN *JOB-SHOP* DINAMIS DENGAN MENGGUNAKAN ALGORITMA GENETIKA

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Komputer

Pada

Program Studi Sistem Informasi

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya

Mengetahui/Menyetujui

osen Pembimbing 1

Dosen Pembimbing 2

3/1/12

ABSTRAK

Dalam sebuah sistem produksi yang kompleks dapat terjadi penumpukan ekerjaan atau barang yang membentuk antrian panjang yang tidak dapat selesaikan secara optimal. Sistem produksi yang melibatkan banyak proses, esin dan juga waktu proses yang bervariasi akan menemui banyak hambatan bila dak ada metode penjadwalan yang tepat. Dalam proses produksi secara eseluruhan, keadaan demikian dapat menyebabkan sistem tidak dapat bekerja ecara efisien dan efektif.

Dalam tugas akhir ini dibuat sebuah aplikasi untuk manangani enjadwalan *job-shop* dinamis dengan menggunakan algoritma genetika. Sebuah enjadwalan *job-shop* dikatakan dinamis apabila tugas yang akan dijadwalkan emiliki aliran proses dan waktu kedatangan yang berbeda-beda. Sebuah enjadwalan *job-shop* dinamis juga harus mampu mengubah hasil penjadwalan etika sebuah tugas baru datang, walaupun penjadwalan sebelumnya telah rbentuk. Untuk mendapatkan hasil penjadwalan yang optimal maka digunakan goritma genetika. Tugas yang masuk akan mengalami proses *encoding* agar apat diproses dengan menggunakan algoritma genetika. Hasil proses *encoding* i adalah sebuah kromsom. Kromosom inilah yang akan diproses oleh operasi enetik untuk mendapatkan solusi penjadwalan yang optimal. Sebuah solusi enjadwalan dikatakan optimal apabila memiliki waktu rata-rata produksi eminimal mungkin. Akhirnya, keluaran atau hasil operasi genetik akan diubah embali menjadi sebuah jadwal yang mudah dipahami oleh pengguna

Hasil uji coba menunjukkan bahwa peningkatan optimasi penjadwalan kan berpengaruh terhadap peningkatan waktu komputasi. Untuk mendapatkan olusi yang semakin optimal, maka waktu komputasi yang dibutuhkan juga emakin meningkat Dari hasil pengujian juga didapatkan bahwa, perubahan arameter genetik akan mempengaruhi optimasi solusi yang dihasilkan. erubahan parameter laju seleksi, jumlah keturunan dan jumlah populasi emberikan pengaruh yang sangat signifikan terhadap optimasi penjadwalan. edangkan perubahan parameter laju tukar silang dan laju mutasi tidak emberikan pengaruh yang signifikan terhadap hasil penjawalan. Penggunaan emori terbesar terdapat pada proses perhitungan nilai *fitness* populasi dan proses

Dipersembahkan Kepada

Tuhan dan Sahabat yang Setia, Yesus Kritus

Semuanya hanya bagiMu

KATA PENGANTAR

Segala puji syukur kepada Tuhan Yesus Kristus, karena anugrahNya kita pat menjalani hari ini dengan segala berkat-berkatNya. Semua hanya karena ıta, apabila kita hari ini mampu berdiri tegak. Diberkatilah orang yang engandalkan dan menaruh harapannya pada Tuhan dalam jalannya, karena hari-rinya tidak akan kering dan tidak akan berhenti menghasilkan buah.

Kurang lebih 4 bulan dilalui dalam penulisan Tugas akhir ini sebagai menuhan syarat kurikulum akademik Program Studi Sistem Infomasi, Fakultas knologi Informasi – Institut Teknologi Sepuluh November. Tugas Akhir ini kkan pernah terwujud tanpa dukungan, saran, dan bantuan dari berbagai pihak. ıtuk itu, penulis menyampaikan rasa terima kasih kepada:

Tuhan dan sahabat saya, Yesus Kristus yang telah setia menemani, menyertai saya senantiasa dalam perngerjaan tugas akhir ini dari mula sampai pada akhirnya. "*Segala perkara dapat kutanggung di dalam Dia yang memberi kekuatan padaku*", Flp 4:13.

Kepada kedua orang tua yang telah dengan lelah dan setia membesarkanku, tugas akhir ini kupersembahkan. Berharap tugas akhir ini dapat menjadi balasan atas kasih dan sayang telah diberikan selama ini.

My Shinning Moon yang telah memberikan suka dan duka dalam kehidupan saya, serta memberikan semangat dalam mengerjakan tugas akhir ini. *"Aku mengucap syukur kepada Allahku setiap kali aku mengingat kamu"*.

Kepada kedua adikku tercinta yang telah memberikan senyuman hangat dan

S.Kom, MT, Wiwik Anggraeni, S.Si, M.Kom, Ir. Akhmad Holil N.A, M.Kom, Bpk. Mudjahidin, ST, MT, Bpk. Faizal Johan A, S.Kom, Bpk. Ir. Aris Tjahyanto, M.Kom.

Karyawan Sistem Informasi dan FTIf pada umumnya: Mas Kadir, Mbak Anita Pak Yudi, Pak Narno, Bu Tutik, Pak Muin, Pak Karmono, Pak Soleh, Mas Sugeng, Mbak Eva, dkk.

Baladewa DSS yang selalu membantu dan menemani pada saat menanti bimbingan, Angga, Ansor, Titus, Riza, Cel-Fatwa, dan yang terakhir Adhi

. Kawan dan rekan seperjuangan 2001 yang telah membantu dan mendukung dalam pengerjaan tugas akhir ini. **VIVAT SI**

. Semua pihak yang telah membantu dalam pengerjaan tugas akhir tidak dapat disebutkan satu persatu. Sekali lagi terima kasih

Penulis menyadari masih banyak kekurangan dalam Tugas Akhir ini. ntuk itu penulis mengharapkan adanya pengembangan, kritik dan saran yang pat menyempurnakan Tugas Akhir ini. Akhirnya, harapan penulis, semoga ugas Akhir ini dapat bermanfaat bagi pembaca.

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SEGMENTASI PROGRAM

BAB I

PENDAHULUAN

BAB 1
PENDAHULUAN

Dalam sebuah sistem produksi yang kompleks dapat terjadi penumpukan ekerjaan atau barang yang membentuk antrian panjang yang tidak dapat iselesaikan secara optimal. Sistem yang melibatkan banyak proses, mesin dan uga waktu proses yang bervariasi akan menemui banyak hambatan bila tidak ada ietode penjadwalan yang tepat. Dan akhirnya mengakibatkan proses produksi ienjadi tidak efisien dan efektif.

Objektif dari masalah penjadwalan *job-shop* adalah bagaimana menyusun emua operasi dari semua *job* pada tiap mesin sehingga keseluruhan *job* dapat iproses menurut urutan pengerjaannya. *Job* atau yang kemudian disebut dengan ugas dapat berupa produk, atau tugas tertentu. Menurut waktu kedatangan sebuah ugas, penjadwalan *job-shop* dapat dibedakan menjadi dua, yaitu penjadwalan *ob-shop* statis dan penjadwalan *job-shop* dinamis . Untuk penjadwalan *job-shop* tatis, seluruh tugas diterima pada waktu yang bersamaan atau waktu kedatangan eluruh tugas adalah sama. Sedangkan penjadwalan *job-shop* dinamis, waktu edatanggan tugas bervariasi atau tidak bersamaan.

Sistem penjadwalan yang bersifat dinamis, adalah sebuah sistem yang ampu menangani perubahan proses produksi, kemudian secara langsung emberikan solusi. Dalam tugas akhir ini, perubahan proses produksi yang imaksud adalah kedatangan suatu tugas yang tidak direncanakan sebelumnya.

ehingga permasalahan penjadwalan dinamis dalam tugas akhir ini dilakukan :cara simulasi.

Solusi permasalahan penjadwalan dapat menjadi sebuah permasalahan aru karena banyaknya kemungkinan dan kombinasi solusi yang berkembang :cara supereksponensial seiring meningkatnya tugas dan sumber daya / mesin. ntuk menjawab persoaalan diatas, dikembangkan sebuah fasilitas atau metode ang disebut dengan algoritma genetik. Dalam implementasinya, algoritma enetik telah terbukti handal dan dapat diaplikasikan secara luas dalam pencarian ang bersifat stochastik (*Stochastic Search*) dan permasalahan optimasi. Dalam eberapa tahun ini, Algoritma Genetik banyak diimplementasikan pada ermasalahan optimasi pada *industrial engineering* dan design sistem manufaktur.

.1 Tujuan

Titik berat atau tujuan dalam tugas akhir ini adalah melakukan pemodelan an simulasi mengenai penjadwalan *job-shop* yang bersifat dinamis sehingga idapatkan suatu jadwal produksi yang optimal. Diharapkan dengan adanya enjadwalan yang optimal dapat meningkatkan efisiensi dan efektivitas produksi.

.2 Permasalahan

Permasalahan yang dihadapi dalam pengembangan tugas akhir ini adalah ebagai berikut :

a. Bagaimana mengatasi alternatif solusi yang besar dan kompleks

Bagaimana mengatasi perubahan yang bersifat dinamis

Pada sebuah penjadwalan yang bersifat dinamis, perubahan selalu terjadi an secara berkala penjadwalan tersebut di update. Sistem ini harus selalu peka rhadap perubahan sehingga penjadwalan dapat diatur atau disesuaikan dengan ermasalahan yang sedang terjadi tetapi tanpa menghentikan proses produksi. ebagai contoh, apabila datang sebuah tugas yang bersifat kritis atau berprioritas nggi yang harus segera diselesaikan maka jadwal yang telah ada harus di update ehingga proses produksi tetap berjalan.

Bagaimana merancang sebuah model yang sesuai untuk simulasi

Dalam merancang sebuah aplikasi dan simulasi kita membutuhkan sebuah odel yang dapat mewakili permasalahan dan penyelesaian masalah. Dengan erancang model yang benar, dapat memudahkan kita dalam mengerti dan enyelesaikan masalah yang terjadi.

Bagaimana mengkodekan solusi ke dalam kromosom

Keberhasilan algoritma genetik dipengaruhi oleh model atau pengkodean olusi ke dalam kromosom yang kemudian dapat diolah oleh algoritma genetik. Model kromosom ini sangat mempengaruhi fungsi evaluasi dan solusi yang idapat.

Bagaimana menentukan fungsi evaluasi yang sesuai

Fungsi evaluasi ini merupakan bagian terpenting dalam algoritma genetik

Bagaimana menyajikan informasi kepada pengguna

Sebuah perangkat lunak yang canggih dan mahal tidak akan berguna bila perangkat lunak tersebut tidak dapat memberi informasi yang berguna, mudah dimengerti bagi user. Penyajian informasi menjadi permasalahan yang cukup berarti dalam pembuatan perangkat lunak. Karena alasan tersebut, dalam tugas akhir ini permasalahan penyajian informasi menjadi masalah yang harus diperhatikan .

1.3 Batasan Masalah

1. Tiap tugas hanya diproses oleh sebuah mesin maksimal sekali, dan proses tidak berulang
2. Tugas yang telah masuk ke dalam sistem tidak dapat dihentikan atau dihapus.
3. Tidak memiliki tenggat waktu setup. Diasumsikan bahwa mesin berjalan terus selama 24 jam sehari tanpa berhenti
4. Tidak ada waktu perpindahan dari satu mesin ke mesin yang lain, diasumsikan waktu perpindahannya adalah 0
5. Tidak ada masalah lain yang timbul dalam proses produksi, selain kedatangan tugas

1.4 Metodologi

Pembuatan tugas akhir ini terbagi menjadi beberapa tahap pengerjaan yang tertera sebagai berikut :

1.4.1 Studi Kepustakaan

.4.2 Perancangan Model dan Simulasi

Tahap ini merupakan proses pemodelan sistem, mengenai masukan, roses, keluaran. Dalam tahap ini juga dilakukan pemodelan pengkodean solusi ke alam kromosom, fungsi evaluasi,

.4.3 Perancangan dan Implementasi Perangkat Lunak

Tahap ini merupakan proses analisa dan desain sistem. Dimana sistem ang dirancang diharapkan mampu menangani permasalahan penjadwalan yang ersifat dinamis. Sehingga bila ada interupsi dari lingkungan diharapkan sistem ni mampu memberikan solusi guna memudahkan pengambilan keputusan. Dengan perancangan dan design yang baik, diharapkan memberikan informasi una kemudahan bagi pihak managemen untuk mengambil keputusan. Sehingga apat dilakukan efisiensi dalam hal sumber daya, baik waktu, sdm, material

.4.4 Uji Coba dan Evaluasi

Pada tahap ini sistem yang telah dikembangkan di atas diuji tingkat kurasi dan kelayakannya, sehingga memberikan solusi yang optimal, akurat, dan alid dalam bebagai situasi dan sesuai dengan batasan yang ada. Pengujian ilakukan secara simulasi dengan melihat input yang diberikan, apakah sistem ampu mengatasi input yang bervariasi. Juga melihat pada output yang ihasilkan dari proses melalui input yang diberikan kepada sistem. Apakah output ang dihasilkan sesuai dengan yang diharapkan dan memenuhi semua batasan ang ada.

.5 Sistematika Laporan

Laporan tugas akhir ini disusun menjadi 6 bab utama dan beberapa bagian ambahan dan pelengkap.

BAB I PENDAHULUAN

Berisi latar belakang, tujuan dan manfaat tugas akhir, perumusan masalah, embatasan dan asumsi masalah, dan sistematika penulisan.

BAB II DASAR TEORI

Berisi dasar teori yang digunakan untuk menunjang pengerjaan tugas akhir ni. Berisi dasar teori mengenai proses produksi, penjadwalan produksi, dan teori lgorima genetik baik secara umum maupun yang digunakan dalam penjadwalan *ob-shop*

BAB III PERANCANGAN PERANGKAT LUNAK

Bab ini menjelaskan mengenai desain perangkat dan algoritma genetika ang secara khusus diimplementasikan pada parangkat lunak. Bab ini juga enjelaskan metode yang digunakan untuk perancangan perangkat lunak.

BAB IV IMPLEMENTASI PERANGKAT LUNAK

Dalam bab ini menjelaskan mengenai bagaimana implementasi perangkat unak pada tugas akhir ini. Implementasi perangkat lunak ini mengacu pada erancangan algoritma pada bab III sebelumnya.

BAB V UJI COBA DAN EVALUASI

BAB II

DASAR TEORI

BAB 2
DASAR TEORI

Dalam bab ini akan dijelaskan mengenai dua permasalah utama yang enjadi pokok pembahasan dalam tugas akhir ini yaitu teori penjadwalan roduksi dan algritma genetika. Banyak kepustakaan yang menjelaskan dan emberikan gagasan atau teori mengenai penjadawalan produksi dan teori goritma ganetika.

Dalam bab ini pembahasan mengenai teori penjadwalan dimulai dari enjelasan mengenai proses produksi. Dimana permasalahan penjadwalan roduksi sering kali ditemui dalam proses produksi perusahaan manufaktur. engan mengerti permasalahan proses produksi dan pembagiannya, maka kita apat dengan mudah memahami tipe penjadwalan produksi. Dimana tipe produksi ang diangkat dalam tugas akhir ini adalah *job-shop*. Tipe *job-shop* ini memiliki arakter yang berbeda dengan tipe proses produksi yang lain.

1 Proses Produksi

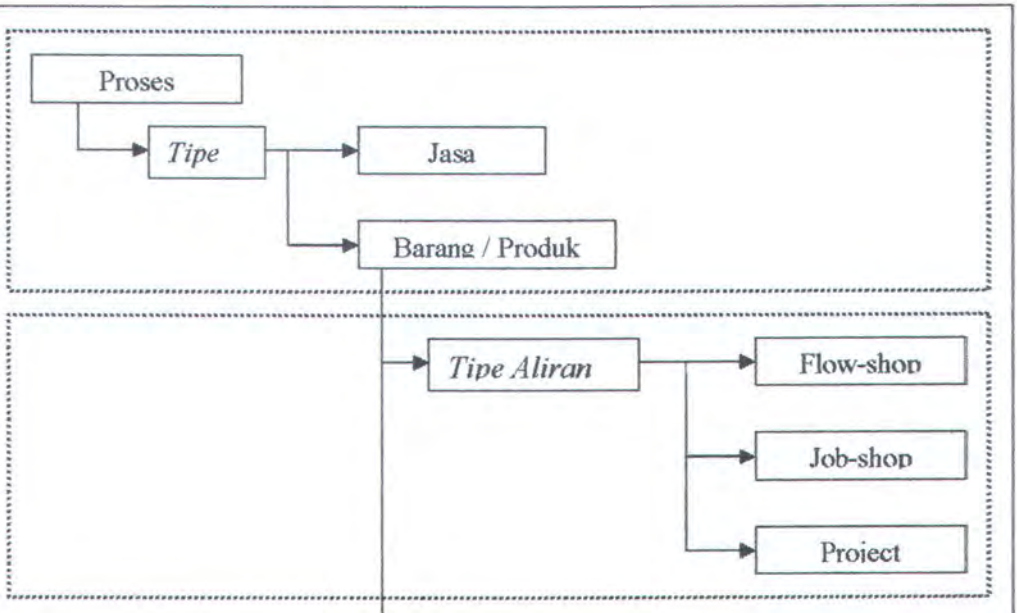
Adanya perbedaan yang besar dan kompleks atas kebutuhan manusia akan arang dan jasa mendorong munculnya berbagai macam jenis proses produksi. Mulai dari proses produksi yang sederhana seperti kerajinan tangan sampai roses produksi yang besar dan kompleks seperti perusahaan minyak.

an proses produksi untuk jasa. Untuk jenis jasa memiliki proses produksi yang elatif lebih sederhana dibandingkan dengan proses produksi barang. Pada umumnya pada proses produksi barang diperlukan sumberdaya yang lebih banyak an teknologi yang lebih kompleks.

Sedangkan proses produksi barang, menurut tipe aliran, dapat dibedakan menjadi tiga bagian besar. Mengenai tipe aliran ini akan dijelaskan lebih lanjut ada bagian berikutnya.

.2 Tipe aliran proses

Menurut aliran proses, sebuah sistem produksi dapat dibagi menjadi tiga, *ow-shop, Job-shop, Project.*



Pola *Project* merupakan proses produksi yang menghasilkan produk yang ıngat kompleks dan memiliki variasi produk yang sangat tinggi dengan engaturan sumberdaya yang kompleks. Untuk menghasilkan satu jenis produk stem produksi harus disesuaikan dengan kebutuhan produk. Sehingga satu jenis roduk memiliki susunan sistem produksi yang berbeda satu dengan yang lainnya. ada umumnya produk yang dihasilkan hanya sedikit untuk satu jenis produk rtentu.

.2.1 Pola *Flow-shop*

Pola *flow-shop* merupakan proses produksi dimana tugas melalui urutan roses yang sama dan tetap antara satu tugas dengan tugas yang lainnya. Dimana eralatan atau mesin yang digunakan mempunyai konfigurasi tetap dan digunakan ntuk kebutuhan yang khusus. Pola ini biasanya untuk produk dengan design ang relatif stabil atau tetap di setiap waktu. Dan biasanya berlaku pada produk ang dibuat untuk memenuhi kebutuhan pasar (*make to stock*).

Flow-shop dapat dibagi lagi menjadi dua macam yaitu *flow-shop* yang ersifat *continuous* dan *flow-shop* yang bersifat *intermittent* Pada pola *flow-shop* ang bersifat *continuous*, konfigurasi atau urutan mesin tetap dan tidak dapat iubah lagi Contoh *flow-shop* yang bersifat *continuous* adalah proses pembuatan okok, pembuatan kemasan makanan. Sedangkan untuk *flow-shop* yang bersifat *ıtermittent* urutan atau konfigurasi mesin pada waktu tertentu diubah untuk emenuhi kebutuhan khusus tetapi tidak mengubah fungsi dasar.

Pada umumnya penjadwalan *job-shop*, merupakan aktifitas engalokasikan beberapa produk atau tugas kedalam beberapa sumber daya engan aturan tertentu. Sumber daya tesebut dapat berupa produk, atau tugas. engan kata lain, pengalokasian sekumpulan tugas kedalam mesin menurut rutan proses atau mesin tertentu.

Apabila dibandingkan tingkat kompleksitas, *job-shop* lebih tinggi ibandinng dengan *flow-shop*. Karena *job-shop* memiliki pola aliran proses yang erbeda-beda antara tugas yang masuk, sedangkan pada *flow-shop* hanya memiliki ola aliran yang tetap. Sumber daya yang ada digunakan secara bergantian oleh erbagai macam tugas atau produk, sedangkan *flow-shop* hanya menangani satu acam tugas atau produk. Pada pola *job-shop* tiap tugas yang masuk memiliki rioritas yang berbeda, sedangkan pada *flow-shop* tiap tugas memiliki prioritas ang sama.

Secara umum permasalahan *job-shop* dijelaskan sebagai berikut. ermasalahan *job-shop* memiliki beberapa atribut, yaitu tugas, mesin, batasan-atasan lingkungan.

Tugas merupakan pekerjan yang harus diselesaikan dengan menggunakan esin-mesin yang ada dengan urutan tertentu. Sehingga tiap tugas dapat memiliki rutan pengerjaan yang berbeda antara tugas satu dengan tugas yang lainnya. engalokasian tugas kedalam mesin menurut urutan pengerjaan merupakan asalah utama yang sering dihadapi dalam penjadwalan proses produksi.

i merupakan indeks yang membedakan tingkat kebutuhan dan kepentingan suatu igas dibanding tugas yang lainnya.

Mesin merupakan sumberdaya yang akan digunakan dalam proses roduksi untuk mengolah atau memproses tugas sehingga menjadi produk atau arang yang diinginkan. Tiap mesin mempunyai fungsi yang berbeda dalam emproses tugas. Pengaturan tugas yang masuk dalam mesin harus mendapat erhatian yang lebih karena adanya batasan kapasitas mesin.

Batasan lingkungan dalam penjadwalan produksi datang dari banyak spek. Antara lain sumber daya manusia, persediaan material, permintaan pasar, ebijakan management.

.2.3 Pola Proyek

Seringkali sebuah pabrik atau perusahaan dituntut untuk menciptakan roduk atau jasa yang unik dengan jangka waktu tertentu,*short-time event*. Pada mumnya tuntutan ini menyangkut tugas yang besar dan rumit, tetapi dengan umberdaya yang terbatas. Untuk dapat menjalankan tugas yang besar dengan umberdaya yang terbatas dibutuhkan manajemen dan koordinasi yang baik.

Pola proyek merupakan sebuah proses penciptaan produk atau jasa yang ompleks dengan sumberdaya terbatas dalam suatu periode waktu tertentu. ontoh perusahaan yang menggunakan pola ini adalah, perusahaan film, ontraktor, *event organizer*. Pola proyek tidak hanya diimplementasikan oleh erusahaan yang berorientasikan proyek, tetapi juga perusahaan lain yang tidak

enjadwalan yang buruk akan menyebabkan penggunaan sumberdaya yang tidak isien, menimbulkan waktu sela (*delay*) dalam suatu aliran produksi.

.4 Penjadwalan Satu Mesin

Permasalahan penjadwalan yang paling sederhana adalah penjadwalan ada satu mesin. Permasalahan ini melibatkan banyak tugas dengan waktu proses ang bervariasi dan harus dikerjakan dengan mesin tunggal. Permasalahannya dalah menentukan tugas atau *job* mana yang harus diprose terlebih dahulu. Ada eberapa variabel dalam penjadwalan satu mesin ini, yaitu :

- Waktu pemrosesan (*processing time*), yaitu waktu yang diperlukan untuk mengerjakan tugas.

- Waktu penyelesaian (*completion time*), yaitu waktu yang dibutuhkan untuk mengerjakan tugas mulai ketika tugas tersebut masuk dalam jadwal sampai selesai. Waktu penyelesaian dirumuskan sebagai berikut,

$$C_i = w_i + t_i \qquad (2\text{-}1)$$

Dimana w_i adalah waktu tunggu tugas i sebelum diproses. Dengan kata lain tugas menunggu mesin selesai memproses tugas yang terlebih dahulu masuk. Sedangkan t_i adalah waktu pemrosesan tugas i

- Batas waktu (*due date*), yaitu batas waktu untuk tugas ke i harus sudah

Dimana d_i adalah batas waktu. Lateness bernilai negatif bila tugas diselesaikan sebelum waktu jatuh tempo, dan positif apabila telah melebihi tanggal jatuh tempo. Keterlambatan yang bernilai negatif menunjukkan kinerja yang baik dari sistem. Nilai keterlambatan yang benilai positif biasa disebut dengan *tardiness*.

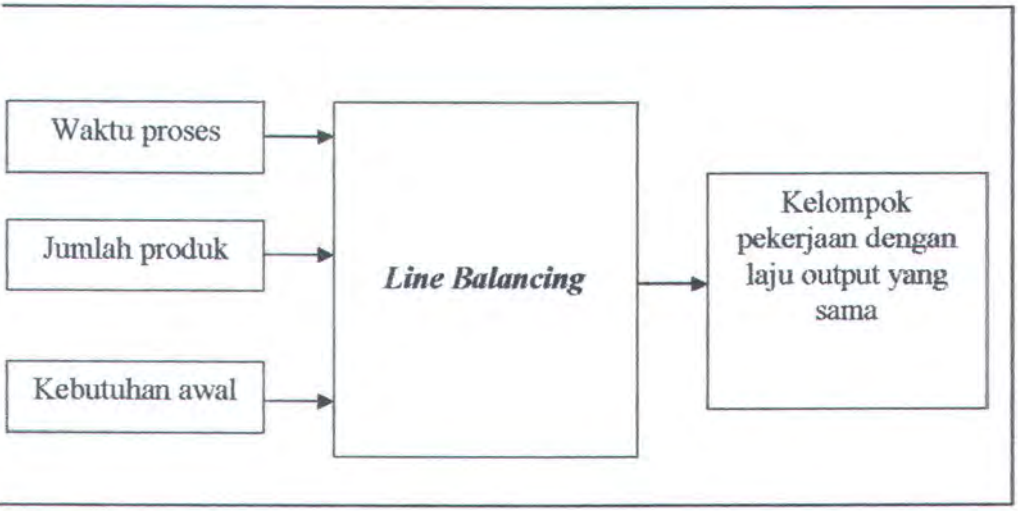
e. *Tardiness*, yaitu waktu keterlambatan dari tugas ***i*** jika gagal mencapai batas akhir atau nol untuk yang lain.

$$T_i = max\ [0, L_i] \qquad (2\text{-}3)$$

.5 Penjadwalan *Flow-shop*

Berbeda dengan penjadwalan satu mesin, penjadwalan ini menyangkut anyak mesin tetapi mempunyai urutan proses yang sama dan tetap. Seperti telah ijelaskan diatas penjadwalan ini ditujukan pada proses produksi dengan onfigurasi mesin tetap, jumlah produk akhir besar. Pola *flow-shop* hanya igunakan dalam memproduksi satu macam produk saja dan tidak dapat elakukan pengembangan yang spesifik dari suatu produk. Sehingga produk yang ihasikan sama satu dengan yang lainnya.

Yang menjadi permasalahan dalam penjadwalan *flow-shop* adalah agaimana mengumpulkan tugas yang sama sehingga sesuai dengan urutan proses



Gambar 2-2 permasalahan *Line-Balancing*

Untuk mengatasi permasalahan *Line-balancing* diperlukan metode khusus, imana metode ini tergantung lama proses. Ada tiga kemungkinan lama proses ang terjadi , yaitu :

- *Mechanically paced produktion line*. Dalam kasus ini waktu produksi tiap bagian produksi adalah konstant.
- *Stochastic production line*. Terjadi dimana waktu produksi tiap bagian produksi tidak tetap, sesuai distibusi statistika.
- *Mixed produktion line*. Merupakan gabungan dari kedua jenis waktu produksi di atas

.6 Penjadwalan Pola *Job-shop*

Secara umum penjadwalan *job-shop* yang *feasible* dapat dikelompokkan enjadi 4 macam, yaitu dapat diterima (*admissible*), *semi-aktif*, aktif, tanpa waktu nda (*non-delay*).

Penjadwalan yang bersifat dapat diterima (*admissible*) merupakan enjadwalan dimana terdapat waktu menganggur yang berlebihan. Mesin banyak enunggu daripada memproses sebuah tugas. Hal ini dapat diatasai dengan erapatkan waktu operasi tiap tugas sehingga dapat memperkecil waktu enganggur dan meningkatkan efisiensi. Tetapi perlu diingat bahwa dalam erapatkan sebuah tugas harus memperhatikan waktu proses dari proses ebelumnya agar tidak terjadi proses yang tumpang tindih dan menyebabkan enjadwalan menjadi salah (*infeasible*).

Pada penjadwalan yang bersifat semi-aktif merupakan penjadwalan imana sudah tidak ada lagi waktu menganggur yang berlebihan Kemungkinan au kombinasi penjadwalan yang *feasible* masih terlalu banyak.

Pada penjadwalan aktif sudah tidak dapat lagi proses yang dapat irapatkan lagi. Pada penjadwalan ini waktu mulai sebuah proses sudah dibatasi eh waktu selesai proses sebelumnya, sehingga tidak mungkin dirapatkan lagi, pabila dirapatkan lagi akan menindih proses sebelumnya.

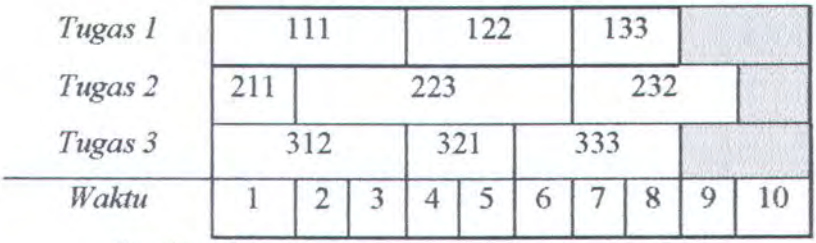
Jadwal yang bersifat tanpa waktu tunda (*non-delay*), sudah tidak ada lagi esin yang dibiarkan menganggur (*idle*) pada saat mengolah tugas. Namun tidak da jaminan akan didapatkan jadwal yang optimal. Karena jadwal yang optimal

ıtuan waktu dan diagram yang kedua mununjukkan aktivitas mesin tiap satuan aktu.

Sebuah contoh, apabila diberikan permasalahan dengan tiga tugas dan tiga esin untuk memprosesnya. Untuk mempermudah menganalisa diagram maka ap proses di beri nama dengan ***jom*** . Dimana ***j*** melambangkan kode atau nama gas (*job*), ***o*** melambangkan kode operasi, dan ***m*** melambangkan nama mesin. ada tabel 2-1 diberikan data permasalahan diatas. Jika permasalahan siatas igambarkan dengan diagram gantt adalah sebagai pada gambar 2-3 dan gambar -4 :

Tabel 2-1 Data permasalahan *job-shop* 3x3

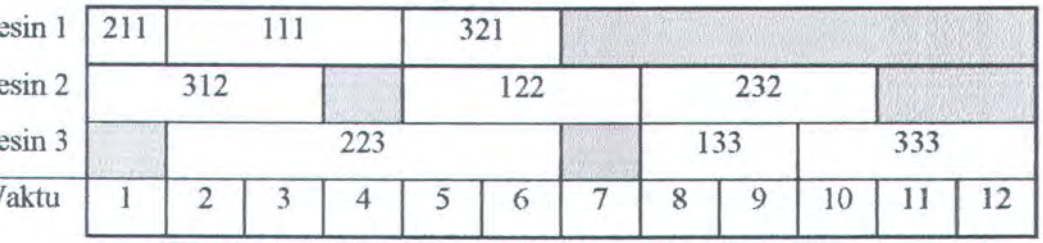
Tugas (job)	**Waktu Proses**			**Urutan Mesin**		
	Operasi			**Operasi**		
	1	**2**	**3**	**1**	**2**	**3**
1	3	3	2	1	2	3
2	1	5	3	1	3	2
3	3	2	3	2	1	3



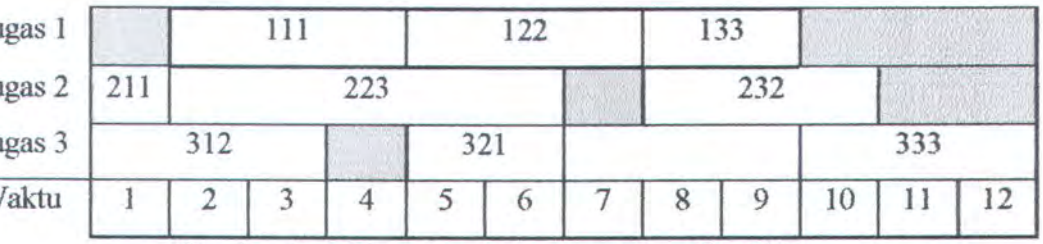
Gambar 2-3 Gantt permasalahan *job-shop* 3x3

Namun diagram 2-4 belum dapat dikatakan jadwal yang layak (***infeasible***) arena beberapa operassi dalam tugas yang dikerjakan oleh satu mesin secara multan, yaitu operasi (***Oij***) pada suatu tugas sudah mulai dikerjakan sedangkan perasi sebelumnya (O_{i-1j}) belum selesai diproses.

Sehingga diperlukan penyusunan operasi yang lebih baik sehingga ihasilkan penjadwalan yang layak (*feasible*). Salah satu contoh penjadwalan ang dapat dianggap layak adalah seperti pada gambar 2-5 :



Gambar 2-5 Diagram gantt aktivitas mesin (*feasible*)



Gambar 2-6 Diagram gantt proses tugas (*feasible*)

Penjadwalan gambar 2-5 dikatakan layak (*feasible*) karena waktu

an tidak saling bergantung atau berhubungan. Dan diasumsikan bahwa mesin ekerja selama 24 jam dengan tidak ada masalah kerusakan mesin Dalam tugas khir ini waktu *setup* dan *downtime* tidak diperhitungkan.

Dalam proses produksi tedapat sejumlah n tugas yaitu, $J_1 \ldots J_n$. Tiap tugas empunyai beberapa operasi dengan jumlah maksimal sebesar m. Tiap proses iolah dalam mesin tertentu maksimal satu kali dan tidak berulang. Jumlah perasi dari tiap tugas dinotasikan dengan m_i. Karena tiap tugas tidak melalui roses dari seluruh mesin maka $m_i \leq m$. Urutan operasi yang harus dilalui tiap igas dinotasikan sebagai $\mu_i(k)$ dimana $k = (1 < \ldots < m_i)$. Operasi ke k dari tugas dinotasikan dengan O_{ik}. Notasi waktu proses yang diperlukan oleh operasi ke k ari J_i adalah P_{ik} .

Sebuah jadwal dapat berupa pencatatan waktu mulai t_{ik} proses O_{ik} kedalam ebuah mesin atau sumber daya, dan disusun sesuai dengan urutan proses tugas J_i Waktu penyelesaian sebuah operasi merupakan waktu mulai ditambah dengan aktu proses ($t_{ik} + P_{ik}$) . Waktu mulai pemrosesan sebuah tugas merupakan nilai aksimal perbandingan antara waktu penyelesaian operasi sebelumnya O_{ik-1} engan waktu menganggur mesin yang bersangkutan.

$$t_{ik} = max\,(t_{ik-1} + P_{ik-1},\ t_{hl} + P_{hl}) \qquad (2\text{-}4)$$

6.3 Model Penjadwalan Pola *Job-shop* Dinamis (Bierwirth,1999)

Pada penjadwalan *job-shop* yang bersifat dinamik kedatangan job ervariasi, tidak bersifat reguler, sehingga penjadwalan dapat berubah sewaktu – aktu. Perubahan ini tidak bersifat reguler tetapi acak walaupun kemungkinan edatangan job sudah diketahui sebelumnya.

Pendekatan penjadwalan ulang *job-shop* dapat dibagi menjadi 3 macam, aitu :

Menjadwalkan kembali dari awal

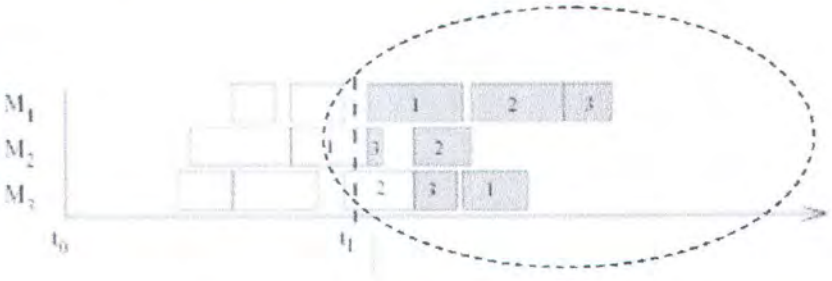
Melakukan perbaikan saat terjadi perubahan. Dilakukan secara berulang-ulang dengan memodifikasi jadwal lengkap sehingga didapat jadwal baru yang memuaskan

Menghilangkan beberapa operasi lama yang telah dikerjakan, kemudian menyusun jadwal baru terhadap operasi yang belum terlaksana.

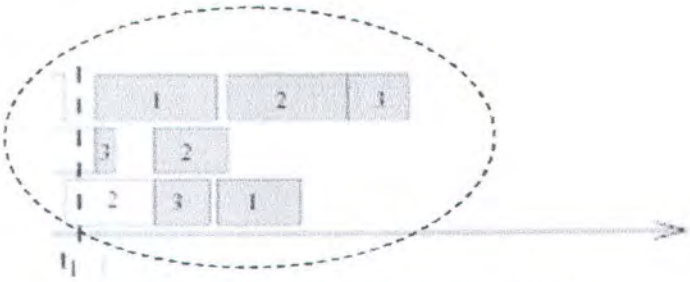
Dalam tugas akhir ini akan digunakan pendekatan ketiga. Operasi yang elah berjalan dipertahankan tidak mengalami perubahan. Tetapi perubahan ilakukan kepada operasi yang belum berjalan.

Perubahan jadwal tersebut adalah sebagai berikut. Apabila terdapat sebuah ermasalahan P_0 yang datang pada saat t_0. Dan pada saat t_0 sistem memiliki n gas. Permasalahan P_0 dapat mulai dikerjakan pada saat $t_1 > t_0$. Apabilah sebuah gas baru datang pada saat t_1, dimana $t_1 > t_0$, maka kita buat sebuah permasalahan aru P_1 yang kita dapatkan dari cuplikan jadwal yang ada dengan operasi yang

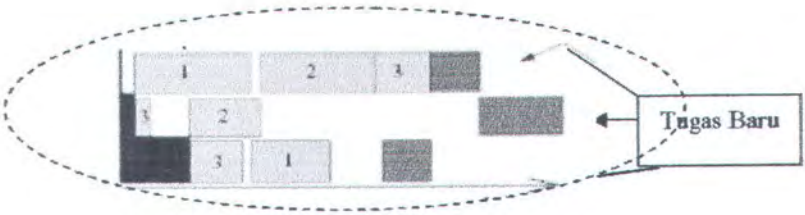
rutan proses penjadwalan di atas dapat digambarkan sebagai berikut,



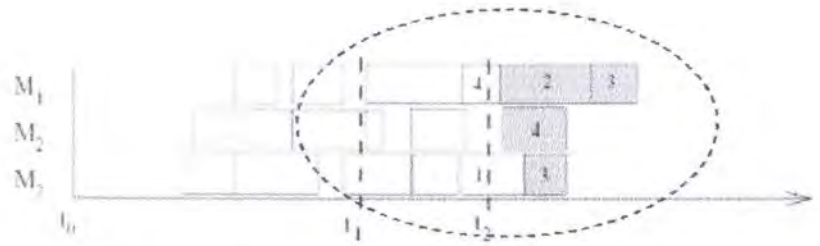
Gambar 2-7 Penjadwalan sebelum modifikasi



Gambar 2-8 Cuplikan jadwal setelah t_1



Gambar 2-9 Penambahan tugas baru



Gambar 2-11 jadwal baru setelah modifikasi

Gambar 2.8 menunjukkan cuplikan jadwal yang didapat dari panjadwalan ng telah berjalan. Cuplikan ini mengambil hanya operasi yang diproses setelah dan menghilangkan atau tidak mengikut sertakan operasi yang dimulai sebelum Setelah kita dapatkan permasalah baru P1 (Gambar 2.9), maka disisipkan tugas ru yang akan dijadwalkan ulang. Kemudian dilakukan penjadwalan dengan gas yang baru, seperti proses sebelumnya. Gabungkan jadwal baru dengan dawal yang lama (Gambar 2.11).

Waktu mulai proses pada pola *Job-shop* dinamis berbeda dengan waktu ulai proses pola *Job-shop* statis. Dimana waktu mulai proses sebuah tugas pada la *Job-shop* dinamis merupakan nilai maksimum dari perbandingan waktu datangan tugas r_i dengan waktu menganggur mesin yang bersangkutan. Waktu ulai proses sebuah tugas diformulasikan seperti pada formula (2-5). Notasi riabel yang digunakan terdapat pada tabel 2-2.

$$t_{ik} = max\,(r_i, t_{hl} + P_{hl}) \qquad (2\text{-}5)$$

Untuk sistem penjadwalan yang bersifat dinamis, sistem diharapkan

Tabel 2-2 Notasi variabel

Notasi	Penjelasan
n	Jumlah tugas
m	Jumlah mesin
m_i	Jumlah operasi tugas ke i , $1 < i < n$, $m_i \leq m$
$\mu_i(k)$	Urutan mesin yang harus dilalui tugas i , $k = (1 < ... < m_i)$
O_{ik}	Operasi ke – k tugas ke i
O_{hl}	Operasi ke -l tugas ke h
C_i	Waktu penyelesaian tugas
P_{ik}	Waktu proses operasi ke k tugas ke i
t_{ik}	Waktu mulai (*starting time*) proses ke – k tugas ke -i
r_i	Waktu kedatangan tugas ke i

7 Penjadwalan Pola Proyek

Seperti telah dijelaskan di atas, bahwa sebuah proyek merupakan suatu egiatan yang bersifat spesifik dan mempunyai jangka waktu pendek. Bersifat pesifik karena pada umumnya proyek dilaksanakan apabila diperlukan perubahan tur tertentu pada produk atau jasa yang dihasilkan. Proyek dalam perusahaan apat berupa pengembangan produk baru, pembangunan fasilitas produksi, engembangan strategi marketing baru. Sebuah proyek yang besar memerlukan erhatian dan manajemen khusus, sehingga setiap kegiatan dan sumber daya dapat ioptimalkan.

. *Aktifitas* adalah pekerjaan yang berhubungan atau menyangkut waktu dan penggunaan sumber daya, dimana start dan finis dari pekerjaan tersebut terencana dan jelas.

. *Even* merupakan sebuah kejadian yang menjadi tanda mulai atau berakhirnya suatu pekerjaan. Event tidak menyangkut waktu dan penggunaan sumber daya, tetapi sebuah titik atau kejadian suatu pekerjaan dapat ditangani. Sbuah even dikatakan *milestones* apabila even tersebut melambangkan penyelesaian suatu fase penting dalam proyek.

. *Perencanaan*. Elemen ini merupakan kegiatan untuk menentukan urutan aktifitas yang akan dilakukan. Untuk melakukan perencanaan ada beberapa hal yang perlu diperhatikan yaitu :

- Tentukan aktifitas yang harus diselesaikan terlebih dahulu sebelum aktifitas lain dimulai.
- Tentukan aktifitas yang harus segera dilakukan setelah aktifitas lain selesai
- Tentukan aktifitas mana yang dapat dilakukan bersama-sama dengan aktifitas lainnya.

.8 Fungsi Pengukuran Optimasi Penjadwalan

Dalam sistem industri, pada umumnya, pelanggan menginginkan etepatan dan kecepatan pengiriman pesanan. Sehingga industri akan engembangkan sebuah penjadwalan produksi yang meminimalisasi waktu eterlambatan, yang mungkin akan menyebabkan penurunan efisiensi

- Prosentase keterlambatan tugas (*percentage of late job*)

Kriteria yang bertujuan unutk menigkatkan utilisasi sumberdaya

- Utilisasi SDM (labor utilization)
- Utilisasi Mesin (machine utilization)
- Biaya inventory (In-process inventory cost)

Banyak literatur yang mengusulkan mengenai pengukuran optimasi dalam enjadwalan. Berikut ini beberapa fungsi pengukuran yang sering digunakan :

8.1 Waktu penyelesaian rata-rata (average flow time)

Permasalahan ini mendasarkan pada kriteria optimasi penjadwalan akan icapai dengan menyusun urutan pemrosesan tugas sedemikian rupa hingga waktu enyelesaian rata-rata untuk setiap tugas dapat diminimalisasi. Waktu enyelesaian rata-rata dapat dirumuskan sebagai berikut :

$$MFT = \frac{\sum_{i=1}^{n} C_i}{n} \qquad (2\text{-}6)$$

MFT = waktu penyelesaian rata-rata

ngan jelas bahwa permasalahan tersebut dapat diselesaikan dengan menyusun utan penyelesaian tugas. Jika urutannya adalah sebagai berikut :

$$J_2 - J_4 - J_5 - J_3 - J_1$$

Waktu penyelesaian dapat dilihat pada tabel berikut :

Tabel 2-3 Waktu penyelesaian tugas MFT

Tugas	Wi	ti	Ci
2	**0**	**3**	**3**
4	**3**	**6**	**9**
5	**9**	**9**	**18**
3	**18**	**5**	**23**
1	**23**	**4**	**27**
Jumlah	53	27	80

MFT = 80 / 5 = 16

Namun jika urutannya adalah, $J_3 - J_4 - J_1 - J_5 - J_2$ Maka hasilnya adalah:

Tabel 2-4 Waktu penyelesaian tugas MFT(2)

Tugas	Wi	ti	Ci
3	**0**	**5**	**5**

Jadi hasilnya adalah sama. Hal ini disebabkan belum ditentukan aturan ng spesifik untuk menyelesaikan permasalahan di atas, sehingga urutan kerjaan dilakukan secara random atau mencoba-coba, sehingga dimungkinkan endapatkan hasil yang tidak optimal. Aturan yang lebih spesifik tersebut isalnya penyelesaian tugas dimulai dari tugas dengan waktu pengerjaan rpendek (*shortest processing time*).

8.2 Waktu Pengerjaan Terpendek (*shortest processing time*)

Jika diberikan permasalahan seperti di atas, maka penyelesaian njadwalan diberikan urutan tugas sebagai berikut :

$$J_2 - J_1 - J_3 - J_4 - J_5$$

Sehingga waktu penyelesaian didapat sebagai berikut :

Tabel 2-5 Waktu penyelesaian tugas SPT

Tugas	Wi	ti	Ci
2	0	3	3
1	3	4	7
3	7	5	12
4	12	6	18
5	18	9	27
Jumlah	**40**	**27**	**67**

katakan jadwal yang dihasilkan adalah optimal. Waktu penyelesaian seluruh gas ini disebut *makespan.*

9 Eksplorasi Dan Eksploitasi

Dalam ilmu komputer, pencarian solusi terbaik suatu masalah dalam kelompok alternatif solusi sering disebut pencarian dalam daerah solusi (*search ace*). Sedangkan daerah solusi (*search space*) berbicara mengenai kumpulan ndidat solusi permasalahan.

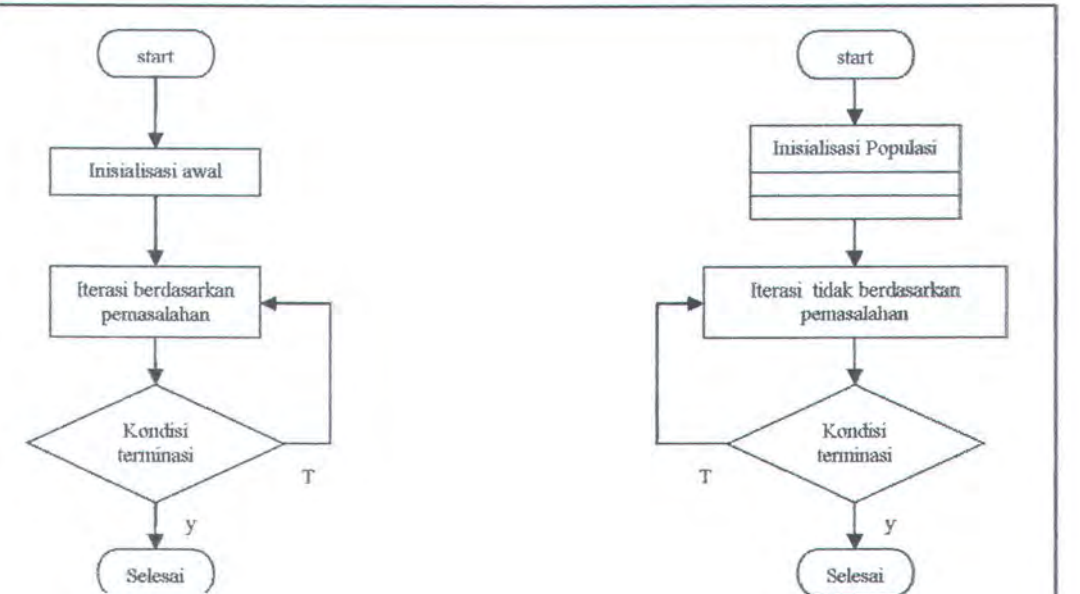
Teknik penyelesaian masalah yang sering digunakan adalah pencarian *earching*). Terdapat beragai macam teknik pencarian yang sering digunakan, tapi secara umum dapat dibedakan, menurut karakteristiknya, menjadi dua tegori besar. Cara pertama disebut dengan *blind search,* yaitu strategi atau cara ncarian tanpa memerlukan informasi mengenai domain permasalahan yang an diselesaikan. Sedangkan cara kedua disebut dengan *heuristik search*, yaitu rategi pencarian yang memerlukan informasi untuk mengarahkan jalannya oses pencarian, sehingga lebih terarah dan optimal.

Terdapat dua strategi penting dalam pemilihan strategi pencarian, ksploitasi solusi terbaik dan eksplorasi daerah pencarian. Teknik yang sering gunakan dalam proses pencarian antara lain, *Hill-climbing*, pencarian acak *andom search*), dan algoritma genetika.

Algoritma genetika, merupakan gabungan kedua teknik pencarian di atas. lgoritma genetika menyeimbangkan eksplorasi daerah pencarian dan eksploitasi olusi terbaik.

10 Pencarian Berbasis Populasi

Pada umumnya algoritma pemecahan masalah optimasi merupakan urutan ngkah menuju optimal. Berawal dari satu nilai yang kemudian dieksploitasi enurut aturan tertentu. Sedangkan pada algoritma genetika berawal dari eberapa nilai yang disebut populasi awal, kemudian melalui iterasi tertentu ergerak dari populasi yang satu kepada populasi keturunan berikutnya. erbedaan antara algoritma konvensional dengan algoritma genetika dapat dilihat ada *Gambar 2.12*.



11 Dasar Algoritma Genetika

Banyak permasalahan dalam dunia perindustrian yang sangat komplek ang tidak dapat diselesaikan dengan cara biasa. Algoritma genetika merupakan alah satu cara yang populer untuk menyelesaikan permasalah optimasi. lgoritma genetika saat ini sudah diaplikasikan dalam penyelesaian permasalahan ptimasi. Banyak permasalahan peridustrian yang dapat diselesaikan dengan enggunakan algoritma genetik, antara lain, kelayakan design, pemilihan jalur ansportasi, *facility layout*, dan penjadwalan.

Algoritma genetika merupakan sebuah pendekatan algoritmik yang ermula dari teori genetika pada bidang biologi. Seluruh mahluk hidup tersusun as ribuan sel , dan tiap sel tersusun atas kromosom. Kromosom ini merupakan susunan atau rangkaian DNA yang menjadi "*blueprint*" dari tiap mahluk hidup. ecara konseptual kromosom tersusun atas gen.

Dalam algoritma genetik kromosom adalah kumpulan kandidat solusi emasalahan, sering kali dikodekan dalam bentuk bit string. Untuk mendapatkan romosom atau alternatif solusi algoritma genetik membangkitkan solusi tersebut ecara acak. Untuk menentukan kromosom atau individu mana yang mampu eradaptasi atau dilakukan seleksi, algoritma genetik harus memiliki fungsi untuk engevaluasi beberapa alternatif tersebut. Apabila alternatif tersebut mampu emenuhi target dari fungsi fitnes tersebut maka alternatif tersebut lolos ke tahap valuasi berikutnya. Begitu seterusnya sampai didapatkan solusi yang paling ptimal.

Menurut teorema yang lazim digunakan dalam ilmu genetika, bahwa eluruh organisme tersusun atas komponen yang menyebabkan perbedaan khas ntara organisme satu dengan yang lainnya.

Komponen yang menyebabkan perbedaan tersebut dinamakan kromosom. atu atau lebih kromosom menyusun sebuah organisme yang khas dan spesifik. romosom yang lengkap beserta unsurnya disebut juga dengan genotip. edangkan penampilan dari genotip secara fisik disebut fenotip. Setiap kromosom irepresentasikan dalam bentuk simbol, biasanya dalam bentuk *bit string*. Setiap romosom tersusun atas bagian- bagian yang terstruktur yang disebut *gen*. Setiap en menempati lokasi yang dinamakan *locus* dan memiliki nilai (*value*) yang isebut *allele*. Setiap nilai gen inilah yang merupakan pengkodean variabel ermasalahan yang akan diselesaikan.

Kumpulan dari berbagai macam dan jenis kromosom tersebut membentuk uatu kumpulan atau populasi. Pada algoritma genetika, istilah populasi diberikan ntuk menamai himpunan solusi. Untuk pertama kali, himpunan solusi dihasilkan ecara acak (*random*).

13 Struktur Umum Algoritma Genetika

Struktur algoritma secara umum pertama kali dijelaskan oleh Goldberg. lgoritma genetik merupakan model pencarian *stochastic* (*stochastic search*) ang didasarkan pada aturan seleksi alam dan sifat genetika. Algoritma genetika

```
Prosedur: Algoritma Genetika
Begin
t <= 0
Initialize P(t);
Evaluate P(t);
  While (not terminate condition) do
     Recombine P(t) to yield C(t);
     Evaluate C(t);
     Select P(t+1) from P(t) and C(t);
     t<= t + 1;
    end
end
```

Program 2-1 Prosedur umum algoritma genetika

Inisialisasi himpunan solusi pada algoritma diatas dibangkitkan secara cak (*random*). Himpunan solusi awal ini disebut juga dengan istilah populasi wal. Populasi ini kemudian akan mengalami proses evaluasi berdasarkan suatu ngsi tertentu yang disebut dengan fungsi *fitness*. Kromosom – kromosom yang rdapat pada populasi awal akan berevolusi menurut proses iterasi dan atuaran rtentu yang disebut generasi.

Pada proses iterasi, berlaku dua jenis operasi untuk menghasilkan generasi erikutnya. Kedua jenis operasi tersebut adalah :

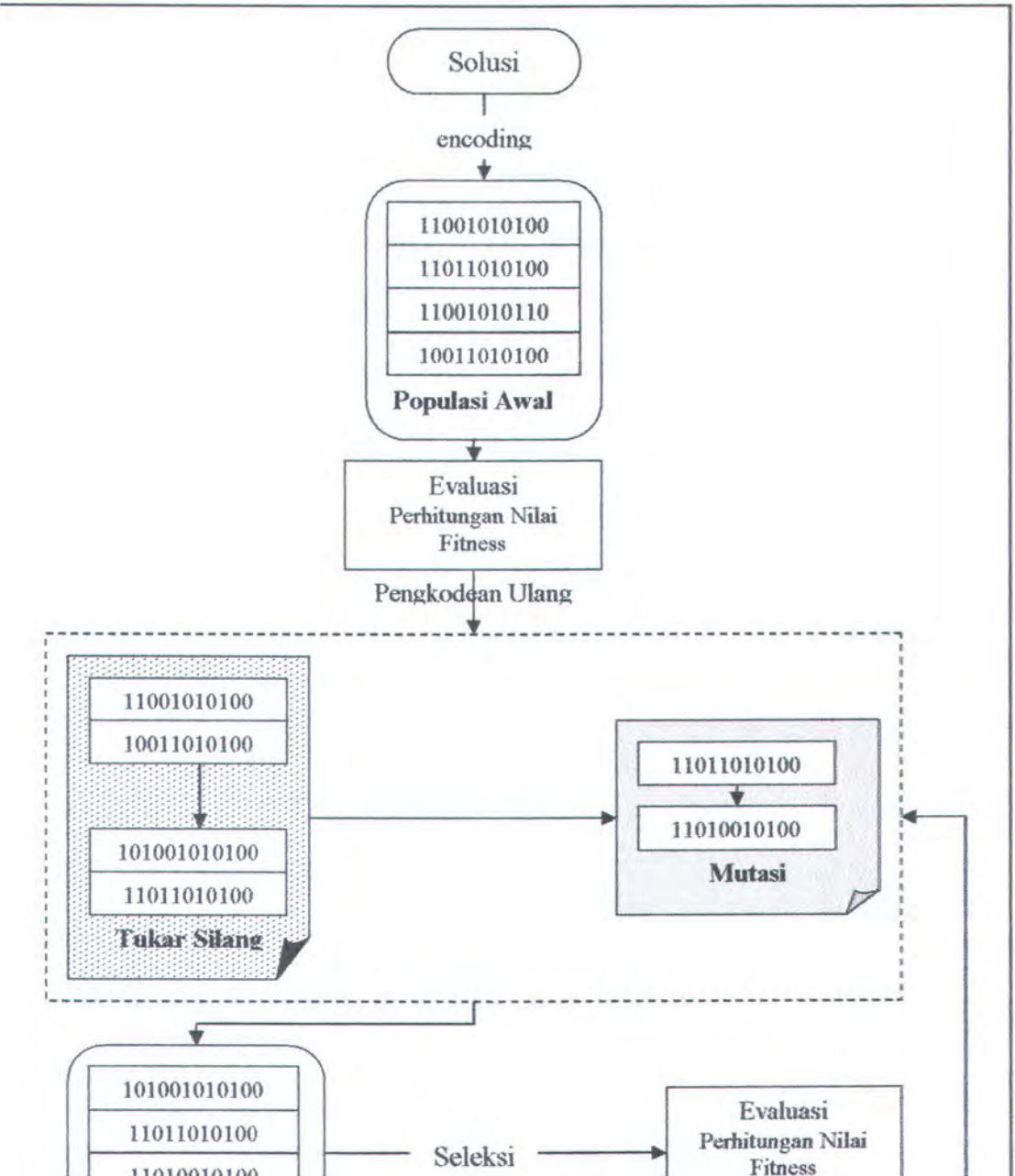
- **Operasi genetik** (*genetic operation*) , yang terdiri dari proses silang (*Crossover*) dan mutasi (*mutation*)
- **Operasi evolusi** (*evolution operation*), yaitu seleksi (*selection*).

Tukar silang merupakan proses unttuk mengubah dua kromosom menjadi tu kromosom baru. Proses ini melibatkan dua kromosom orang tua (*parent*) ntuk mendapatkan keturunan baru (*offspring*). Proses tukar silang dilakukan engan menukar gen dari kromosom orang tua pertama dengan gen dari romosom orang tua kedua.

Mutasi merupakan sebuah proses mendapatkan kromosom baru dari satu romosom orang tua. Proses ini hanya melibatkan satu kromosom orang tua saja. roses mutasi dilakukan dengan jalan menukar posisi dua gen atau mengubah lah satu gen menjadi gen lain dari kromosom orang tua.

Proses evaluasi dilakukan untuk menetapkan nilai kromosom-kromosom ng disebut nilai *fitness*. Seluruh kromosom yang ada dalam populasi harus empunyai nilai *fitness*. Fungsi untuk memberikan nilai *fitness* disebut juga engan fungsi *fitness*. Fungsi *fitness* dapat berbeda-beda sesuai dengan rmasalahan yang dihadapi. Nilai *fitness* inilah yang menjadi parameter optimasi romosom. Kromosom yang memiliki nilai *fitness* tinggi, lebih optimal dalam enyelesaikan permasalahan daripada kromosom yang memiliki nilai *fitness* yang bih rendah.

Proses seleksi adalah pemilihan keturunan dari populasi lama (*parent*) rdasarkan nilai *fitness*-nya. Kromosom yang memiliki nilai *fitness* lebih tinggi empunyai kemungkinan untuk dipilih menjadi keturunan berikutnya. Sedangkan romosom dengan nilai *fitness* kecil akan mengalami proses genetik kembali. romosom yang memiliki nilai *fitness* terbesar diharapkan menjadi solusi yang


Solusi
encoding
11001010100
11011010100
11001010110
10011010100
Populasi Awal
Evaluasi
Perhitungan Nilai
Fitness
Pengkodean Ulang
11001010100
10011010100
101001010100
11011010100
Tukar Silang
11011010100
11010010100
Mutasi
101001010100
11011010100
Seleksi
Evaluasi
Perhitungan Nilai
Fitness

.4 Algoritma Genetika Untuk Penjadwalan Dinamis

Cara kerja algoritma genetik pada pembahasan berikut ini merupakan ıtan proses sesuai dengan struktur umum algoritma genetik. Proses diawali ngan representasi masalah, iterasi dengan operator genetik dan operasi evolusi, ta fungsi evluasi untuk menilai apakah kromosom masih dapat dipertahankan generasi berikutnya. Cara kerja algoritma genetik dapat diilustrasikan seperti da *gambar 2.13*

14.1 Representasi Kromosom

Representasi atau mengkodekan masalah ke dalam bentuk kromosom erupakan dasar dari seluruh proses algoritma genetik. Representasi ini dapat rbeda-beda, sesuai dengan cara mengkodekan masalah dan permasalahan itu ndiri. Sehingga sebuah kromosom representasi merupakan sebuah solusi yang ungkin terjadi. Pada umunya representasi yang sering digunakan berupa *bit ring*. Tetapi tidak menutup kemungkinan untuk merepresentasikan masalah ke lam tipe data lain.

Seperti telah dijelaskan di atas, bahwa representasi kromosom merupakan nentu keberhasilan algoritma genetik. Representasi kromosom yang berbeda an menyebabkan perbedaan dalam proses *encoding*, *decoding* dan perhitungan ai *fittnes*.

Representai kromosom yang digunakan dalam tugas akhir ini berdasarkan da operasi dari tugas yang masuk atau disebut juga *Operation-based*

Sebagai ilustrasi penggambaran kromosom di atas adalah sebagai berikut. pabila terdapat perasalahan *job-shop* 3 tugas dan 3 mesin dengan daftar urutan esin dan lama proses tertera pada tabel 2-6. Diberikan contoh kromosom dengan n sebagai berikut [***3 2 2 1 1 2 3 1 3***], maka angka ***3*** melambangkan tugas ke ga (J_3) dan angka ***1*** melambangkan tugas ke satu (J_1). Karena setiap tugas emiliki operasi sebanyak 3 maka kode masing-masing tugas muncul sebanyak ga kali dalam kromosom. Gen pertama contoh kromosom di atas adalah ***3*** maka i melambangkan tugas ke tiga. Karena angka tiga tersebut muncul pada urutan rtama dari angka tiga yang lain maka ini berarti operasi pertama dari tugas ke ga. Dengan melihat pada daftar di atas maka operasi pertama dari tugas ke tiga proses pada mesin 2 (m_2). Begitu pula berlaku pada gen ke dua dari kromosom atas berarti tugas dua operasi pertama, karena muncul pada urutan pertama dari ngka ***2*** yang lain. Dilihat dari tabel di atas maka proses pertama dari tugas tiga proses di mesin 1 (m_1).

Apabila contoh kromosom di atas diterjemahkan ke dalam operasi dan esin adalah sebagai berikut,

romosom	[***3 2 2 1 1 2 3 1 3***]
perasi	$[O_{31}, O_{21}, O_{22}, O_{11}, O_{12}, O_{23}, O_{32}, O_{13}, O_{33}]$
lesin	$[m_2, m_1, m_3, m_1, m_2, m_2, m_1, m_3, m_3]$

Tabel 2-6 Contoh permasalahan 3 tugas dan 3 mesin

Tugas (job)	Waktu Proses	Urutan Mesin
	Operasi	Operasi

14.2 Encoding

Dalam algoritma genetika, *encoding* merupakan sebuah proses untuk engubah solusi kedalam kromosom untuk diproses lebih lanjut. Seperti telah jelaskan di atas, representasi kromosom akan disusun menurut operasi *peration-based*). Sehingga gen pembentuk kromosom berupa operasi.

Dalam sistem penjadwalan, masing masing operasi dari tiap tugas emiliki status yang menandakan keadaanya. Operasi berstatus 0 berarti operasi rsebut telah tiba tetapi belum dijadwalkan. Operasi berstatus 1 berarti bahwa erasi tersebut telah dijadwalkan tetapi sedang menunggu giliran untuk diproses lam mesin yang bersangkutan. Operasi dengan status 2 berarti bahwa operasi rsebut sedang dalam proses. Operasi berstatus 3 berarti operasi tersebut telah lesai diproses.

Prosedur *encoding* ini menambahkan proses dari seluruh tugas yang rstatus 1 dan 0 saja. Karena untuk operasi dengan status 2 dan 3 dianggap telah elalui atau sedang dalam proses, sehingga tidak perlu dilakukan penjadwalan ang. Operasi yang telah atau sedang diproses dianggap tidak dapat dihentikan, au bersifat tetap.

14.3 Pembangkitan Populasi

Dalam algoritma genetika pembangkitan populasi merupakan suatu proses ng penting dan start awal untuk mendapatkan solusi. Dari proses pembangkitan pulasi ini akan dihasilkan kromosom atau calon solusi yang kemudian akan

gas akhir ini populasi dibangkitkan dari pengacakan kromosom awal yang dapat dari proses *encoding*. Dari proses pengacakan kromosom awal ini akan dapatkan kromosom baru. Proses pengacakan kromosom awal ini dilakukan cara terus menerus sampai didapatkan jumlah kromosom yang diinginkan.

14.4 Fungsi Evaluasi

Fungsi evaluasi dalam algoritma genetika adalah sebuah fungsi yang emberikan penilaian kepada kromosom yang disebut nilai *fitness* (*fitnes value*). ilai *fitness* inilah yang kemudian menjadi nilai bobot suatu kromosom, apakah atu kromosom layak untuk dipertahankan dalam keturunan berikutnya. Oleh rena itulah fungsi *fitness* menjadi masalah atau penentu utama keberhasilan goritma genetik.

Fungsi evaluasi dilakukan dalam 3 tahap, yaitu :

- Tahap 1 : konversikan genotip menjadi fenotip.
- Tahap 2 : evaluasi fungsi tujuan
- Tahap 3 : konversikan nilai fungsi tujuan menjadi nilai *fitness*

Dalam sistem industri, pada umumnya, pelanggan menginginkan ketepatan an kecepatan pengiriman pesanan. Sehingga industri akan mengembangkan ebuah penjadwalan produksi yang meminimalisasi waktu keterlambatan, yang ungkin akan menyebabkan penurunan efisiensi sumberdaya.

enilai suatu penjadwalan *job-shop* yang bersifat dinamis dibandingkan dengan etode yang lain seperti *minimal makespan.* Karena dalam sistem penjadwalan ng dinamis sebuah tugas datang dengan waktu yang berbeda dan tidak menentu hingga waktu total penyelesaian proses memiliki rentang yang sangat panjang hingga metode *minimal makespan* akan gagal dalam menilai suatu penjadwalan namis.

Fungsi evaluasi *average flow time* yang digunakan dalam tugas akhir ini alah sebagai berikut,

$$MFT = \frac{\sum_{i=1}^{n} C_i}{n} \qquad (2\text{-}7)$$

FT = waktu penyelesaian rata-rata
= waktu penyelesaia tugas *i*
= jumlah tugas

Karena tujuan metode *average flow time* adalah meminimalisasi waktu oses, sedangkan dalam algoritma genetika adalah maksimasi maka fungsi aluasi diubah menjadi,

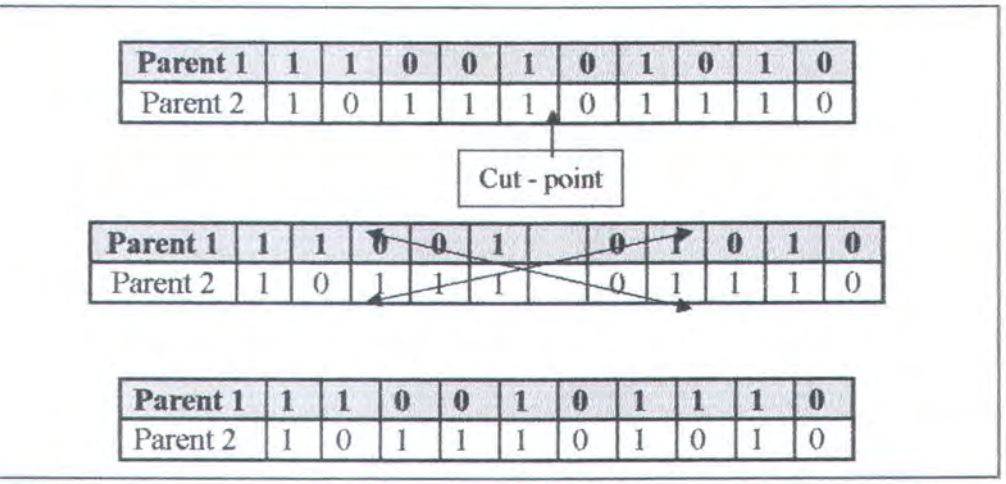
$$fitness = \frac{1}{MFT} \qquad (2\text{-}8)$$

14.5 Operasi Genetik

14.5.1 Tukar Silang (*crossover*)

Operasi tukar silang merupakan operasi yang sering digunakan dalam erasi genetik. Operasi ini bekerja dengan menggabungkan dua kromosom orang a (*parent*) menjadi kromosom baru (*offspring*). Operasi tukar silang mempunyai berapa metode yaitu, *one-point crossover, Uniform, Precedence Preservative rossover* (PPC), dan sebagainya.

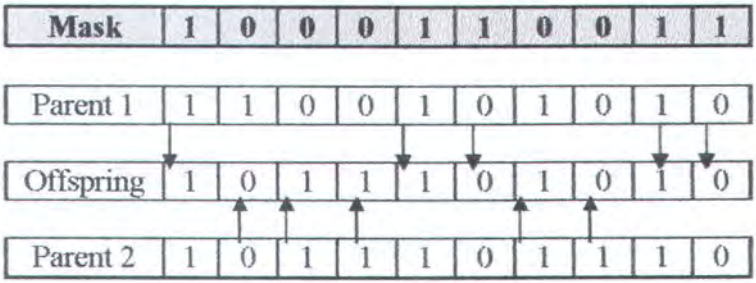
Metode tukar silang yang paling sederhana, one-point, memiliki aturan bagai berikut :



Gambar 2-14 One-point Crossover

Untuk two-point crossover dan muti-point crossover berlaku cara yang ma seperti di atas hanya berbeda dalam jumlah titik atau *pointi* tukar silang.

Selain metode yang telah disebutkan diatas, terdapat juga metode tukar



Gambar 2-15 Operasi Tukar Silang Uniform

Banyak proses atau metode yang telah diajukan oleh para ahli algoritma enetik, salah satunya adalah metode *Precedence Preservative Crossover* (PPC). Metode PPC inilah yang akan digunakan daalm tugas akhir ini. Pada metode PPC i dilakukan pembangkitan bilangan acak yang digunakan untuk memilih gen omosom orang tua pertama atau kedua. Gen yang terpilih akan menjadi gen ng menyusun kromosom baru. Metode PPC dapat digambarkan sebagai berikut,

Parent 1

A B C D E F

Parent 2

C A B F D E

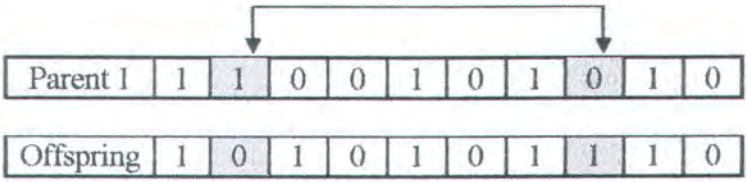
Langkah pertama pada metode PPC adalah membangkitkan bilangan acak ntuk menentukan gen *parent* manakah yang akan dipilih untuk diteruskan kepada eturunannya. Apabila bilangan acak yang dihasilkan adalah ***1*** maka ambil gen aling kiri dari *parent* 1, kemudian masukkan dalam gen tersebut pada kromosom aru. Dan sebaliknya, apabila bilangan acak yang dihasilkan adalah ***2*** maka ambil en paling kiri dari *parent* 2 untuk menjadi gen kromosom baru. Dalam metode i perlu dilakukan pengecekan, apakah gen *parent* yang terpilih telah ada pada romosom baru. Sehingga gen yang menyusun kromosom baru tetap valid dan dak terjadi perulangan gen. Hal ini dilakukan untuk mencegah terjadinya plikasi dan tertukarnya urutan operasi suatu job.

Pada proses tukar silang dikenal istilah tingkat tukar silang (*crossover te*), yaitu besarnya kemungkinan terjadinya tukar silang antara dua kromosom alam populasi. Dengan kata lain bahwa terjadi *crossover rate x pop_size* romosom yang melakukan tukar silang, dimana *pop_size* adalah jumlah romosom dalam populasi.

14.5.2 Mutasi

Mutasi merupakan operasi untuk membentuk keturunan (*offspring*) dengan lan mengubah susunan bit / gen dari orang tua (*parent*). Metode yang sering gunakan dalam operasi mutasi adalah mutasi inversi (*invertion mutation*) dan utasi pertukaran (*mutation exchange*).

Pada mutasi inversi terjadi perubahan gen pada kromosom, misal gen "1"



Gambar 2-18 Operasi Mutasi Pertukaran

Dalam operasi mutasi dikenal istilah tingkat mutasi (*mutation rate*) yaitu esarnya kemungkinan tiap gen dalam kromosom untuk bermutasi. Sehingga rjadinya mutasi adalah *mutation rate x pop_size* dalam tiap generasi atau opulasi. Dimana *pop_size* adalah jumlah kromosom dalam populasi. Dengan ata lain bahwa banyak kromosom yang mengalami mutasi adalah *mutation rate x op_size*. Misal *mutation rate=0.5* dan *pop_size=10* maka jumlah kromosom yan engalami mutasi sebanyak 0.5 x 10 = 5 kromosom. Metode mutasi yang gunakan dalam tugas akhir ini adalah mutasi pertukaran

14.5.3 Seleksi

Dasar pemikiran seleksi dalam algoritma genetika adalah teori Darwin engenai seleksi alam. Sampai saat ini banyak metode seleksi yang telah diajukan eh para ahli. Ada permasalahan penting dalam operasi seleksi, yaitu :

- Daerah sample (sampling space)

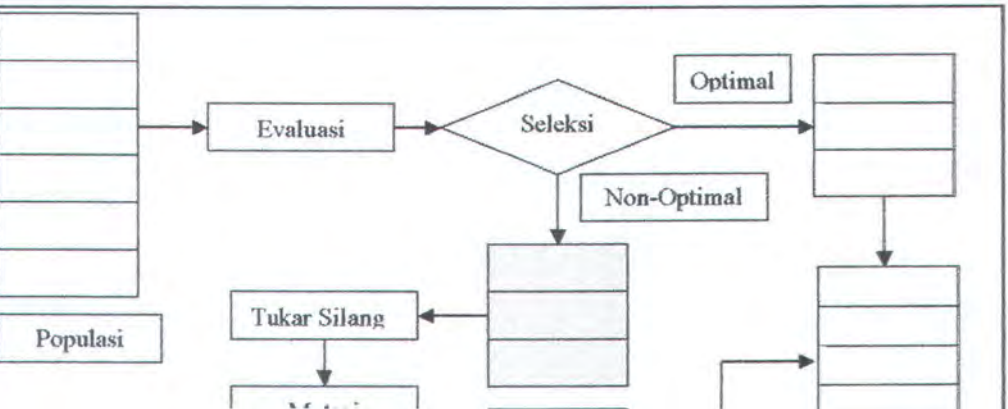
n keturunannya mempunyai kemungkinan untuk tetap bertahan pada keturunan rikutnya. Dalam pengerjaan tugas akhir ini metode seleksi yang digunakan alah metode *regular*.

Sebelum diseleksi, kromosom dalam populasi diurutkan menurut nilai *ness*, dari nilai tertinggi sampai nilai terendah. Proses pemilihan kromosom ini lakukan menurut komposisi atau laju seleksi (*selection rate*). Komposisi ini rbicara mengenai berapa jumlah kromosom terbesar dalam populasi yang lolos periode berikutnya. Jumlah kromosom yang lolos ke periode berikutnya dapatkan sebagai berikut,

$$n_{\text{optimal}} = \textit{selection rate} \times \textit{jumlah populasi} \qquad (2\text{-}9)$$

Sedangkan jumlah kromosom yang tidak lolos adalah,

$$n_{\text{non-optimal}} = (1\text{-}\textit{selection rate}) \times \textit{jumlah populasi} \qquad (2\text{-}10)$$



14.6 Decoding

Decoding adalah sebuah proses menerjemahkan kromosom ke dalam buah jadwal yang mudah dibaca. Metode *decoding* yang digunakan adalah *peration-based decoding*. Proses *decoding* dilakukan berdasarkan pada urutan perasi pada kromosom. Pada proses *decoding* ini, setiap operasi akan masukkan kedalam antrian mesin yang bersangkutan. Pengalokasian operasi dalam mesin dilakukan terlabih dahulu terhadap operasi berstatus 3 dan 2, mudian operasi berstatus 0 dan 1. Proses ini dilakukan agar operasi berstatus 2 n 3 tidak dijadwalkan ulang dan tetap berada dalam mesin. Untuk menentukan aktu mulai proses sebuah operasi terdapat beberapa kriteria, yaitu :

Apabila waktu idle mesin lebih besar daripada waktu selesai operasi sebelumnya(***Oi-1,j***)maka waktu mulai proses adalah idle, dan sebaliknya.

Apabila waktu kedatangan tugas lebih besar daripada waktu idle mesin maka waktu mulai proses adalah waktu kedatangan demikian sebaliknya.

Apabila waktu selesai operasi sebelumnya (Oi-1,j) lebih besar daripada waktu kedatangan tugas maka waktu mulai proses adalah waktu selesai operasi sebelumnya (Oi-1,j).

BAB III

RANCANGAN PERANGKAT LUNAK

BAB 3
PERANCANGAN PERANGKAT LUNAK

Bab ini menjelaskan mengenai perancangan aplikasi penjadwalan yng an dibuat. Rancangan aplikasi dilakukan dengan pemodelan use-case. modelan use-case merupakan sebuah pemodelan khusus yang menitik beratkan da fungsi sistem. Pemodelan use-case pada umumnya dilakukan untuk mencatat a yang akan dilakukan oleh pengguna. Setiap use-case direalisasi dengan enggunakan 3 jenis diagram, yaitu diagram sekuensi, diagram partisipasi kelas n diagram kelas. Perancangan proses aplikasi ini dilakukan dengan enggunakan notasi UML (Unified Modelling Language) yang dimiliki oleh rangkat lunak Rational Rose Enterprise Edition 2000.

1 Analisa *Use-Case*

Seperti telah dijelaskan di atas bahwa pemodelan *use-case* merupakan modelan mengenai fungsi sistem. Oleh karena itu dalam melakukan analisa *use-se* perlu dilakukan pencatatan atau identifikasi fungsi sistem. Dengan kata lain alisa *use-case* seperti membuat daftar fungsi yang akan dilakukan sistem.

Tiap-tiap *use case* memiliki sebuah deskripsi yang menjelaskan fungsi ng akan dibangun pada sistem yang baru. *Use case* pada dasarnya berhubungan ngan aktor. Aktor adalah pengguna atau sistem lain yang berinteraksi dengan stem. Aktor yang berinteraksi dengan sistem dijelaskan pada tabel 3-1.

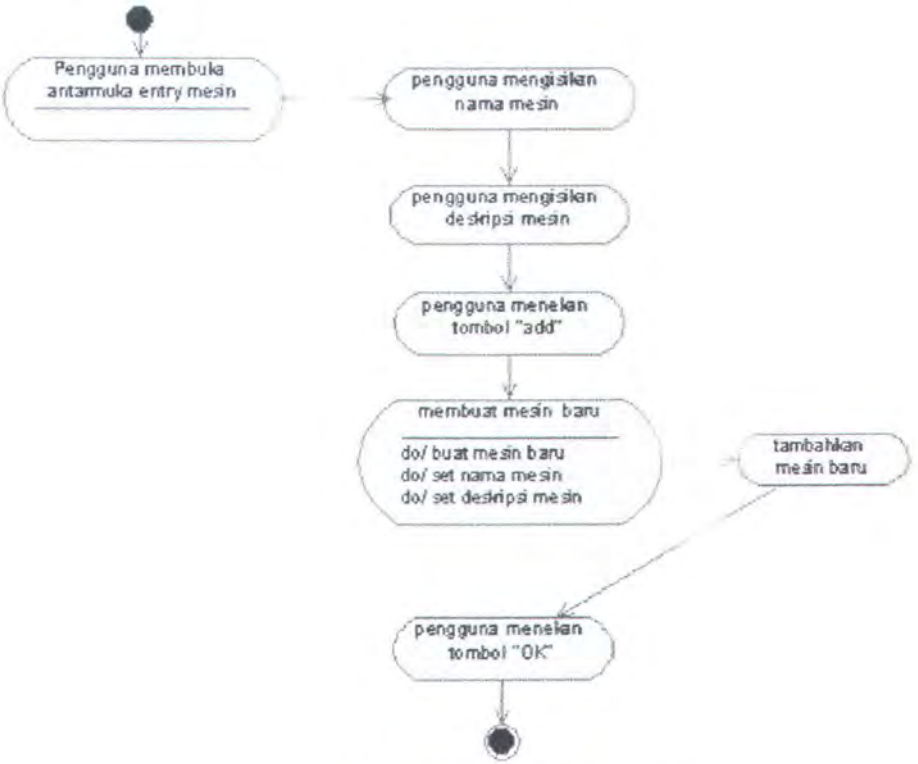
1.1 *Use-Case* Entry Mesin

Sebelum melakukan penjadwalan user harus melakukan entry mesin yang an digunakan untuk memproses tugas yang masuk. Mesin inilah yang nantinya an menjadi sumberdaya pengalokasian tugas agar dapat diproses. Detil engenai use-case ini akan dijelaskan pada tabel 3-2.

Tabel 3-2 spesifikasi *use-case* entry mesin

Spesifikasi	Penjelasan
Aktor	▪ Pengguna
Tujuan	▪ Memasukkan mesin untuk produksi
Ringkasan	▪ Pengguna melakukan entry mesin yang nantinya akan digunakan untuk mengalokasikan tugas yang masuk
Masukan	▪ Nama Mesin ▪ Deskripsi mesin
Luaran	▪ Mesin baru untuk proses produksi
Kondisi Awal	▪ Pengguna membuka antarmuka entry mesin
Kondisi Akhir	▪ Terdapat tambahan mesin baru pada aplikasi
Aliran Aksi normal	▪ Pengguna membuka antarmuka entry mesin melalui FrameUtama ▪ Pengguna mengisikan nama mesin ▪ Pengguna memasukkan deskripsi mengenai mesin yang akan ditambahakan ▪ Pengguna memekan tombol "add"

jadi di dalamnya. Proses ini diawali dengan membuka antarmuka entry mesin eh pengguna. Dilanjutkan dengan memesukkan nama dan deskripsi mesin. oses ini diakhiri dengan menekan tombol "OK" atau "Apply". Setelah mesin ru telah terbentuk maka mesin tersebut akan disimpan dalam kumpulan mesin.

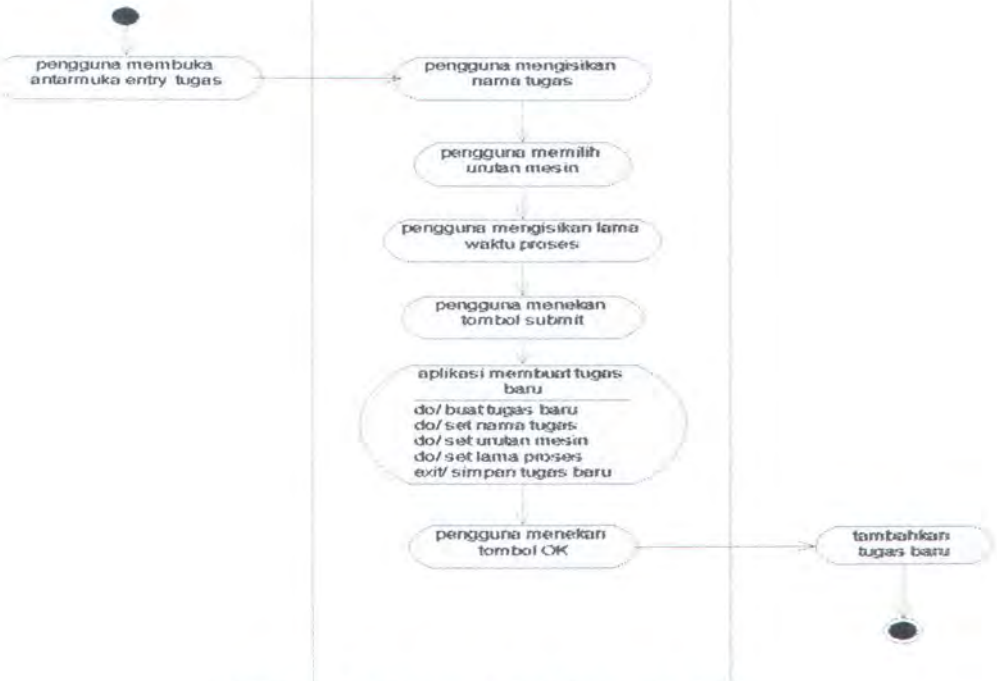


Gambar 3-1 diagram aktivitas entry mesin

Tabel 3-3 spesifikasi *use-case* entry tugas awal

Spesifikasi	Penjelasan
Aktor	▪ Pengguna
Tujuan	▪ Memasukkan tugas untuk dijadwalkan
Ringkasan	▪ Pengguna melakukan entry tugas yang nantinya akan dijadwalkan dan dialokasikan ke dalam mesin
Masukan	▪ Nama tugas ▪ Urutan mesin yang digunakan untuk proses ▪ Lama waktu proses tiap operasi
Luaran	▪ Tugas baru untuk dijadwalkan
Kondisi Awal	▪ Pengguna membuka antarmuka entry tugas
Kondisi Akhir	▪ Terdapat tambahan tugas baru pada aplikasi
Aliran Aksi normal	▪ Pengguna membuka antarmuka tugas melalui antarmuka frame utama ▪ Entry nama tugas baru yang akan dibuat ▪ Pilih urutan mesin yang akan digunakan untuk proses ▪ Tentukan lama waktu proses untuk tiap operasi. ▪ Submit tugas baru ▪ Aplikasi membuat sebuah tugas baru ▪ Klik tombol "OK" untuk menempatkan tugas baru dalam kumpulan tugas lama.

oses entry tugas baru dapat digambarkan dengan diagram aktifitas seperti pada mbar 3-2



Gambar 3-2 diagram aktifitas entry tugas

1.3 *Use-Case* Pengaturan Parameter Ganetik

Pengaturan ini dilakukan terhadap parameter *crossover rate, mutation rate, lection rate*, jumlah populasi, jumlah keturunan, laju acak pembangkitan

Tabel 3-4 spesifikasi *use-case* pengaturan parameter genetik

Spesifikasi	Penjelasan
Aktor	▪ Pengguna
Tujuan	▪ Melakukan perubahan terhadap nilai parameter algoritma genetic.
Ringkasan	▪ Pengaturan ini bertujuan untuk meningkatkan kinerja algoritma genetik agar didapatkan solusi yang optimal.
Masukan	▪ Nilai parameter genetik
Luaran	▪ Nilai parameter genetik
Kondisi Awal	▪ Nilai parameter genetik awal adalah nilai *default* ▪ Nilai default ditetapkan pada saat pembuatan aplikasi
Kondisi Akhir	▪ Perubahan nilai parameter genetik
Aliran Aksi normal	▪ Pengguna membuka antarmuka pengaturan parameter GA melalui antarmuka frame utama ▪ Masukkan nilai parameter yang dikehendaki. ▪ Klik tombol OK atau tombol Submit untuk mengubah parameter algoritma genetic sistem

Proses pengaturan parameter genetik dapat digambarkan dengan diagram tifitas seperti pada gambar 3-3.

Proses pengaturan parameter genetik dimulai dengan membuka antarmuka rameter genetik. Setelah antarmuka parameter GA terbuka, pengguna dapat lakukan perubahan terhadap parameter yang diinginkan. Untuk merealisasikan rubahan yang dibuat pengguna harus menekan tombol "OK" atau "Submit".

1.4 *Use-Case* Waktu Sistem

Kembali merujuk pada tujuan tugas akhir ini, yaitu melakukan simulasi njadwalan *job-shop* dinamis, maka sistem ini mensimulasikan kedatangan tugas ngan waktu kedatangan yang berbeda beserta hasil penjadwalannya. Waktu ng digunakan dalam sistem ini berbeda dengan waktu yang dimiliki *desktop* mputer. Waktu dalam sistem ini disimulasikan sehingga dapat dengan mudah tuk mengatur kecepatan, menghentikan, dan menjalankan kembali.

Waktu sistem ini merupakan sebuah kelas, yang dalam implementasinya, rupa proses *background* (*background process*). Object ini nantinya akan njadi patokan waktu yang digunakan dalam penjadwalan bukan waktu yang ng dimiliki *desktop* komputer.

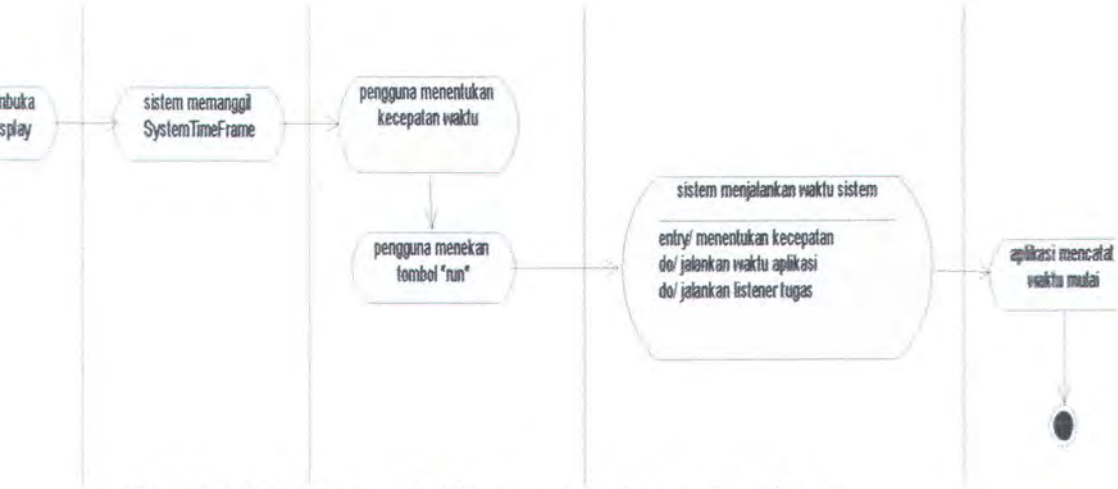
Tabel 3-5 spesifikasi *use-case* menjalankan waktu sistem

esifikasi	Penjelasan
tor	▪ Pengguna
juan	▪ Menjalankan waktu aplikasi
ngkasan	▪ Proses ini dilakukan untuk menjalankan waktu sistem agar

Tabel 3-5 spesifikasi *use-case* menjalankan waktu sistem (lanjutan)

esifikasi	Penjelasan
iran Aksi normal	▪ Pengguna membuka antarmuka display melalui antarmuka frame utama ▪ Tentukan kecepatan *refresh* waktu sistem penjadwalan ▪ Klik tombol run untuk menjalankan sistem dengan kecepatan yang diinginkan

Proses menjalankan waktu sistem dapat digambarkan melalui diagram tifitas. Diagram aktifitas yang menjelaskan aliran proses menjalankan waktu tem digambarkan pada gambar 3-4. Untuk menjalankan waktu sistem pengguna rus membuka antarmuka display. Kemudian pengguna dapat menentukan cepatan waktu, menjalankan dan menghentikan waktu.

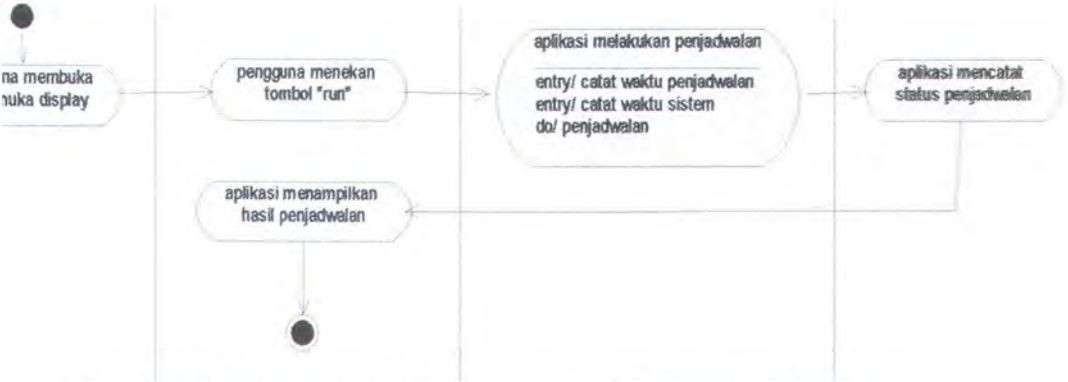


Gambar 3-4 diagram aktifitas menjalankan waktu sistem

Tabel 3-6 spesifikasi *use-case* menjalankan penjadwalan

esifikasi	Penjelasan
tor	▪ Pengguna
juan	▪ Menjalankan penjadwalan terhadap tugas yang masuk
ngkasan	▪ Proses ini dilakukan untuk mengalokasika operasi dari setiap tugas kedalam mesin yang bersangkutan
asukan	▪ Waktu penjadwalan baru ▪ Tugas baru
aran	Hasil alokasi operasi kedalam mesin
ndisi Awal	▪ Antrian mesin kosong ▪ Waktu mulai operasi tiap tugas adalah 0
ndisi Akhir	▪ Perubahan pada waktu mulai operasi ▪ Perubahan antrian mesin
iran Aksi normal	▪ Pengguna membuka antarmuka display melalui antarmuka frame utama ▪ Jalankan waktu sistem untuk mengaktifkan tombol *run* ▪ Apbila waktu telah berjalan, hentikan sejenak waktu sistem ▪ Klik tombol *run* untuk menjalankan penjadwalan dan mengalokasikan operasi yang ada ke dalam mesin. ▪ Lihat hasil penjadwalan pada diagram gantt

Agar lebih jelas proses menjalankan penjadwalan dapat digambarkan ngan diagram aktifitas. Diagram aktifitas yang menggambarkan proses



Gambar 3-5 diagram aktifitas proses menjalankan penjadwalan

1.6 *Use-Case* Menjalankan Operasi Genetik

Fungsi ini dijalankan oleh proses penjadwalan untuk mendapatkan solusi njadwalan yang optimal. Operasi genetik ini berupa tukar silang, mutasi, leksi. Operasi ini mengolah kromosom, yang merupakan hasil *encoding* dari gas-tugas yang ada, sehingga didapatkan kromosom dengan nilai *fitness* tinggi. makin besar nilai *fitness* maka kromosom tersebut semakin optimal.

Tabel 3-7 spesifikasi *use-case* menjalankan operasi genetik

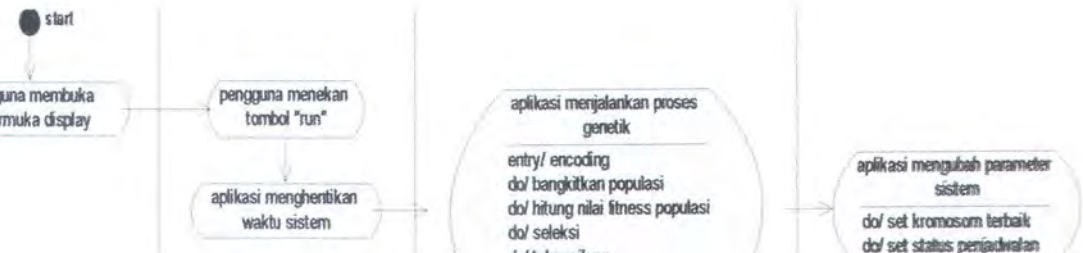
esifikasi	Penjelasan
ktor	▪ Pengguna
ujuan	▪ Melakukan alokasi tugas ke dalam mesin dan mencari kombinasi penjadwalan yang optimal
ngkasan	▪ Proses ini dilakukan untuk mendapatkan kombinasi penjadwalan yang optimal dengan menerapakan algoritma

Tabel 3-7 spesifikasi *use-case* menjalankan operasi genetik (lanjutan)

esifikasi	Penjelasan
ran Aksi normal	▪ Pengguna menjalankan penjadwalan melalui antarmuka display ▪ Tombol *run* menjalankan prosesdur jadwalUlang() ▪ Prosedur jadwalUlang() menghentikan waktu sistem serta menjalankan prosedur algoritma genetik

Untuk menjelaskan aliran proses menjalankan operasi genetik ini maka gambarkan melalui diagram aktivitas. Diagram aktivitas proses menjalankan erasi genetik ini digambarkan melalui gambar 3-6.

Awal dari proses ini tidak berbeda dengan proses menjalankan njadwalan karena proses menjalankan operasi genetik dilakukan oleh proses njalankan penjadwalan. Setelah pengguna menekan tombol run maka aplikasi an menghentikan waktu sistem dan kemudian aplikasi akan menjalankan proses netik. Hasil proses ini berupa kromosom dengan nilai *fitness* terbaik, yang rarti solusi yang optimal.



.7 *Use-Case* Menampilkan Hasil Penjadwalan

Proses ini dilakukan untuk manampilkan hasil proses penjadwalan melalui gram gantt. Proses ini dilakukan agar memudahkan pengguna untuk mahami hasil penjadwalan.

Tabel 3-8 spesifikasi *use-case* menampilkan hasil penjadwalan

esifikasi	Penjelasan
tor	▪ Aplikasi
juan	▪ Menampilkan hasil penjadwalan kedalam bentuk diagram gantt
ngkasan	▪ Proses ini dilakukan untuk menampilkan hasil penjadwalan kedalam diagram gantt. Diagram ini menggambarkan waktu mulai, lama proses dan mesin yang memproses masing-masin operasi.
asukan	▪ Hasil *decoding* kromsom terbaik
aran	▪ Diagram gantt
ndisi Awal	▪ Dataset untuk menggambarkan diagram masih kosong
ndisi Akhir	▪ Terbentuk dataset yang berisi operasi
iran Aksi normal	▪ Pengguna membuka diagram gantt melalui antarmuka Display ▪ Kelas GanttChartPanel akan menampilkan hasil penjadwalan melalui antrian operasi yang dimiliki masing-masing mesin. ▪ Kelas GanttChartPanel akan membuat sebuah dataset yang

unjukkan pada gambar 3-7. Proses menampilkan hasil penjadwalan dilakukan
h sistem berdasarkan hasil penjadwalan yang didapat.

Gambar 3-7 diagram aktifitas proses menampilkan hasil penjadwalan

1.8 *Use-Case* Entry Tugas Baru

Proses ini dilakukan apabila terdapat tugas baru yang datang dan harus gera diproses, tetapi proses produksi telah berjalan. Proses ini hampir sama ngan proses entry tugas awal, hanya saja pada proses ini terjadi ketika jadwal oduksi telah terbentuk sebelumnya.

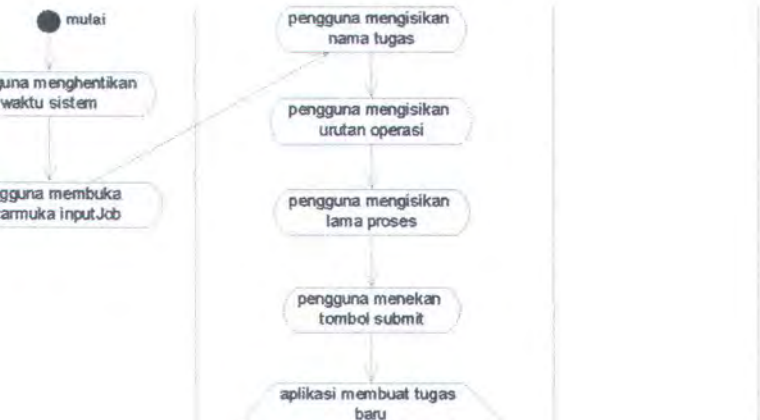
Tabel 3-9 spesifikasi *use-case* entry tugas baru

esifikasi	Penjelasan
tor	▪ Pengguna
juan	▪ Memasukkan tugas baru untuk dijadwalkan
ngkasan	▪ Pengguna melakukan entry tugas baru sementara penjadwalan telah terbentuk sebelumnya.
asukan	▪ Nama tugas

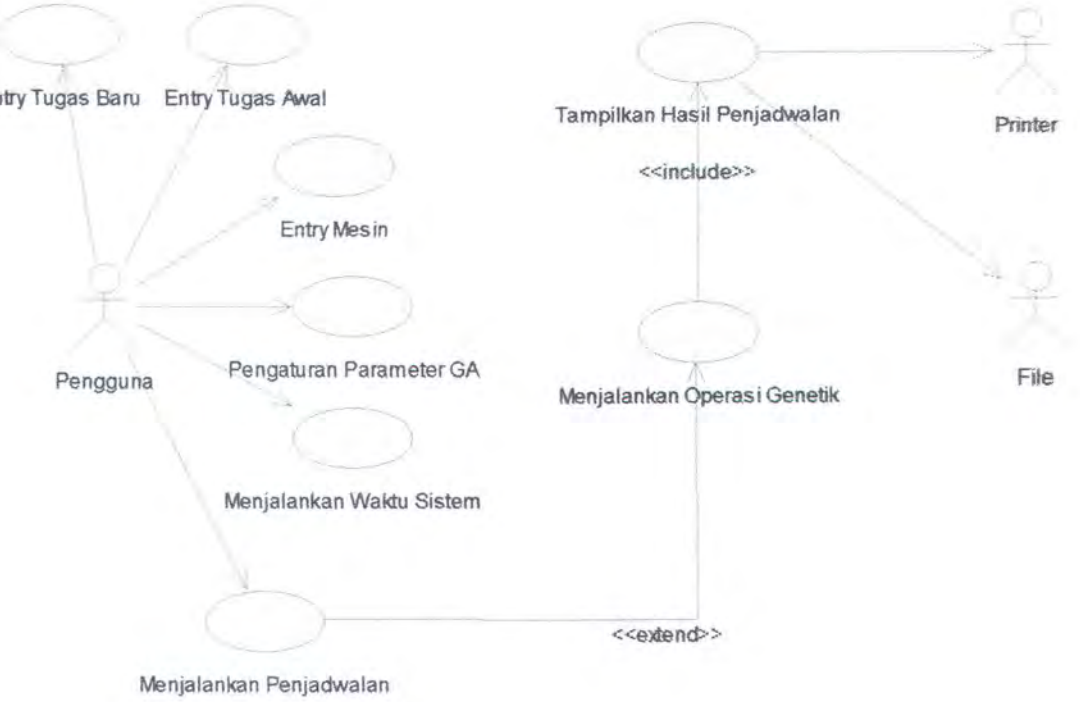
Tabel 3-9 spesifikasi *use-case* entry tugas baru (lanjutan)

esifikasi	Penjelasan
iran Aksi normal	▪ Hentikan sementara waktu sistem ▪ Buka antarmuka entry tugas/*job* melalui FrameUtama ▪ Entry nama tugas baru yang akan dibuat ▪ Pilih urutan mesin yang akan digunakan untuk proses ▪ Tentukan lama waktu proses untuk tiap operasi. ▪ Setelah selesai mamasukkan seluruh tugas baru, klik tombol ok untuk menempatkan tugas baru dalam kumpulan tugas lama.

Proses entry tugas baru dapat digambarkan dengan diagram aktivitas. agram aktivitas proses ini digambarkan pada gambar 3-8. Proses ini diawali ngan menghentikan waktu sistem oleh pengguna, kemudian diikuti dengan embuka antarmuka input job. Proses ini diakhir dengan menjalankan kembali ktu sistem.



Setelah didapatkan seluruh fungsi yang akan diakukan oleh sistem, maka oses berikutnya adalah menggambarkan fungsi-fungsi tersebut, aktor dan keraksinya. Diagram *use-case* yang menggambarkan fungsi-fungsi tersebut rdapat pada gambar 3-9.



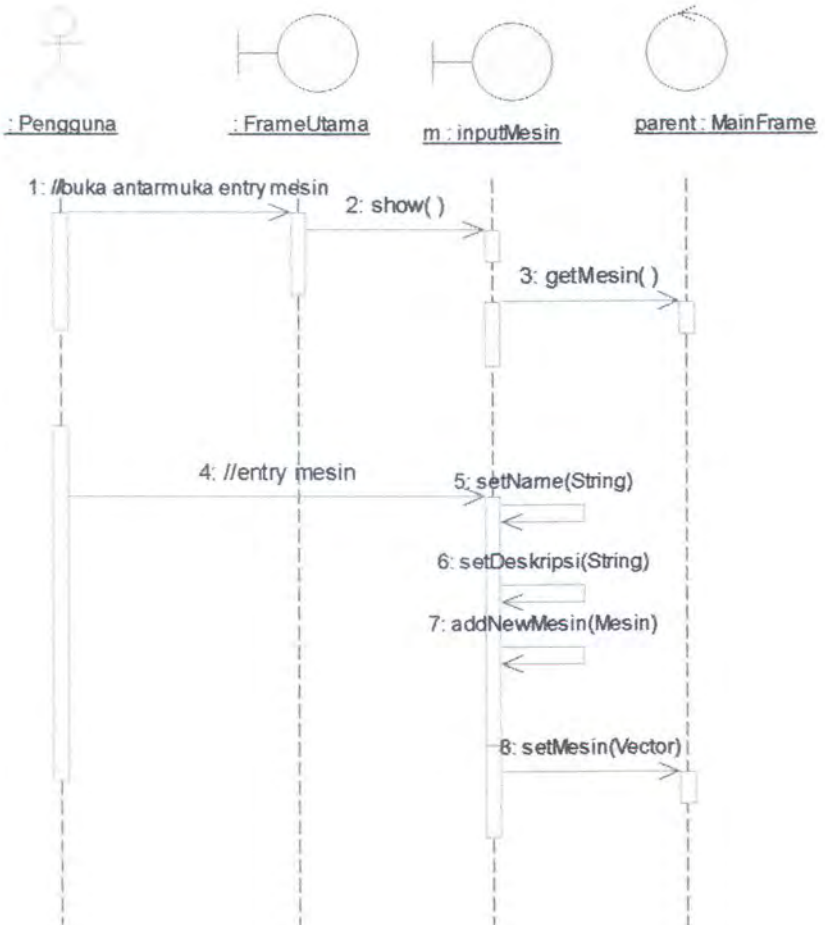
Gambar 3-9 diagram *use-case* sistem penjadwalan

2 Realisasi *Use-Case*

Bagian ini akan membahas realisasi analisa *use-case*. Diagram aktivitas

2.1 Realisasi *Use-Case* Entry Mesin

Realisasi dan penjelasan lebih detil mengenai fungsi dan proses entry esin akan dijelaskan melalui diagram sekuensi. Proses entry mesin beserta ngsi yang terkait dapat digambarkan dalam diagram sekuensi yang terdapat da gambar 3-10.

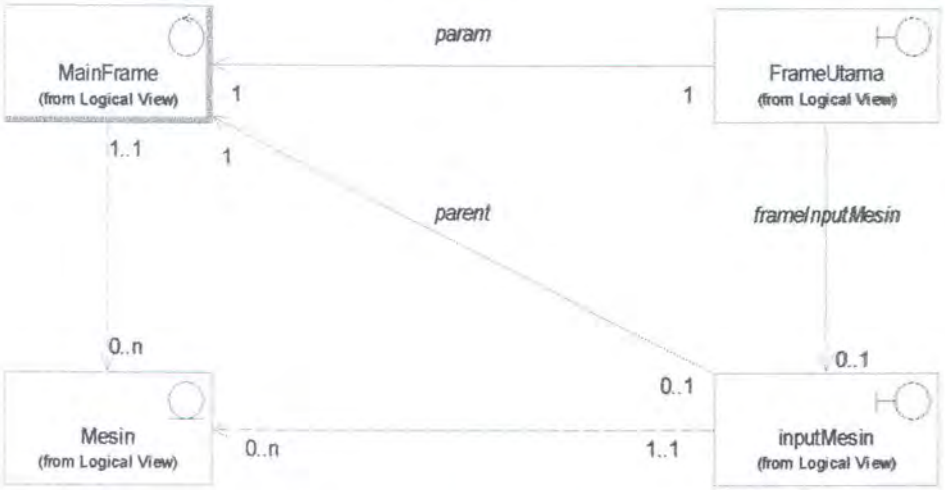


Tabel 3-10 deskripsi fungsi proses entry mesin

Fungsi	Deskripsi
//membuka antarmuka entry mesin	Proses agar pengguna dapat melakukan entry mesin
Show ()	Fungsi yang panggil untuk membuka antarmuka input job
getMesin()	Fungsi ini dipanggil untuk mendapatkan mesin yang telah ada.
// entry mesin	Proses mengisikan mesin baru kedalam aplikasi
setName(String)	Fungsi untuk memberikan nama mesin baru sesuai yang diberikan pengguna
setDeskripsi(String)	Fungsi untuk memberikan deskripsi mesin baru sesuai yang diberikan pengguna
addNewMesin (Mesin)	Menyimpan sementara mesin yang telah dibuat
setMesin(Vector)	Menggabungkan mesin yang telah dibuat dengan mesin yang telah ada sebelumnya

Diagram sekuensi entry mesin diatas terdiri dari dua proses utama yang an dilakukan oleh pengguna. Proses pertama adalah proses membuka tarmuka entry mesin. Proses ini dilakukan agar pengguna dapat mengisikan esin baru dalam aplikasi. Proses kedua adalah proses entry mesin, yaitu proses ntuk menciptakan mesin baru dalam aplikasi. Gambar 3-10 menunjukan kelas ng terlibat dalam proses entry mesin. Kelas tersebut adalah FrameUtama, putMesin, MainFrame. Hubungan antara kelas yang terlibat dalam proses entry esin dijelaskan melalui diagram VOPC (View of Participated Class) seperti da gambar 3-11.

Dalam hubungan ini terdapat kelas mesin yang merupakan kelas entity,



Gambar 3-11 diagram partisipasi proses entry mesin

2.2 Realisasi *Use-Case* Entry Tugas Awal

Penjelasan lebih lanjut mengenai fungsi dan proses entry tugas awal jelaskan melalui diagram sekuensi. Proses entry tugas awal dapat digambarkan lam diagram sekuensi yang terdapat pada gambar 3-12.

Menurut gambar 3-12, terdapat beberapa fungsi yang terlibat dalam proses try tugas awal. Pada proses entry tugas awal terdapat dua aktifitas utama yang rus dilakukan pengguna, yaitu membuka antarmuka entry tugas dan melakukan try tugas. Proses membuka entry tugas dilakukan agar pengguna dapat emasukkan tugas baru kedalam aplikasi. Sedangkan proses entry tugas

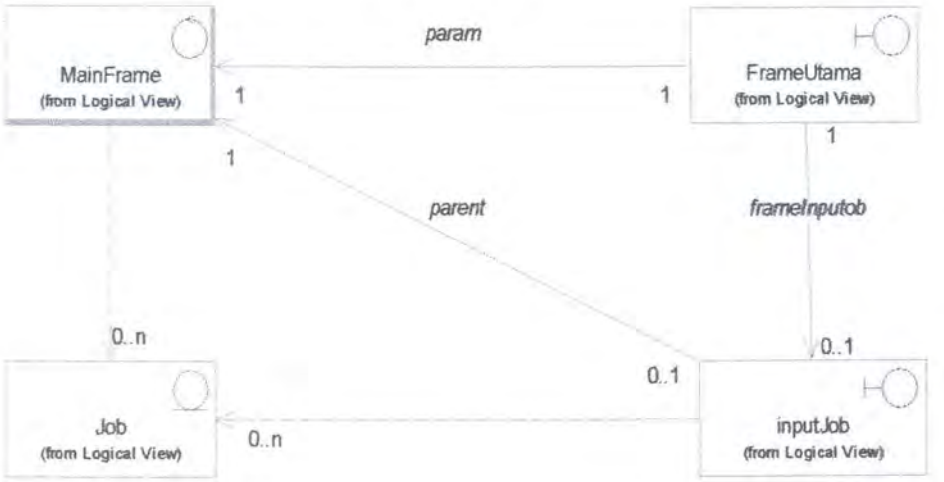
Gambar 3-12 diagram sekuensi entry job awal

Tabel 3-11 deskripsi fungsi proses entry tugas awal

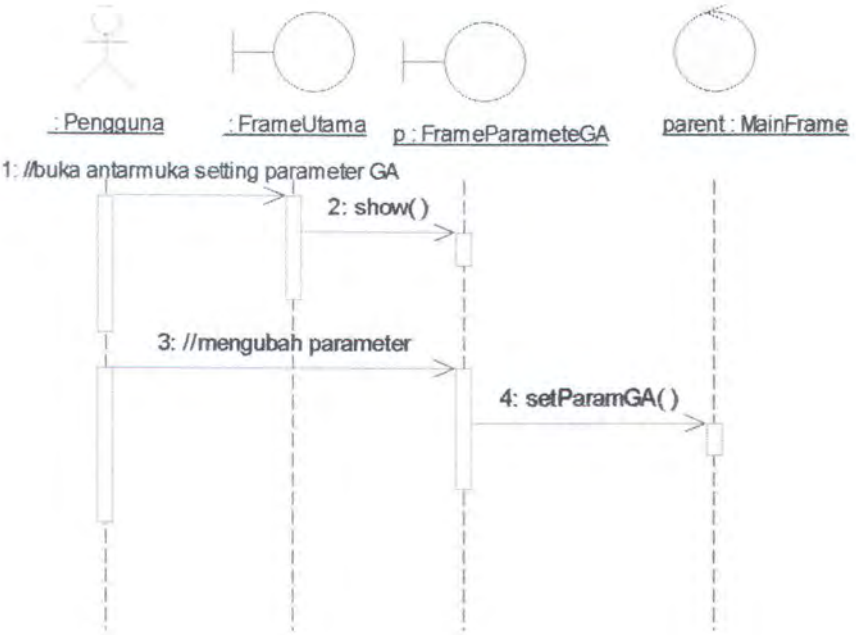
Fungsi	Deskripsi
//membuka antarmuka entry job	proses agar pengguna dapat melakukan entry job
Show ()	Fungsi yang panggil untuk membuka antarmuka input job
getMesin()	Fungsi ini dipanggil untuk mendapatkan mesin yang telah ada.
	Fungsi ini dipanggil untuk mendapatkan

Diagram sekuensi di atas menggambarkan kelas yang terlibat dalam proses ry tugas awal. Kelas tersebut adalah FrameUtama, inputJob, MainFrame. bungan antara kelas yang terlibat dalam proses entry mesin dijelaskan melalui gram VOPC (*View of Participated Class*) seperti pada gambar 3-13.

Hubungan antara kelas yang terlibat dalam proses entry tugas awal elaskan pada gambar 3-13. Dalam hubungan ini terdapat kelas Job yang rupakan kelas yang merepresentasikan tugas yang masuk, jumlah objek job ng dapat diciptakan antar 1 sampai n. Jumlah kelas MainFrame dalam diagram adalah 1, yang berarti objek MainFrame harus ada dalam proses entry tugas. gitu juga dengan FrameUtama, frame ini merupakan antarmuka utama yang alankan pertama kali oleh sistem. Sedangkan objek inputJob boleh bernilai ***null*** n maksimal 1 objek.



Gambar 3-13 diagram partisipasi proses entry tugas awal



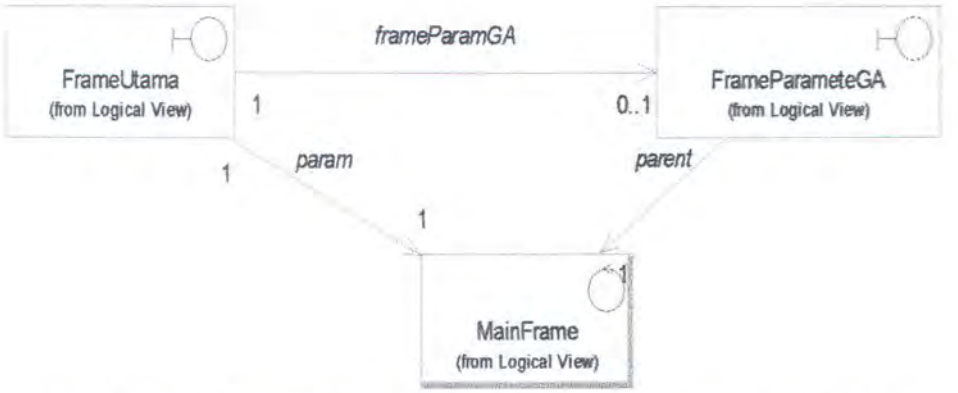
Gambar 3-14 diagram sekuensi pengaturan parameter genetik

Fungsi yang terdapat pada diagram sekuensi pengaturan parameter genetik elaskan lebih lanjut pada tabel 3-12

Tabel 3-12 deskripsi fungsi proses pengaturan parameter GA

	Fungsi	Deskripsi
1	//buka antarmuka setting parameter GA	Proses untuk membuka antarmuka setting GA
2	Show()	Fungsi untuk membuka parameter GA
3	//mengubah parameter	Proses dimana pengguna melakukan perubahan terhadap parameter GA
4	setParamGa()	Fungsi untuk merealisasikan perubahan nilai parameter GA

Hubungan kelas yang terdapat dalam diagram sekuensi pengaturan goritma genetik jelaskan lebih lanjut pada gambar 3-15. Pada diagram tersebut ameParameterGA mempunyai jumlah 0..1 hal ini berarti jumlah objek boleh rnilai *null* dengan jumlah maksimal 1. Sedangkan objek FrameUtama dan ainFrame harus ada dan tidak boleh tidak.

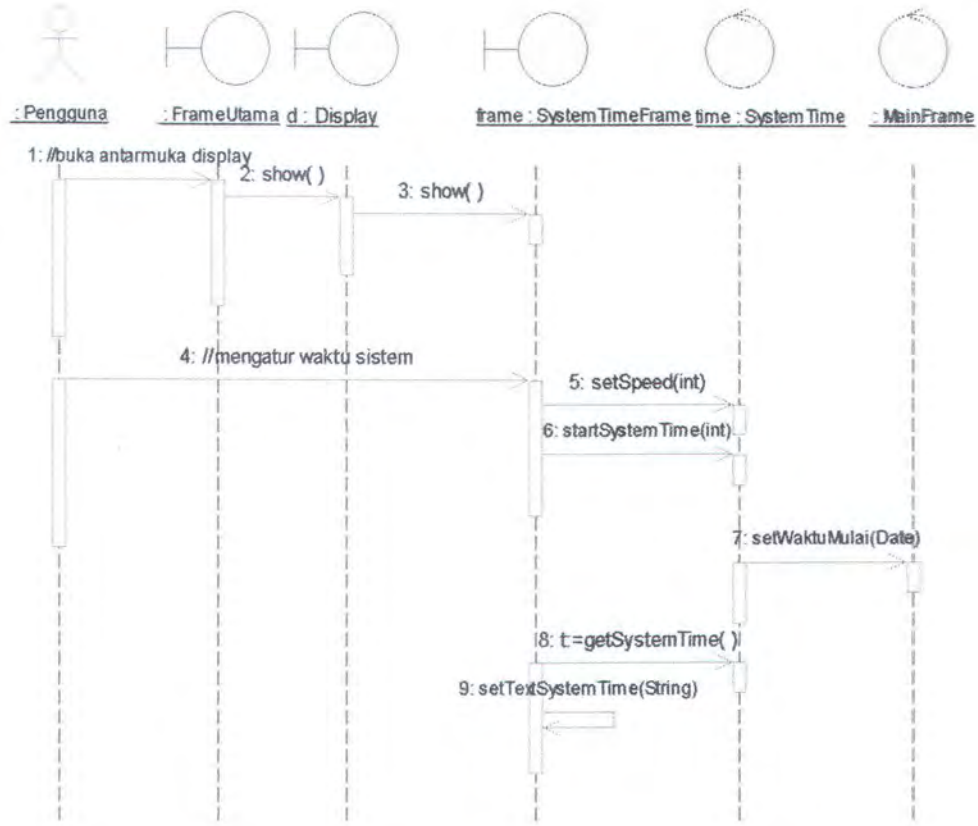


Gambar 3-15 diagram partisipasi proses pengaturan parameter GA

2.4 Realisasi *Use-Case* Menjalankan Waktu Sistem

Sebelum dapat menjalankan penjadwalan pengguna harus menjalankan aktu sistem. Karena waktu sistem ini diperlukan oleh seluruh proses njadwalan dalam sistem. Proses menjalankan waktu sistem dijelaskan lebih ajut melalui diagram sekuensi seperti pada gambar 3-16. Seluruh fungsi yang emerlukan data waktu mereferensi waktu dari Kelas SystemTime ini.

Proses menjalankan waktu sistem dilakukan dengan memanggil fungsi



Gambar 3-16 diagram sekuensi menjalankan waktu system

Tabel 3-13 deskripsi fungsi proses menjalankan waktu sistem

Fungsi	Deskripsi
//buka antarmuka display	Proses agar pengguna dapat melakukan entry job
Show ()	Fungsi yang panggil untuk membuka antarmuka display
	Fungsi yang panggil untuk membuka

Tabel 3-13 deskripsi fungsi proses menjalankan waktu sistem (lanjutan)

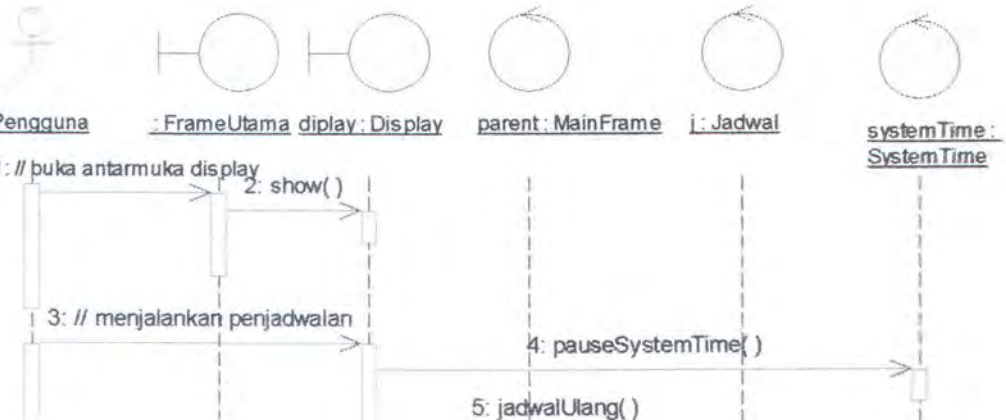
Fungsi	Deskripsi
setWaktuMulai(Date)	Fungsi untuk mencatat waktu mulai penjadwalan kedalam kelas MainFrame
getSystemTime()	Fungsi ini dipanggil untuk mendapatkan waktu sistem.
setTextSystemTime()	Fungsi yang dipanggil untuk mengubah label pada antarmuka SystemTimeFrame sebagai Informasi bagi pengguna

Dalam proses menjalankan waktu sistem terdapat beberapa kelas yang libat, yaitu FrameUtama, MainFrame, Display, SystemTimeFrame, stemTime, MainFrame. Hubungan antara kelas yang terlibat dalam proses ngaturan parameter genetik dijelaskan melalui diagram VOPC (*View of rticipated Class*) seperti pada gambar 3-17.

Proses ini melibatkan 5 kelas yaitu FrameUtama, MainFrame, stemTime, Display, SystemTimeFrame. Secara global objek MainFrame harus a dalam setiap hubungan (*association*), karena MainFrame merupakan kelas ng berisi parameter sistem. Sedangkan jumlah objek Display boleh bernilai *null* n maksimal 1. Dalam hubungan antara SystemTimeFrame dengan SystemTime, nlah objek SystemTime harus ada dan tidak boleh tidak. Objek Display boleh emiliki objek SystemTimeFrame minimal *null* dan maksimal 1.

.5 Realisasi *Use-Case* Menjalankan Penjadwalan

Proses menjalankan penjadwalan dilakukan untuk mengalokasikan tugas ıg telah masuk kedalam mesin untuk diproses. Penjadwalan dilakukan terhadap ɛrasi dengan status menunggu dan belum terjadwal. Sedangkan operasi dengan tus dalam proses dan selesai tidak dijadwalkan ulang. Dalam menjalankan ses ini pengguna terlebih dahulu harus membuka frame display. Dimana dalam me display ini terdapat berbagai fitur untuk menjalankan penjadwalan, ngatur waktu sistem, melihat hasil penjadwalan serta memantau status tiap erasi. Sehingga untuk menjalankan penjadwalan serta mengatur waktu sistem akukan melalui frame display. Proses menjalankan penjadwalan dijelaskan ih lanjut pada gambar 3-18. Gambar 3-18 menjelaskan proses dan fungsi yang libat dalam proses menjalankan penjadwalan. Fungsi dan kelas yang terlibat n dijelaskan lebih lanjut.

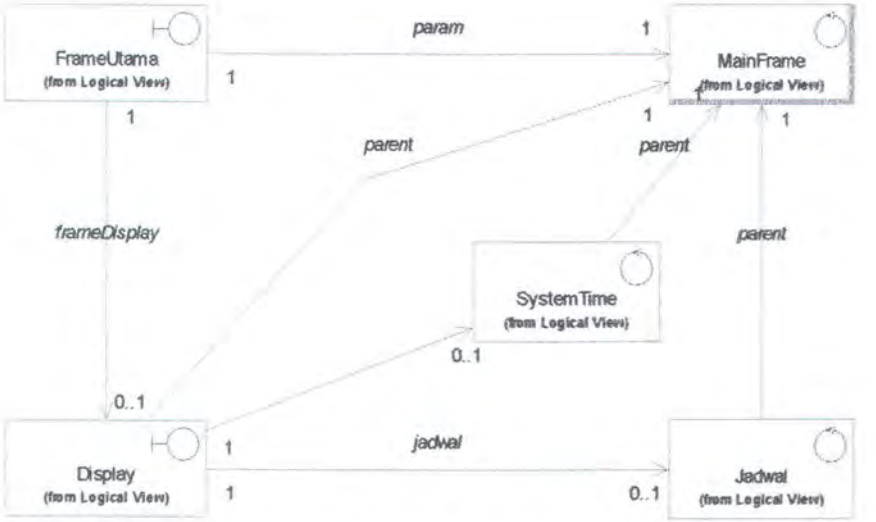


Fungsi yang terlibat dalam proses menjalankan penjadwalan dijelaskan
ih lanjut pada tabel 3-14.

Tabel 3-14 deskripsi fungsi proses menjalankan penjadwalan

Fungsi	Deskripsi
//buka antarmuka display	Proses agar pengguna dapat melakukan entry job
Show ()	Fungsi yang panggil untuk membuka antarmuka display
Show()	Fungsi yang panggil untuk membuka antarmuka SystemTimeFrame yang menangani proses yang berkenaan dengan waktu sistem
//menjalankan penjadwalan	Proses yang dilakukan pengguna untuk menjalankan penjadwalan
pauseSystemTime()	Fungsi yang dipanggil untuk menghentikan waktu sistem
jadwalUlang ()	Fungsi yang dipanggil menjalankan penjadwalan terhadap tugas yang masuk
buildChart()	Fungsi untuk menampilkan hasil penjadwalan kedalam diagram gantt
getSystemTime()	Fungsi ini dipanggil untuk mendapatkan waktu sistem.
setStatusJadwal(boolean)	Fungsi yang dipanggil untuk mengubah status jadwal, apakan terjadi penjadwalan ulang.

Dalam proses menjalankan penjadwalan terdapat beberapa kelas yang
peran yaitu, FrameUtama, MainFrame, Display, SystemTime, Jadwal.
bungan antara kelas yang terlibat dalam proses pengaturan parameter genetik
elaskan melalui diagram VOPC (*View of Participated Class*) seperti pada
nbar 3-19

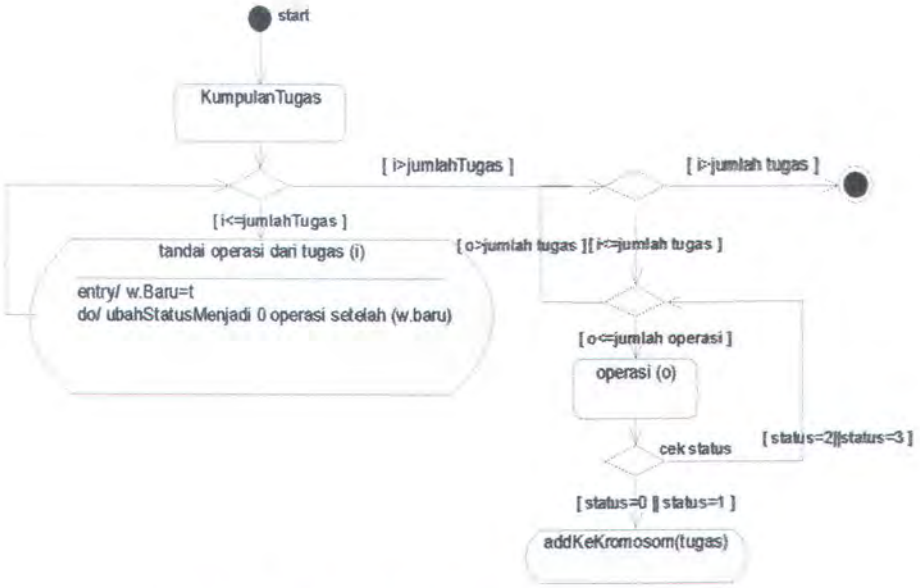


Gambar 3-19 diagram partisipasi proses menjalankan penjadwalan

2.6 Realisasi *Use-Case* Menjalankan Operasi Genetik

Proses menjalankan operasi genetik merupakan bagian dari proses enjalankan penjadwalan. Proses menjalankan operasi genetik ini dipanggil oleh ngsi *jadwalUlang*() pada proses menjalankan penjadwalan. Sedangkan fungsi *dwalUlang()* dipanggil melalui tombol "run". Proses menjalankan operasi netik melibatkan berbagai fungsi, antara lain *encoding*, pembangkitan populasi, rhitungan nilai *fitness*, seleksi, tukar silang, mutasi. Fungsi – fungsi tersebut an dijelaskan lebih detil.

2.6.1 Realisasi Proses Encoding



Gambar 3-20 diagram aktivitas proses *encoding*

2.6.2 Realisasi Proses Pembangkitan Populasi

Prosedur *generatePopulasi()* akan membangkitkan sejumlah *n* kromosom lam populasi secara acak. Dimana *n* merupakan parameter yang telah entukan oleh pengguna. Proses pembangkitan ini dilakukan melalui kromosom rtama yang terbentuk dalam proses *encoding()*. Kromosom tersebut kemudian urunkan menjadi kromosom lain. Dimana posisi gen kromosom baru tersebut ah diubah secara acak. Proses pembangkitan populasi dapat digambarkan lalui diagram aktivitas seperti pada gambar 3-21.

3.6.3 Realisasi Proses Perhitungan Nilai *Fitness*

Proses selanjutnya adalah menghitung nilai *fitness* dari seluruh kromosom ng terdapat dalam populasi. Perhitungan nilai *fitness* dilakukan oleh fungsi aluasi. Nilai *fitness* ini yang menandakan nilai optimasi masing-masing omosom. Proses perhitungan nilai *fitness* digambarkan pada gambar 3-22

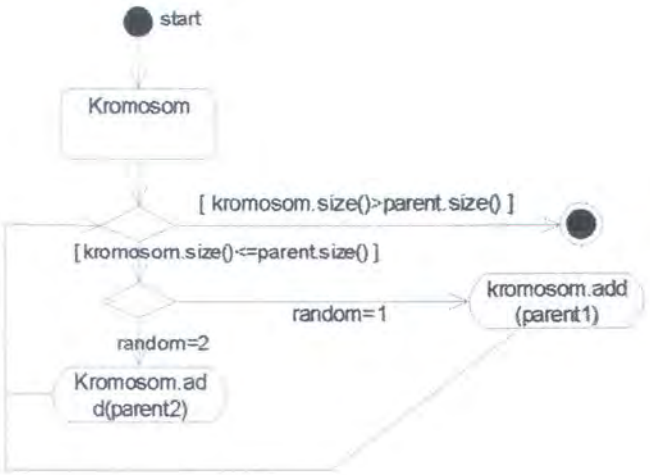
Gambar 3-22 diagram aktivitas perhitungan nilai *fitness*

3.6.4 Realisasi Proses Seleksi

Proses seleksi ini memilih sebanyak *n* kromosom yang memiliki nilai *ness* terbaik. Nilai *n* ditentukan oleh pengguna. Kromosom yang tidak terpilih lam proses seleksi akan menjalani proses tukar silang dan mutasi untuk ndapatkan kromosom yang lebih optimal. Proses seleksi dijelaskan pada gram aktifitas seperti pada gambar 3-23.

2.6.5 Realisasi Proses Tukar Silang

Proses tukar silang (*crossover*) merupakan proses genetik yang ngubahkan dua kromosom menjadi satu kromosom baru. Metode yang gunakan dalam proses tukar silang adalah *Precedence Preservative Crossover* PC). Pemilihan gen kromosom parent 1 atau parent 2 dilakukan secara acak. oses pemilihan dilakukan sampai jumlah gen kromosom baru sama dengan mlah kromosom orang tua. Proses tukar silang ini dijelaskan melalui diagram ivitas seperti pada gambar 3-24.



Gambar 3-24 diagram aktivitas proses tukar silang

2.6.6 Realisasi Proses Mutasi

Sedangkan mutasi (*mutation*) merupakan proses genetik yang

Gambar 3-25 diagram aktivitas proses mutasi

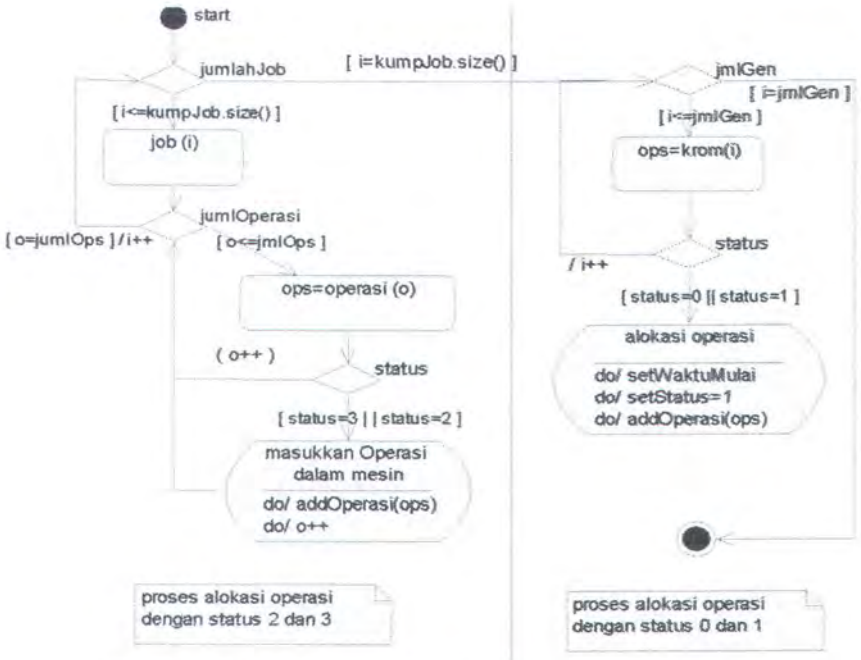
2.6.7 Realisasi Proses Decoding

Kromosom dengan nilai *fitness* terbaik, yang didapatkan dari operasi netik, akan diubah menjadi suatu jadwal produksi. Proses ini dibagi menjadi a proses utama. Proses pertama, mengalokasikan operasi berstatus 2 dan 3 dalam mesin. Proses pertama dilakukan agar operasi dengan status 2 dan 3 tetap rada dalam antrian mesin. Kedua, mengalokasikan operasi berstatus 0 dan 1 dalam mesin serta mencatat waktu mulai produksi.

Untuk menentukan waktu mulai proses sebuah operasi terdapat beberapa teria yang harus dipenuhi, yaitu :

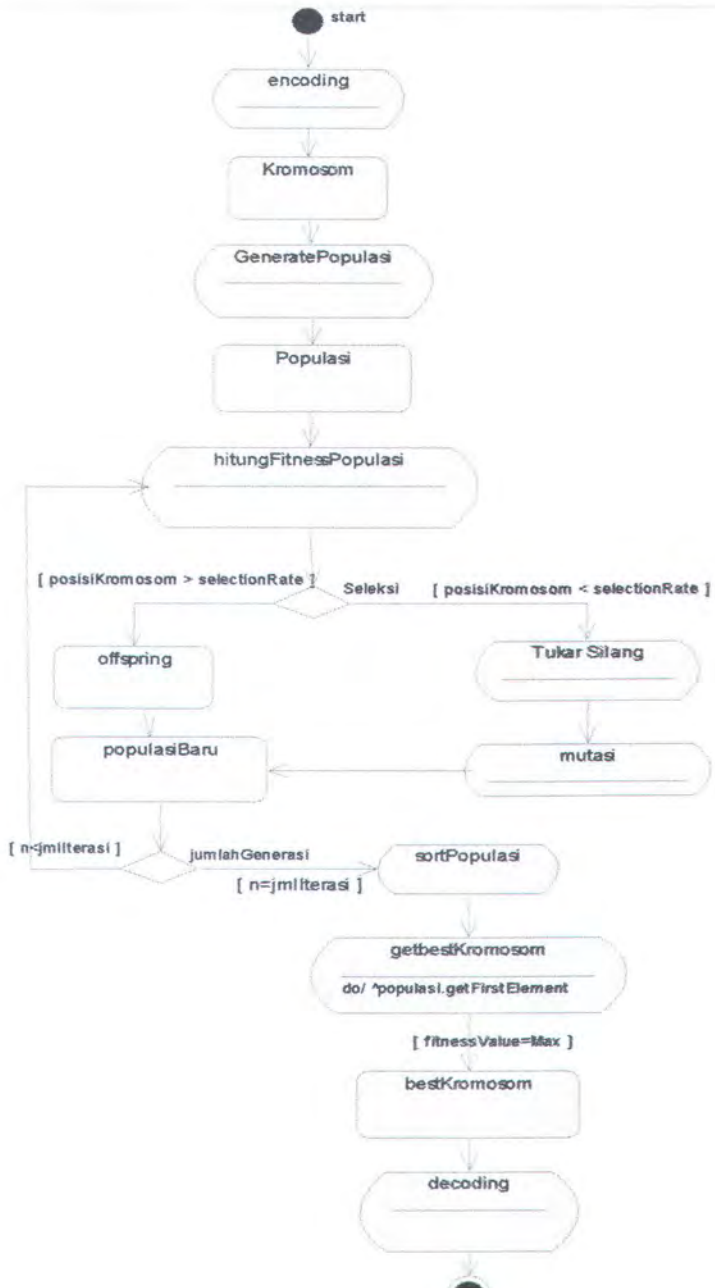
Apabila waktu idle mesin lebih besar daripada waktu selesai operasi

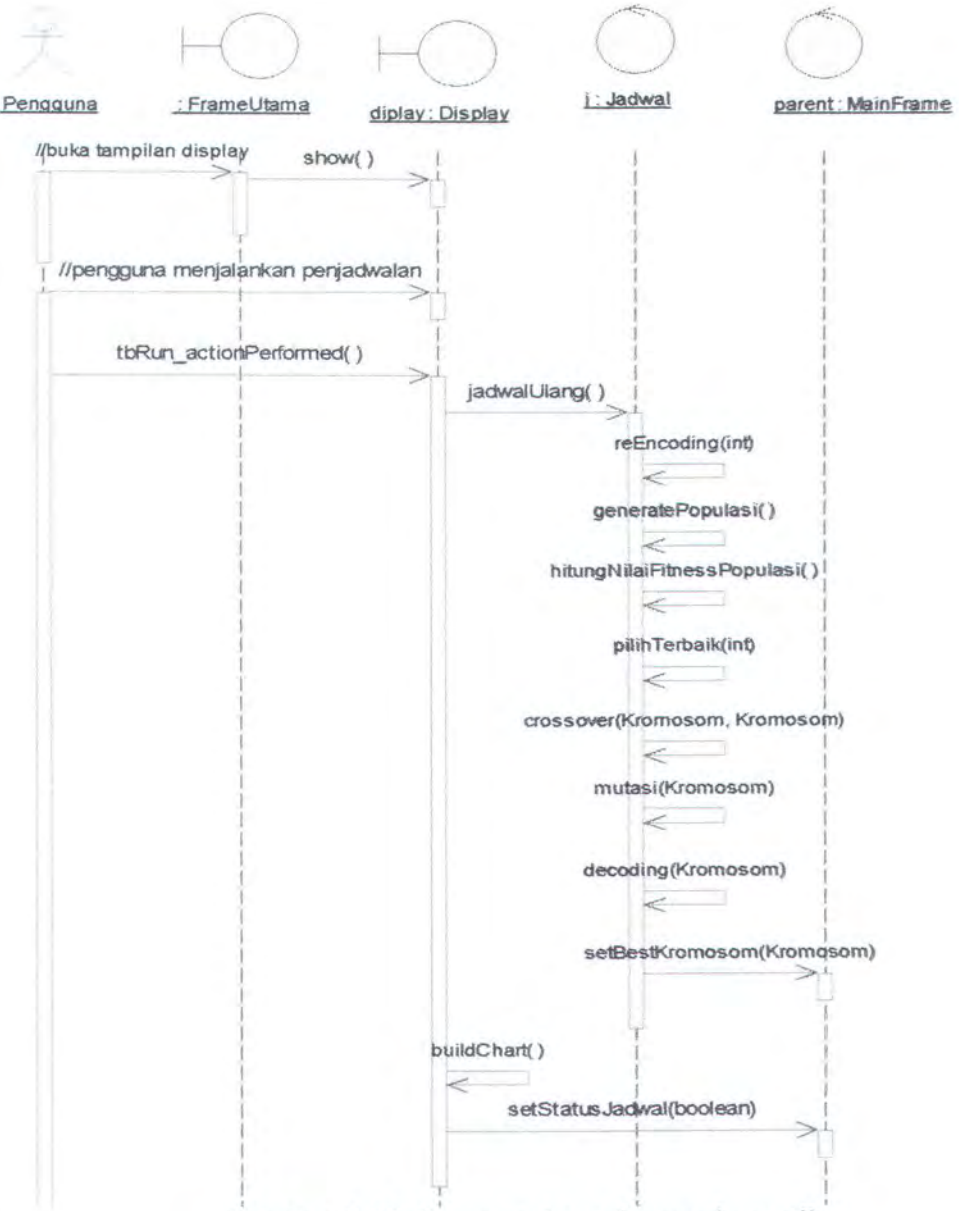
Proses *decoding* dapat digambarkan melalui diagram aktivitas seperti pada
nbar 3-26.



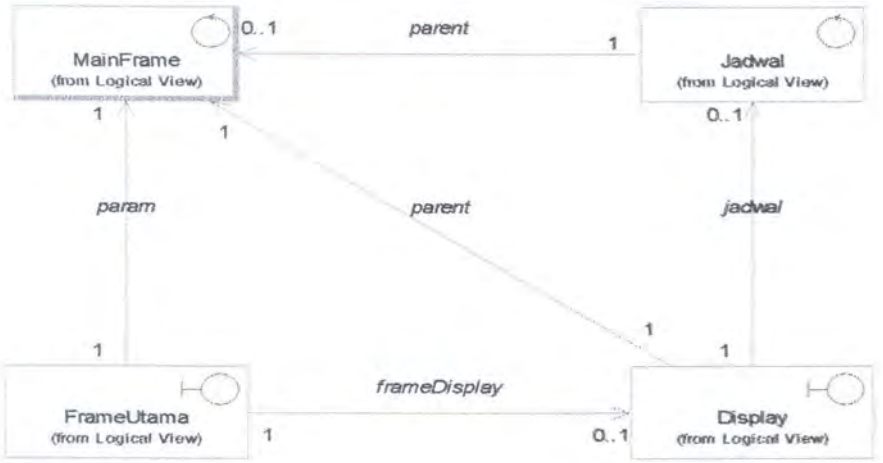
Gambar 3-26 diagram aktivitas proses *decoding*

Seluruh proses operasi genetik ini dilakukan terus menerus sampai *i*
rasi. Iterasi ini merepresentasikan proses pembentukan generasi baru dengan *i*
alah jumlah keturunan. Setelah *i* iterasi atau keturunan terselesaikan, proses
anjutnya adalah menentukkan kromosom yang memiliki nilai *fitness* terbesar.
omosom inilah yang menjadi kromosom yang optimal.


start
encoding
Kromosom
GeneratePopulasi
Populasi
hitungFitnessPopulasi
[posisiKromosom > selectionRate]
Seleksi
[posisiKromosom < selectionRate]
offspring
Tukar Silang
populasiBaru
mutasi
[n<jmlIterasi]
jumlahGenerasi
[n=jmlIterasi]
sortPopulasi
getbestKromosom
do/ ^populasi.getFirstElement
[fitnessValue=Max]
bestKromosom
decoding



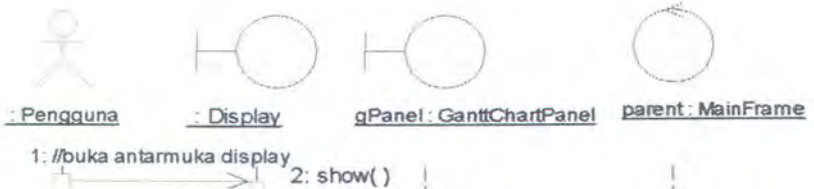
Gambar 3-28 diagram sekuensi operasi genetik



Gambar 3-29 diagram partisipasi proses operasi genetik

2.7 Realisasi *Use-Case* Menampilkan Hasil Penjadwalan

Proses menampilkan hasil penjadwalan merupakan proses mengubah hasil njadwalan kedalam diagram gantt. Proses ini dilakukan agar hasil penjadwalan adah dimengerti oleh pengguna. Penjelasan lebih lanjut mengenai fungsi dan las yang terlibat dalam proses menampilkan hasil penjadwalan digambarkan lam diagram sekuensi seperti pada gambar 3-30. Penjelasan mengenai fungsi ng terlibat dalam proses menampilkan hasil penjadwalan terdapat pada tabel 3-

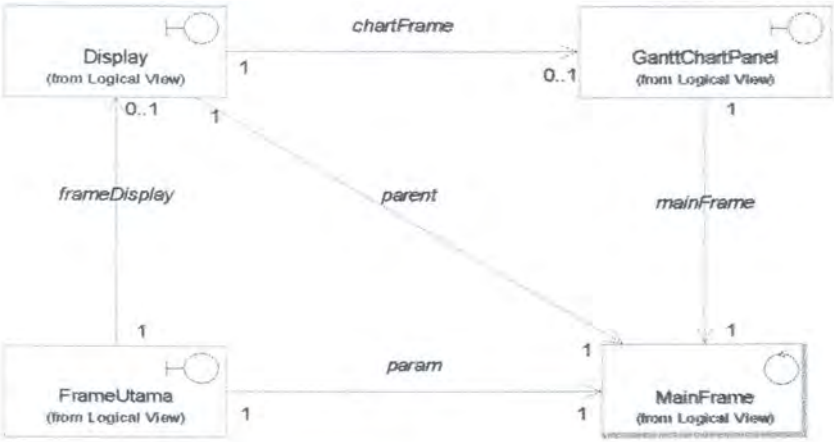


Tabel 3-15 deskripsi fungsi proses menampilkan hasil penjadwalan

Fungsi	Deskripsi
//buka antarmuka display	Proses agar pengguna dapat melihat diagram hasil penjadwalan
Show ()	Fungsi yang panggil untuk membuka antarmuka panel diagram gantt
getMesin()	Fungsi yang panggil untuk mendapatkan mesin-mesin yang ada
createDataset()	Fungsi untuk membuat dataset yang digunakan untuk menggambar diagram gantt. Dataset ini berisi operasi yang ada dalam antrian mesin.
createChart()	Fungsi yang dipanggil untuk mengambar diagram gantt dari dataset yang telah dibentuk

Proses menampilkan hasil penjadwalan melibatkan kelas Display, ainFrame dan kelas GanttChartPanel. Kelas yang akan menampilkan diagram ntt adalah kelas GanttChartPanel. GanttChartPanel memerlukan data mesin ng dimiliki dari kelas MainFrame. Data mesin ini nantinya akan digunakan tuk membuat dataset dalam pembuatan diagram gantt.Diagram gantt yang erupakan hasil pembangkitan kelas GanttChartPanel akan ditampilkan dalam tarmuka Display.

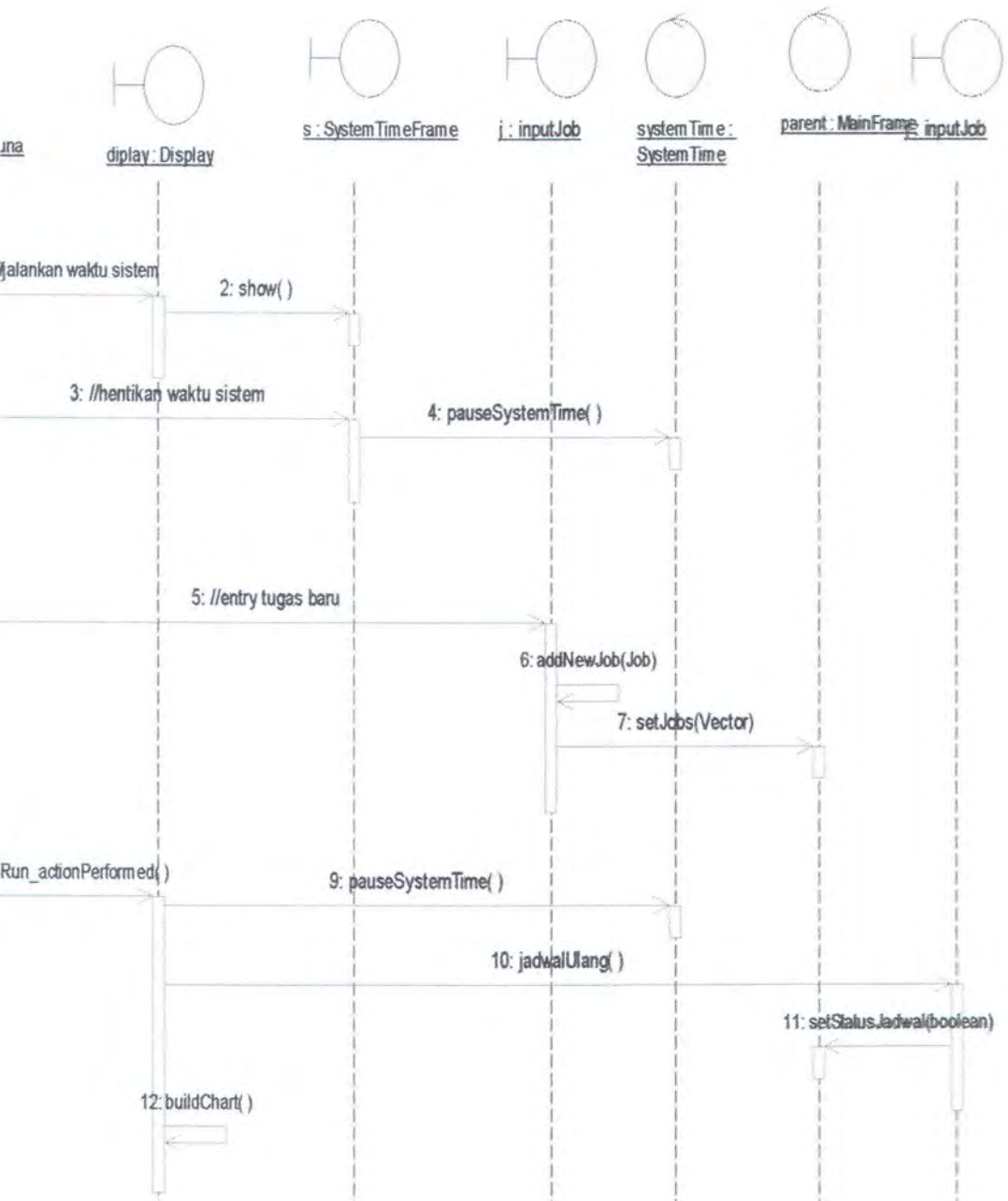
Hubungan kelas-kelas yang terlibat proses ini digambarkan pada gambar



Gambar 3-31 diagram partisipasi proses menampilkan hasil penjadwalan

2.8 Realisasi *Use-Case* Entry Tugas Baru

Proses entry tugas baru dilakukan apabila terdapat tugas baru yang datang n harus segera diproses. Proses ini mirip dengan proses entry tugas awal, tetapi da proses ini sudah terbentuk jadwal sebelumnya. Proses lebih detil mengenai ry tugas baru dijelaskan dalam diagram sekuensi. Digram sekuensi yang nggambarkan proses operasi genetik ini digambarkan pada gambar 3-33. bungan antara kelas yang terlibat dalam proses pengaturan parameter genetik elaskan melalui diagram VOPC (*View of Participated Class*) seperti pada mbar 3-32.

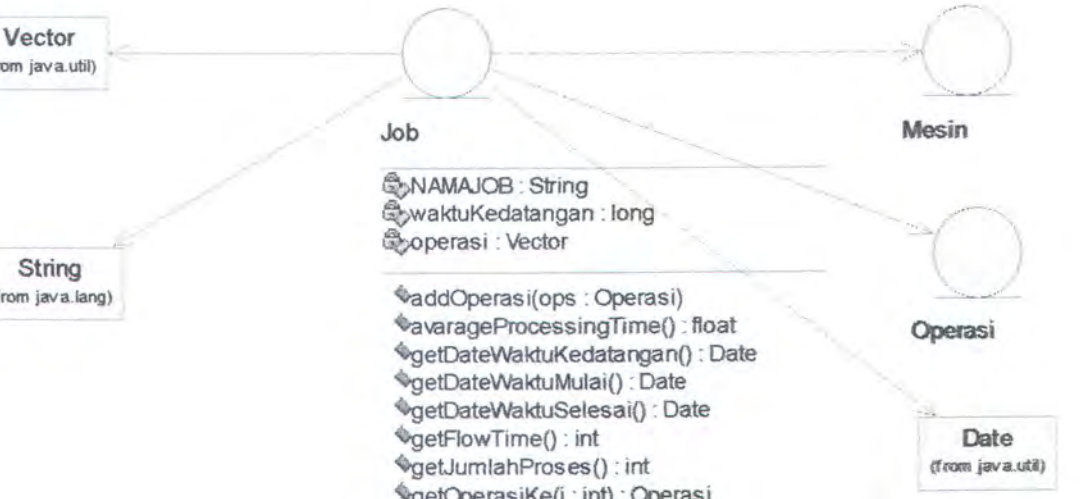

una
diplay : Display
s : SystemTimeFrame
j : inputJob
systemTime : SystemTime
parent : MainFrame
inputJob
jalankan waktu sistem
2: show( )
3: //hentikan waktu sistem
4: pauseSystemTime( )
5: //entry tugas baru
6: addNewJob(Job)
7: setJobs(Vector)
Run_actionPerformed( )
9: pauseSystemTime( )
10: jadwalUlang( )
11: setStatusJadwal(boolean)
12: buildChart( )

Perancangan Kelas

Seperti telah dijelaskan dalam diagram partisipasi di atas, terdapat berapa kelas yang digunakan dalam aplikasi ini. Perancangan kelas ini njelaskan kelas-kelas yang digunakan dalam aplikasi ini beserta hubungan tar kelas. Penjelasan kelas-kelas yang digunakan adalah sebagai berikut,

.1 Kelas Tugas.

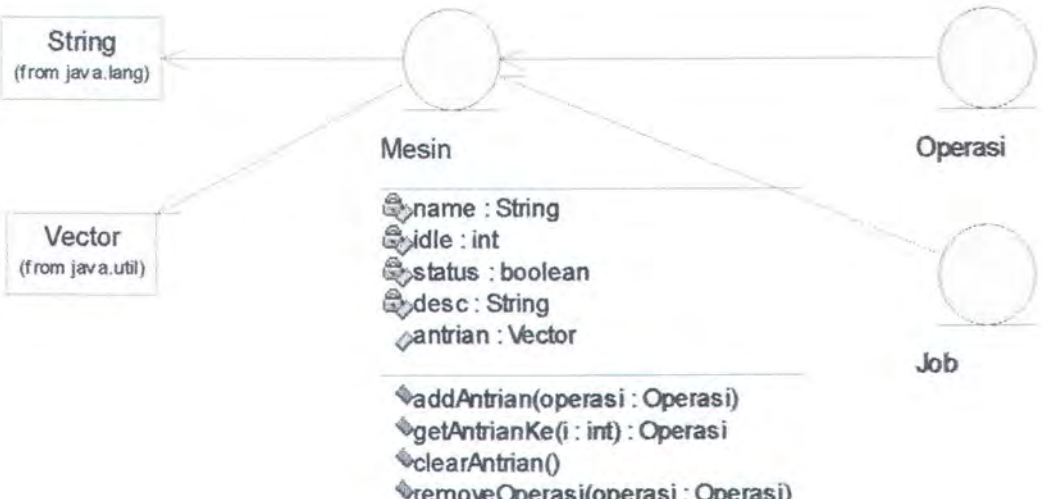
Kelas tugas ini merupakan kelas yang merepresentasi tugas yang masuk n yang akan diproses oleh sistem. Kelas Job ini merupakan entity bagi rangkat lunak ini dan data inilah yang akan diolah oleh sistem. Secara lengkap las tugas (*job*) ini digambarkan melalui diagram kelas seperti pada gambar 3- . Gambar 3-34 menggambarkan atribut, fungsi dan *dependencies* kelas Tugas *ob*).



3.2 Kelas Mesin

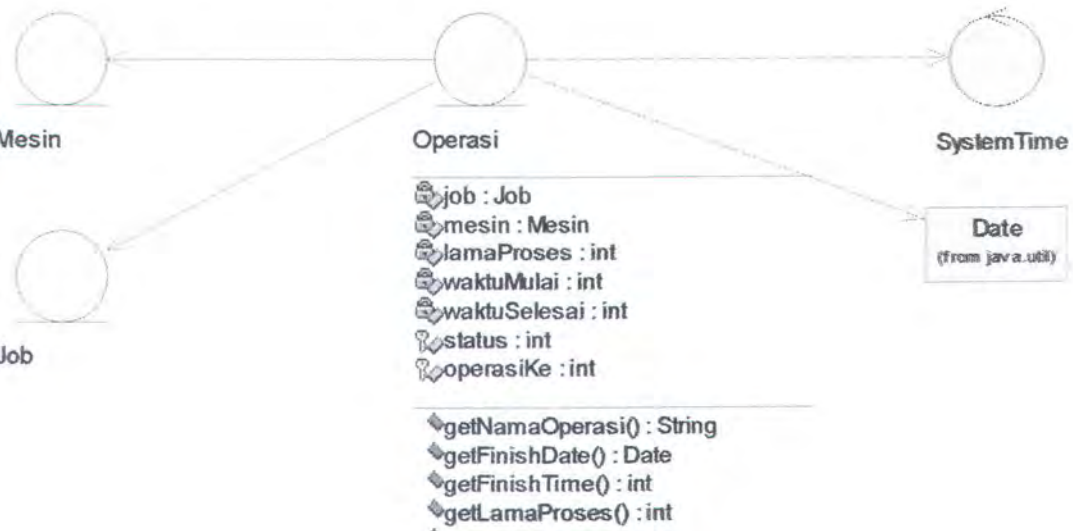
Kelas ini melambangkan mesin yang digunakan dalam proses produksi. elas mesin ini dibangkitkan setelah pengguna melakukan entry mesin ke dalam likasi. Kelas Job ini merupakan entity bagi perangkat lunak ini dan data inilah ng akan diolah oleh sistem.

Kelas mesin mempunyai atribut nama dan deskripsi yang membedakan jek mesin satu dengan objek mesin yang lainnya. Kelas ini memiliki sebuah *ctor* yang digunakan untuk menyimpan operasi yang menunggu untuk diproses lam mesin ini. Atribut getIdle() digunakan untuk mendapatkan waktu idle esin yang berasal dari waktu selesai tugas terakhir dalam antrian. Diagram kelas tuk menggambarkan data mesin ini terdapat pada gambar 3-35.



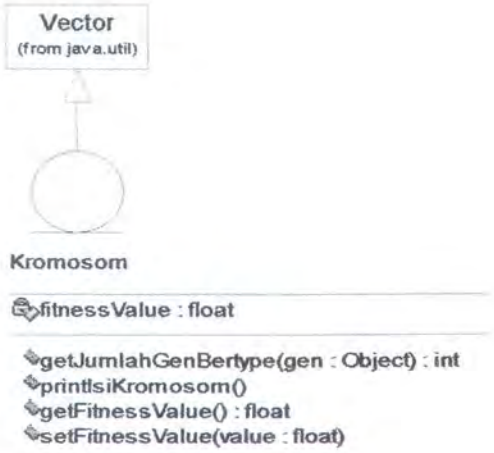
.3 Kelas Operasi

Kelas ini merupakan representasi proses-proses produksi yang akan dilalui i sebuah tugas. Kelas operasi ini merupakan kelas utama yang akan mengalami njadwalan. Setelah melewati proses penjadwalan, atribut waktu mulai, waktu esai, lama proses, status dari kelas operasi ini akan diubah oleh proses njadwalan. Seperti telah dijelaskan sebelumnya bahwa diagram gantt hasil njadwalan menampilkan waktu mulai proses, lama proses,waktu selesai proses buah operasi. Kelas operasi ini ber-*stereotype* entity. Diagram kelas yang nggambarkan kelas operasi terdapat pada gambar 3-36. Melalui diagram sebut dapat dengan mudah mengetahui atribut, fungsi/*method*, dan *pendencies* yang dimiliki kelas operasi.



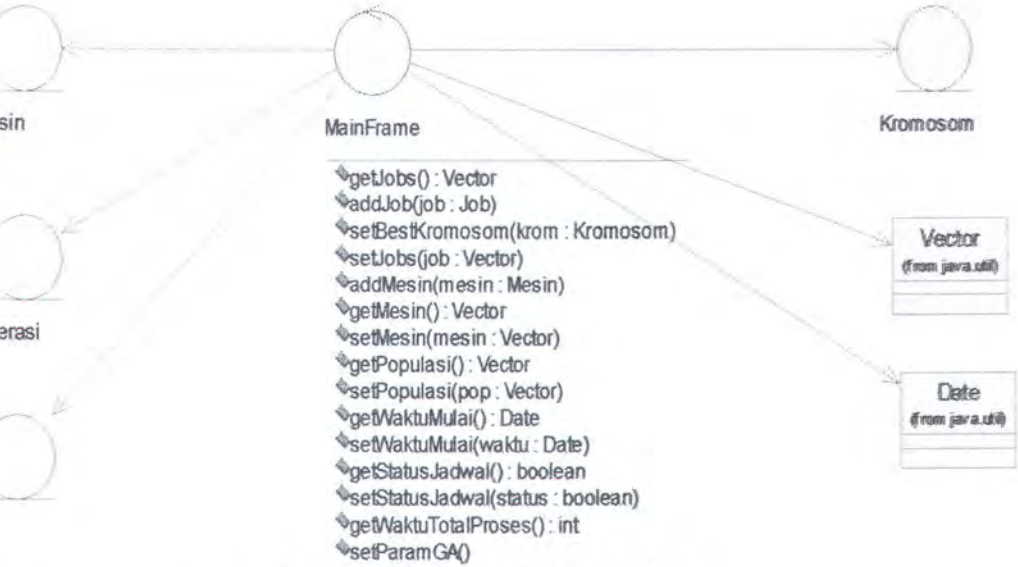
3.4 Kelas Kromosom

Kromosom ini merupakan kelas hasil proses *encoding* dari tugas yang asuk. Kelas ini mempunyai atribut untuk menyimpan nilai *fitness* yang didapat ri perhitungan fungsi evaluasi. Kelas ini juga memiliki method *tJumlahGenBertype(Object)* yang berfungsi untuk mendapatkan jumlah gen *ject* yang dikandung kromosom Kelas Kromosom digambarkan melalui agram kelas seperti pada gambar 3-37 . Pada gambar tersebut tertera juga ibut dan method yang dimiliki Kromosom.



Gambar 3-37 diagram kelas Kromosom

3.5 Kelas MainFrame

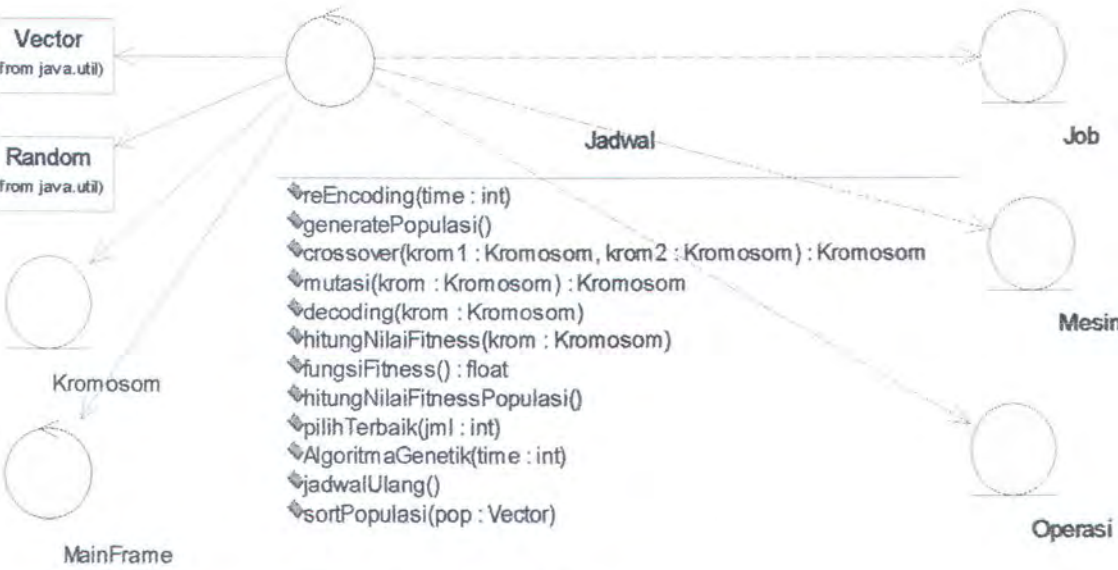


Gambar 3-38 diagram kelas MainFrame

.6 Kelas Jadwal

Jadwal ini merupakan kelas dengan *stereotype control.* Kelas ini nyimpan seluruh prosedur atau fungsi untuk melakukan penjadwalan beserta ngsi – fungsi algoritma genetik. Kelas jadwal ini bertanggung jawab mengatur gala aktifitas yang berkaitan dengan penjadwalan dan juga algoritma genetik. tika pengguna menjalankan perintah penjadwalan maka sistem akan manggil fungsi *jadwalUlang()* yang berada pada kelas ini. Struktur kelas dwal dapat digambarkan melalui diagram kelas. Diagram kelas yang nggambarkan kelas Jadwal ini digambarkan pada gambar 3-39.

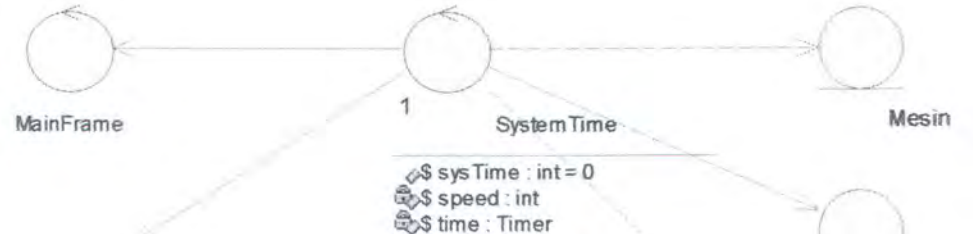
Kelas Jadwal menyimpan fungsi yang berkenan dengan algoritma genetik. ngsi algortma genetik yang terdapat pada kelas ini meliputi, *reEncoding()* yang



Gambar 3-39 diagram kelas jadwal

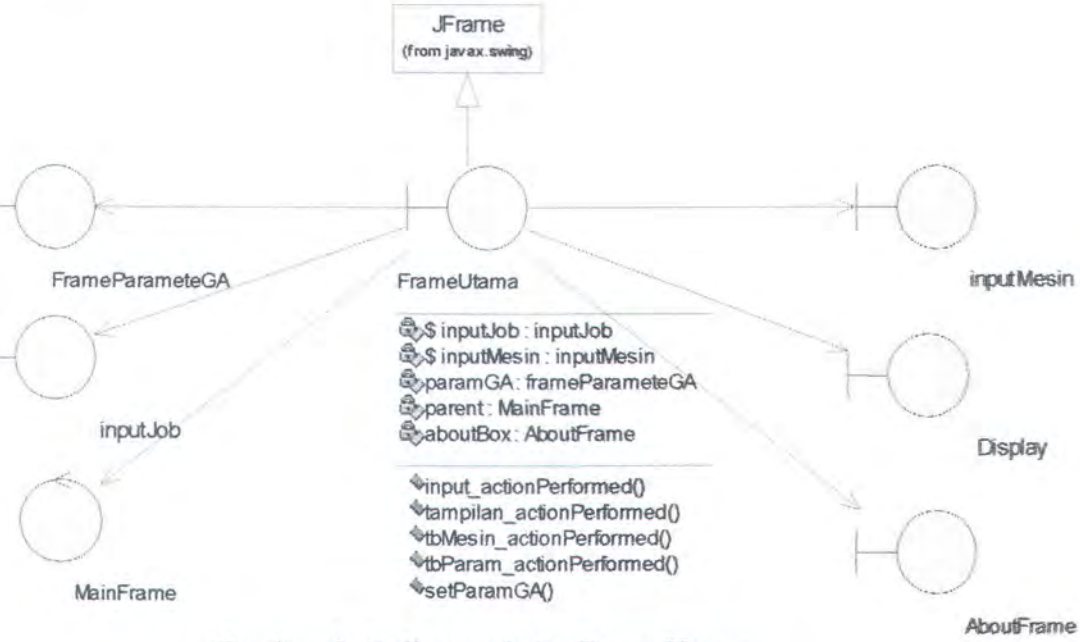
3.7 Kelas System Time

Seperti halnya pada kelas jadwal, System Time ini mengatur kinaerja tem waktu perangkat lunak. Kelas ini memiliki atribut dan fungsi untuk ministrasi waktu sistem. Ketika pengguna menjalankan atau menghentikan ktu sistem maka sistem akan memanggil fungsi yang ada di kelas SystemTime . Stuktur kelas SystemTime digambarkan pada gambar 3-40.



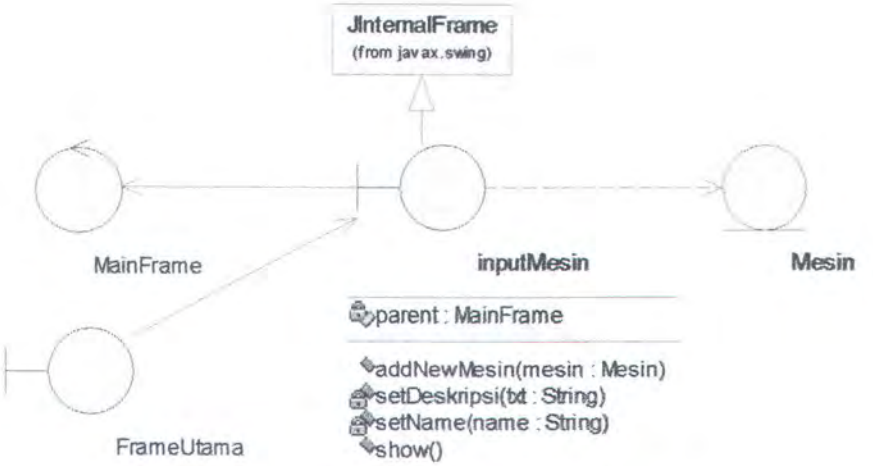
3.8 Kelas Frame Utama

FrameUtama ini merupakan antarmuka pertama yang muncul ketika likasi dijalankan. Melalui FrameUtama ini pengguna dapat menjalankan fungsi–ngsi yang lain seperti entry tugas, entry mesin dan sebagainya. Frame Utama ini bagai antarmuka yang memfasilitasi pengguna untuk berinteraksi dengan tem. Diagram kelas yang menggambarkan kelas FrameUtama ini digambarkan da gambar 3-41.



Gambar 3-41 diagram kelas Frame Utama

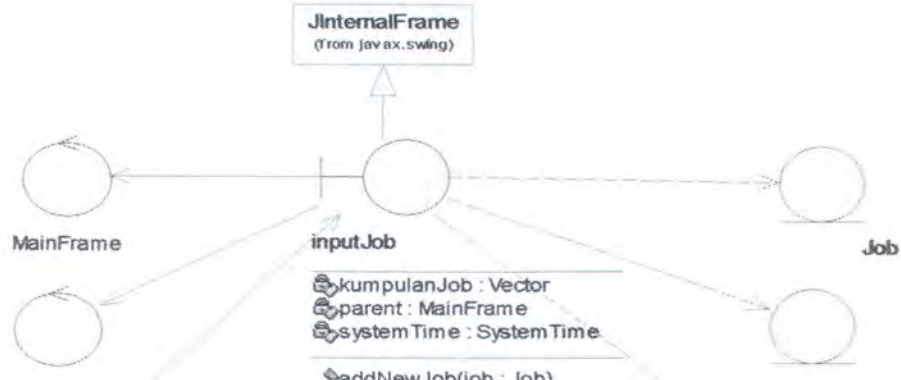
3.9 Kelas Entry Mesin



Gambar 3-42 diagram kelas Input Mesin

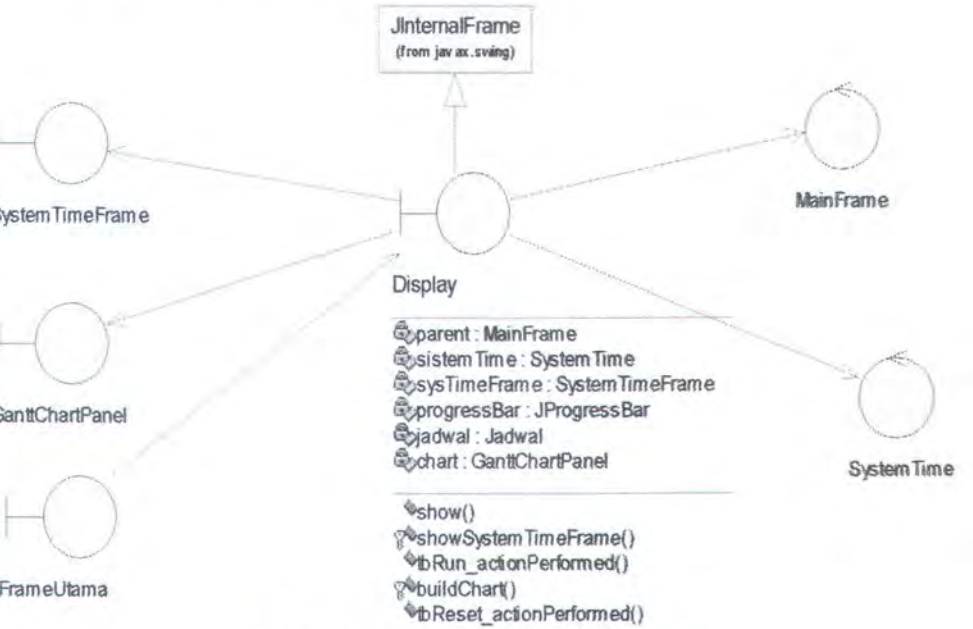
3.10 Kelas Input Job

Seperti halnya pada entry mesin, entry job merupakan antarmuka yang emfasilitasi pengguna untuk memasukkan tugas baru yang akan diproses. elaui antarmuka ini pengguna dapat mengatur urutan mesin untuk proses beserta na waktu proses. Diagram kelas InputJob digambarkan pada gambar 3-43.



.11 Kelas Display Penjadwalan

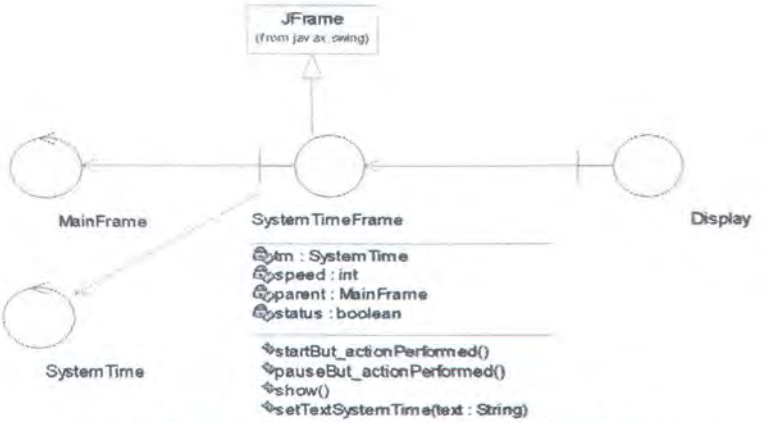
Antarmuka ini merupakan output hasil penjadwalan beserta beberapa gsi untuk pengaturan waktu sistem, menjalankan penjadwalan dan fungsi yang nya. Antarmuka ini memuat beberapa kelas untuk mendukung sistem njadwalan. Sehingga melalui antarmuka ini pengguna mendapatkan fasilitas adukung tanpa harus membuka antarmuka yang lain. Diagram kelas Display njadwalan digambarkan pada gambar 3-44.



Gambar 3-44 diagram kelas Display

.12 Kelas Frame System Time

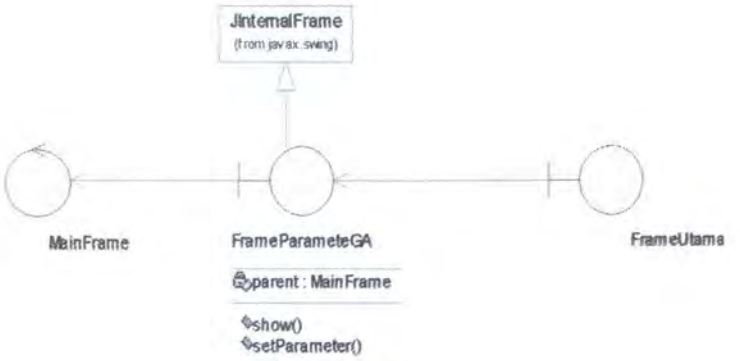
Antarmuka ini merupakan fasilitas bagi pengguna untuk administrasi

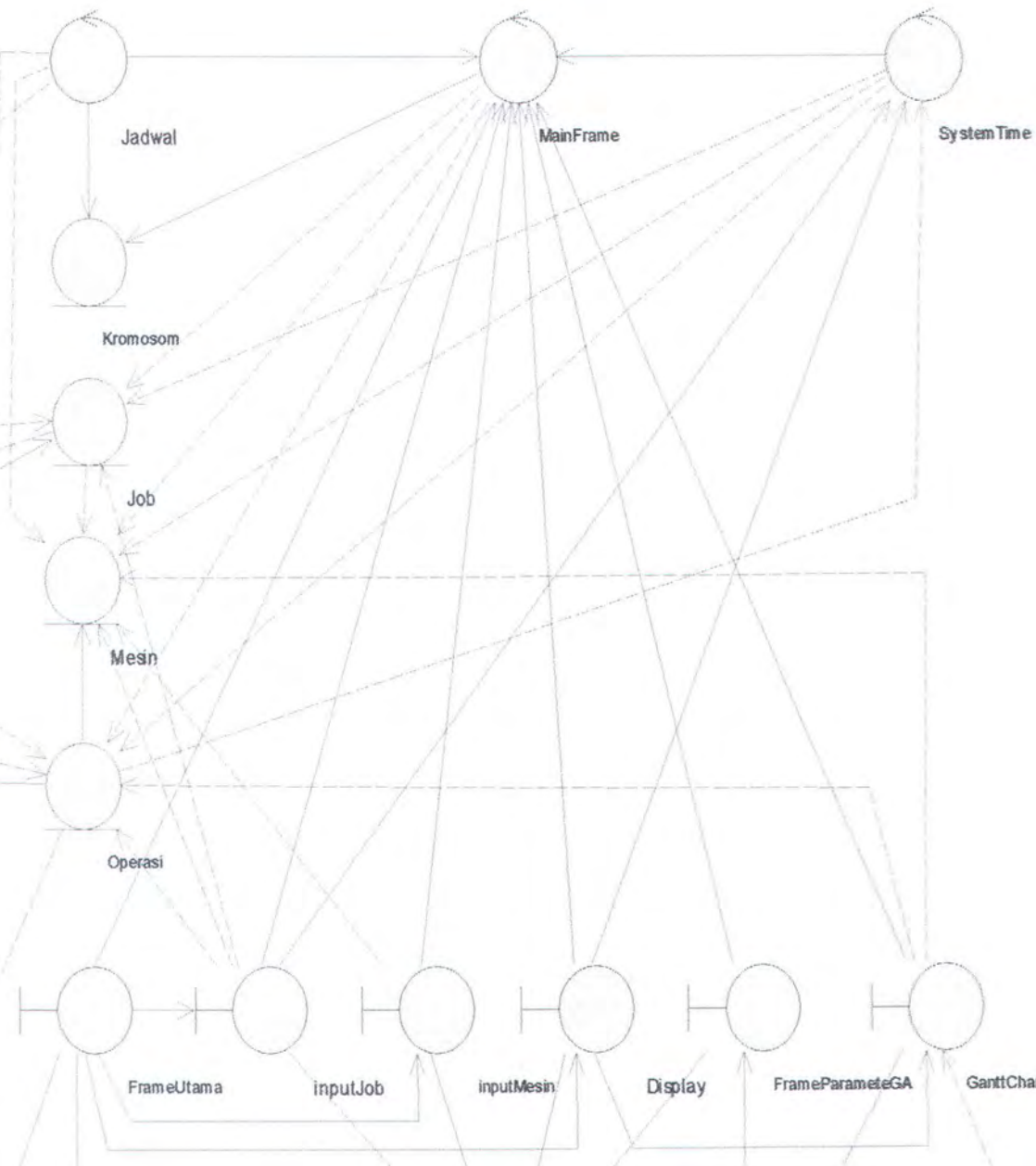


Gambar 3-45 diagram kelas SystemTimeFrame

.13 Kelas Frame Parameter Genetik

Antarmuka ini memfasilitasi pengguna untuk melakukan perubahan hadap parameter algoritma genetik. Parameter algoritma genetik ini tersimpan am kelas lain yaitu kelas MainFrame. Antarmuka ini dipanggil melalui kelas meUtama. Diagram kelas Frame Parameter GA digambarkan pada gambar 6.

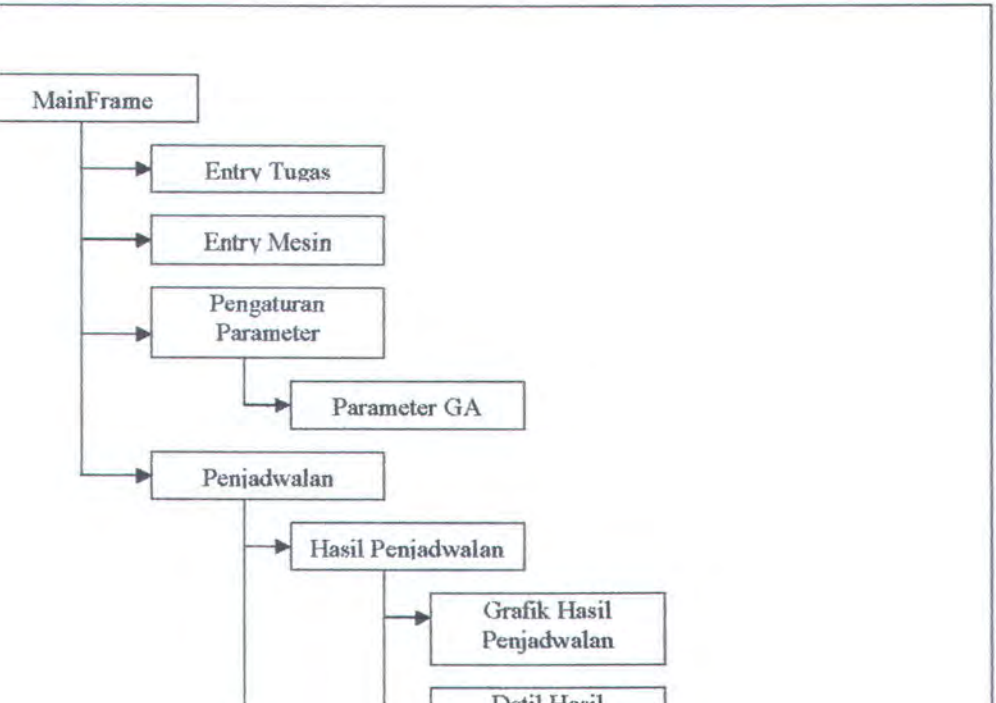



Jadwal
MainFrame
SystemTime
Kromosom
Job
Mesin
Operasi
FrameUtama
inputJob
inputMesin
Display
FrameParameteGA
GanttCha

Perancangan Antarmuka

Pada bagian ini akan dijelaskan mengenai struktur dan dan perencanaan tarmuka dari perangkat lunak ini. Antarmuka merupakan bagian yang sangat nting dalam penggunaan perangkat lunak. Antarmuka yang baik akan mudahkan pengguna untuk berinteraksi dengan sistem atau perangkat lunak.

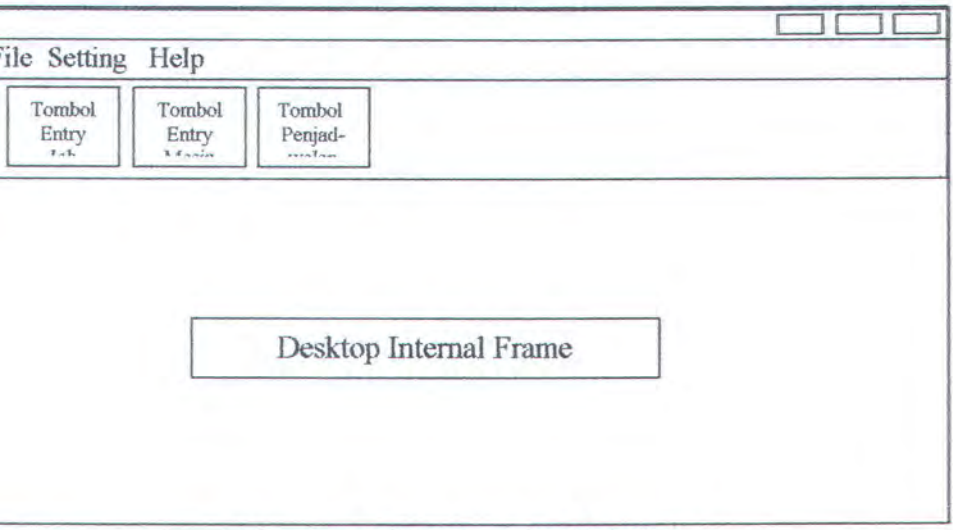
Perangkat lunak / sistem penjadwalan ini mempunyai sebuah struktur atau ran proses tertentu yang akan memudahkan pengguna untuk berinteraksi. cara umum rancangan struktur antarmuka sistem penjadwalan ini digambarkan da gambar 3-48.



Gambar 3-48 menjelaskan mengenai struktur perangkat lunak secara seluruhan. Untuk mengetahui secara detil mengenai tampilan antarmuka rangkat lunak akan dijelaskan berikut ini.

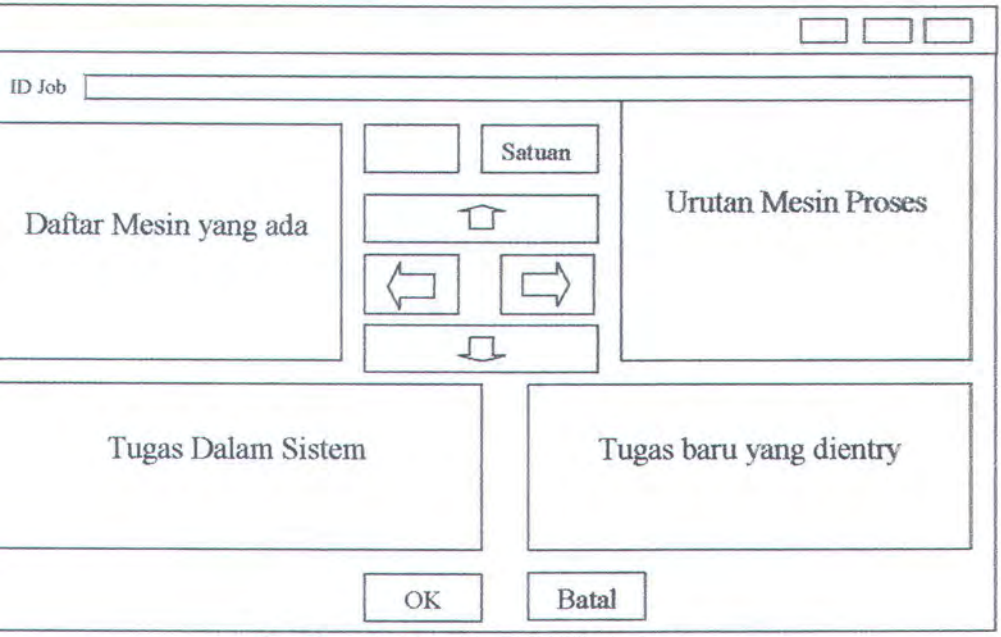
4.1 Main Frame

Antarmuka MainFrame ini menjadi awalan dan tampilan utama dari seluruhan sistem. Dari antarmuka ini terdapat beberapa tombol untuk membuka tarmuka yang lainnya, seperti entry tugas, entry mesin, dll. MainFrame ini ain sebagai antarmuka awal tetapi juga, secara pemrograman, berupa data yang nyimpan seluruh parameter – parameter yang diperlukan oleh sistem. Diagram mpilan antar muka mainFrame digambarkan pada gambar 3-49.



Gambar 3-49 antarmuka MainFrame

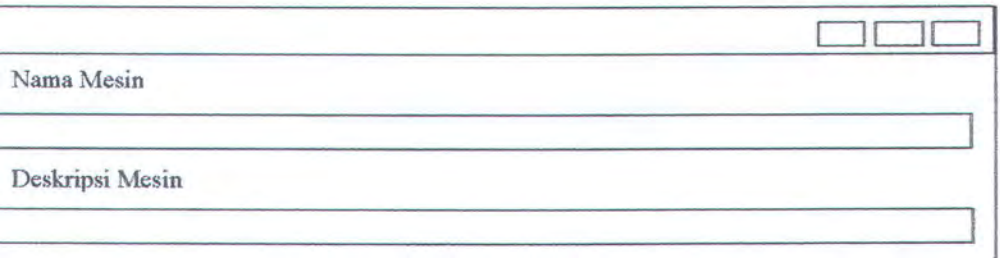
4.2 Entry Tugas



Gambar 3-50 antarmuka entry tugas

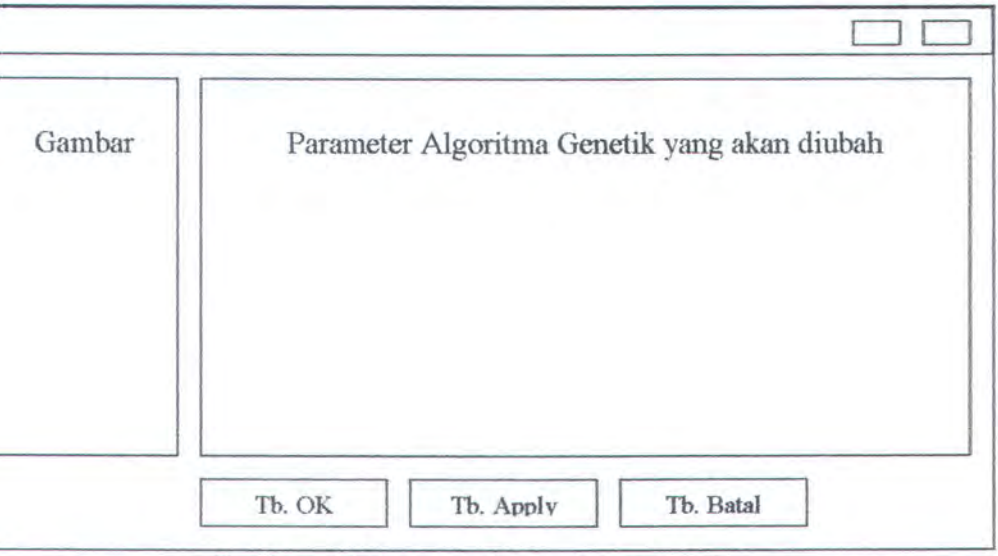
4.3 Entry Mesin

Selain dapat memasukkan tugas baru, pengguna juga dapat memasukkan esin yang digunakan untuk proses produksi sesuai dengan keinginan. Mesin ini ntinya berupa sumberdaya tempat dimana tugas yang masuk dialokasikan. agram antarmuka entry mesin digambarkan pada gambar 3-51 .

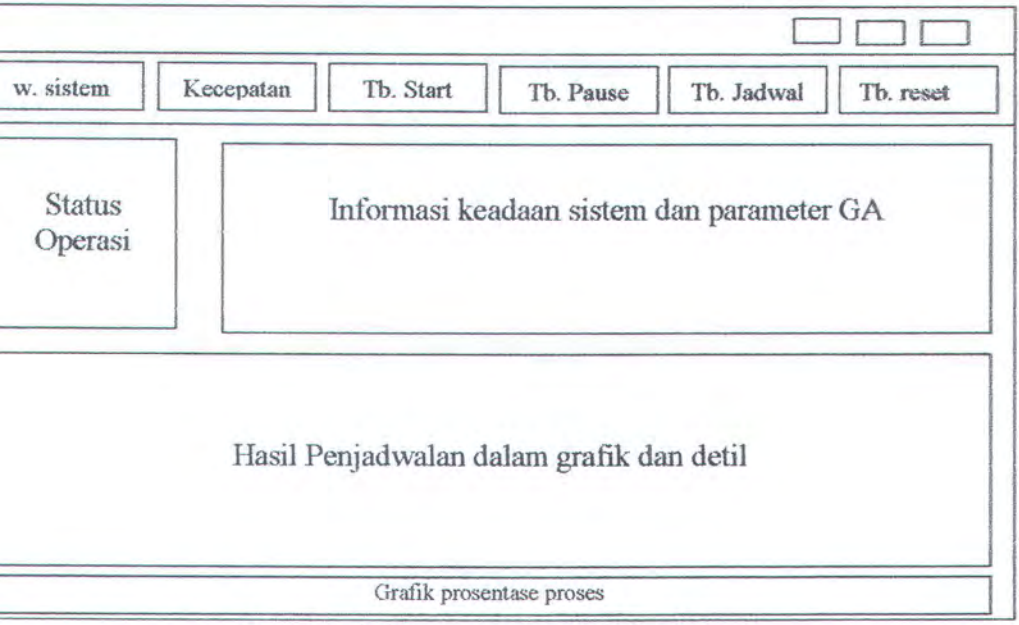


.4 Setting Parameter Algoritma Genetik

Anatarmuka ini merupakan fasilitas bagi pengguna unutk melakukan ubahan parameter algoritma genetik. Melalui anatarmuka ini pengguna dapat ngubah crossover rate, mutation rate, jumlah populasi dan sebagainya. rameter algoritma genetik akan segera berubah setelah pengguna melakukan mit atau mengklik tombol OK atau tombol Apply. Apabila pengguna mbatalkan perubahan parameter pengguna dapat mengklik tombol batal. agram atau layout antarmuka setting parameter digambarkan pada gambar 3-52.



Gambar 3-52 antarmuka setting parameter



Gambar 3-53 antarmuka penjadwalan

BAB IV

IPLEMENTASI PERANGKAT LUNAK

BAB 4
IMPLEMENTASI PERANGKAT LUNAK

Dalam bab ini akan dijelaskan mengenai bagaimana implementasi rangkat lunak pada tugas akhir ini. Pembuatan perangkat lunak ini mengacu da perancangan algoritma pada bab sebelumnya. Penjelasan dalam bab ini eliputi masukan data, pengolahan data, dan penyajian hasil keluaran dari sistem.

1 Implementasi Data

Implementasi data yang akan dijelaskan pada bagian ini terdiri dari tiga acam data yaitu implementasi data masukan, data proses, data keluaran.

1.1 Implementasi Data Masukan

Data masukan ini merupakan data yang dientry oleh pengguna, dimana ta inilah yang nantinya akan diproses lebih lanjut untuk menghasilkan solusi. ata yang harus dientry adalah tugas (*job*), operasi, mesin. Implementasi ketiga ta masukan berupa sebuah kelas. Penjelasan implementasi tersebut adalah bagai berikut,

,getDateWaktuSelesai(),unflagOperasiSetelah(),setStatusOperasi()

n beberapa fungsi lain. Implementasi kelas tugas ini adalah sebagai berikut,

```
01)  public class Job extends Object
02)  {
03)        private String NAMAJOB;
04)        private Vector operasi;
05)        private long waktuKedatangan;

06)        public void addOperasi(Operasi ops)
07)        {this.operasi.addElement(ops);}

08)        public Date getDateWaktuKedatangan()
09)        {return new Date(waktuKedatangan);}

10)        public Date getDateWaktuMulai(){
11)       return ((Operasi)operasi.firstElement()).getStartDate();
12)        }

13)        public Date getDateWaktuSelesai(){
14)              long time = getDateWaktuMulai().getTime() +
15)              (getFlowTime()*60000);
16)              return new Date(time);
17)        }

18)  public void UnflagOperasiSesudah(int t)
19)  {
20)        for (int i = 0; i < operasi.size(); i++)
21)        {
22)              Operasi ops=(Operasi)operasi.elementAt(i);
23)              int start =
                 ((Operasi)operasi.elementAt(i)).getStartTime();
24)              int finish =
                 ((Operasi)operasi.elementAt(i)).getFinishTime();
25)              if(ops.getStatus()!=0)
26)              {
27)                    if(start<t&&finish<t)ops.setStatus(3);
28)                    if(start<t&&finish>t)ops.setStatus(2);
29)                    if(start>t&&finish>t)ops.setStatus(1);
30)              }
31)        }
```

.1.2 Mesin

Kelas mesin ini merupakan data yang digunakan untuk mengalokasika gas yang masuk kedalam sistem. Kelas ini memiliki beberapa atribut dan etode yang digunakan untuk mengolah tugas yang masuk. Kelas ini merupakan unan dari object java yang bernama *Object* . Atribut yang dimiliki kelas ini alah nama, antrian, idle, status, deskripsi. Sedangkan fungsi yang dimiliki tara lain, `setName()`, `addAntrian()`, `getAntrianKe()`, `clearAntrian()`, `moveOperasi()`, `getIdle()`, `getStatus()`. Implementasi data Mesin alah sebagai berikut,

```
01)  public class Mesin extends Object
02)  {
03)        public Vector antrian=new Vector();
04)        private int idle=0
05)        private String name;
06)        private boolean status=true;
07)        private String desc;
08)        public void setName(String name){this.name=name;}
09)        public void addAntrian(Operasi operasi)
10)        {
11)           this.antrian.addElement(operasi);
12)           Operasi lstOp=(Operasi)antrian.lastElement();
13)            idle=operasi.getFinishTime();
14)         }
15)         public Operasi getAntrian(int i)
16)         {return (Operasi)antrian.elementAt(i);}
17)         public void clearAntrian()
18)         {antrian.removeAllElements();}
19)         public void removeOperasi(Operasi op)
20)         {
21)              if(antrian.contains(op))
22)              antrian.removeElement(op);
23)         }
24)         public int getIdle()
```

4.1.3 Operasi

kelas ini merupakan representasi proses yang harus dilalui dari sebuah gas. Kelas ini memiliki beberapa atribut yaitu, job, status, urutan, startTime, naProses, waktuSelesai, mesin. Sedangkan operasi yang dimiliki kelas ini alah `getStartDate()`, `getFinishDate()`, `setStatus()`, `getMesin()`. plementasi kelas Operasi ini adalah sebagai berikut,

```
01)  public class Operasi
02)  {
03)        private Job job;
04)        int status=0;
05)        int operasi;
06)        private int startTime=-1;
07)        private int lamaProses;
08)        private int waktuSelesai
09)        private Mesin mesin;

10)        public Date getStartDate()
11)        {
12)          return new date
             (SystemTime.firstStart+getMinuteOfStart());
13)        }

14)        public Date getFinishDate()
15)        {
16)          return
17)       new Date(SystemTime.firstStart+getMinuteOfFinish());
18)        }

19)       public void setStatus(int status)
20)        {this.status = status;}

21)        public Mesin getMesin(){return mesin;}
22)  }
```

Program 4-3 implementasi kelas Operasi

1.2.1 Kromosom

Kelas kromsom ini merupakan hasil *encoding* dari tugas yang masuk untuk engalami proses lebih lanjut . Ada sebuah atribut yang ditambahkan untuk endukung data kromosom, yaitu, *fittnesValue* . Atribut *fittnesValue* menyimpan ai fittnes yang digunakan untuk menilai kromosom pada saat seleksi. Fungsi ng dimiliki data ini adalah `getFitnessValue()`, `getJumlahGenBertype()`, `intIsiKromosom()`. Implementasi kelas kromosom ini adalah sebagai berikut,

```
01)  public class Kromosom extends Vector
02)  {
03)     private float fittnesValue;

04)     public Kromosom()
05)     {super();}

06)     public float getFittnesValue(){return fittnesValue;}

07)     public int getJmlGenBertype(Object gen)
08)     {
09)        int count=0;
10)        for (int i = 0; i < size(); i++)
11)        {
12)          if(elementAt(i).equals(gen))count++;
13)        }
14)        return count;
15)     }

16)     public void printIsiKromosom()
17)     {
18)        for (int i = 0; i < size(); i++)
19)        {
20)  System.out.print(((Job)elementAt(i)).getName()+";");
21)        }
22)        System.out.println("");
23)     }
24)  }
```

belumnya bahwa seluruh fungsi yang memerlukan data waktu harus merujuk pada kelas SystemTime ini.

Untuk menjalankan waktu sistem maka perlu dibuat sebuah objek yang erupakan proses *background* (*background process*), sehingga data ini dapat rjalan bersamaan dengan keseluruhan sistem. Dapat dikatakan bahwa data ini pat berjalan sendiri tanpa mempengaruhi jalannya sistem. Oleh sebab itu data tem waktu ini mengimplementasikan sebuah *instance* dari objek vax.swing.Timer. Objek ini menjalankan sebuah fungsi tertentu dengan ktu tenggang tertentu, dimana waktu tenggang tersebut dapat diatur. Dengan nggunakan objek ini maka dapat dengan mudah mengatur kecepatan *refresh*, njalankan dan menghentikan waktu. Implementasi instance objek vax.swing.Timer dapat dilihat pada Program 4-5 berikut ini.

```
1)   Timer time = new Timer(speed,new
     java.awt.event.ActionListener())
2)   {
3)     public void actionPerformed(ActionEvent evt)
4)     {
5)        sysTime++;
6)        jobListener();
7)        parent.startTabelStatus=true
8)     }
9)   }
0)   );
```

Program 4-5*instance* javax.swing.Timer

Dalam kelas sistem waktu terdapat beberapa fungsi yang digunakan dalam ngatur waktu sistem. Fungsi yang digunakan untuk menjalankan waktu adalah

eed). Implementasi fungsi `StartSystemTime(int speed)` terdapat pada ogram 4-6.

Sedangkan fungsi yang akan dijalankan oleh *instance* time adalah bListener (). Dimana fungsi `jobListener()` bertugas untuk selalu engupdate status seluruh job yang ada dalam sistem. Implementasi fungsi `bListener()` terdapat pada Program 4.7.

```
01)   public void StartSystemTime(int speed)
02)   {
03)     if(!hasStart)
04)     {
05)         firstStart=getDesktopTime();
06)         hasStart=true;
07)     }
08)     time=new Timer(speed,new java.awt.event.ActionListener()
09)     {
10)       public void actionPerformed(ActionEvent evt)
11)       {
12)         sysTime++;
13)         jobListener();
14)         parent.startTabelStatus=true; }
15)      });
16)   time.start();
17)   }
```

Program 4-6 fungsi `StartSystemTime`

```
01)   public void jobListener(){
02)     Vector jobs=parent.getJobs();
03)     for (int i = 0; i < jobs.size(); i++){
04)       Job j=(Job)jobs.elementAt(i);
05)       for (int o = 0; o < j.getJumlahProses(); o++){
06)         Operasi op=j.getOperasiKe(o);
07)         Mesin m=op.getMesin();
08)         int finis=op.getFinishTime();
```

Menghentikan Waktu Sistem

Seperti telah dijelaskan di atas bahwa data SystemTime juga mempunyai ngsi / *method* yang menangani penghentian waktu. Fungsi tersebut adalah useSystemTime(). Fungsi tersebut diimplementasikan sebagai berikut,

```
01)   public void pauseSystemTime()
02)   {
03)      if(time!=null&&time.isRunning())
04)      {
05)        this.time.stop();
06)        parent.startTabelStatus=false;
07)       }
08)   }
```

Program 4-8 fungsi pauseSystemTime

Fungsi pauseSystemTime() ini memanggil fungsi stop() yang dimiliki eh *instance* time. Selain itu sistem ini juga mengubah beberapa parameter tem. Sehingga untuk menghentikan waktu sistem dan mengubah beberapa rameter hanya dengan memanggil fungsi ini.

Mengatur Kecepatan Waktu Sistem

Data sistem waktu ini selain berfungsi untuk menjalankan dan enghentikan waktu juga dapat digunakan untuk mengatur kecepatan waktu. ngaturan ini ditujukan untuk mensimulasikan waktu dan mengatur kecepatan suai dengan keinginan. Implementasi fungsi pengatur waktu terdapat pada ogram 4-9.

Selain fungsi yang telah dijelaskan di atas data SystemTime ini juga enyimpan informasi mengenai waktu, baik waktu yang dimiliki desktop mputer atau waktu yang dimiliki oleh sistem penjadwalan.

1.3 Implementasi Data Keluaran

Data keluaran yang dihasilkan perangkat lunak ini berupa grafik yang enggambarkan isi antrian yang dimiliki tiap mesin. Antrian ini berupa operasi ng telah dan akan diproses oleh mesin. Selain itu grafik ini juga enggambarkan kapan sebuah operasi mulai diproses dan kapan waktu lesainya. Data keluaran ini berupa kumpulan mesin yang ada dalam sistem dan buah kelas untuk menggambarkan isi antrian mesin.

1.3.1 Kumpulan Mesin

Kumpulan mesin ini merupakan data yang dimiliki oleh kelas MainFrame ng digunakan pada keseluruhan sistem. Data mesin ini berisi mesin – mesin ng ada dalam sistem Implementasi data ini terdapat dalam kelas MainFrame. plementasinya adalah sebagai berikut,

```
private static Vector mesin=new Vector();
```

1.3.2 Diagram Gantt

Untuk menggambarkan diagram gantt dilayar maka perangkat lunak ini

```
public class GanttChartPanel extends JPanel
{
   BorderLayout borderLayout1 = new BorderLayout();
   MainFrame parent;
   XYDataset dataset = null;
   JFreeChart chart = null;
   ChartPanel chartPanel =null;

  public GanttChartPanel(MainFrame prt)
  {
     try
     {
       this.parent=prt;
       init();
     }
     catch(Exception ex){ex.printStackTrace();}
   }
  void init() throws Exception
  {
     this.setLayout(borderLayout1);
     XYDataset dataset = createDataset();
     JFreeChart chart = createChart(dataset);
     ChartPanel chartPanel = new ChartPanel(chart, false);
     chartPanel.repaint();
     chartPanel.setMouseZoomable(true, false);
     this.add(chartPanel);
  }
  private XYDataset createDataset()
  {
     Vector mesin=parent.getMesin();
     TimeSeriesCollection dataset = new TimeSeriesCollection();
     dataset.removeAllSeries();
     if(mesin.size()>0){
       for (int i = 0; i < mesin.size(); i++)
       {
          Mesin m=(Mesin)mesin.elementAt(i);
          if(m.antrian.size()>0){
            for (int a = 0; a <m.antrian.size(); a++)
            {
              Operasi op=m.getOperasi(a);
```

```
}  private static JFreeChart createChart(XYDataset dataset)
}  {
}      JFreeChart chart = createTimeSeriesChart(
}          "Jadwal Mesin",  // title
}          "Waktu",              // x-axis label
}          "Mesin",    // y-axis label
}          dataset,             // data
}          true,                // create legend?
}          true,                // generate tooltips?
}          true                // generate URLs?
}      );
}      XYPlot plot = (XYPlot) chart.getPlot();
}      plot.setBackgroundPaint(Color.lightGray);
}      plot.setDomainGridlinePaint(Color.white);
}      plot.setRangeGridlinePaint(Color.white);
}      plot.setDomainCrosshairVisible(true);
}      plot.setRangeCrosshairVisible(true);
}      plot.setDomainCrosshairValue(10.0);
}      XYItemRenderer r = plot.getRenderer();
}      DateAxis axis = (DateAxis) plot.getDomainAxis();
}      chart.fireChartChanged();
}      return chart;
}    }
```

Program 4-10 Implementasi kelas GanttChartPanel (lanj)

2 Implementasi Proses

Pada bagian ini akan dijelaskan mengenai implementasi proses yang jadi pada waktu eksekusi perangkat lunak ini.

2.1 Entry Mesin

Implementasi proses ini melibatkan beberapa kelas atau objek. Seperti ah digambarkan dalam diagram kolaborasi (gambar 3-4) pada bab 3

belumnya atau jumlah antar muka InputMesin sama dengan 0 . Sebaliknya abila objek *inputMesin* telah diciptakan sebelumnya fungsi ini hanya akan emanggil *method show()*, untuk menampilkan frame entry mesin. Implementasi ngsi *tbMesin_actionPerformed()* dijelaskan pada Program 4-11 .

```
01)   void tbMesin_actionPerformed(ActionEvent e)
02)   {
03)      if(inputMesin==null||numInputMesin==0)
04)      {
05)        try
06)        {
07)           inputMesin=new InputMesin(param);
08)           desktop.add(inputMesin);
09)           inputMesin.setVisible(true);
10)           inputMesin.setSelected(true);
11)           inputMesin.toFront();
12)            numInputMesin ++;
13)        }
14)        catch(Exception ex){ex.printStackTrace();}
15)      }
16)      else if(numInputMesin>0)
17)      {
18)        try
19)        {
20)         inputMesin.show();
21)         inputMesin.setSelected(true);
22)        }
23)        catch (Exception e1){e1.printStackTrace();}
24)      }
25)   }
```

Program 4-11 implementasi *method* tombol entry mesin

Setelah *instance* inputMesin terbentuk, maka untuk pertama kali *instance* akan menciptakan sebuah *instance parent* dimana *instance* ini mereferensi as MainFrame. Objek *parent* ini digunakan oleh kelas InputMesin untuk

mpulan mesin yang telah ada pada MainFrame. Implementasi kelas InputMesin elaskan pada Program 4-13.

```
1)  public class InputMesin extends JInternalFrame
2)  {
3)     MainFrame parent;
4)     public InputMesin(MainFrame prt){
5)        try
6)        {
7)          parent=prt;
8)          jbInit();
9)        }
0)        catch(Exception ex){ex.printStackTrace();}
1)     }
2)     void jbInit() throws Exception{
3)        tombolOk.addActionListener(
4)        new java.awt.event.ActionListener()
5)        {
6)            public void actionPerformed(ActionEvent e)
7)            {tombolOk_actionPerformed(e);}
8)        });
9)        tombolSubmit.addActionListener(
0)        new java.awt.event.ActionListener()
1)        {
2)           public void actionPerformed(ActionEvent e)
3)             {tombolSubmit_actionPerformed(e);}
4)         });
5)        listMesin.addListSelectionListener(
6)        new javax.swing.event.ListSelectionListener() {
7)        public void valueChanged(ListSelectionEvent e)
8)          {listMesin_valueChanged(e);}
9)        });
0)   }
1)   private Vector getNamaMesin(){
2)      Vector m=parent.getMesin();
3)      Vector data=new Vector();
4)      for (int i = 0; i < m.size(); i++)
5)      data.addElement(((Mesin)m.elementAt(i)).getName());
6)      return data;
7)   }
8)   void tombolSubmit_actionPerformed(ActionEvent e){
```

```
47          else if(txNama.getText().length()<1)
48          txNama.setText("");
49          txDesc.setText("");
50      }
51      void tombolOk_actionPerformed(ActionEvent e)
52      {
53            this.doDefaultCloseAction();
54        }
55      void listMesin_valueChanged(ListSelectionEvent e)
56      {
57      int indx=listMesin.getSelectedIndex();
58      if(indx>-1)
59      {
60          Mesin m=(Mesin)parent.getMesin().elementAt(indx);
61          txJobDesc.setText(m.getDesc());
62      }
63      }
64  }
```

Program 4-12 implementasi kelas InputMesin (lanjutan)

2.2 Entry Tugas

Proses entry tugas awal ini dilakukan oleh pengguna untuk memasukkan gas yang akan dijadwalkan. Proses ini melibatkan beberapa kelas, yaitu kelas ameUtama, MainFrame, InputJob.

Untuk dapat memasukkan tugas baru pengguna harus mengklik tombol try tugas dari antarmuka FrameUtama. Tombol ini akan menjalankan fungsi *out_actionPerformed*() , dimana fungsi tersebut akan menciptakan *instance* nput baru yang mereferesi ke kelas InputJob apabila *instance* ini belum pernah

```
)1)   void input_actionPerformed()
)2)   {
)3)     if (numInput==0)
)4)     {
)5)       try {
)6)        Jinput=new InputJob(param);
)7)        Jinput.setVisible(true);
)8)        Jinput.setSelected(true);
)9)        Jinput.toFront();
.0)        desktop.add(Jinput);
.1)        numInput++;
.2)        }
.3)       catch (Exception ie) {}
.4)     }
.5)     else if(numInput>0)
.6)     {
.7)       try{
.8)       Jinput.show();
.9)       Jinput.setSelected(true);
:0)       }
:1)       catch (Exception e1){e1.printStackTrace();}
:2)      }
:3)   }
```

Program 4-13 implementasi fungsi tombol entry tugas

Setelah instance Jinput yang mereferensi kelas MainFrame terbentuk, ıka untuk pertama kali JInput ini akan membentuk *instance parent* yang ereferensi kepada kelas MainFrame. Dimana *parent* ini nantinya akan gunakan oleh kelas InputJob untuk mendapatkan parameter sistem yang erlukan, seperti kumpulan job dan mesin yang ada. Melalui antarmuka utTugas pengguna diminta memasukkan nama tugas dan urutan mesin yang an digunakan untuk proses. Pengguna juga diminta memasukkan lama waktu ses dari masing-masing operasi. Dalam implementasinya, fungsi yang

```
01)   public class InputJob extends JInternalFrame
02)   {
03)      private MainFrame parent;
04)      private SystemTime systemTime;
05)      private Vector kumpulanJob;
06)      private Vector urutanOperasi=new Vector();
07)      public InputJob(MainFrame prt)
08)      {
09)        try
10)        {
11)          parent=prt;
12)          systemTime=new SystemTime(parent);
13)          kupulanJob=prt.fixJob.clone();
14)          jbInit();
15)        }
16)        catch(Exception e){e.printStackTrace();}
17)      }
18)      private void jbInit() throws Exception
19)      {
20)         tbOk.addActionListener(
21)         new java.awt.event.ActionListener()
22)         {
23)             public void actionPerformed(ActionEvent e)
24)             {tbOk_actionPerformed(e);}
25)         });
26)         tbSubmit.addActionListener(
27)         new java.awt.event.ActionListener()
28)         {
29)             public void actionPerformed(ActionEvent e)
30)             {tbSubmit_actionPerformed(e);}
31)         });
32)      }
33)      void tbSubmit_actionPerformed(ActionEvent e)
34)      {
35)         if(urutanMesin.size()!=0)
36)         {
37)            String id=txIdJob.getText();
38)            kumpulanJob.addElement(new
   Job(id,(Vector)urutanOperasi.clone(),1,0,systemTime.getSysDa
   te().getTime()));
39)         }
40)         else JOptionPane.showConfirmDialog()
41)      }
```

2.3 Mengatur Parameter Algoritma Genetik

Proses ini dilakukan untuk mengatur parameter algoritma agar didapatkan sil yang lebih optimal. Pengguna dapat mengubah nilai parameter algoritma netik melalui antarmuka FrameParamGA. Proses ini melibatkan beberapa kelas itu FrameUtama, FrameParamGA dan MainFrame.

Untuk memulai proses ini pengguna harus membuka antarmuka ameUtama melalui menu setting. Implementasi fungsi *rameterGA_actionPerformed()* untuk membuka antarmuka setting parameter goritma genetik terdapat pada program 4-15. Fungsi *rameterGA_actionPerformed()* akan menciptakan *instance* baru dengan nama ramGA yang mereferensi kelas FrameParamGA apabila *instance* ini belum bentuk sebelumnya. Apabila instance paramGA telah terbentuk maka fungsi an memanggil fungsi *show()* yang dimiliki FrameParamGA.

```
01)  void parameterGA_actionPerformed()
02)  {
03)     try{
04)       paramGA=new FrameParamGA(param);
05)       desktop.add(paramGA);
06)       paramGA.setVisible(true);
07)       paramGA.setSelected(true);
08)       paramGA.toFront();
09)     }
10)     catch(Exception e){e.printStackTrace();}
11)  }
```

Program 4-15 implementasi fungsi membuka antarmuka setting Parameter

```
01)  public class FrameParamGA extends JInternalFrame{
02)     private MainFrame parent;
03)     public FrameParamGA(MainFrame prt) {
04)      try{
05)        parent=prt;
06)        jbInit();
07)      }
08)     catch(Exception e){e.printStackTrace();}
09)    }
10)    private void jbInit() throws Exception{
11)       loadParam();
12)       tbBatal.addActionListener(
13)       new java.awt.event.ActionListener() {
14)          public void actionPerformed(ActionEvent e)
15)          {tbBatal_actionPerformed(e);}
16)       });
17)      tbApply.addActionListener(
18)      new java.awt.event.ActionListener() {
19)         public void actionPerformed(ActionEvent e)
20)         {tbApply_actionPerformed();}
21)      });
22)      tbOK.addActionListener(
23)      new java.awt.event.ActionListener() {
24)         public void actionPerformed(ActionEvent e)
25)         {tbOK_actionPerformed();}
26)      });
27)    }
28)  void tbOK_actionPerformed(){
29)     tbApply_actionPerformed() ;
30)     doDefaultCloseAction();
31)  }
32)  private void loadParam(){
33)      txJmlIterasi.setText(
34)      String.valueOf(parent.jmlIterasi));
35)      txLajuCross.setText(
36)      String.valueOf(parent.crossoverRate*100));
37)      txLajuMutasi.setText(
38)      String.valueOf(parent.mutationRate*100));
39)      txLajuSeleksi.setText(
40)      String.valueOf(parent.selectionRate*100));
41)  }
42)  void tbApply_actionPerformed(){
43)     try  {
```

2.4 Menjalankan Waktu Sistem

Proses ini ditujukan untuk menjalankan waktu sistem. Secara *default*, aktu sistem berstatus berhenti atau tidak berjalan. Sebelum melakukan njadwalan pengguna harus menjalankan waktu sistem yang ditangani oleh kelas stemTime. Karena seluruh fungsi dalam sistem yang membutuhkan data waktu, engacu kelas SystemTime ini.

Untuk menjalankan waktu sistem pengguna harus membuka antarmuka splay melalui FrameUtama. Antarmuka Display akan memanggil stemTimeFrame yang memfasilitasi pengguna untuk dapat mengatur waktu tem. SystemTimeFrame ini akan mengubah parameter dan menjalankan fungsi ng terdapat pada kelas SystemTime. Implementasi proses untuk membuka tamuka Display terdapat pada program 4-17.

```
01)   void tampilan_actionPerformed()
02)   {
03)      try
04)      {
05)          display=new Display(param);
06)          desktop.add(display);
07)          display.setVisible(true);
08)          display.setSelected(true);
09)          display.toFront();
10)          numDisplay++;
11)       }
12)      catch(Exception ex){ex.printStackTrace();}
13)   }
```

Program 4-17 implementasi fungsi membuka antarmuka Display

```
01)  public class Display extends JInternalFrame{
02)      SystemTimeFrame sysTime;
03)      JMenuBar menuBar = new JMenuBar();
04)      public Display(MainFrame prn)
05)      {
06)       try{
07)        parent=prn;
08)        sysTime=new SystemTimeFrame(prn);
09)        menuBar.add(sysTime);
10)      }catch(Exception e){e.printStackTrace();}
11)  }}
```

Program 4-18 implementasi proses membuka antarmuka SystemTimeFrame

Setelah memanggil antarmuka SystemTimeFrame, kelas Display akan eletakannya pada toolbar sebagai *utility*. Implementasi kelas SystemTimeFrame alah sebagai berikut,

```
01)  public class SystemTimeFrame extends JPanel{
02)      private SystemTime tm;
03)      private MainFrame parent;
04)      public SystemTimeFrame(MainFrame prt){
05)        parent=prt;
06)        jbInit();
07)      }
08)      private void jbInit() throws Exception{
09)        tm=new SystemTime(parent);
10)        bt.addActionListener(
11)        new java.awt.event.ActionListener()
12)        {
13)           public void actionPerformed(ActionEvent e)
14)           {bt_actionPerformed(e);}
15)         });
16)        pauseBut.addActionListener(
17)        new java.awt.event.ActionListener()
18)        {
19)           public void actionPerformed(ActionEvent e)
20)           {pauseBut_actionPerformed(e);}
```

2.5 Menjalankan Penjadwalan

Setelah semua data masukan telah di entry, maka proses selanjutnya alah melakukan penjadwalan terhadap tugas yang masuk. Proses penjadwalan akukan dengan mengalokasikan operasi-operasi tiap tugas yang masuk kedalam esin. Proses penjadwalan melibatkan beberapa kelas, antara lain kelas ainFrame, FrameUtama, Display, Jadwal, SistemTime.

Seperti telah dijelaskan pada bab sebelumnya bahwa proses ini diawali ngan membuka antarmuka display melalui FrameUtama. Proses membuka tarmuka Display ini sama dengan proses membuka antarmuka Display proses enjalankan waktu sistem. Proses penjadwalan ini dapat dimulai hanya setelah aktu sistem dijalankan, proses menjalankan waktu sistem telah dijelaskan belumnya. Setelah waktu sistem berjalan, proses penjadwalan dapat dilakukan.

Proses penjadwalan dilakukan dengan jalan menekan tombol run oleh nggguna. Tombol run ini memanggil fungsi *tbRun_actionPerformed*(). dangkan fungsi *tbRun_actionPerformed*() akan menghentikan waktu sistem enak melalui fungsi *pauseSystemTime()* dari kelas SystemTime. Dan kemudian manggil fungsi *JadwalUlang()* yang berasal dari kelas Jadwal. Setelah proses njadwalan selesai fungsi akan menampilkan hasil penjadwalan melalui fungsi *ildChart()*, fungsi ini akan menampilkan hasil penjadwalan melalui diagram ntt. Implementasi fungsi *tbRun_actionPerformed()* dijelaskan pada progam 4-

Implementasi fungsi *pauseSystemTime()* dari kelas Jadwal dijelaskan pada ogram 4-21 .

```
01)   public void pauseSystemTime()
02)   {
03)     if(time!=null&&time.isRunning())
04)     {
05)       this.time.stop();
06)       this.runStatus=false;
07)       parent.startTabelStatus=false;
08)     }
09)   }
```

Program 4-21 implementasi fungsi *pauseSystemTime()*

Setelah waktu sistem dihentikan untuk sementara, proses selanjutnya alah memanggil fungsi *jadwalUlang()* dari kelas Jadwal. Fungsi *jadwalUlang()* akan menjalankan operasi genetika untuk menghasilkan penjadwalan yang timal. Fungsi *jadwalUlang()* mempunyai parameter masukkan yaitu berupa ıktu penjadwalan baru. Waktu penjadwalan baru ini didapatkan dari waktu tem (kelas SystemTime). Implementasi fungsi jadwalUlang() dari kelas Jadwal elaskan dalam program 4-22.

```
01)   public void JadwalUlang(int t)
02)   {
03)      AlgoritmaGenetik(t);
04)   }
```

Program 4-22 implementasi fungsi *jadwalUlang()*

Setelah hasil penjadwalan didapatkan proses selanjutnya adalah nampilkan hasil penjadwalan dalam diagram gantt melalui fungsi *buildChart()*. plementasi fungsi ini adalah sebagai berikut

2.6 Menjalankan Operasi Genetik

Algoritma genetik merupakan bagian terpenting dalam sistem penjadwalan . Karena untuk melakukan penjadwalan dan mendapatkan hasil penjadwalan ng optimal sistem ini menggunakan algortma genetika. Algoritma genetika nyak digunakan untuk menyelesaikan permasalahan optimasi.

Proses menjalankan operasi genetik ini merupakan bagian dari proses enjalankan penjadwalan. Proses ini dilakukan supaya didapatkan hasil njadwalan yang optimal. Proses menjalankan operasi genetik ini melibatkan berapa kelas yaitu Jadwal, MainFrame, SystemTime.

Seperti telah dijelaskan dalam bab sebelumnya, operasi genetik terdiri dari berapa proses, yaitu *encoding*, *decoding*, seleksi. Implementasi proses-proses sebut adalah sebagai berikut.

2.6.1 Encoding

Dalam algoritma genetika, *encoding* merupakan sebuah proses untuk engubah solusi kedalam kromosom untuk diproses lebih lanjut. Seperti telah elaskan di atas, representasi kromosom akan disusun menurut operasi *peration-based*). Sehingga gen pembentuk kromosom berupa operasi. Tetapi da implemenatasinya yang menyusun kromosom adalah Job. Objek kromosom entuk pada baris 04 program 4-24. Objek ini akan diisi dengan tugas yang suk dalam aplikasi. Proses pengisian kromosom diimplementasikan pada baris – 11 program 4-24. Untuk mendapatkan operasi apa dari gen dilakukan

```
01)   public void encoding()
02)   {
03)      parent.statusJadwal=true;
04)      kromosom=new Kromosom();
05)      for (int i = 0; i < kumpulanJob.size(); i++)
06)      {
07)         Job j=(Job)kumpulanJob.elementAt(i);
08)         for (int a = 0; a <j.getJumlahProses(); a++)
09)         {
10)         kromosom.addElement(((Job)kumpulanJob.elementAt(i)));
11)         }
12)   }
```

Program 4-24 fungsi / method encoding

Jumlah job yang dikandung oleh kromsom sebanyak operasi yang dimiliki b tersebut. Sebagai contoh, apabila terdapat sebuah job ***a*** yang memiliki 3 (tiga) uah operasi (o_{a1}, o_{a2}, o_{a3}) yang berbeda maka kromosom akan mengandung 3 iga) buah job ***a***. Hal ini dilakukan untuk menghindari tetukarnya urutan operasi atu job dalam operasi genetik (tukar silang, mutasi).

2.6.2 Pembangkitan Populasi

Pembangkitan populasi sangat berpegaruh terhadap keluasan daerah solusi. roses ini akan membangkitkan sejumlah ***n*** kromosom sesuai dengan parameter ng diberikan. Semakin besar jumlah populasi maka semakin besar/luas pula aerah solusi.

Implementasi proses pembangkita populasi dilakukan dengan mengacu ada kromosom pertama yang dibentuk dari proses *encoding*, ditunjukkan pada aris 04. Pembangkitan dilakukan dengan mereposisi gen dari kromosom pertama ecara acak, ditunjukkan pada baris 23-35 Sebelum mereposisi kromsom terlebih

```
01)    public void generatePopulasi()
02)    {
03)       boolean sama=true
04)       populasi.addElement(kromosom);
05)       Kromosom tmp =(Kromosom)
   ((Kromosom)populasi.firstElement()).clone();
06)       int kSize=tmp.size();
07)       if (kSize==0)kSize=1;
08)       Job tmp1=null;
09)       Job tmp2=null;
10)       Job tmp3=null;
11)       //mengecek apakah job yang masuk lebih dari satu
12)       for (int i = 0; i < kumpulanJob.size()-1; i++)
13)       {
14)          Job j=(Job)kumpulanJob.elementAt(i);
15)          if(kromosom.contains(j)){sama=false;break;}
16)       }
17)       //apabila job yang masuk lebih dari satu
18)       if(!sama){
19)        for (int i = 0; i <jmlPopulasi-1; i++)
20)        {
21)          tmp=((Kromosom) populasi.firstElement()).clone();
22)          int iRandom= Math.round(tmp.size()* parent.
   initialRandomRate)
23)          for (int n = 0; n <iRandom; n++)
24)          {
25)             int indx1=0;
26)             int indx2=0;
27)             while(indx1==indx2||
28)    tmp.elementAt(indx1).equals(tmp.elementAt(indx2)))
29)             {
30)                 indx1=rand.nextInt(kSize);
31)                 indx2=rand.nextInt(kSize);
32)             }
33)             tmp1=(Job)tmp.remove(indx1);
34)             tmp.insertElementAt(tmp1,indx2);
35)         }
36)         populasi.addElement(tmp);
37)         parent.setPopulasi(populasi);
38)       }
39)       }
40)    //apabila job yang masuk hanya satu
41)      else
```

.2.6.3 Fungsi Evaluasi

Dalam memilih atau menyeleksi kromosom, algoritma genetik akan emilih kromosom yang dinilai efektif dan efisien dalam mencapai tujuan yang iinginkan. Fungsi evaluasi yang digunakan dalam tugas akhir ini adalah rata-rata aktu produksi (*average flow time*). Implementasi fungsi evaluasi ini terdapat ada program 4-26.

```
01)    public float fungsiFittnes()
02)    {
03)       float fit=0;
04)       for (int i = 0; i < kumpulanJob.size(); i++)
05)       {
06)         fit=fit+
    (Job)kumpulanJob.elementAt(i)).getTotalProcessingTime();
07)       }
08)       return kumpulanJob.size()/fit;
09)    }
```

Program 4-26 implementasi fungsi evaluasi (*fitness*)

Proses perhitungan nilai *fitness* dilakukan dengan menjumlahkan waktu roduksi dari seluruh tugas yang ada, dan kemudian membagi jumlah tugas engan hasil penjumlahan waktu produksi tugas.

.2.6.4 Seleksi

Seleksi adalah sebuah proses memilih individu atau kromosom yang emiliki efektifitas dan efisiensi yang tinggi. Seperti telah dijelaskan diatas ahwa efektivitas dan efisiensi kromosom dinilai dengan fungsi evaluasi.

```
01)   public void pilihTerbaik(int jumlah)
02)   {
03)      int indx=0;
04)      offspring=new Vector();
05)      float max=-1;
06)      Kromosom maxK=new Kromosom();
07)      while(offspring.size()<=jumlah-1)
08)      {
09)        for (int i = 0; i < populasi.size(); i++) {
10)   if(((Kromosom)populasi.elementAt(i)).getFittnesValue()>=max)
11)      {
12)       max=((Kromosom)populasi.elementAt(i)).getFittnesValue();
13)       maxK=(Kromosom)populasi.elementAt(i);
14)       indx=i;
15)      }
16)     }
17)   offspring.addElement(maxK);
18)   max=-1;
19)   populasi.removeElementAt(indx);
20)   }
21)   }
```

Program 4-27 implementasi fungsi seleksi

Setelah mengalami seleksi, jumlah populasi menjadi sebesar (1-*selection ate)* x *jumlah populasi.* Sedangkan jumlah offspring menjadi *selection rate* x *mlah populasi.* Kromosom yang terdapat dalam populasi akan mengalami perasi genetik selanjutnya yaitu mutasi dan tukar silang. Sedangkan kromosom ang terdapat dalam offspring lolos dari operasi genetik.

2.6.5 Mutasi

```
01)   public Kromosom mutasi(Kromosom krom)
02)   {
03)     Object tmp=new Object();
04)     rand.setSeed(System.currentTimeMillis());
05)     int in=0;
06)     int pos=0;
07)     while(krom.elementAt(in).equals(krom.elementAt(pos))&&
08)   pos==in)
09)     {
10)        in=rand.nextInt(krom.size());
11)        pos=rand.nextInt(krom.size());
12)     }
13)     tmp=krom.remove(in);
14)     krom.insertElementAt(tmp,pos);
15)     return krom;
16)   }
```

Program 4-28 fungsi / method mutasi

Proses pertukaran posisi gen dilakukan secara acak. Untuk menentukkan osisi gen yang akan dipindah ditunjukkan pada baris 10. Sedangkan posisi tujuan enempatan gen ditunjukkan pada baris 11. Fungsi ini mempunyai nilai embalian berupa kromosom yang telah mengalami mutasi.

2.6.6 Tukar Silang (*Crossover*)

Seperti halnya pada proses mutasi, tukar silang dilakukan untuk endapatkan kromosom baru (*offspring*) dari kromosom dengan kategori tidak ptimal. Proses tukar silang ini melibatkan dua kromosom, dimana gen-gen dari edua kromosom tersebut dipilih dan kemudian dijadikan gen pada kromosom

lang secara lebih lengkap telah dijelaskan pada bab sebelumnya. Implementasi *ethod* tukar silang terdapat pada program 4-29.

```
01)   public Kromosom crossOver(Kromosom krom1,Kromosom krom2)
02)   {
03)      Kromosom off=new Kromosom();
04)      Enumeration enum1=krom1.elements();
05)      Enumeration enum2=krom2.elements();
06)      rand.setSeed(System.currentTimeMillis());
07)      while (enum1.hasMoreElements()||enum2.hasMoreElements()){
08)        if(rand.nextBoolean())
09)        {
10)           if(enum1.hasMoreElements()){
11)              Job job=(Job)enum1.nextElement();
12)              int i=job.getJumlahProses();
13)             if(off.getJmlGenBertype(job)<i)off.addElement(job);
14)           }
15)        }
16)        else{
17)         if(enum2.hasMoreElements())
18)         {
19)           Job job=(Job)enum2.nextElement();
20)           int i=job.getJumlahProses();
21)           if(off.getJmlGenBertype(job)<i)off.addElement(job);
22)         }
23)       }
24)       if(off.size()==krom1.size())break;
25)      }
26)   return off;
27)   }
```

Program 4-29 fungsi / method crossOver

2.6.7 Decoding

Decoding merupakan proses untuk mengubah solusi genotip menjadi notip. Atau dengan kata lain mengubah solusi dari representasi gen menjadi buah solusi yang berupa jadwal. Implementasi *decoding* terdapat pada program

```
01)   public void decoding(Kromosom krom)
02)   {
03)   //meletakkan operasi yang telah tuntas ke dalam antrian
04)     for (int j = 0; j < kumpulanJob.size(); j++)
05)     {
06)       Job job=(Job)kumpulanJob.elementAt(j);
07)       for (int x = 0; x < job.getJumlahProses(); x++)
08)       {
09)         Operasi o=job.getOperasiKe(x);
10)         int sta=o.getStatus();
11)         Mesin m=o.getMesin();
12)         if(sta==3)
13)         {if(!m.antrian.contains(o))m.addAntrian(o);}
14)        else if(sta==2)
15)         {if(!m.antrian.contains(o))m.addAntrian(o);}
16)       }
17)   }
18)   //menjadwalkan operasi yang belum terjadwal
19)   for (int i = 0; i < krom.size(); i++)
20)   {
21)      Job job=(Job)krom.elementAt(i);
22)      int wDatang=job.getDateWaktuKedatangan().getMinutes();
23)      int indx=job.getIndexTidakTerjadwal();
24)      Operasi ops=job.getOperasiKe(indx);
25)      if(ops.getStatus()==0||ops.getStatus()==1)
26)      {
27)         Mesin mesin=job.getMesinOperasiKe(indx);
28)         int idle=mesin.getIdle();
29)         int lst=0;
30)         if(indx>0)lst=indx-1;
31)         int fin = ((Operasi)job.getOperasiKe(lst)).
32)          getFinishTime();
33)         if(idle<tBaru)idle=tBaru;
34)         if(idle==0&&ops.getUrutanOperasi()==1)
35)         {
36)           ops.setStartTime(idle);
37)           ops.setStatus(1);
38)           mesin.addAntrian(ops);
39)         }
40)         else
41)         {
42)           ops.setStartTime((idle>=fin)?idle:fin);
43)           ops.setStatus(1);
```

Proses penjadwalan dilakukan hanya kepada operasi dengan status (1) dan 2). Penjadwalan dilakukan secara urut mulai dari gen pertama dalam kromosom ampai gen yang terakhir, baris 19. Penentuan waktu mulai ditentukan menurut aktu idle mesin dan waktu selesai proses sebelumnya, baris 36 dan 42. Untuk perasi dengan urutan proses pertama, waktu mulainya adalah waktu idle mesin ang bersangkutan karena tidak memiliki operasi sebelumnya. Proses engalokasian operasi kedalam mesin ditunjukkan pada baris 38 dan 44.

.2.7 Menampilkan Hasil Penjadwalan

Proses ini merupakan proses untuk menampilkan hasil proses enjadwalan. Hasil ini penjadwalan ini ditampilkan dalam diagram gantt melalui ntarmuka Display. Kelas yang bertanggung jawab untuk menampilkan diagram antt ini adalah GanttChartPanel. Implementasi kelas GanttChartPanel dijelaskan alam program 4-31. Untuk menampilkan diagram kelas GanttChartPanel emerlukan sebuah dataset, baris 17. Dataset tersebut berisi antrian operasi yang ari tiap-tiap mesin, ditunjukkan pada baris 25. Setelah dataset terbentuk proses elanjutnya adalah menggambar diagram gantt dari dataset yang telah terbentuk., aris 18

```
01)   public class GanttChartPanel extends JPanel
02)   {
03)       MainFrame parent;
04)       XYDataset dataset = null;
05)       JFreeChart chart = null;
06)       ChartPanel chartPanel =null;
07)       public GanttChartPanel(MainFrame prt){
08)         try{
09)           this.repaint();
```

```
15)   void init() throws Exception
16)   {
17)      XYDataset dataset = createDataset();
18)      JFreeChart chart = createChart(dataset);
19)      ChartPanel chartPanel = new ChartPanel(chart, false);
20)      chartPanel.setMouseZoomable(true, false);
21)       this.add(chartPanel);
22)   }
23)   private XYDataset createDataset()
24)   {
25)      Vector mesin=parent.getMesin();
26)      TimeSeriesCollection dataset = new TimeSeriesCollection();
27)      dataset.removeAllSeries();
28)      if(mesin.size()>0){
29)        for (int i = 0; i < mesin.size(); i++)
30)        {
31)           Mesin m=(Mesin)mesin.elementAt(i);
32)           if(m.antrian.size()>0){
33)             for (int a = 0; a <m.antrian.size(); a++)
34)             {
35)                Operasi op=m.getOperasi(a);
36)                TimeSeries s1=new TimeSeries
   (op.getParentJob().getName()+"-"+op.getUrutanOperasi(),
   Minute.class);
37)                s1.add(new Minute(op.getStartDate()),i+1);
38)                s1.add(new Minute(op.getFinishDate()),i+1);
39)                dataset.addSeries(s1);
40)             }
41)           }
42)         }
43)       }
44)      dataset.setDomainIsPointsInTime(true);
45)      return dataset;
46)   }
47)   private static JFreeChart createChart(XYDataset dataset)
48)   {
49)      JFreeChart chart = createTimeSeriesChart(
50)      "Jadwal Mesin","Waktu","Mesin",dataset);
51)       XYPlot plot = (XYPlot) chart.getPlot();
52)       XYItemRenderer r = plot.getRenderer();
53)      if (r instanceof XYLineAndShapeRenderer) {
54)         XYLineAndShapeRenderer renderer =
   (XYLineAndShapeRenderer) r;
```

.3 Implementasi Antarmuka

Bagian ini menjelaskan mengenai implementasi antarmuka yang igunakan oleh pengguna untuk berinteraksi dengan aplikasi. Tampilan antarmuka nengacu pada perancangan antarmuka pada bab sebelumnya.

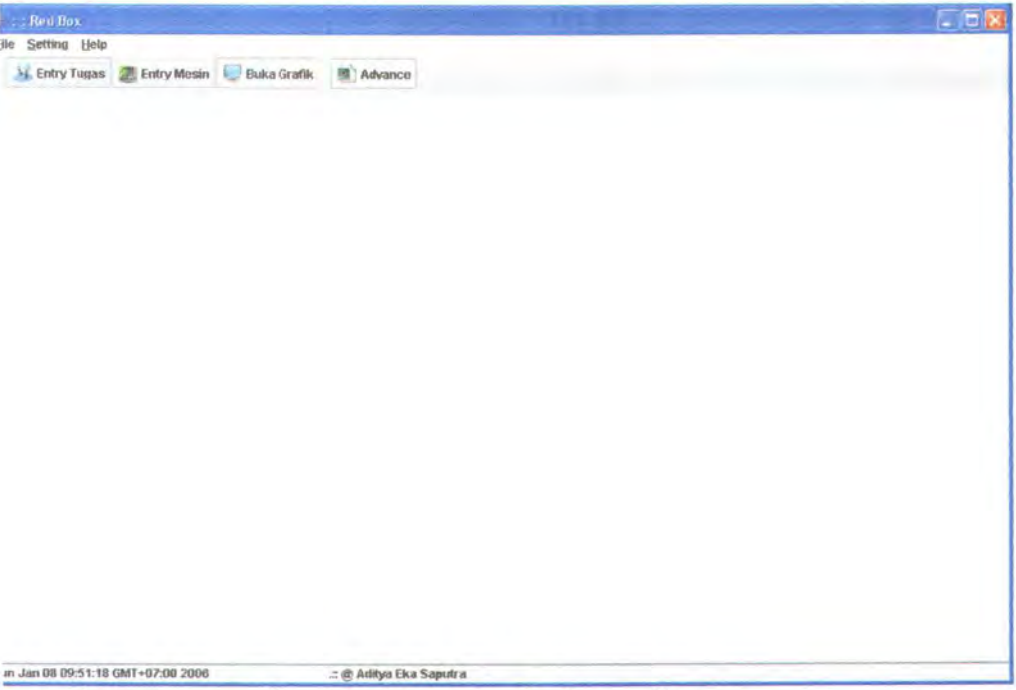
.3.1 Frame Utama

Frame utama ini merupakan tampilan utama perangkat lunak ini. Dimana etika aplikasi dijalankan akan memanggil kelas MainFrame. Melalui antarmuka ni pengguna dapat membuka antarmuka yang lain. Implementasi antarmuka rame Utama dijelaskan pada program 4-32.

```
01)   public class FrameUtama extends JFrame{
02)      Timer time;Date now;InputJob Jinput;InputMesin inputMesin;
03)      FrameParamGA paramGA;Display display;MainFrame param;
04)      public static MonitorFrame monitor;
05)      JButton input = new JButton();
06)      JButton tampilan = new JButton();
07)      JButton tbMesin = new JButton();
08)      JButton tbAdvance = new JButton();
09)   public FrameUtama() {
10)      jbInit();
11)   }
12)   void input_actionPerformed(){
13)      Jinput=new InputJob(param);
14)      desktop.add(Jinput);
15)   }
16)   void tampilan_actionPerformed(){
17)      display=new Display(param);
18)      desktop.add(display);
19)   }
20)   void parameterGA_actionPerformed(){
21)      paramGA=new FrameParamGA(param);
22)      desktop.add(paramGA);
```

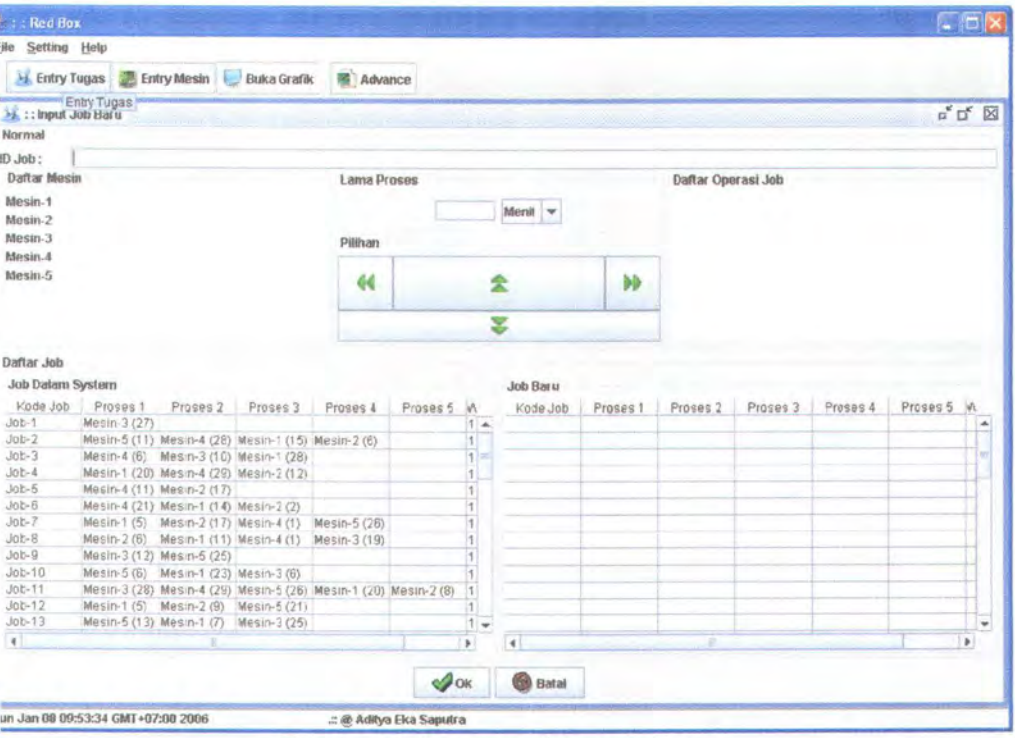
Pada baris 09 – 32 program 4-32 di atas dapat dilihat bahwa kelas Frame Utama mempunyai beberapa fungsi yang digunakan untuk membuka antarmuka lainnya. Fungsi –fungsi tersebut dipanggil ketika penggunan menekan tombol yang terdapat pada antarmuka frame utama. Fungsi tersebut dijalankan untuk membuka antarmuka entry tugas, entry mesin, display dan antarmuka lainnya.

Hasil tampilan program 4-32 digambarkan pada gambar 4-1. Pada tampilan ini tampak beberapa tombol dan menu untuk memanggil antarmuka entry tugas, entry mesin, penjadwalan.



Gambar 4-1 antarmuka Frame Utama

```
01)   public class InputJob extends JInternalFrame
02)   {
03)      private MainFrame parent;
04)      private Vector namaMesin;
05)      private Vector mesin;
06)      private Vector urutanMesin=new Vector()
07)      private Vector urutanOperasi=new Vector();
08)      private Vector kumpulanJob=new Vector();
09)      private SystemTime systemTime;
10)      JButton tbBatal = new JButton();
11)      JButton tbOk = new JButton();
12)      JButton tbSubmit = new JButton();
13)      public InputJob(MainFrame prt)
14)      {
15)         parent=prt;
16)         systemTime=new SystemTime(parent);
17)         mesin=(Vector)prt.getMesin().clone();
18)         jbInit();
19)      }
20)      void tbPilih_actionPerformed(ActionEvent e)
21)      {
22)         int i=listMesin.getSelectedIndex();
23)         Mesin m=(Mesin)mesin.elementAt(i);
24)         int w=getWaktuProses();
25)         urutanMesin.addElement(m);
26)         urutanOperasi.addElement(new Operasi
    (urutanOperasi.size()+1,m,w));
27)         mesin.removeElementAt(i);
28)         listMesin.setListData(namaMesin(mesin));
29)         listOperasi.setListData(namaMesin(urutanMesin));
30)      }
31)      void tbSubmit_actionPerformed(ActionEvent e)
32)      {
33)         kumpulanJob.addElement(
34)         new Job (id, (Vector)urutanOperasi.clone(), 1, 0,
  systemTime.getSysDate().getTime()));
35)      }
36)      void tbOk_actionPerformed(ActionEvent e)
37)      {
38)        for (int i = 0; i < kumpulanJob.size(); i++)
39)        {
40)           parent.addJob((Job)kumpulanJob.elementAt(i));
41)        }
```



Gambar 4-2 antarmuka entry tugas

Proses entry tugas baru dilakukan dengan mengisikan kode tugas, urutan nesin yang digunakan, dan lama waktu proses dari setiap mesin. Tugas baru erbentuk setelah pengguna menekan tombol submit, ditunjukkan pada baris 31-5.

.3.3 Entry Mesin

Seperti halnya pada antarmuka entry tugas, antarmuka ini juga bertujuan

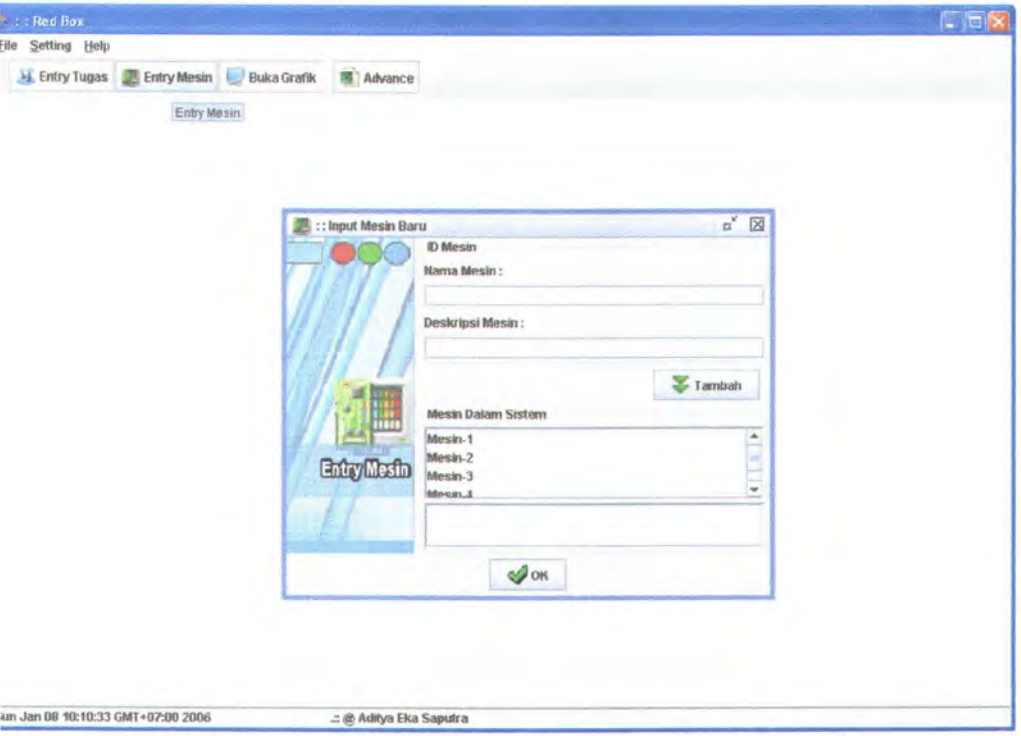
```
public class InputMesin extends JInternalFrame
{
    MainFrame parent;
    JButton tombolOk = new JButton();
    JButton tombolSubmit = new JButton();
    JList listMesin = new JList();

    public InputMesin(MainFrame prt)
    {
        parent=prt;
        jbInit();
    }

    void tombolSubmit_actionPerformed(ActionEvent e)
    {
      if(txNama.getText().length()>0)
      {
          Mesin m=new Mesin(txNama.getText());
          m.setDesc(txDesc.getText());
          parent.addMesin(m);
          parent.mesinBaru=true;
          this.listMesin.setListData(getNamaMesin());
       }
       else if(txNama.getText().length()<1)
       {
            JOptionPane.showMessageDialog(this,"Nama Mesin Tidak
Boleh Kosong","Peringatan",JOptionPane.ERROR_MESSAGE);
            txNama.setText("a");
       }
       txNama.setText("");
       txDesc.setText("");
    }

    void tombolOk_actionPerformed(ActionEvent e)
    {
      this.doDefaultCloseAction();
    }

    void listMesin_valueChanged(ListSelectionEvent e)
    {
      int indx=listMesin.getSelectedIndex();
      if(indx>-1)
      {
```



Gambar 4-3 implementasi antarmuka entry mesin

.3.4 Setting Parameter Algoritma Genetik

Setting parameter algoritma genetik ini dilakukan untuk meningkatkan ptimasi hasil penjadwalan. Antarmuka setting parameter algoritma genetik ini ipanggil dengan memilih menu setting > Parameter GA. Parameter yang dapat iubah oleh pengguna adalah, jumlah populasi, laju acak populasi, laju ıkasilang, laju mutasi, laju seleksi, jumlah keturunan. Perubahan parameter

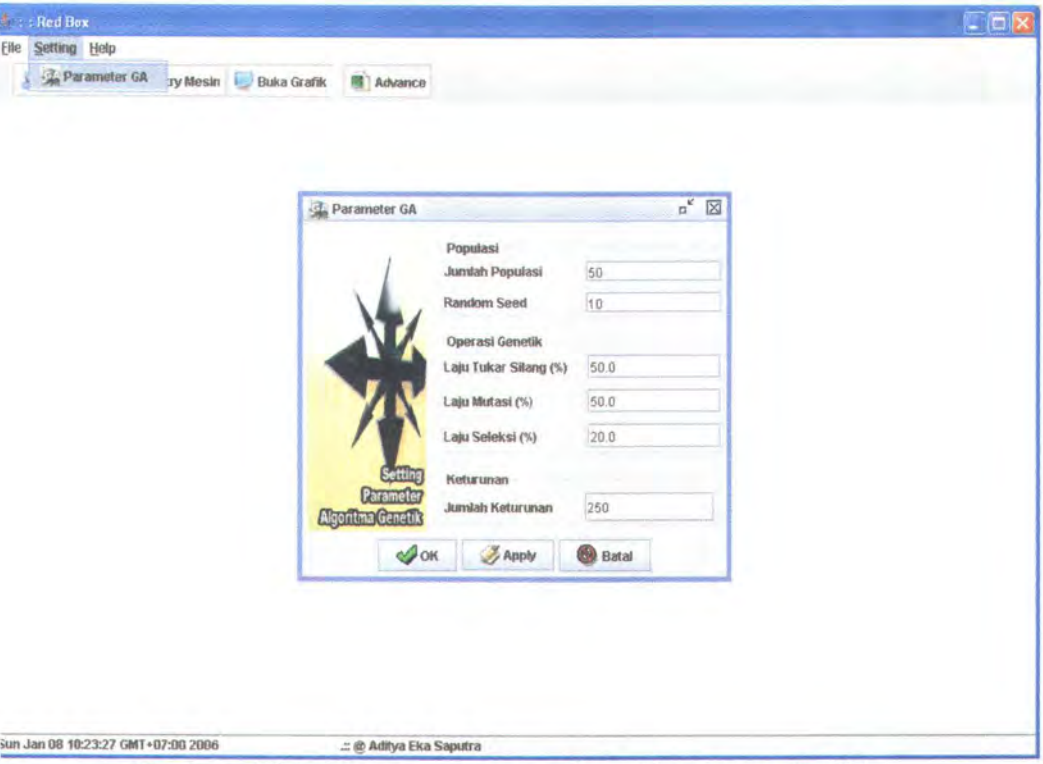
```
public class FrameParamGA extends JInternalFrame
{
    private MainFrame parent;
    JTextField txJmlIterasi = new JTextField();
    JTextField txLajuSeleksi = new JTextField();
    JTextField txLajuCross = new JTextField();
    JTextField txLajuMutasi = new JTextField();
    JTextField txJmlPop = new JTextField();
    JTextField txLajuAcak = new JTextField();
    JButton tbBatal = new JButton();
    JLabel gbPanel = new JLabel();
    JPanel panel2 = new JPanel();
    JButton tbApply = new JButton();
    JButton tbOK = new JButton();

   public FrameParamGA(MainFrame prt){
     parent=prt;
     jbInit();
   }

    private void jbInit() throws Exception{
        loadParam();
    }

    private void loadParam(){
     txJmlIterasi.setText(String.valueOf(parent.jmlIterasi));
     txLajuCross.setText(String.valueOf(parent.crossoverRate*100));
     txLajuMutasi.setText(String.valueOf(parent.mutationRate*100));
   txLajuSeleksi.setText(String.valueOf(parent.selectionRate*100));
    }

    void tbApply_actionPerformed(){
        parent.jmlPopulasi=Integer.parseInt(txJmlPop.getText());
        parent.jmlIterasi=Integer.parseInt(txJmlIterasi.getText());
        parent.crossoverRate=
          Float.parseFloat(txLajuCross.getText())/100;
        parent.mutationRate =
          Float.parseFloat(txLajuMutasi.getText())/100;
        parent.selectionRate=
          Float.parseFloat(txLajuSeleksi.getText())/100;
        parent.initialRandomRate=
          Long.parseLong(txLajuAcak.getText());
     }
```



Gambar 4-4 implementasi antarmuka setting parameter GA

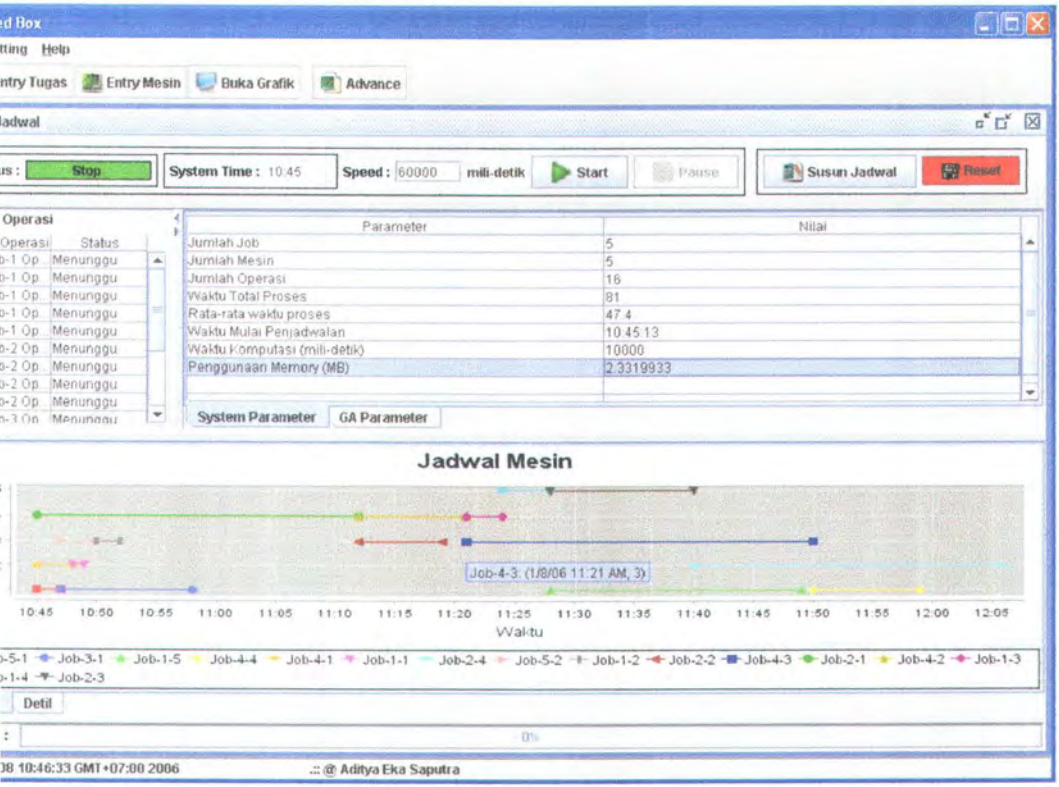
4.3.5 Penjadwalan

Antarmuka ini menampilakan hasil penjadwalan, informasi mengenai keadaan sistem, status operasi yang masuk, grafik hasil penjadwalan, informasi mengenai tugas yang masuk. Melalui antarmuka ini pengguna dapat juga dengan mudah menjalankan dan menghentikan waktu simulasi. Antarmuka ini

```
public class Display extends JInternalFrame
{
    SystemTimeFrame sysTime;
    SystemTime systemTime;
    MainFrame parent;
    Jadwal jadwal;
    SystemParameterPanel sysPanel;
    DetilPanel detilPanel;

    public Display(MainFrame prn){
          parent=prn;
          sysTime=new SystemTimeFrame(prn);
          systemTime=new SystemTime(prn);
          control=new PanelControl(prn);
          jadwal=new Jadwal(prn);
          sysPanel=new SystemParameterPanel(prn);
          detilPanel=new DetilPanel(this.parent);
          jobInfoPanel=detilPanel;
          jbInit();
    }
    public void jadwalUlang(){
      systemTime.pauseSystemTime();
      jadwal.setWaktuBaru(systemTime.getSysTime());
      jadwal.run();
      if(parent.getStatusJadwal()==false)
      {
          parent.setStatusJadwal(true;
          parent.startTabelStatus=true;
      }
      buildChart();
    }
    void tbRun_actionPerformed(ActionEvent e)
    {
        jadwalUlang
    }
    protected void buildChart(){
       if(tabPane.getComponentCount()!=0)tabPane.remove(0);
       ganttPanel =new GanttChartPanel(parent);
       tabPane.add(ganttPanel,  "Grafik",0);

    }
    void tbReset_actionPerformed(ActionEvent e)
    {
```



Gambar 4-5 implementasi antarmuka penjadwalan

BAB V

UJI COBA DAN EVALUASI

BAB 5
UJI COBA DAN EVALUASI

Bab ini menjelaskan mengenai pengujian dan evaluasi terhadap aplikasi penjadwalan. Bab ini menjelaskan mengenai lingkungan uji coba, data uji coba, pelaksanaan uji coba, evaluasi uji coba. Uji coba dilakukan untuk mengetahui kebenaran dan beberapa performa dari aplikasi yang telah dibuat.

5.1 Lingkungan Uji Coba

Aplikasi ini diuji coba diatas perangkat keras Personal Computer (PC) dengan Prosesor Intel Pentium 1,70 Ghz, memori 256 Mb RAM. Sedangkan untuk sistem operasi yang digunakan adalah Microsoft Windows Xp edisi profesional versi 5.1.2600.

5.2 Data Uji Coba

Dalam pengujian aplikasi simulasi penjadwalan ini data yang digunakan berupa tugas dengan beberapa operasi yang berbeda dan waktu kedatangan tugas yang berbeda-beda. Urutan proses dan waktu kedatangan tugas yang akan menjadi data uji coba dibuat secara acak (*random*).

5.3 Pelaksanaan dan Evaluasi Uji Coba

Uji coba ini dibagi menjadi dua macam pengujian yaitu pengujian validitas

5.3.1 Uji Coba Kebenaran

Pengujian ini dilakukan untuk menguji validitas atau kebenaran algoritma aplikasi yang dibuat. Pengujian ini dilakukan dengan membandingkan hasil perhitungan manual dengan hasil yang didapat aplikasi.

Ada beberapa batasan yang harus dipenuhi oleh sebuah penjadwalan yang valid. Batasan tersebut harus dipenuhi, tidak dapat tidak, agar sebuah penjadwalan menjadi masuk akal dan dapat diterima. Batasan tersebut adalah sebagai berikut,

a. Urutan operasi yang dijadwalkan harus sesuai dengan urutan proses dari sebuah tugas
b. Sebuah operasi tidak boleh dijalankan sebelum operasi sebelumnya selesai. Atau waktu mulai sebuah operasi t_{ik} tidak boleh lebih kecil dari waktu selesai operasi sebelumnya t_{ik-1}.

$$t_{ik-1} < t_{ik}$$

c. Sebuah operasi harus dijadwalkan pada mesin yang memprosesnya
d. Operasi yang selesai diproses dan sedang dalam proses tidak dijadwalkan ulang. Hanya operasi yang menunggu dan belum terjadwal yang dijadwalkan ulang
e. Waktu mulai operasi pertama dari sebuah tugas tidak boleh lebih kecil dari waktu kedatangan tugas.

$$r_i < t_{ij}$$

kan dijadwalkan pada mesin 2, 1, 5, 3, dengan lam proses, dalam menit, 4, 4, 3, . Hasil penjadwalan *(t=0)* ditunjukkan pada gambar 5-1.

Tabel 5-1 tugas pada posisi *t=0*

	Operasi 1		Operasi 2		Operasi 3		Operasi 4		Operasi 5	
	Mesin	W.P	Mesin	W.P	Mesin	W.P	Mesin	W.P	Mesin	W.P
b 1	2	4	1	4	5	3	3	7		
b 2	5	3	3	7	2	4	1	6	4	4
b 3	4	4	3	6	2	3	1	5	5	4
b 4	2	3	3	4	5	3				
b 5	2	7	4	3	3	5	1	5	5	4

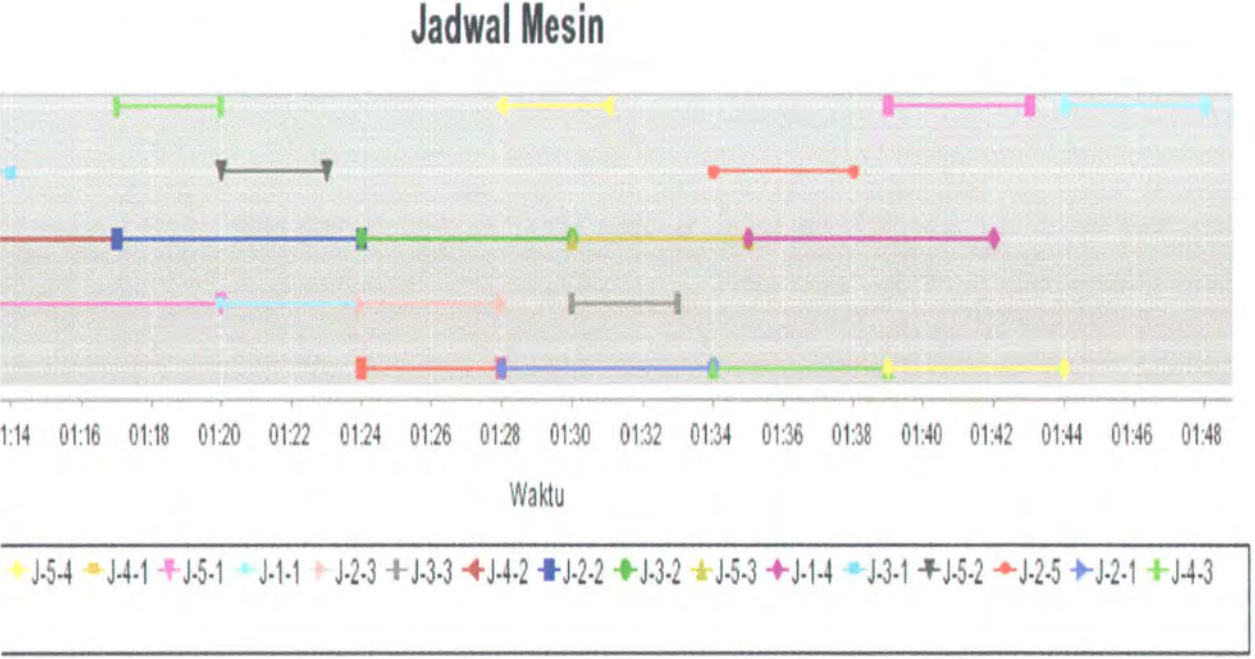
Dari hasil penjadwalan yang ditunjukkan pada gambar 5-1, dapat isimpulkan bahwa hasil penjadwalan tidak melanggar batasan yang ditentukan aitu :

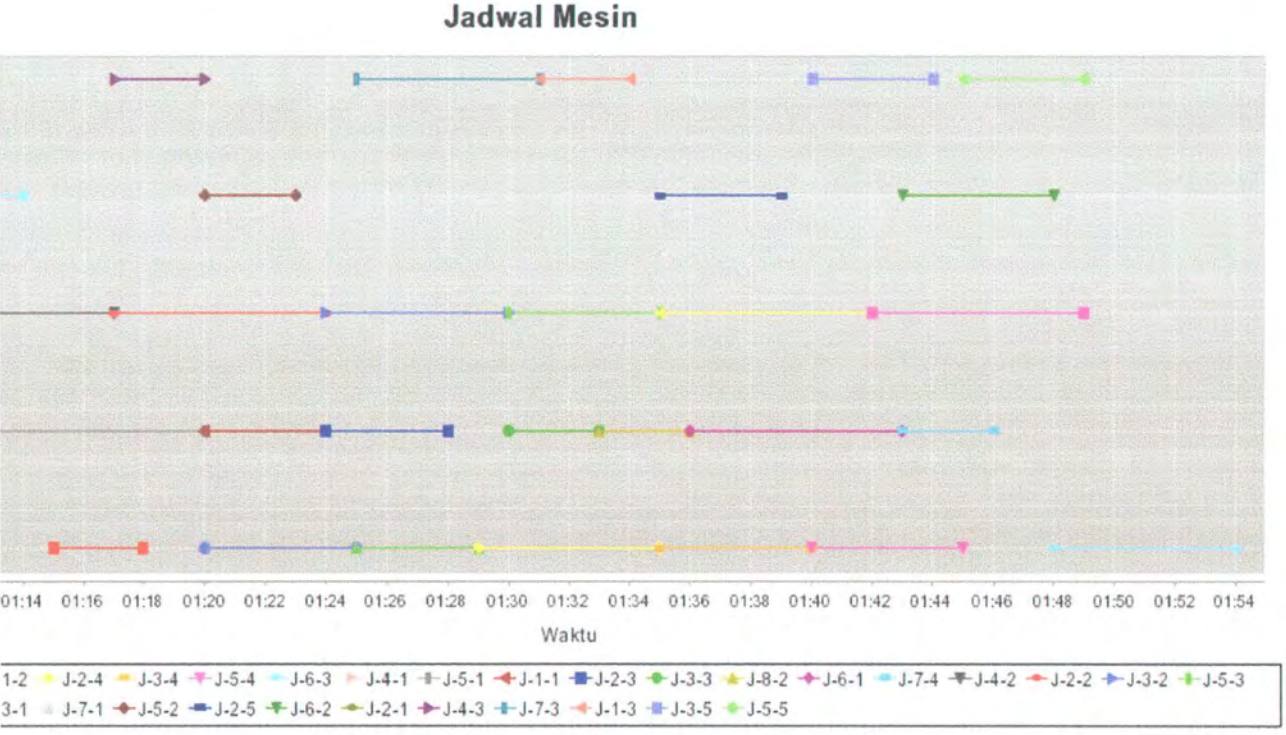
. Urutan operasi tugas hasil penjadwalan sesuai dengan urutan operasi tugas dari dataset.

. Hasil penjadwalan menujukkan bahwa waktu mulai sebuah operasi t_{ik} tidak boleh lebih kecil dari waktu selesai operasi sebelumnya t_{ik-1}. Contoh, operasi *Job 1-3* tidak dimulai sebelum waktu selesai operasi *Job 1-2*.

. Hasil penjadwalan menujukkan bahwa setiap operasi dijadwalkan pada mesin yang sesuai. Contoh, operasi Job 3-2 dijadwalkan pada mesin 3.

Batasan d, e merupakan batasan yang harus dipenuhi apabila terdapat ugas baru yang masuk setelah penjadwalan *(t=0)* telah dibuat. Untuk menguji atasan d maka penjadwalan yang telah dibuat *(t=0)* ditambahkan tugas baru pada t=5), dengan spesifikasi tugas pada tabel 5-2.


Jadwal Mesin
01:14
01:16
01:18
01:20
01:22
01:24
01:26
01:28
01:30
01:32
01:34
01:36
01:38
01:40
01:42
01:44
01:46
01:48
Waktu
J-5-4
J-4-1
J-5-1
J-1-1
J-2-3
J-3-3
J-4-2
J-2-2
J-3-2
J-5-3
J-1-4
J-3-1
J-5-2
J-2-5
J-2-1
J-4-3


Jadwal Mesin
01:14 01:16 01:18 01:20 01:22 01:24 01:26 01:28 01:30 01:32 01:34 01:36 01:38 01:40 01:42 01:44 01:46 01:48 01:50 01:52 01:54
Waktu
1-2 J-2-4 J-3-4 J-5-4 J-6-3 J-4-1 J-5-1 J-1-1 J-2-3 J-3-3 J-8-2 J-6-1 J-7-4 J-4-2 J-2-2 J-3-2 J-5-3
3-1 J-7-1 J-5-2 J-2-5 J-6-2 J-2-1 J-4-3 J-7-3 J-1-3 J-3-5 J-5-5

Hasil penjadwalan dari penambahan job 6, 7, 8 ditunjukkan pada gambar 5-2 Dalam gambar 5-2 terlihat bahwa job 6, 7, dan 8 dapat ditambahkan dengan benar ke dalam jadwal yang sudah ada. Penambahan tugas 6, 7, 8. memenuhi batasan d, e yaitu :

1. Operasi yang selesai diproses yaitu Job 2-1, Job 3-1, Job 4-1 dan sedang dalam proses, Job 5-1, Job 4-2, tidak dijadwalkan ulang. Hanya operasi yang menunggu dan belum terjadwal yang dijadwalkan ulang

e. Waktu mulai operasi Job 8-1 lebih besar atau sama dengan waktu kedatangan Job 8. Waktu mulai proses Job 8-1 adalah t_{81}= 01:15, sedangkan waktu kedatangan Job 8 adalah r_i= 01:15.

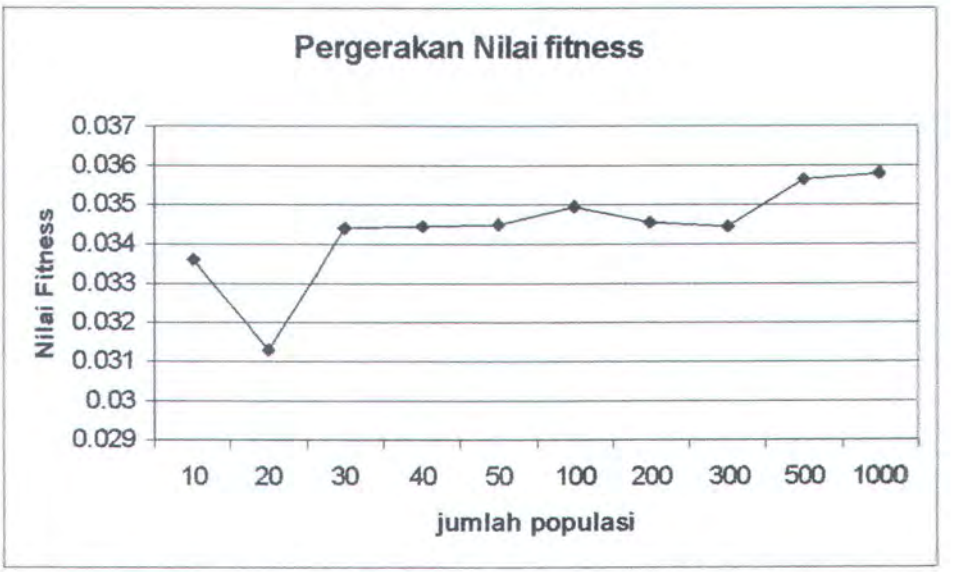
5.3.2 Uji Coba Kinerja

Uji coba ini dilakukan untuk menguji kehandalan aplikasi dengan berbagai skenario dan optimasi penjadwalan. Pengujian optimasi dilakukan dengan mengubah parameter algoritma genetik

Salah satu kelebihan penggunaan algoritma genetik adalah bersifat fleksibel sehingga kita dapat mendapatkan hasil yang berbeda dengan mengatur parameter genetik. Pengujian ini dilakukan dengan beberapa skenario untuk pengujian parameter yang berbeda. Dataset yang digunakan untuk pengijian ini menggunakan dataset tabel 5-1. Fungsi *fitness* yang digunakan dalam pengujian ini adalah rata-rata waktu proses (*average procesing time*).

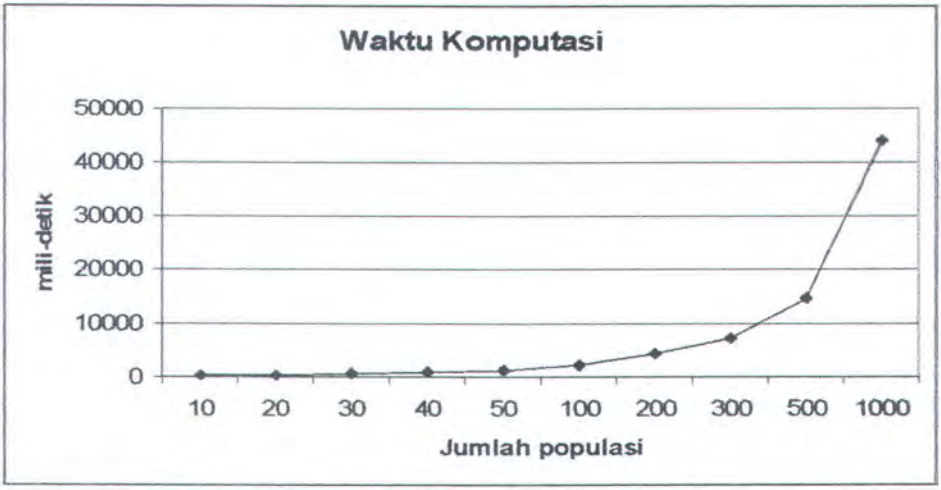
- Random seed = 10
- Laju mutasi = 50%
- Laju tukar silang = 50%
- Laju seleksi = 50%
- Jumlah Keturunan = 100

Pergerakan nilai *fitness* yang didapatkan ditunjukkan pada gambar 5-3. Data detil mengenai uji coba parameter jumlah populasi dapat dilihat pada lampiran.



Gambar 5-3 nilai *fitness* uji parameter jumlah populasi

Menurut gambar 5-3 dapat disimpulkan bahwa semakin tinggi jumlah



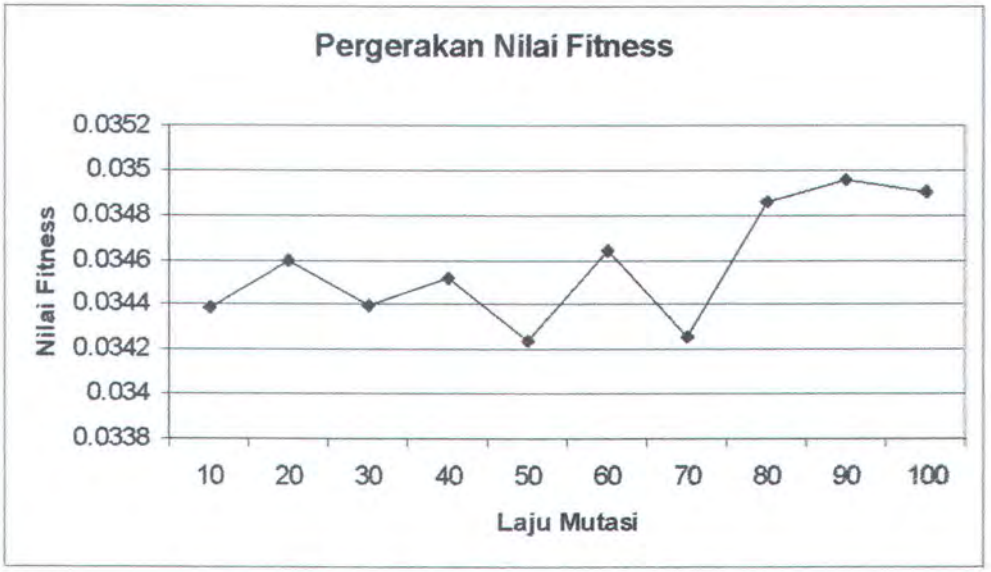
Gambar 5-4 waktu komputasi pengujian parameter jumlah populasi

Sehingga nilai parameter jumlah populasi diambil nilai 50 sebagai *default* arena memiliki waktu komputasi yang relatif kecil tetapi nilai *fitness* yang cukup inggi.

.3.2.2 Uji Parameter Laju Mutasi

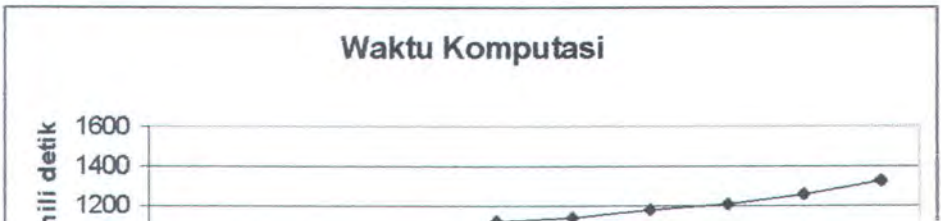
Parameter laju mutasi ini merupakan parameter yang mengatur tingkat nutasi dalam setiap generasi atau iterasi. Pengujian ini dilakukan berulang kali engan skenario yang sama tetapi dengan nilai parameter laju mutasi yang erbeda. Skenario pengujian adalah sebagai berikut,

- Jumlah populasi =50
- Random seed = 10
- Laju tukar silang = 50%



Gambar 5-5 nilai *fitness* uji parameter laju mutasi

Dari hasil pengujian diatas didapatkan rata nilai *fitness* bergerak disekitar ilai 0.034578 dengan standart deviasi=0.000966 dan pergerakan tersebut tidak ignifikan. Sehingga dapat disimpulkan bahwa perubahan laju mutasi tidak emberikan pengaruh yang signifikan terhadap nilai *fitness*. Apabila dilihat dari erubahan waktu komputasi pengujian diatas, maka didapat waktu komputasi eperti pada gambar 5-6.



Dalam gambar 5-6 dapat dilihat bahwa waktu komputasi bertambah eiring bertambahnya nilai laju mutasi. Pengguna dapat mengubah parameter laju nutasi sesuai dengan tujuan yang ingin dicapai, yaitu optimasi hasil atau ecepatan proses. Sebagai default maka ditetapkan nilai tengah antara optimasi lan waktu komputasi yaitu 50%.

5.3.2.3 Uji Parameter Laju Tukar Silang

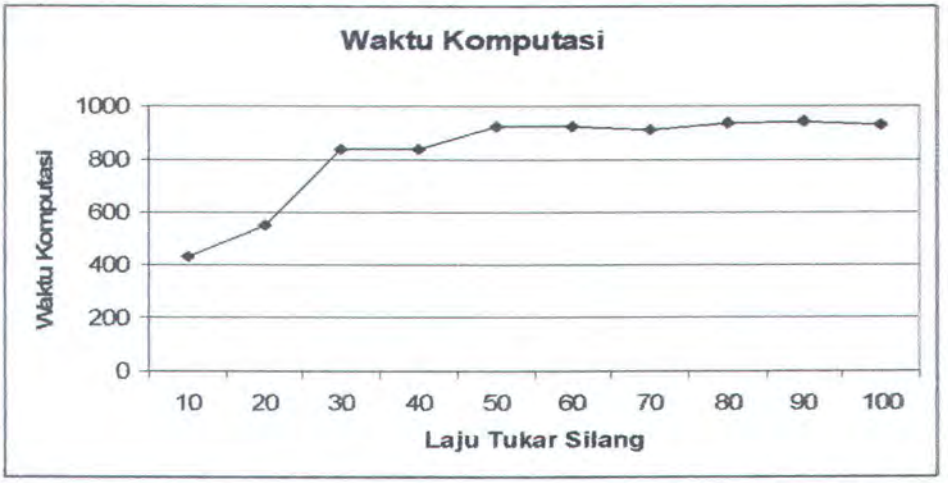
Parameter laju tukar silang ini merupakan parameter yang mengatur ingkat terjadinya tukar silang dalam tiap generasi. Pengujian ini dilakukan erulang kali dengan skenario yang sama tetapi dengan nilai parameter laju tukar ilang yang berbeda. Skenario pengujian adalah sebagai berikut,

- Jumlah populasi =50
- Random seed = 10
- Laju mutasi = 50%
- Laju seleksi = 50%
- Jumlah keturunan = 100

Hasil nilai *fitness*, ditunjukkan pada gambar 5-7. Data detil mengenai uji oba parameter tukar silang dapat dilihat pada lampiran.

Dari hasil pengujian diatas didapatkan bahwa rata-rata nilai fitnes bergerak diantra nilai 0.03451 dengan standart deviasi=0.000704. Sehingga dapat disimpulkan bahwa perubahan laju tukar silang tidak perpengaruh besar pada nilai *fitness*.

Waktu komputasi percobaan diatas ditunjukkan pada gambar 5-8. Pada gambar tersebut ditunjukkan bahwa semakin besar laju tukar silang maka semakin besar pula waktu komputasi. Tetapi dapat terjadi kemungkinan, hasil yang optimal akan didapat melalui laju tukar silang yang besar.



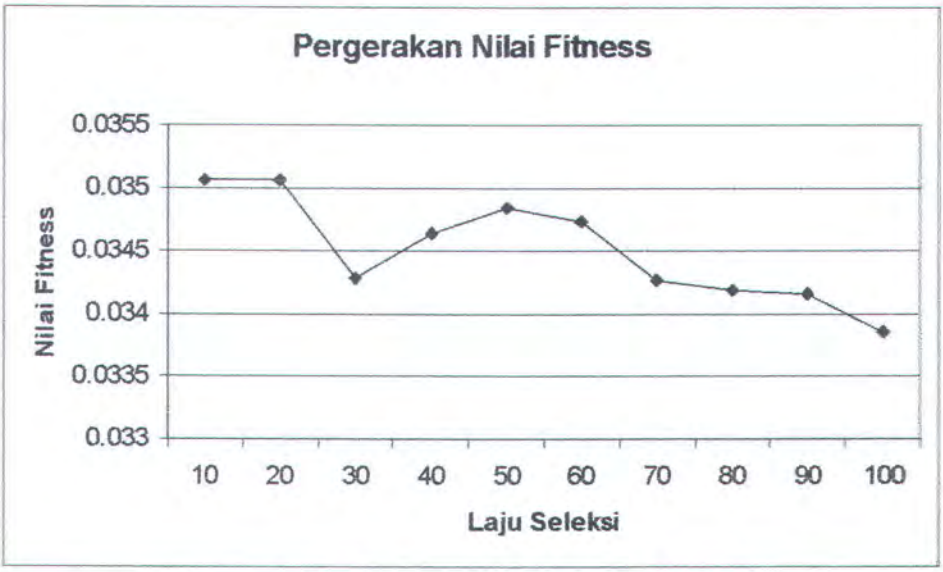
Gambar 5-8 waktu komputasi pengujian parameter laju tukar silang

Dari hasil percobaan laju tukar silang didapatkan bahwa perubahan laju tukar silang tidak terlalu memberikan perubahan pada nilai *fitness*. Sehingga untuk nilai laju tukar silang secara *default* adalah 50%. Dimana nilai *default* ini

perasi genetik. Pengujian ini dilakukan berulang kali dengan skenario yang ama tetapi dengan nilai parameter laju mutasi yang berbeda. Skenario pengujian dalah sebagai berikut,

- Jumlah populasi =50
- Random seed = 10
- Laju mutasi = 50%
- Laju tukar silang=50%
- Jumlah keturunan= 100

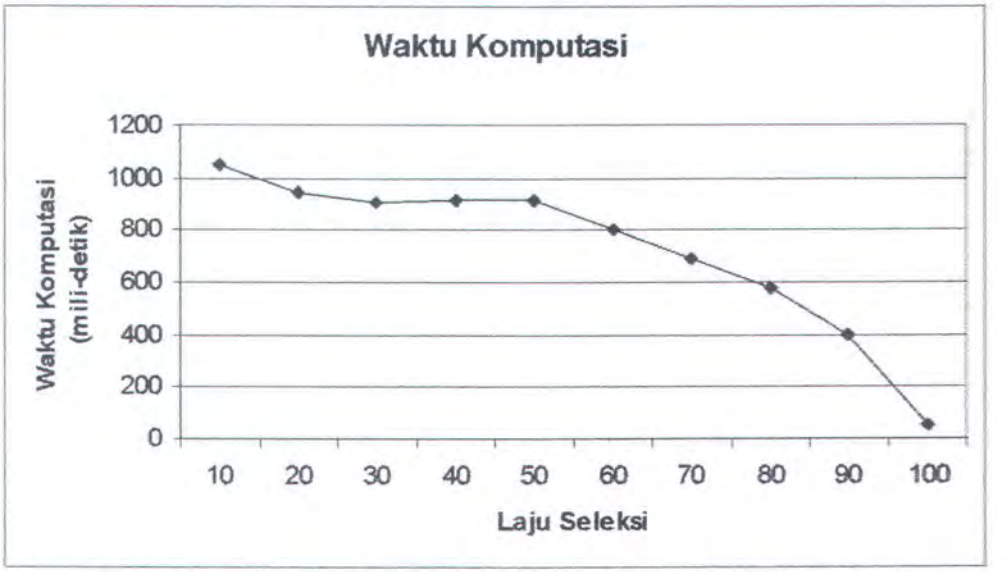
Nilai *fitness* yang didapat dari hasil pengujian ini terdapat pada gambar 5-. Data detil mengenai uji coba parameter laju selaksi dapat dilihat pada lampiran.



Gambar 5-9 nilai *fitness* uji parameter laju seleksi

menyebabkan kromosom yang lemah lolos dari seleksi dan masuk ke generasi berikutnya.

Apabila dilihat dari segi waktu komputasi maka didapatkan pergerakan komputasi seperti pada gambar 5-10. Dapat ditarik kesimpulan bahwa semakin besar laju seleksi maka waktu komputasi akan semakin menurun. Hal ini disebabkan karena jumlah kromosom yang mengalami operasi genetik sedikit sehingga waktu komputasi juga kecil.



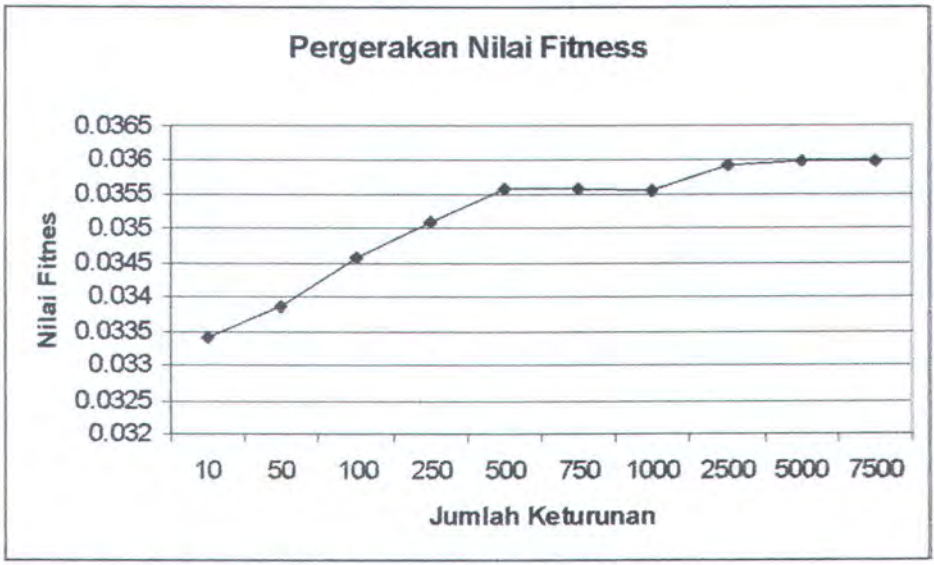
Gambar 5-10 waktu komputasi pengujian parameter laju seleksi

Dapat disimpulkan bahwa semakin besar laju seleksi, maka semakin kecil nilai *fitness* yang didapat dan semakin kecil pula waktu komputasi. Pengguna dapat menentukan tujuan yang akan dicapai, apakah waktu komputasi minimal

kenario yang sama tetapi dengan nilai parameter jumlah keturunan yang berbeda. Skenario pengujian adalah sebagai berikut,

- Jumlah populasi =50
- Random sedd = 10
- Laju mutasi = 50%
- Laju tukar silang=50%
- Laju seleksi = 20 %

Pergerakan nilai *fitness* yang dihasilkan dalam percobaan ini dapat dilihat pada gambar 5-11. Data yang lebih detil mengenai percobaan uji parameter jumlah keturunan dapat dilihat pada lampiran.



Gambar 5-11 nilai *fitness* uji parameter jumlah keturunan

Dapat ditarik kesimpulan bahwa nilai *fitness* maksimal dicapai pada parameter umlah keturunan lebih dari 5000.

Waktu komputasi percobaan ini dapat dilihat pada gambar 5-12 . Dapat ditarik kesimpulan bahwa waktu komputasi akan meningkat seiring dengan meningkatnya jumlah keturunan. Peningkatan waktu komputasi ini dirasa cukup ajam. Pengguna dapat memilih untuk meningkatkan optimasi atau kecepatan komputasi. Jumlah keturunan ini secara *default* ditentukan sebesar 250 keturunan gar hasil yang didapat cukup optimal dengan waktu komputasi yang relatif kecil.

Gambar 5-12 waktu komputasi pengujian jumlah keturunan

5.3.2.6 Uji Perubahan Parameter Genetik

Pengujian ini dilakukan dengan menguji beberapa skenario. Dimana kenario merupakan kombinasi parameter genetik yang berbeda-beda. Kombinasi arameter genetik tersebut dibangkitkan secara acak berdistribusi *uniform*. Dataset ang digunakan dalam pengujian ini terdapat pada tabel 5-3.

Pada tabel 5-3 merupakan permasalahan *job-shop* 5x5, dimana tugas yang kan dijadwalkan berjumlah 5 dan mesin yang digunakan berjumlah 5. Masing-nasing tugas pada tabel 5-3 diproses maksimal 5 kali operasi. Masing-masing perasi diproses dengan mesin yang berbeda dan waktu yang berbeda. Misal, ugas dengan kode *J-1* memiliki 4 operasi berbeda. Operasi 1 tugas *J-1* diproses ada mesin 2 dengan lama proses 4 menit. Operasi 2 tugas *J-1* diproses pada nesin 1 dengan lama proses 4 menit dan seterusnya. Pengujian dilakukan dengan nenggubah parameter genetik secara bersamaan terhadap dataset yang tetap. engujian ini dilakukan terhadap 15 skenario yang berbeda-beda.

Tabel 5-3 dataset pengujian perubahan parameter

	Operasi 1		Operasi 2		Operasi 3		Operasi 4		Operasi 5	
	Mesin	W.P	Mesin	W.P	Mesin	W.P	Mesin	W.P	Mesin	W.P
ob 1	2	4	1	4	5	3	3	7		
ob 2	5	3	3	7	2	4	1	6	4	4
ob 3	4	4	3	6	2	3	1	5	5	4
ob 4	2	3	3	4	5	3				
ob 5	2	7	4	3	3	5	1	5	5	4

Skenario yang digunakan dalam pengujian perubahan parameter ini erdapat pada tabel 5-4. Pada tabel 5-4 terdapat 15 skenario yang berbeda, masing-nasing skenario memiliki nilai parameter genetik yang berbeda. Misal skenario 1

iness pengujian ini digambarkan pada gambar 5-13. Sedangkan pergerakan waktu komputasi pengujian ini digambarkan pada gambar 5-14.

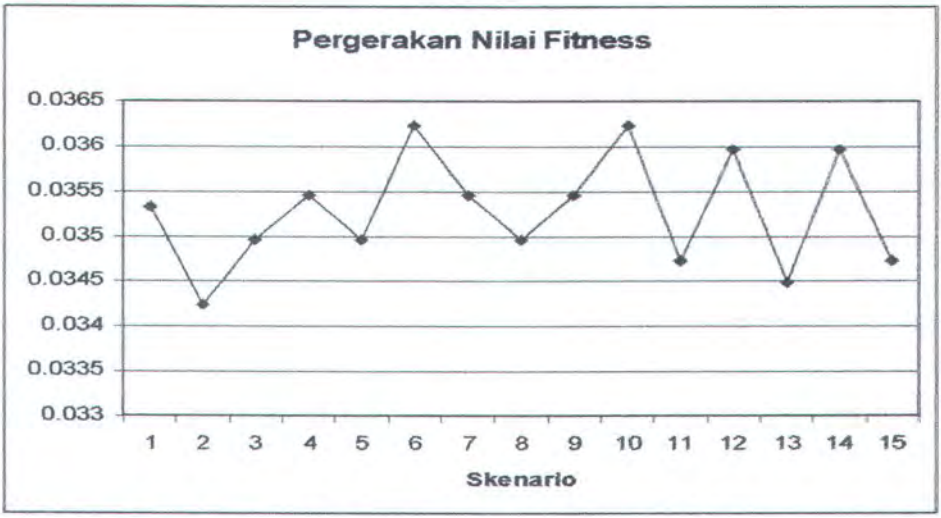
Tabel 5-4 skenario pengujian perubahan parameter

Skenario	Jumlah Populasi	Jumlah Keturunan	Random Seed	Laju Tukar Silang	Laju Mutasi	Laju Seleksi
kenario 1	369	38	-53	0.76	0.69	0.58
kenario 2	153	92	-20	0.22	0.22	0.64
kenario 3	240	576	-47	0.58	0.44	0.62
kenario 4	11	530	18	0.32	0.14	0.45
kenario 5	184	388	21	0.65	0.57	0.66
kenario 6	27	660	-71	0.28	0.88	0.01
kenario 7	125	250	-66	0.83	0.73	0.92
kenario 8	57	688	83	0.22	0.68	0.33
kenario 9	100	475	25	0.83	0.67	0.77
kenario 10	65	538	77	0.29	0.48	0.12
kenario 11	83	466	-39	0.29	0.02	0.42
kenario 12	343	156	-45	0.65	0.02	0.73
kenario 13	70	383	49	0.75	0.37	0.35
kenario 14	496	594	-43	0.03	0.34	0.56
kenario 15	119	300	-87	0.89	0.83	0.23

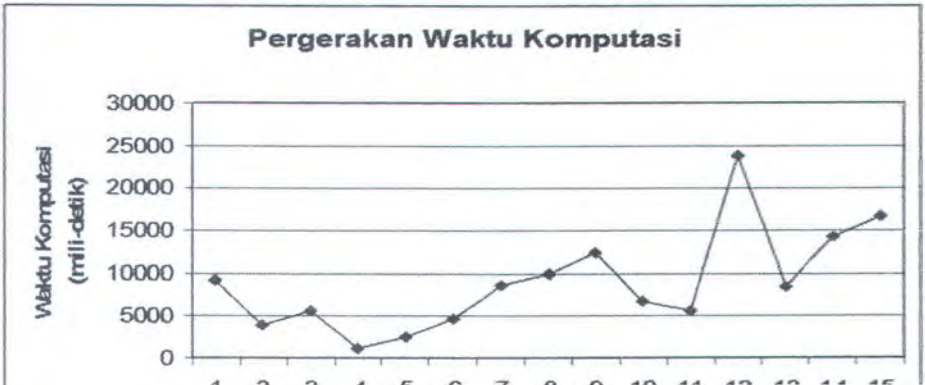
Tabel 5-5 hasil pengujian perubahan parameter genetik

Skenario	Nilai *Fitness*	Waktu Komputasi (mili-detik)
Skenario 1	0.035333335	9140
Skenario 2	0.034246575	4015
Skenario 3	0.034967533	5656
Skenario 4	0.035460994	1094
Skenario 5	0.034965035	2344
Skenario 6	0.036231883	4609

Pada tabel 5-5 ini menunjukkan bahwa setiap skenario akan menghasilkan nilai *fitness* dan waktu komputasi yang berbeda-beda. Nilai rata-rata pergerakan nilai *fitness* adalah 0,035279594 dengan STD sebesar 0,000626253. Sedangkan rata-rata pergerakan waktu komputasi sebesar 8903,8 dan STD sebesar 5964,45.



Gambar 5-13 pergerakan nilai *fitness* pengujian perubahan parameter



angat kecil maka dapat disimpulkan bahwa perubahan parameter secara erempak tidak mempengaruhi nilai *fitness* secara signifikan.

Berdasarkan gambar 5-14 didapatkan bahwa waktu komputasi bergerak ukup fluktuatif dan cenderung tidak stabil. Sehingga dapat disimpulkan bahwa erubahan parameter genetik cukup mempengaruhi waktu komputasi.

5.3.2.7 Uji Penjadwalan Ulang

Uji penjadwalan dilakukan untuk menguji kemampuan aplikasi untuk engatasi permasalah kedatangan tugas yang dinamis. Skenario yang digunakan alam pengujian ini adalah scenario dinamis agar dapat melihat perubahan yang erjadi. Pengujian penjadwalan ulang ini dilakukan melalui dua skenario. kenario tersebut dibagi menurut jumlah tugas yang masuk, yaitu data kecil dan ata besar. Tugas beserta operasinya dibangkitkan secara acak dengan distribusi niform. Sedangkan waktu kedatangan ditentukan secara acak dengan distribusi ksponensial.

Parameter algoritma genetik yang digunakan dalam skenario ini adalah ilai *default* yaitu,

- Jumlah Mesin = 5
- Jumlah populasi = 50
- Random seed = 10

- Skenario 1 – data kecil

Dalam skenario ini jumlah tugas yang akan masuk adalah 20 dengan waktu kedatangan 0, 5, 8, 10, 11. Penjelasan data tugas skenario 1 terdapat pada tabel 5-6. Tabel 5-6 menunjukkan jumlah operasi dan waktu kedatangan tugas baru. Misal pada saat menit ke-0 terdapat 5 tugas baru yang masuk kedalam sistem dan pada saat menit ke-8 terdapat 4 tugas baru yang akan masuk kedalam sistem. Jumlah operasi Job 1 adalah 4, ini berarti bahwa job 1 akan diproses kedalam 4 mesin yang berbeda. Begitu juga dengan Job 2 akan diproses kedalam 5 mesin yang berbeda. Hasil yang didapatkan peda pengujian ini dijelaskan pada tabel 5-7

Tabel 5-6 data tugas scenario 1

	Jumlah Operasi	W.datang (menit)		Jumlah Operasi	W.datang (menit)
Job 1	4	0	Job 11	4	8
Job 2	5	0	Job 12	5	8
Job 3	5	0	Job 13	1	10
Job 4	3	0	Job 14	4	10
Job 5	4	0	Job 15	4	10
Job 6	3	5	Job 16	3	10
Job 7	4	5	Job 17	5	10
Job 8	3	5	Job 18	3	11
Job 9	2	8	Job 19	4	11
Job 10	3	8	Job 20	3	11

Tabel 5-7 hasil pengujian skenario 1

	t=0	t=5	t=8	t=10	t=11
Jumlah Tugas	5	8	12	17	20
Jumlah Mesin	5	5	5	5	5

memperlihatkan bahwa terjadi perubahan nilai parameter sistem (waktu total proses, rata-rata waktu proses, waktu komputasi) terhadap perubahan masukkan yang terjadi. Waktu total proses dan waktu rata-rata proses meningkat seiring meningkatnya jumlah tugas yang masuk. Hal ini menunjukkan bahwa sistem mampu menangani tugas yang masuk dalam jumlah kecil.

- Skenario 2 – data besar

Dalam skenario ini jumlah tugas yang akan masuk kedalam sistem sejumlah 40 tugas baru dengan waktu kedatangan 0, 15, 25, 30, 34. Perincian data tugas skenario ini terdapat pada tabel 5-8 Penjelasan tabel 5-8 adalah sebagai berikut, Job 1 dengan jumlah operasi 3 akan datang pada menit ke-0. Hasil yang didapatkan pada pengujian ini dijelaskan pada tabel 5-9.

Tabel 5-8 data tugas skenario 2

	Jumlah Operasi	W.datang (menit)		Jumlah Operasi	W.datang (menit)
Job 1	3	0	Job 21	4	30
Job 2	4	0	Job 22	2	30
Job 3	3	0	Job 23	3	30
Job 4	2	0	Job 24	4	30
Job 5	5	0	Job 25	3	30
Job 6	3	0	Job 26	2	30
Job 7	3	0	Job 27	2	30
Job 8	4	0	Job 28	3	30
Job 9	1	0	Job 29	2	34
Job 10	2	0	Job 30	3	34

Tabel 5-9 hasil pengujian skenario 2

	t=0	t=15	t=25	t=30	t=34
Jumlah Job	10	13	20	28	40
Jumlah Mesin	5	5	5	5	5
Jumlah Operasi	30	44	76	99	136
Waktu Total Proses	155	195	334	434	558
Rata-rata waktu proses	105.9	132.5385	223.05	271.2143	347
Waktu Komputasi (mili-detik)	3422	4672	12125	18188	33625

Pada tabel 5-9 memperlihatkan bahwa terjadi perubahan nilai parameter istem terhadap perubahan masukkan yang terjadi. Hal ini menunjukkan bahwa istem mampu menangani tugas yang masuk dalam jumlah besar.

.3.3 Uji Penggunaan Memori

Pengujian penggunaan memori dilakukan untuk mengetahui penggunaan nemori operasi genetik. Pengukuran memori dilakukan dengan menggunakan elas Runtime yang miliki paket java. Skenario yang digunakan untuk pengujian ni adalah skenario statis bukan skenario yang dinamis karena akan sulit untuk nengukur penggunaan dalam lingkungan dinamis yang selalu berubah. Hasil engujian dilakukan dengan iterasi masing-masing skenario sebanyak 5 kali. arameter algoritma genetik yang digunakan adalah sebagai berikut,

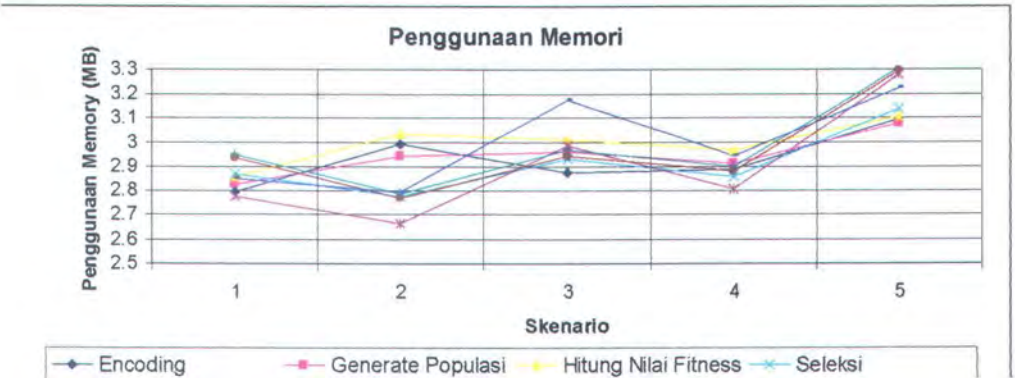
- Jumlah populasi = 50

Sedangkan jumlah tugas yang masuk bervariasi. Data tugas yang digunakan terdapat pada tabel 5-10. Pada skenario 1 pengujian memori dilakukan terhadap 10 buah job dengan total operasi sebanyak 31 dan jumlah mesin sebanyak 5 mesin yang berbeda. Pada skenario 2 pengujian memori dilakukan terhadap 20 buah job dengan total operasi sebanyak 58 operasi. Jumlah tugas dan operasi skenario 2 bertambah dibandingkan dengan skenario 1.

Tabel 5-10 data tugas pengujian memori

	Skenario 1	Skenario 2	Skenario 3	Skenario 4	Skenario 5
Jml Job	10	20	30	40	50
Jml Op.	31	58	88	94	156
Jml Mesin	5	5	5	5	5

Hasil pengujian ini ditampilkan pada tabel 5-11 .Pergerakan penggunaan memori ditunjukkan oleh gambar 5-15. Dari gambar tersebut dapat disimpulkan bahwa penggunaan memori meningkat seiring meningkatnya tugas yang masuk.



Tabel 5-11 hasil penggujian penggunaan memori

	1	2	3	4	5	Rata2	STD
	10	20	30	40	50		
	5	5	5	5	5		
	31	58	88	94	156		
	2.794066	2.994198	2.877186	2.888484	3.096167	2.93002	0.116965
	2.823215	2.941605	2.964307	2.909709	3.081455	2.944058	0.093644
	2.856592	3.036768	3.010358	2.967906	3.108862	2.996097	0.093314
	2.869989	2.778918	2.928859	2.859256	3.139241	2.915253	0.136132
	2.776154	2.665944	2.984182	2.80876	3.283342	2.903677	0.241047
	2.938797	2.76805	2.945033	2.88396	3.294527	2.966073	0.196866
	2.949057	2.786932	2.968569	2.901519	3.305546	2.982325	0.193956
	2.350533	2.798235	3.177307	2.842025	3.227394	2.979099	0.19336

BAB VI

SIMPULAN DAN SARAN

BAB 6
SIMPULAN DAN SARAN

Bab ini berisi beberapa simpulan dari tugas akhir yang dibuat berdasarkan hasil uji coba yang telah dilakukan. Selain itu disertakan pula saran pengembangan lebih lanjut dari tugas akhir ini.

6.1 Simpulan

1. Aplikasi yang dibuat dapat menangani penjadwalan yang dinamis dan mampu melakukan penjadwalan ulang setiap kali datang tugas baru tanpa diketahui kapan tugas tersebut akan datang. Aplikasi yang dibuat dapat menyimulasikan waktu produksi dan kedatangan tugas dengan mengubah kecepatan waktu sistem. Waktu sistem juga dapat dihentikan maupun dijalankan kembali. Dalam tugas akhir ini telah berhasil dibuat sebuah aplikasi penjadwalan *job-shop* dinamis dengan menggunakan algoritma genetika. Algoritma genetika yang digunakan berhasil membentuk suatu penjadwalan dengan waktu rata-rata produksi minimal. Sesuai dengan kriteria sebuah penjadwalan dinamis, aplikasi yang dibuat dapat mengubah penjadwalan ketika tugas baru masuk, walaupun penjadwalan sebelumnya telah terbentuk. Dengan menggunakan algoritma genetika ini pengguna dapat dengan mudah dan cepat mendapatkan hasi penjadwalan.
2. Simpulan akhir yang didapatkan dari beberapa pengujian yang telah dilakukan adalah sebagai berikut,
   - Dari uji coba kebenaran, didapatkan bahwa aplikasi yang dibuat berhasil

komputasi yang dibutuhkan juga semakin meningkat. Perubahan parameter genetik akan berpengaruh juga pada hasil penjadwalan. Beberapa parameter genetik memberikan pengaruh yang cukup signifikan terhadap hasil penjadwalan, dan beberapa parameter yang lain memberikan sedikit pengaruh terhadap hasil penjadwalan. Perubahan parameter laju seleksi, jumlah keturunan, jumlah populasi memberikan pengaruh yang sangat signifikan terhadap optimasi penjadwalan. Sedangkan perubahan parameter laju tukar silang, laju mutasi tidak memberikan pengaruh yang signifikan terhadap hasil penjawalan.

- Dari uji penggunaan memori didapatkan bahwa fungsi perhitungan nilai *fitness* populasi dan fungsi iterasi untuk menurunkan generasi berikutnya membutuhkan sumber daya memori yang lebih besar dibandingkan dengan proses genetika lainnya. Sedangkan proses tukar silang dan seleksi mengkonsumsi sumber daya memori yang paling kecil dibandingkan dengan proses yang lainnya.

.2 Saran

Salah satu pengembangan yang dapat dilakukan terhadap aplikasi yang elah dibuat berkenaan dengan strategi penanganan gangguan penjadwalan, ehingga jadwal yang dihasilkan menjadi lebih reaktif terhadap gangguan yang erjadi. Gangguan tersebut dapat berupa kerusakan mesin, penghapusan tugas ang telah masuk dan sebagainya. Kerusakan mesin sering terjadi dalam sebuah roses produksi. Hal ini menyebabkan perubahan yang cukup signifikan terhadap

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[BIER 1999] Bierwirth, C. 1999, *Production Sceduling and Rescheduling with Genetic Algorithms,* University of Bremen

[DERV 1984] Dervitsiotis, N Kostas. 1984, *Operations Management*, McGraw-Hill,Int. Singapore.

[EVN 2002] Evans, James R, 2002, *Introducing To Simulation And Risk Analysis*, Prentice Hall, USA

[GEN 1997] Gen, M. Cheng, R. 1997, *Genetic Algorithms And Engineering Design,* John Wiley & Sons, INC. New York.

[MRN 2002] Martin CM, 2002, *UML for Java Programer*, Pretince Hall, New Jersey

[MCL 1998] Mitchell, M. 1998, *An Introducing to Genetic Algorithm,* First MIT Press paperback edn, MIT Press, Inggris

[MNT 1998] Montana, David. 1998, *Genetic Algorithm for Complex, Real Time Scheduling,* BBN Technologies / GTE Internetworking,.

[SPT 2004] Saputra, Nico. et al. 2004, *Pemakaian Algoritma Genetik Untuk Penjadwalan Job Shop Dinamis non Deterministik*, Jurusan Ilmu Komputer FMIPA, Univesitas Katholik Parahyangan.

[UWA 2005] UWA Library, 2005, *Citing your sources - Harvard style*, [online], The University of Western Australia Library, bersumber dari <http://www.library..uwa.edu.au/>, [14 Maret 2005].

LAMPIRAN

Lampiran A

Hasil Uji Coba

LAMPIRAN A - HASIL UJI COBA

Hasil Pengujian Perameter Jumlah Populasi

Pada tabel 7-1 diberikan hasil pengujian parameter jumlah populasi. Tabel 7-1 menunjukkan nilai fitnes yang didapatkan menurut nilai parameter jumlah populasi dalam berbagai percobaan.

Tabel 7-1 Pergerakan Nilai Fitness Pengujian Parameter Jumlah Populasi

Percobaan	Parameter Jumlah Populasi				
	10	20	30	40	50
1	0.03311	0.03145	0.03448	0.03497	0.03521
2	0.03521	0.03067	0.03425	0.03378	0.03378
3	0.03425	0.03205	0.03521	0.03521	0.03546
4	0.03289	0.0303	0.03425	0.03425	0.03425
5	0.03472	0.03472	0.03356	0.03472	0.03425
6	0.03425	0.03205	0.03597	0.03425	0.03401
7	0.03289	0.03289	0.03356	0.03425	0.03472
8	0.03333	0.02976	0.03378	0.03356	0.03546
9	0.03425	0.03086	0.03401	0.03425	0.03378
10	0.03125	0.02809	0.03472	0.03546	0.03425
Rata	0.03362	0.03128	0.03438	0.03447	0.03452
STD	0.001153	0.001815	0.000761	0.000609	0.000655
Percobaan	**Parameter Jumlah Populasi**				
	100	**200**	**300**	**400**	**1000**
1	0.03472	0.03546	0.03597	0.03597	0.03546
2	0.03472	0.03597	0.03546	0.03546	0.03597
3	0.03472	0.03546	0.03546	0.03546	0.03546
4	0.03378	0.03521	0.03226	0.03546	0.03521
5	0.03597	0.03521	0.03521	0.03597	0.03597
6	0.03425	0.03472	0.03546	0.03597	0.03597

Pada tabel 7-2 diberikan hasil pengujian parameter jumlah populasi. Tabel -2 menunjukkan waktu komputasi yang didapatkan menurut nilai parameter umlah populasi dalam berbagai percobaan.

Tabel 7-2 Pergerakan Waktu Koputasi Pengujian Parameter Jumlah Populasi (mili-detik)

Percobaan	Parameter Jumlah Populasi				
	10	20	30	40	50
1	297	468	703	844	1093
2	203	391	515	766	1125
3	188	343	531	734	1094
4	172	359	547	907	1078
5	171	360	531	906	1130
6	172	329	719	703	1085
7	313	485	672	828	1035
8	187	343	688	891	1059
9	172	359	578	750	1152
10	313	360	656	922	1132
Rata	218.8	379.7	614	825.1	1098.3
STD	62.29642	53.65538	80.90735	81.4391	36.399176
Percobaan	Parameter Jumlah Populasi				
	100	200	300	400	1000
1	2125	4530	7391	14435	43859
2	2109	4656	7219	14565	44406
3	2156	4498	7325	14687	43609
4	2098	4437	7312	14654	43579
5	2115	4391	7198	14502	44547
6	2090	4380	7235	14760	45157
7	2095	4456	7264	14843	44844
8	2165	4650	7305	14937	44329

Hasil Pengujian Perameter Laju Mutasi

Pada tabel 7-3 diberikan hasil pengujian parameter laju mutasi. Tabel 7-3 menunjukkan nilai *fitness* yang didapatkan menurut nilai parameter laju mutasi dalam berbagai percobaan.

Tabel 7-3 Pergerakan Nilai Fitness Pengujian Parameter Laju Mutasi

Percobaan	Parameter Laju Mutasi				
	10	20	30	40	50
1	0.03497	0.03378	0.03378	0.03497	0.03106
2	0.03378	0.03472	0.03425	0.03521	0.03268
3	0.03546	0.03378	0.03472	0.03546	0.03521
4	0.03472	0.03521	0.03378	0.03356	0.03597
5	0.03546	0.03546	0.03425	0.03448	0.03425
6	0.03378	0.03378	0.03472	0.03425	0.03597
7	0.03472	0.03597	0.03472	0.03378	0.03425
8	0.03378	0.03521	0.03378	0.03472	0.03378
9	0.03289	0.03378	0.03521	0.03425	0.03497
10	0.03425	0.03425	0.03472	0.03448	0.03425
Rata	0.03438	0.0346	0.03439	0.03452	0.03424
STD	0.00083	0.000828	0.000501	0.000598	0.0015
Percobaan	Parameter Laju Mutasi				
	100	200	300	400	1000
1	0.03597	0.03067	0.03521	0.03497	0.03356
2	0.03597	0.03247	0.03472	0.03597	0.03546
3	0.03472	0.03425	0.03521	0.03425	0.03521
4	0.03378	0.03597	0.03311	0.03597	0.03546
5	0.03497	0.03356	0.03521	0.03597	0.03597
6	0.03378	0.03597	0.03546	0.03378	0.03425
7	0.03378	0.03546	0.03497	0.03521	0.03521

Pada tabel 7-4 diberikan hasil pengujian parameter laju mutasi. Tabel 2-2 menunjukkan waktu komputasi yang didapatkan menurut nilai parameter laju mutasi dalam berbagai percobaan.

Tabel 7-4 Pergerakan Waktu Komputasi Pengujian Parameter Laju Mutasi (mili-detik)

Percobaan	Parameter Laju Mutasi				
	10	20	30	40	50
1	875	938	969	1250	1297
2	719	766	860	1203	1031
3	766	797	797	985	1250
4	703	719	797	1000	1265
5	735	750	781	1016	1031
6	750	781	844	935	1000
7	735	797	781	953	1062
8	750	796	766	1032	1109
9	687	734	828	1015	1032
10	703	750	828	1062	1093
Rata	742.3	782.8	825.1	1045.1	1117
STD	52.79531	61.05881	58.93584082	102.9438	111.2375
Percobaan	Parameter Laju Mutasi				
	100	200	300	500	1000
1	1344	1391	1453	1406	1516
2	1312	1203	1187	1313	1407
3	1125	1391	1218	1250	1250
4	1078	1140	1234	1203	1297
5	1094	1109	1157	1312	1297
6	1063	1109	1157	1234	1297
7	1063	1109	1172	1250	1282
8	1109	1094	1140	1188	1343

Hasil Pengujian Parameter Laju Tukar Silang

Pada tabel 7-5 diberikan hasil pengujian parameter laju tukar silang. Tabel 7-5 menunjukkan nilai *fitness* yang didapatkan menurut nilai parameter laju tukar silang dalam berbagai percobaan.

Tabel 7-5 Pergerakan Nilai Fitness
Pengujian Parameter Laju Tukar Silang

Percobaan	Parameter Laju Tukar Silang				
	10	20	30	40	50
1	0.03378	0.03425	0.03597	0.03546	0.03497
2	0.03472	0.03401	0.03521	0.03425	0.03472
3	0.03546	0.03356	0.03521	0.03425	0.03378
4	0.03472	0.03356	0.03546	0.03378	0.03472
5	0.03546	0.03472	0.03401	0.03546	0.03497
6	0.03289	0.03378	0.03546	0.03546	0.03497
7	0.03425	0.03425	0.03521	0.03356	0.03521
8	0.03378	0.03425	0.03546	0.03472	0.03378
9	0.03546	0.03472	0.03546	0.03597	0.03356
10	0.03378	0.03378	0.03378	0.03378	0.03401
Rata	0.03443	0.03409	0.03512	0.03467	0.03447
STD	0.000882	0.000426	0.000685	0.000865	0.000614
Percobaan	Parameter Laju Tukar Silang				
	100	200	300	500	1000
1	0.03425	0.03448	0.03546	0.03472	0.03472
2	0.03472	0.03546	0.03425	0.03425	0.03378
3	0.03378	0.03521	0.03378	0.03472	0.03378
4	0.03425	0.03448	0.03546	0.03401	0.03472
5	0.03472	0.03425	0.03401	0.03425	0.03401
6	0.03546	0.03521	0.03597	0.03546	0.03425
7	0.03378	0.03425	0.03425	0.03472	0.03425

Pada tabel 7-6 diberikan hasil pengujian parameter laju tukar silang. Tabel 7-6 menunjukkan waktu komputasi yang didapatkan menurut nilai parameter laju tukar silang dalam berbagai percobaan.

Tabel 7-6 Pergerakan Waktu Komputasi
Pengujian Parameter Laju Tukar Silang (mili-detik)

Percobaan	Parameter Laju Tukar Silang				
	10	20	30	40	50
1	547	703	1015	922	1063
2	437	531	828	825	953
3	407	531	859	813	875
4	407	516	781	828	1078
5	422	515	828	865	906
6	438	531	798	750	906
7	406	515	813	812	875
8	406	562	828	953	859
9	406	563	797	812	844
10	406	562	843	812	907
Rata	428.2	552.9	839	839.2	926.6
STD	43.71066	56.24441	66.03029608	59.31236	81.76824
Percobaan	Parameter Laju Tukar Silang				
	100	200	300	500	1000
1	1094	1078	1047	1047	1015
2	985	969	906	984	922
3	907	875	953	937	938
4	891	907	1110	953	875
5	875	860	890	906	922
6	906	843	938	937	922
7	875	875	906	890	875
8	922	860	891	937	890

Hasil Pengujian Parameter Laju Seleksi

Pada tabel 7-7 diberikan hasil pengujian parameter laju seleksi. Tabel 7-7 menunjukkan nilai *fitness* yang didapatkan menurut nilai parameter laju seleksi dalam berbagai percobaan.

Tabel 7-7 Pergerakan Nilai Fitness
Pengujian Parameter Laju Seleksi

Percobaan	Parameter Laju Seleksi				
	10	20	30	40	50
1	0.03546	0.03546	0.03425	0.03472	0.03311
2	0.03448	0.03521	0.03521	0.03425	0.03546
3	0.03472	0.03597	0.03425	0.03311	0.03497
4	0.03546	0.03356	0.03333	0.03521	0.03597
5	0.03425	0.03497	0.03472	0.03521	0.03597
6	0.03521	0.03521	0.03425	0.03597	0.03497
7	0.03546	0.03497	0.03378	0.03425	0.03378
8	0.03497	0.03378	0.03546	0.03472	0.03546
9	0.03546	0.03597	0.03401	0.03472	0.03356
10	0.03521	0.03546	0.03356	0.03425	0.03521
Rata	0.03507	0.03506	0.03428	0.03464	0.03485
STD	0.000448	0.000812	0.000683	0.000763	0.0010135
Percobaan	Parameter Laju Seleksi				
	100	200	300	500	1000
1	0.03521	0.03378	0.03472	0.03378	0.03289
2	0.03378	0.03311	0.03425	0.03497	0.03521
3	0.03472	0.03597	0.03425	0.03546	0.03472
4	0.03472	0.03472	0.03472	0.03425	0.03289
5	0.03472	0.03472	0.03356	0.03378	0.03378
6	0.03497	0.03472	0.03425	0.03378	0.03247
7	0.03425	0.03333	0.03425	0.03425	0.03546

Pada tabel 7-8 diberikan hasil pengujian parameter laju seleksi. Tabel 7-8 menunjukkan waktu komputasi yang didapatkan menurut nilai parameter laju seleksi dalam berbagai percobaan.

Tabel 7-8 Pergerakan Waktu Komputasi
Pengujian Parameter Laju Seleksi (mili-detik)

Percobaan	Parameter Laju Seleksi				
	10	20	30	40	50
1	1234	1265	1125	1063	1047
2	1125	1094	875	875	922
3	1015	1078	891	875	922
4	968	843	890	859	907
5	1000	828	875	860	922
6	953	860	907	891	922
7	1016	860	860	1074	890
8	1016	844	890	891	891
9	1031	875	890	890	859
10	1109	907	907	890	860
Rata	1046.7	945.4	911	916.8	914.2
STD	85.2344	148.0902127	76.55354	80.90845	52.74635
Percobaan	Parameter Laju Seleksi				
	100	200	300	500	1000
1	844	812	687	547	110
2	813	688	547	422	47
3	765	640	719	391	31
4	766	672	719	375	32
5	797	625	531	375	16
6	797	641	516	375	15
7	781	656	515	391	31
8	906	813	547	391	31

Hasil Pengujian Parameter Jumlah Keturunan

Pada tabel 7-9 diberikan hasil pengujian parameter jumlah keturunan. Tabel 7-9 menunjukkan nilai *fitness* yang didapatkan menurut nilai parameter jumlah keturunan dalam berbagai percobaan.

Tabel 7-9 Pergerakan Nilai Fitness Pengujian Parameter Jumlah Keturunan

Percobaan	Parameter Jumlah Keturunan				
	10	50	100	250	500
1	0.03268	0.03185	0.03472	0.03472	0.03597
2	0.03185	0.03012	0.03356	0.03425	0.03597
3	0.03378	0.03401	0.03448	0.03521	0.03546
4	0.03356	0.03546	0.03378	0.03521	0.03521
5	0.03521	0.03597	0.03497	0.03597	0.03425
6	0.03497	0.03401	0.03497	0.03401	0.03546
7	0.03205	0.03378	0.03521	0.03521	0.03597
8	0.03268	0.03546	0.03472	0.03497	0.03597
9	0.03425	0.03401	0.03521	0.03597	0.03546
10	0.03311	0.03401	0.03425	0.03546	0.03597
Rata	0.03341	0.03387	0.03459	0.0351	0.03557
STD	0.001153	0.001755	0.000571	0.000646	0.00055
Percobaan	Parameter Jumlah Keturunan				
	750	1000	2500	5000	7500
1	0.03546	0.03546	0.03597	0.03597	0.03597
2	0.03546	0.03546	0.03597	0.03597	0.03597
3	0.03597	0.03546	0.03546	0.03597	0.03597
4	0.03546	0.03521	0.03597	0.03597	0.03597
5	0.03597	0.03597	0.03597	0.03597	0.03597
6	0.03497	0.03546	0.03597	0.03597	0.03597
7	0.03597	0.03597	0.03597	0.03597	0.03597

Pada tabel 7-10 diberikan hasil pengujian parameter jumlah keturunan. Tabel 7-10 menunjukkan waktu komputasi yang didapatkan menurut nilai parameter jumlah keturunan dalam berbagai percobaan.

Tabel 7-10 Pergerakan Waktu Komputasi
Pengujian Parameter Jumlah Keturunan (mili-detik)

Percobaan	Parameter Jumlah Keturunan				
	10	50	100	250	500
1	219	593	1063	2391	4438
2	109	469	938	2422	4343
3	109	609	844	2141	4265
4	93	484	891	2187	4281
5	94	468	937	2187	4219
6	93	453	1063	2125	4281
7	94	422	843	2094	4172
8	94	453	859	2078	4250
9	250	453	906	2204	4203
10	94	453	1016	2203	4203
Rata	124.9	485.7	936	2203.2	4265.5
STD	58.56136	62.93392	84.74144	116.0707	78.31596
Percobaan	Parameter Jumlah Keturunan				
	750	1000	2500	5000	7500
1	6563	8765	21109	42047	63109
2	6343	9234	21594	41891	62953
3	6500	8532	21235	41547	62203
4	6406	8375	21360	41797	63016
5	6281	8922	21297	43079	62609
6	6250	8406	21344	42734	62015
7	6297	8422	21203	42734	63015
8	6265	8578	21187	43719	62953

Data Pengujian Jadwal Ulang

Data pada lampiran ini merupakan yang akan diujikan dalam pengujian jadwal ulang. Dataset yang dilampirkan berisikan data mengenai tugas beserta lama waktu proses dan mesin yang digunakan.. Data yang dilampirkan merupakan dataset kecil dan dataset besar. Misal, Job 1 pada dataset kecil akan diproses pada mesin 2, 1, 5, 3 dengan lama proses, dalam menit, 4, 4, 3, 7.

Tabel 7-11 dataset kecil pengujian jadwal ulang

	Operasi 1		Operasi 2		Operasi 3		Operasi 4		Operasi 5	
	Mesin	W.P	Mesin	W.P	Mesin	W.P	Mesin	W.P	Mesin	W.P
1	2	4	1	4	5	3	3	7		
2	5	3	3	7	2	4	1	6	4	4
3	4	4	3	6	2	3	1	5	5	4
4	2	3	3	4	5	3				
5	2	7	4	3	3	5	1	5	5	4
6	2	7	4	5	1	6				
7	4	5	1	5	5	6	2	3		
8	1	3	2	3	3	7				
9	2	16	3	10						
10	4	13	3	8	5	3				
11	1	6	3	2	5	7	2	7		
12	1	8	5	17	4	16	3	28	2	18
13	3	20								
14	4	28	2	5	3	28	1	6		
15	1	17	4	8	5	15	3	17		
16	2	22	5	10	1	10				
17	1	2	4	3	2	26	5	1	3	4
18	2	7	4	5	1	6				
19	4	5	1	5	5	6	2	3		
20	1	3	2	3	3	7				

Tabel 7-12 dataset besar pengujian jadwal ulang

	Operasi 1		Operasi 2		Operasi 3		Operasi 4		Operasi 5	
	Mesin	W.P	Mesin	W.P	Mesin	W.P	Mesin	W.P	Mesin	W.P
1	5	28	4	20	1	10				
2	3	26	1	3	2	24	5	4		
3	4	15	5	25	1	14				
4	3	24	5	24						
5	1	8	5	6	2	27	3	10	4	18
6	2	20	1	14	3	27				
7	3	25	1	19	5	19				
8	1	2	4	15	5	16	3	14		
9	3	23								
10	1	27	4	27						
11	3	11	4	26	1	26	2	25	5	6
12	1	10	2	25	5	9	3	18		
13	4	16	3	3	2	8	5	22	1	5
14	3	19	1	11	4	26	5	12		
15	5	26	1	1	4	16	3	10		
16	2	19	1	11	4	10	3	29	5	14
17	4	16	2	20	1	5	3	10	5	23
18	4	2	1	9	3	26	5	6	2	28
19	4	28	3	24	5	25	2	12		
20	5	28	4	4	3	23	1	21	2	29
21	3	12	5	23	2	11	1	22		
22	1	3	3	18						
23	4	14	3	12	5	2				
24	5	16	3	8	4	24	2	8		
25	4	15	5	10	3	26				
26	3	1	4	22						
27	5	16	2	23						
28	2	11	4	15	5	12				
29	5	16	2	23						
30	2	11	4	15	5	12				

Data Pengujian Penggunaan Memori

Data pada lampiran ini merupakan skenario yang diujikan dalam pengujian penggunaan memori. Dataset yang dilampirkan berisikan tugas beserta lama waktu proses dan mesin yang digunakan. Misal, Job 1 pada scenario 1 akan diproses pada mesin 1, 3, 5, 4, 2 dengan lama proses, dalam menit, 7, 7, 21, 8,1.

Tabel 7-13 skenario 1 pengujian penggunaan memori

Skenario 1					
gas	Operasi 1	Operasi 2	Operasi 3	Operasi 4	Operasi 5
	Mesin-1 (7)	Mesin-3 (7)	Mesin-5 (21)	Mesin-4 (8)	Mesin-2 (1)
	Mesin-5 (21)	Mesin-3 (13)	Mesin-1 (1)	Mesin-2 (7)	
	Mesin-1 (7)	Mesin-3 (15)	Mesin-4 (17)	Mesin-2 (1)	
	Mesin-5 (2)	Mesin-4 (27)			
	Mesin-2 (7)	Mesin-4 (17)	Mesin-5 (21)		
	Mesin-4 (17)	Mesin-3 (10)			
	Mesin-3 (7)	Mesin-4 (8)	Mesin-1 (28)		
	Mesin-1 (28)	Mesin-5 (16)			
	Mesin-2 (1)	Mesin-1 (11)	Mesin-4 (13)		
)	Mesin-4 (8)	Mesin-5 (2)	Mesin-3 (7)		

Tabel 7-14 skenario 2 pengujian penggunaan memori

Skenario 2					
gas	Operasi 1	Operasi 2	Operasi 3	Operasi 4	Operasi 5
	Mesin-5 (10)	Mesin-1 (25)	Mesin-2 (24)	Mesin-4 (15)	
	Mesin-5 (11)				
	Mesin-2 (9)	Mesin-5 (8)			
	Mesin-3 (5)	Mesin-5 (10)	Mesin-2 (23)	Mesin-4 (9)	Mesin-1 (22)
	Mesin-4 (25)				
	Mesin-5 (15)				
	Mesin-4 (26)	Mesin-5 (8)	Mesin-3 (12)	Mesin-1 (22)	Mesin-2 (11)
	Mesin-5 (6)	Mesin-4 (21)	Mesin-3 (22)	Mesin-1 (18)	
	Mesin-2 (6)	Mesin-1 (25)			

Tabel 7-15 skenario 3 pengujian penggunaan memori

Skenario 3					
gas	Operasi 1	Operasi 2	Operasi 3	Operasi 4	Operasi 5
	Mesin-4 (1)	Mesin-5 (13)	Mesin-2 (24)	Mesin-1 (8)	Mesin-3 (23)
	Mesin-3 (22)	Mesin-4 (12)	Mesin-5 (28)	Mesin-1 (25)	Mesin-2 (15)
	Mesin-4 (7)	Mesin-5 (21)	Mesin-1 (3)		
	Mesin-5 (7)	Mesin-3 (4)	Mesin-4 (8)	Mesin-2 (1)	Mesin-1 (1)
	Mesin-1 (13)	Mesin-5 (12)			
	Mesin-4 (22)	Mesin-2 (4)	Mesin-5 (3)	Mesin-3 (21)	
	Mesin-3 (7)	Mesin-1 (21)	Mesin-4 (3)		
	Mesin-3 (13)	Mesin-1 (6)	Mesin-2 (11)	Mesin-5 (19)	
	Mesin-3 (26)	Mesin-1 (24)			
0	Mesin-3 (17)	Mesin-4 (15)			
1	Mesin-3 (16)	Mesin-5 (2)	Mesin-4 (16)		
2	Mesin-3 (10)	Mesin-5 (8)	Mesin-1 (5)	Mesin-2 (7)	
3	Mesin-4 (10)	Mesin-5 (29)	Mesin-1 (8)		
4	Mesin-3 (19)				
5	Mesin-5 (13)	Mesin-2 (9)	Mesin-1 (7)	Mesin-3 (20)	Mesin-4 (12)
6	Mesin-1 (3)	Mesin-2 (20)	Mesin-4 (18)	Mesin-5 (28)	
7	Mesin-3 (3)	Mesin-5 (9)	Mesin-4 (15)	Mesin-1 (22)	Mesin-2 (16)
8	Mesin-5 (11)	Mesin-2 (1)			
9	Mesin-1 (16)	Mesin-5 (18)			
0	Mesin-5 (5)				
1	Mesin-4 (1)	Mesin-5 (9)	Mesin-1 (19)	Mesin-2 (1)	
2	Mesin-4 (16)	Mesin-2 (24)	Mesin-1 (2)		
3	Mesin-2 (20)	Mesin-1 (3)			
4	Mesin-4 (7)				
5	Mesin-4 (9)	Mesin-3 (3)	Mesin-2 (23)	Mesin-1 (14)	Mesin-5 (20)
6	Mesin-3 (29)	Mesin-2 (12)	Mesin-1 (2)	Mesin-5 (29)	Mesin-4 (10)
7	Mesin-5 (29)	Mesin-3 (16)	Mesin-4 (14)	Mesin-2 (1)	Mesin-1 (25)
8	Mesin-1 (6)	Mesin-2 (8)			
9	Mesin-5 (7)	Mesin-4 (11)	Mesin-2 (12)	Mesin-3 (22)	Mesin-1 (4)
0	Mesin-2 (14)	Mesin-1 (12)	Mesin-3 (10)		

Tabel 7-16 skenario 4 pengujian penggunaan memori

Skenario 4					
gas	Operasi 1	Operasi 2	Operasi 3	Operasi 4	Operasi 5
	Mesin-3 (29)	Mesin-4 (18)	Mesin-5 (8)		
	Mesin-5 (10)	Mesin-3 (26)	Mesin-4 (14)		
	Mesin-2 (2)	Mesin-4 (2)	Mesin-3 (3)	Mesin-1 (23)	Mesin-5 (24)
	Mesin-4 (8)	Mesin-2 (25)			
	Mesin-3 (13)	Mesin-1 (28)	Mesin-5 (9)	Mesin-4 (9)	
	Mesin-3 (17)	Mesin-1 (8)	Mesin-2 (6)	Mesin-4 (28)	Mesin-5 (18)
	Mesin-5 (19)	Mesin-1 (12)			
	Mesin-2 (10)				
	Mesin-2 (18)				
0	Mesin-5 (18)				
1	Mesin-2 (18)	Mesin-3 (6)	Mesin-1 (27)	Mesin-5 (20)	
2	Mesin-5 (3)	Mesin-4 (3)	Mesin-1 (14)	Mesin-3 (26)	
3	Mesin-3 (24)	Mesin-4 (12)	Mesin-2 (4)		
4	Mesin-4 (15)	Mesin-1 (5)	Mesin-5 (24)	Mesin-2 (7)	Mesin-3 (20)
5	Mesin-1 (23)	Mesin-2 (13)			
6	Mesin-1 (17)	Mesin-5 (17)	Mesin-4 (26)	Mesin-2 (23)	Mesin-3 (16)
7	Mesin-4 (9)				
8	Mesin-5 (2)	Mesin-2 (24)	Mesin-4 (6)		
9	Mesin-1 (29)	Mesin-4 (26)			
0	Mesin-3 (16)				
1	Mesin-2 (16)	Mesin-1 (29)	Mesin-3 (11)	Mesin-4 (10)	Mesin-5 (7)
2	Mesin-1 (18)				
3	Mesin-4 (19)				
4	Mesin-2 (26)	Mesin-4 (29)	Mesin-5 (28)		
5	Mesin-4 (27)				
6	Mesin-2 (28)				
7	Mesin-1 (18)	Mesin-2 (18)			
8	Mesin-3 (8)	Mesin-4 (18)			
9	Mesin-5 (13)	Mesin-2 (15)	Mesin-4 (23)	Mesin-3 (3)	
0	Mesin-2 (17)	Mesin-3 (6)	Mesin-5 (23)		
1	Mesin-3 (26)	Mesin-5 (6)	Mesin-1 (19)	Mesin-4 (4)	Mesin-2 (18)