



**PENERAPAN ALGORITMA GENETIKA
DAN ALGORITMA *SIMULATED ANNEALING*
PADA PENJADWALAN *FLOWSHOP***

SKRIPSI

Oleh

**Weny Duwi Nurjanah
NIM 091810101054**

**JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015**



**PENERAPAN ALGORITMA GENETIKA
DAN ALGORITMA *SIMULATED ANNEALING*
PADA PENJADWALAN *FLOWSHOP***

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarjana Sains

Oleh

**Weny Duwi Nurjanah
NIM 091810101054**

**JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015**

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Ibunda Samroten dan ayahanda Supriyanto serta keluarga yang tanpa putus mendoakan dan memotivasi;
2. saudara-saudaraku Hermawan dan Sofiyana yang selalu menjadi semangat;
3. guru-guruku sejak taman kanak-kanak hingga perguruan tinggi;
4. Almamater Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember, MAN 1 Jember, SMP 2 Wuluhan-Jember, SDN Tegal Harjo IX Krikilan-Banyuwangi, TK Dahlia Karangsono-Jember.

MOTTO

Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari suatu urusan), tetaplah bekerja keras (untuk urusan yang lain), dan hanya kepada Tuhan-mulah engkau berharap. (terjemahan surat *Al-Insyirah* ayat 6-8) ^{*)}

Dua pejuang yang paling berkuasa adalah kesabaran dan waktu
(Leo Nikolaevich Tolstoy (1828-1910)) ^{**)}

^{*)} Departemen Agama Republik Indonesia. 2005. *Al-Qur'an dan Terjemahannya*. Bandung: CV Penerbit Diponegoro.

^{**)} ibenx. 2008. *Kata-kata Mutiara dan Bijak dari Para Pakar*.

<http://ibenxs.wordpress.com/other/kata-kata-mutiara-dan-bijak-dari-para-pakar/> [7 Desember 2014]

PERNYATAAN

Saya yang bertanda tangan dibawah ini :

Nama : Weny Duwi Nurjanah

NIM : 091810101054

Menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul “Penerapan Algoritma Genetika dan Algoritma *Simulated Annealing* pada Penjadwalan *Flowshop*” adalah benar-benar hasil karya sendiri, kecuali kutipan yang telah saya sebutkan sumbernya, belum pernah diajukan pada institusi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata dikemudian hari pernyataan ini tidak benar.

Jember, 2 Maret 2015

Yang menyatakan,

Weny Duwi Nurjanah
091810101054

SKRIPSI

**PENERAPAN ALGORITMA GENETIKA DAN
ALGORITMA *SIMULATED ANNEALING* PADA
PENJADWALAN *FLOWSHOP***

Oleh

Weny Duwi Nurjanah
NIM 091810101054

Pembimbing

Dosen Pembimbing Utama : Kiswara Agung Santoso, S.Si., M.Kom.

Dosen Pembimbing Anggota : Ahmad Kamsyakawuni, S.Si., M.Kom.

PENGESAHAN

Skripsi berjudul “Penerapan Algoritma Genetika dan Algoritma *Simulated Annealing* pada Penjadwalan *Flowshop*” telah diuji dan disahkan pada :

Hari, tanggal :

Tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji :

Ketua,

Sekretaris,

Kiswara Agung Santoso, S.Si., M.Kom.
NIP 197209071998031003

Ahmad Kamsyakawuni S.Si, M.Kom.
NIP 197211291998021001

Anggota I,

Anggota II,

Kusbudiono, S.Si., M.Si.
NIP 197704302005011001

Dian Anggraeni, S.Si., M.Si.
NIP 198202162006042002

Mengesahkan
Dekan,

Prof. Drs. Kusno, DEA, Ph.D.
NIP 196101081986021001

RINGKASAN

Penerapan Algoritma Genetika dan Algoritma *Simulated Annealing* pada Penjadwalan *Flowshop*; Weny Duwi Nurjanah, 091810101054; 2015: 57 halaman: Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penjadwalan adalah suatu proses pengambilan keputusan yang memainkan peranan penting dalam kebanyakan bidang manufaktur dan pelayanan industri. Penjadwalan juga merupakan alat ukur yang baik bagi perencanaan industri, untuk jangka pendek dalam rentang periode beberapa hari sampai satu bulan, perusahaan harus melakukan penjadwalan produksi untuk memenuhi order atau permintaan konsumen. Secara umum penjadwalan merupakan suatu proses dalam perencanaan dan pengendalian produksi serta pengalokasian sumber daya pada suatu waktu tertentu dengan memperhatikan kapasitas sumber daya yang ada, sehingga diperlukan adanya pengaturan sumber daya yang ada secara efisien. Penjadwalan *flowshop* adalah salah satu jenis penjadwalan produksi dimana setiap *job* akan melalui setiap mesin dengan urutan yang seragam.

UD. Samjaya merupakan salah satu industri yang memproduksi kerupuk jenis padat dan cair. Dalam proses pembuatan kerupuk terdapat 9 mesin yang digunakan dalam proses produksinya dengan 10 pekerjaan. Dalam proses produksinya, industri UD.Samjaya tidak memiliki jadwal yang tetap, penumpukan pesanan dari banyaknya permintaan pelanggan, sistem penjadwalan perusahaan yang kurang tepat sehingga dapat mengakibatkan keterlambatan dalam pengiriman barang.

Tujuan yang ingin dicapai dalam penulisan skripsi ini adalah mencari solusi terbaik berdasarkan *makespan* optimal dengan algoritma Genetika dan algoritma *Simulated Annealing*. Tujuan berikutnya yaitu mengetahui hasil perbandingan kedua algoritma tersebut berdasarkan efektifitas algoritma dan tingkat kecepatan konvergensi.

Penelitian dilakukan melalui beberapa langkah yang terdiri dari mengolah data yang diperoleh menjadi data urutan mesin dan waktu proses kemudian menjadwalkan dengan kedua algoritma, dan menentukan algoritma yang cepat konvergen. Langkah selanjutnya membandingkan hasil kedua algoritma berdasarkan *makespan*, kompleksitas waktu, dan kecepatan konvergensi yang diperoleh. Langkah terakhir adalah menentukan kesimpulan berdasarkan perbandingan sebelumnya.

Hasil penelitian yang dilakukan dapat dilihat bahwa *makespan* terbaik dari 14 kali pengujian dengan menggunakan algoritma Genetika dan algoritma *Simulated Annealing* menghasilkan nilai *makespan* sebesar 8878 menit. Artinya penggunaan kedua algoritma memiliki efektifitas yang sama dalam penjadwalan produksi kerupuk.

Apabila ditinjau dari perhitungan kompleksitas waktu yang dihasilkan, algoritma Genetika dan algoritma *Simulated Annealing* memiliki kompleksitas waktu yang sama pada produksi kerupuk. Dengan kata lain menurut kompleksitas waktu yang diperoleh dapat dikatakan algoritma Genetika dan algoritma *Simulated Annealing* mempunyai tingkat efisiensi yang sama. Sedangkan berdasarkan tingkat konvergensi, algoritma Genetika lebih cepat konvergen dibanding dengan algoritma *Simulated Annealing*.

PRAKATA

Puji syukur ke hadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul “Penerapan Algoritma Genetika dan Algoritma *Simulated Annealing* pada Penjadwalan *Flowshop*”. Skripsi ini disusun untuk memenuhi salah satu syarat untuk menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penyusun skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada :

1. Prof. Drs. Kusno, DEA., Ph.D. selaku Dekan Fakultas MIPA;
2. Kiswara Agung Santoso, S.Si, M.Kom. selaku Dosen Pembimbing Utama dan Ahmad Kamsyakawuni, S.Si, M.Kom. selaku Dosen Pembimbing Anggota yang telah banyak memberikan arahan dan bimbingan sehingga skripsi ini terselesaikan dengan baik;
3. Kusbudiono, S.Si., M.Si. dan Dian Anggraeni, S.Si., M.Si., selaku Dosen Penguji yang telah memberikan kritik dan saran sehingga skripsi ini menjadi lebih baik;
4. ibunda dan ayahanda serta keluarga yang selalu memberikan doa dan dukungan;
5. teman-teman kos Kalimantan 39 serta teman-teman angkatan 2009 yang telah menemani dan membantu dalam menyelesaikan skripsi ini.

Penulis menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap semoga skripsi ini dapat bermanfaat.

Jember, 2 Maret 2015

Penulis

DAFTAR ISI

	Halaman
HALAMAN JUDUL	ii
HALAMAN PERSEMBAHAN	iii
HALAMAN MOTTO	iv
HALAMAN PERNYATAAN	v
HALAMAN PEMBIMBING	vi
HALAMAN PENGESAHAN	vii
RINGKASAN	viii
PRAKATA	x
DAFTAR ISI	xi
DAFTAR TABEL	v
DAFTAR GAMBAR	vi
DAFTAR LAMPIRAN	vii
BAB 1. PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	4
1.4 Tujuan	4
1.5 Manfaat	4
BAB 2. TINJAUAN PUSTAKA	5
2.1 Definisi Penjadwalan	5
2.2 Penjadwalan <i>Flowshop</i>	6
2.3 <i>Permutation Flowshop Scheduling Problem (PFSP)</i>	7
2.4 Algoritma, Efisiensi Algoritma, dan Notasi <i>Big-O</i>	8
2.4.1 Algoritma	8
2.4.2 Efisiensi Algoritma	9
2.4.3 Notasi <i>Big-O</i>	10

2.5	Algoritma Genetika	11
2.5.1	Pengertian Individu.....	12
2.5.2	Komponen-Komponen Utama Algoritma Genetika	13
2.6	Algoritma <i>Simulated Annealing</i>	17
BAB 3.	METODE PENELITIAN	22
3.1	Data Penelitian	22
3.2	Pengolahan Data dengan Algoritma Genetika.....	23
3.3	Pengolahan Data dengan Algoritma <i>Simulated Annealing</i>	26
3.4	Membuat <i>flowchart</i> dari kedua Algoritma	27
3.5	Menghitung Kompleksitas Waktu dari kedua Algoritma	27
3.6	Menghitung Performa dari kedua Algoritma	27
BAB 4.	HASIL DAN PEMBAHASAN	29
4.1	Penyelesaian Penjadwalan Secara Manual	29
4.1.1	Penjadwalan Menggunakan Algoritma Genetika	29
4.1.2	Penjadwalan Menggunakan Algoritma <i>Simulated Annealing</i>	34
4.2	Penyelesaian dengan Menggunakan Program	37
4.3	Perbandingan Algoritma Genetika dan Algoritma <i>Simulated Annealing</i>	44
4.3.1	Perhitungan Kompleksitas Waktu	44
4.3.2	Kecepatan Konvergensi	57
BAB 5.	PENUTUP	58
5.1	Kesimpulan	58
5.2	Saran	59
DAFTAR PUSTAKA		
LAMPIRAN		

DAFTAR TABEL

	Halaman
2.1 Daftar Notasi <i>Big-O</i>	10
2.2 Perbandingan Pertumbuhan $T(n)$ dengan n^2	10
2.3 Contoh Pengkodean Permutasi.....	14
2.4 Analogi Proses Fisika dan <i>SA</i> pada Penjadwalan <i>Flowshop</i>	21
3.1 Data Waktu Pembuatan Kerupuk (menit)	22
3.2 Pencarian Nilai <i>Makespan</i> dengan i job dan j mesin.....	24
4.1 Waktu Proses Produksi dengan 5 job dan 4 Mesin (detik).....	29
4.2 Perhitungan <i>Makespan</i> Kromosom Pertama yang dibangkitkan pada Solusi Awal.....	30
4.3 Perhitungan <i>Makespan</i> Kromosom Kedua yang dibangkitkan pada Solusi Awal.....	30
4.4 Perhitungan <i>Makespan</i> Kromosom Ketiga yang dibangkitkan pada Solusi Awal.....	31
4.5 Perhitungan <i>Makespan</i> Kromosom Keempat yang dibangkitkan pada Solusi Awal.....	31
4.6 Perhitungan <i>Makespan</i> Kromosom Kelima yang dibangkitkan pada Solusi Awal.....	31
4.7 Penentuan Induk	32
4.8 Kromosom induk 1 dan induk 2	32
4.9 Calon anak 1 dan anak 2.....	32
4.10 anak 1 dan anak 2 dengan Relasi Pemetaan	33
4.11 Generasi Baru Setelah Seleksi.....	34
4.12 Populasi pada Generasi kedua	34
4.13 Hasil Pertukaran untuk Setiap Iterasi	36

DAFTAR GAMBAR

	Halaman
2.1 Jalur proses <i>pure flowshop</i>	6
2.2 Jalur proses <i>general flowshop</i>	6
2.3 Diagram <i>gantt</i>	8
2.4 Ilustrasi representasi penyelesaian permasalahan dalam algoritma genetika	13
2.5 Ilustrasi Prosedur <i>PMX</i>	16
2.6 Skema prosedur algoritma Genetika	17
2.7 Ilustrasi metode <i>insertion</i>	19
2.8 Skema prosedur algoritma <i>Simulated Annealing</i>	21
3.1 Interpretasi kromosom	23
3.2 Pengodean <i>crossover two point</i>	25
3.3 Pengodean dengan metode <i>insertion</i>	26
3.4 Skema langkah penyelesaian masalah	28
4.1 Relasi pemetaan	33
4.2 Tampilan program RESET	38
4.3 Konvergensi GA dan SA	39
4.4 Tingkat konvergensi algoritma Genetika dan <i>Simulated Annealing</i>	40
4.5 Penurunan temperatur pada algoritma <i>Simulated Annealing</i>	40
4.6 Hasil perhitungan manual nilai <i>makespan</i>	41
4.7 Diagram <i>gant</i> hasil akhir dari algoritma Genetika	42
4.8 Diagram <i>gant</i> hasil akhir dari algoritma <i>Simulated Annealing</i>	42
4.9 Gambar <i>flowchart</i> algoritma Genetika	44
4.10 Gambar <i>flowchart</i> algoritma <i>Simulated Annealing</i>	51
4.11 Tingkat kecepatan konvergensi algoritma	57

DAFTAR LAMPIRAN

	Halaman
A. Gambar proses produksi UD. Samjaya	62
B. Tampilan <i>insert</i> data dengan jumlah pesana yang ditambahkan	63



BAB 1. PENDAHULUAN

1.1 Latar Belakang

Adanya kompetisi pasar global yang semakin kuat telah menimbulkan tantangan dalam perusahaan manufaktur untuk beroperasi dengan biaya produksi yang rendah, pada kondisi persaingan perusahaan industri yang semakin ketat semua manajemen perusahaan pasti menginginkan pekerjaannya dapat terlaksanakan secara efektif dan efisien sehingga memenuhi kepuasan konsumen, yaitu dengan menyelesaikan order tepat waktu, kualitas bagus, dan harga yang terjangkau. Salah satu cara yang digunakan adalah dengan memanfaatkan penggunaan mesin-mesin secara efektif. Untuk mendapatkan hasil yang optimal dengan keterbatasan mesin yang dimiliki, maka perlu dilakukan penjadwalan.

Penjadwalan merupakan suatu kegiatan pengalokasian sumber daya yang terbatas untuk mengerjakan sejumlah pekerjaan. Proses penjadwalan timbul jika terdapat keterbatasan sumber daya yang dimiliki. Permasalahan penjadwalan biasanya berkaitan dengan pengurutan pembuatan atau pengerjaan produk secara menyeluruh terhadap sejumlah mesin yang tersedia. Penjadwalan jenis ini biasa disebut dengan penjadwalan produksi.

Penjadwalan produksi merupakan kegiatan perencanaan produksi yang terdapat pada perusahaan manufaktur. Penjadwalan produksi melibatkan n pekerjaan dan m mesin dalam proses produksinya, dimana setiap pekerjaan mengandung informasi tentang jenis-jenis produk. Adapun tujuan dari penjadwalan produksi adalah minimasi *makespan* (total waktu yang dibutuhkan untuk menyelesaikan semua pekerjaan). Penjadwalan produksi dibedakan menjadi dua, yaitu penjadwalan *jobshop* dan *flowshop*. Penjadwalan produksi *flowshop* merupakan proses penjadwalan produksi dimana setiap pekerjaan akan diproses tepat satu kali pada setiap mesin

dengan urutan yang seragam, yaitu dari mesin pertama, mesin kedua sampai mesin ke- m . Penyelesaian masalah penjadwalan *flowshop* dapat menggunakan metode heuristik.

Metode heuristik adalah metode yang dimulai dari sebuah atau sekumpulan solusi awal, kemudian melakukan pencarian terhadap solusi yang lebih baik hingga mendekati solusi optimal. Kelebihan metode heuristik adalah tidak perlu menganalisa semua kemungkinan solusi untuk mendapatkan solusi optimal dan mampu mendapatkan solusi yang mendekati solusi optimal dengan waktu komputasi yang relatif singkat. Metode heuristik meliputi beberapa algoritma diantaranya *Tabu Search*, *Simulated Annealing*, Genetika, *Memetic*, Hamid, FRB, *Ho and Chang* dan *Ant Colony*. Mariani (2009) menggunakan metode Hamid, *Simulated Annealing*, dan metode FRB (*Fuzzy Rule-Base*) untuk penjadwalan *flowshop* dengan data simulasi diperoleh kesimpulan bahwa *Simulated Annealing* yang paling baik dibandingkan dengan kedua metode yang lain. Laila (2014) menggunakan metode heuristik yaitu algoritma *Simulated Annealing* dan *Tabu Search* untuk penjadwalan *flowshop* dengan data primer dan didapatkan kesimpulan bahwa pada algoritma *Simulated Annealing* pada saat membangkitkan solusi baru dengan menggunakan metode *swap* memiliki tingkat kekonvergenan lebih cepat dibanding dengan algoritma *Tabu Search*. Dari perbandingan penggunaan metode heuristik tersebut didapatkan bahwa algoritma *Simulated Annealing* lebih baik dari pembanding algoritma yang lain. Sedangkan ide dari algoritma *Simulated Annealing* tersebut merupakan analogi proses *annealing* (pendinginan) material panas. Algoritma ini melakukan peningkatan iteratif untuk memperbaiki solusi yang dihasilkan teknik-teknik penjadwalan heuristik ataupun random.

Selain algoritma *Simulated Annealing*, algoritma Genetika juga dapat digunakan dalam penjadwalan *flowshop*. Tessa (1999) menggunakan algoritma *Ho and Chang* dan algoritma Genetika untuk penjadwalan *flowshop* dengan data simulasi diperoleh kesimpulan bahwa algoritma Genetika lebih unggul daripada hasil penjadwalan oleh algoritma *Ho and Chang*. Andriyanti (2014) pada penelitiannya

menggunakan algoritma Genetika dan *Tabu Search* untuk penjadwalan *flowshop* dengan data primer dimana pada algoritma Genetika saat proses evolusinya menggunakan mutasi namun didapatkan kesimpulan bahwa algoritma *Tabu Search* memiliki nilai makespan yang lebih kecil sehingga algoritma *Tabu Search* lebih baik diterapkan pada penelitiannya. Sedangkan, ide dari algoritma Genetika adalah algoritma pencarian solusi tetangga berdasarkan mekanisme seleksi alam dan genetika alam. Algoritma genetika dapat mengacu pada semua metode pencarian solusi tetangga dengan mensimulasikan proses evolusi alam. Dalam penelitian ini, peneliti akan mengembangkan penelitian sebelumnya (Andriyanti 2014) yang mana pada perhitungan *makespan* jumlah job tidak harus sama dengan jumlah mesin yang tersedia.

UD. Samjaya merupakan salah satu industri yang memproduksi kerupuk jenis padat dan cair, berdiri sejak tahun 1984 yang terletak di Jalan Galunggung-Klatangan-Tanggul, Jember. Dalam proses pembuatan kerupuk terdapat 9 mesin yang digunakan dalam proses produksinya dengan 10 pekerjaan. pekerjaan yang dimaksud merupakan jenis kerupuk adonan padat yang diproduksi, dengan setiap pekerjaan diproses tepat satu kali pada setiap mesin dengan urutan yang sama. Dalam proses produksinya, industri ini tidak memiliki jadwal yang tetap. Penumpukan pesanan dari banyaknya permintaan pelanggan, sistem penjadwalan perusahaan yang kurang tepat, dapat mengakibatkan keterlambatan penyelesaian produksi, sehingga terjadi keterlambatan dalam pengiriman barang. Oleh karena itu, diharapkan dengan menjadwalkan produksi menggunakan kedua algoritma Genetika dan *Simulated Annealing* diperoleh jadwal yang optimal dalam proses produksi kerupuk adonan cair.

1.2 Rumusan Masalah

Permasalahan yang akan dibahas dalam skripsi ini adalah:

- a. bagaimana mengaplikasikan algoritma *Simulated Annealing* dan algoritma genetika pada permasalahan penjadwalan *flowshop* pada UD. Samjaya;

- b. bagaimana hasil perbandingan kedua algoritma yang diterapkan pada UD. Samjaya berdasarkan nilai *makespan*, efektifitas algoritma dan tingkat kecepatan kekonvergenan.

1.3 Batasan Masalah

Permasalahan dalam penulisan skripsi ini dibatasi oleh beberapa hal. Batasan tersebut adalah sebagai berikut:

- a. diasumsikan setiap bahan mentah yang dibutuhkan selalu tersedia;
- b. semua *job* mempunyai *ready time* yang sama dan bersifat *independent* terhadap yang lain;
- c. setiap mesin selalu dalam keadaan siap untuk mengerjakan *job-job* yang ada tanpa adanya gangguan;
- d. pada saat produksi berjalan, diasumsikan tidak terjadi penambahan pesanan baru;
- e. pada saat membawa *job* ke mesin lainnya, diasumsikan waktu diabaikan.

1.4 Tujuan

Adapun tujuan dari penulisan skripsi ini adalah:

- a. menerapkan algoritma *Simulated Annealing* dan algoritma Genetika dalam menyelesaikan permasalahan penjadwalan produksi *flowshop* pada UD. Samjaya;
- b. mengetahui hasil perbandingan kedua algoritma yang diterapkan pada UD. Samjaya berdasarkan nilai *makespan*, efektifitas algoritma dan tingkat kecepatan kekonvergenan.

1.5 Manfaat

Adapun manfaat yang diharapkan dari penulisan skripsi ini adalah mendapatkan jadwal produksi kerupuk pada UD. Samjaya yang efektif dengan meminimumkan *makespan* menggunakan algoritma *Simulated Annealing* dan genetika beserta memberikan alternatif metode untuk penyelesaian masalah penjadwalan sehingga dapat meningkatkan keuntungan bagi industri tersebut.

BAB 2. TINJAUAN PUSTAKA

2.1 Definisi Penjadwalan

Penjadwalan merupakan suatu proses pengambilan keputusan yang memainkan peranan penting dalam kebanyakan bidang manufaktur dan pelayanan industri. Penjadwalan digunakan dalam pengadaan bahan dan produksi dalam bidang transportasi dan distribusi serta dalam proses informasi dan komunikasi. Penjadwalan juga merupakan alat ukur yang baik bagi perencanaan industri, untuk jangka pendek dalam rentang periode beberapa hari sampai satu bulan, perusahaan harus melakukan penjadwalan produksi untuk memenuhi order atau permintaan konsumen.

Secara umum penjadwalan merupakan suatu proses dalam perencanaan dan pengendalian produksi yang merencanakan produksi serta pengalokasian sumber daya pada suatu waktu tertentu dengan memperhatikan kapasitas sumberdaya yang ada.

Menurut Barry Render (1997) pentingnya penjadwalan bagi perusahaan antara lain (Hartanto, 2007) :

- a. Dengan penjadwalan secara efektif, perusahaan menggunakan asetnya dengan efektif dan menghasilkan kapasitas uang yang diinvestasikan menjadi lebih besar dan dapat mengurangi biaya.
- b. Penjadwalan menambah kapasitas dan fleksibilitas yang terkait, memberikan waktu pengiriman yang lebih cepat dan dengan demikian pelayanan kepada pelanggan menjadi lebih baik.
- c. Keuntungan ketiga dari penjadwalan yang baik adalah keunggulan kompetitif dengan pengiriman yang dapat diandalkan.

Penjadwalan dibedakan menjadi dua, yaitu penjadwalan *flowshop* dan *jobshop*.

Penjadwalan *flowshop* merupakan proses pengurutan pekerjaan dalam setiap

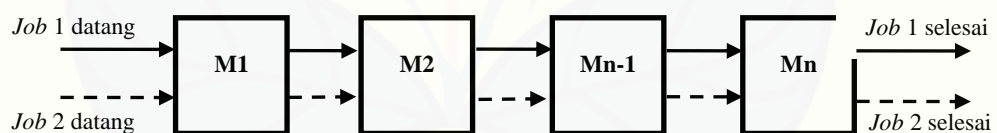
mesin dimana urutan mesin pada setiap pekerjaan sama. Serupa dengan penjadwalan *flowshop*, penjadwalan *jobshop* merupakan proses pengurutan pekerjaan dalam setiap mesin tetapi masing-masing pekerjaan dapat mempunyai urutan mesin yang berbeda, dan juga bisa terjadi satu mesin dilewati lebih dari satu kali oleh pekerjaan yang sama. Perbedaan utama antara *flowshop* dan *jobshop* adalah urutan mesin pada *jobshop* alirannya tidak searah.

2.2 Penjadwalan *Flowshop*

Proses produksi dengan aliran *flowshop* berarti proses produksi dengan pola aliran identik dari satu mesin ke mesin lain. Dalam *flowshop* setiap pekerjaan dari n job harus di proses melalui m mesin untuk permintaan yang sama dan setiap *job* diproses satu kali untuk setiap mesin.

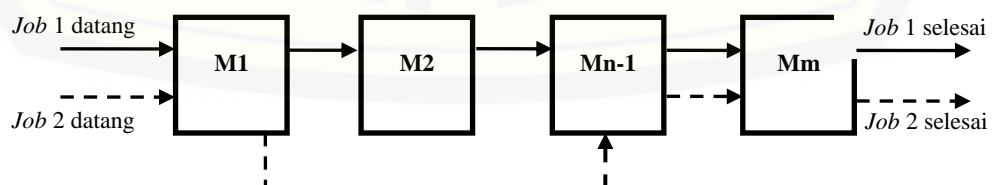
Flowshop terbagi menjadi dua macam pola, yaitu *pure flowshop* dan *general flowshop* (Hartanto, 2007).

- Pure flowshop*, berbagai *job* akan mengalir pada jalur produksi yang sama atau bisa diartikan setiap *job* akan melewati setiap mesin yang terdapat pada aliran proses yang sama. Gambar 2.1 berikut menunjukkan ilustrasi untuk *pure flowshop*.



Gambar 2.1 Jalur proses *pure flowshop*

- General flowshop*, *job* yang datang ke dalam *flowshop* tidak harus dikerjakan pada semua jenis mesin. Gambar 2.2 berikut menunjukkan ilustrasi untuk *genera flowshop*.



Gamabar 2.2 Jalur proses *general flowshop*

2.3 *Permutation Flowshop Scheduling Problem (PFSP)*

PFSP adalah salah satu kasus dalam masalah penjadwalan, dimana n job harus diproses pada m mesin dengan urutan yang sama, yaitu melalui mesin 1 ke mesin 2 dan seterusnya hingga mesin ke- m serta n job diproses dalam urutan yang sama pada setiap mesin. Karena pada PFSP diketahui bahwa pemrosesan n job pada setiap mesin memiliki urutan yang sama, maka cara penempatan pekerjaan pada setiap mesin adalah sama, yaitu $n!$, sehingga ruang solusi untuk PFSP sebesar $n!$.

Tujuan PFSP dalam tugas akhir ini adalah untuk menentukan urutan n job yang meminimumkan total waktu penyelesaian semua job atau *makespan* yang dilambangkan dengan E_{ij} . PFSP dapat dimodelkan dalam bentuk model matematika yang diperoleh dari fungsi rekursif. Berikut bentuk model matematikanya (Andriyanti, 2014) ditunjukkan pada persamaan (2.1).

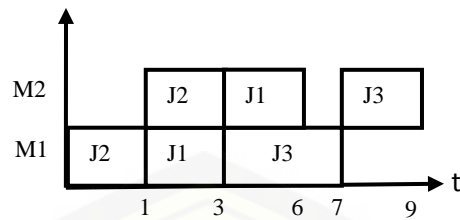
$$\begin{aligned}
 S_{1,1} &= 0 \\
 E_{1,1} &= t_{1,1} \\
 S_{1,j} &= E_{1,j-1} \\
 E_{1,j} &= S_{1,j} + t_{1,j} \quad j = 1, \dots, m \\
 S_{i,j} &= \max \{E_{i-1,j}, E_{i,j-1}\} \quad i = 2, \dots, n \\
 E_{i,j} &= S_{i,j} + t_{i,j}
 \end{aligned} \tag{2.1}$$

dimana $S_{i,j}$ = waktu mulai job i pada mesin j

$t_{i,j}$ = waktu proses job i pada mesin j

$E_{i,j}$ = waktu berhenti job i pada mesin j

Untuk mempermudah perhitungan *makespan*, digunakan *gant chart*, yang diperkenalkan oleh Henry Gantt pada tahun 1916. *Gantt chart* merupakan representasi grafis dari *job-job* yang harus diselesaikan dan digambarkan dalam bentuk batang dan analog dengan waktu dan penyelesaian job tersebut. Gambar 2.3 berikut menunjukkan ilustrasi dari *gant chart* :

Gambar 2.3 Diagram *gantt*

Pada Gambar 2.3 digambarkan dengan sumbu horisontal sebagai waktu dan sumbu vertikal sebagai urutan mesin yang digunakan. Dengan kata lain, *Machine Oriented Gantt Chart* adalah diagram *gantt* yang berorientasi pada mesin.

Keuntungan menggunakan diagram *gantt* adalah (Ginting, 2009):

- Dalam situasi awal mengenai penggunaan diagram *gantt* memungkinkan evaluasi lebih awal mengenai penggunaan sumber daya seperti yang telah direncanakan;
- Kemajuan pekerjaan mudah diperiksa pada setiap waktu karena sudah tergambar dengan jelas;
- Semua pekerjaan diperlihatkan secara grafis dalam suatu diagram yang mudah dipahami.

2.4 Algoritma, Efisiensi Algoritma, dan Notasi *Big-O*

2.4.1 Algoritma

Algoritma adalah urutan langkah-langkah logis untuk menyelesaikan masalah yang disusun secara sistematis (Hasad, 2011). Dalam kehidupan sehari-hari banyak terdapat proses yang dinyatakan dalam suatu algoritma. Contohnya, cara membuat masakan yang dinyatakan dalam suatu resep juga dapat disebut sebagai sebuah algoritma. Pada setiap resep selalu ada urutan langkah-langkah membuat masakan. Bila langkah-langkahnya tidak logis, maka tidak akan dapat dihasilkan masakan yang diinginkan. Secara umum, pihak (benda) yang mengerjakan proses disebut pemroses. Pemroses tersebut dapat berupa manusia, komputer, robot atau alat-alat elektronik lainnya.

Pemroses melakukan suatu proses dengan melaksanakan atau mengeksekusi algoritma yang menjabarkan proses tersebut.

Dengan kata lain, algoritma merupakan deskripsi langkah-langkah penyelesaian masalah yang tersusun secara logis (jelas dan pasti) atau urutan logis pengambilan keputusan untuk pemecahan suatu masalah. Algoritma ditulis dengan notasi khusus yang mudah dimengerti dan notasi dapat diterjemahkan menjadi suatu bahasa pemrograman.

Suatu algoritma akan memerlukan masukan (input) tertentu untuk memulainya, dan akan menghasilkan keluaran (output) tertentu pada akhirnya. Hal-hal yang perlu diperhatikan dalam algoritma adalah mencari langkah-langkah yang paling sesuai untuk penyelesaian suatu masalah, karena setiap algoritma memiliki karakteristik tertentu yang memiliki kelebihan dan kekurangan (Nugraha, 2012).

Umumnya sebuah algoritma sering dikaitkan dengan perhitungan dan mengingat proses perhitungan dilakukan dengan cepat oleh sebuah komputer. Komputer merupakan suatu alat elektronik yang mempunyai kemampuan untuk melakukan perhitungan dan membuat keputusan logika dengan jumlah waktu yang jauh lebih cepat dari kemampuan manusia.

Komputer memproses data dengan menggunakan kumpulan instruksi (langkah) yang disebut program. Program ini memandu komputer untuk melaksanakan kerja (mengeksekusi perintah) yang secara sistematis dan berstruktur. Program yang disusun secara sistematis dan logis dan dapat menyelesaikan masalah (menghasilkan output) merupakan sebuah algoritma.

2.4.2 Efisiensi Algoritma

Algoritma yang bagus adalah algoritma yang efisien, dimana dikatakan efisien karena dinilai dari aspek kebutuhan waktu dan ruang membutuhkan waktu yang sedikit. Keefisienan algoritma juga berguna dalam membandingkan algoritma. Keefisienan algoritma dapat diukur dari orde yang terdapat didalam persamaan kompleksitas waktu. Kompleksitas waktu diukur

dari jumlah tahapan komputasi yang dibutuhkan dalam menjalankan algoritma yang tergantung pada jumlah data yang akan diproses.

2.4.3 Notasi *Big-O*

Notasi *Big-O* adalah notasi matematika yang digunakan untuk menggambarkan tingkah laku asimtotik dari fungsi. Notasi ini berguna menganalisa kompleksitas waktu dari suatu algoritma. Selain itu, notasi *Big-O* juga berguna untuk membandingkan beberapa algoritma untuk masalah yang sama sehingga dapat menentukan algoritma terbaik.

Berikut ini adalah daftar kelas fungsi yang umum dijumpai dalam menganalisa algoritma. Semua ini diandaikan n mendekati tak terhingga. Urutan berdasarkan dari kinerja yang tercepat ke yang terlambat.

Tabel 2.1 Daftar Notasi *Big-O*

Notasi	Nama
$O(1)$	Konstanta
$O(\log n)$	Logaritma
$O([\log n]^c)$	Polilogaritma
$O(n)$	Linier
$O(n^2)$	Kuadratik
$O(n^c)$, dimana $c > 1$	Polinomial (aljabar)
$O(c^n)$	Eksponensial
$O(n!)$	Faktorial

Notasi *Big-O* sangat berguna didalam menganalisa efisiensi dari suatu algoritma. Tinjauan perbandingan $T(n) = 2n^2 + 6n + 1$ dengan n pada tabel 2.2 berikut.

Tabel 2.2 Perbandingan Pertumbuhan $T(n)$ dengan n^2

N	$T(n) = 2n^2 + 6n + 1$	n^2
10	261	100
100	2.061	10.000
1.000	2.006.001	1.000.000
10.000	200.060.001	100.000.000

Menurut definisi :

$T(n) = O(f(n))$, artinya $T(n)$ berorde paling besar $f(n)$ bila terdapat konstanta C dan n_0 sehingga $T(n) \leq C (f(n))$.

Artinya, jika sebuah algoritma mempunyai kompleksitas waktu $O(f(n))$, maka jika n dibuat semakin besar waktu yang dibutuhkan tidak akan melebihi satu tetapan C dikali $f(n)$ (Wibowo, 2011).

Contoh :

$$T(n) = 2n^2 + 6n + 1 = O(n^2)$$

Penyelesaiannya :

$$2n^2 + 6n + 1 = O(n^2)$$

Karena

$$2n^2 + 6n + 1 \leq 2n^2 + 6n^2 + n^2 = 9n^2 \text{ untuk semua } n \geq 1 (C = 9).$$

Untuk n yang besar, pertumbuhan $T(n)$ sebanding dengan n^2 . Pada kasus diatas laju pertumbuhan $T(n)$ adalah sama seperti n^2 . $T(n)$ bertambah seperti n^2 pada saat n bertambah. Maka dapat dikatakan bahwa $T(n)$ berorde n^2 atau $T(n) = n^2$.

2.5 Algoritma Genetika

Algoritma genetika sebagai cabang dari algoritma evolusi merupakan metode *adaptive* yang biasa digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah optimasi. Algoritma genetika didasarkan pada proses genetik yang ada dalam makhluk hidup, yaitu perkembangan generasi dalam sebuah populasi yang alami, secara lambat laun mengikuti prinsip seleksi alam atau “siapa yang kuat, dia yang bertahan (*survive*).” Dengan meniru teori evolusi ini, algoritma genetika dapat digunakan untuk mencari solusi permasalahan-permasalahan dalam dunia nyata.

Peletak prinsip dasar sekaligus pencipta algoritma genetika adalah John Holland. Algoritma genetika menggunakan analogi secara langsung dari kebiasaan yang alami yaitu seleksi alam. Algoritma ini bekerja dengan sebuah populasi yang terdiri dari individu-individu, yang masing-masing individu merepresentasikan sebuah solusi yang mungkin bagi persoalan yang ada. Dalam kaitan ini, individu dilambangkan dengan sebuah nilai *fitness* yang akan digunakan untuk mencari solusi terbaik dari persoalan yang ada.

Pertahanan yang tinggi dari individu memberikan kesempatan untuk melakukan reproduksi melalui perkawinan silang dengan individu yang lain dalam populasi tersebut. Individu baru yang dihasilkan dalam hal ini dinamakan keturunan, yang membawakan beberapa sifat dari induknya. Sedangkan individu dalam populasi yang tidak terseleksi dalam reproduksi akan mati dengan sendirinya. Dengan jalan ini, beberapa generasi dengan karakteristik yang baik akan bermunculan dalam populasi tersebut, untuk kemudian dicampur dan ditukar dengan karakter yang lain. Dengan mengawinkan semakin banyak kemungkinan terbaik yang dapat diperoleh.

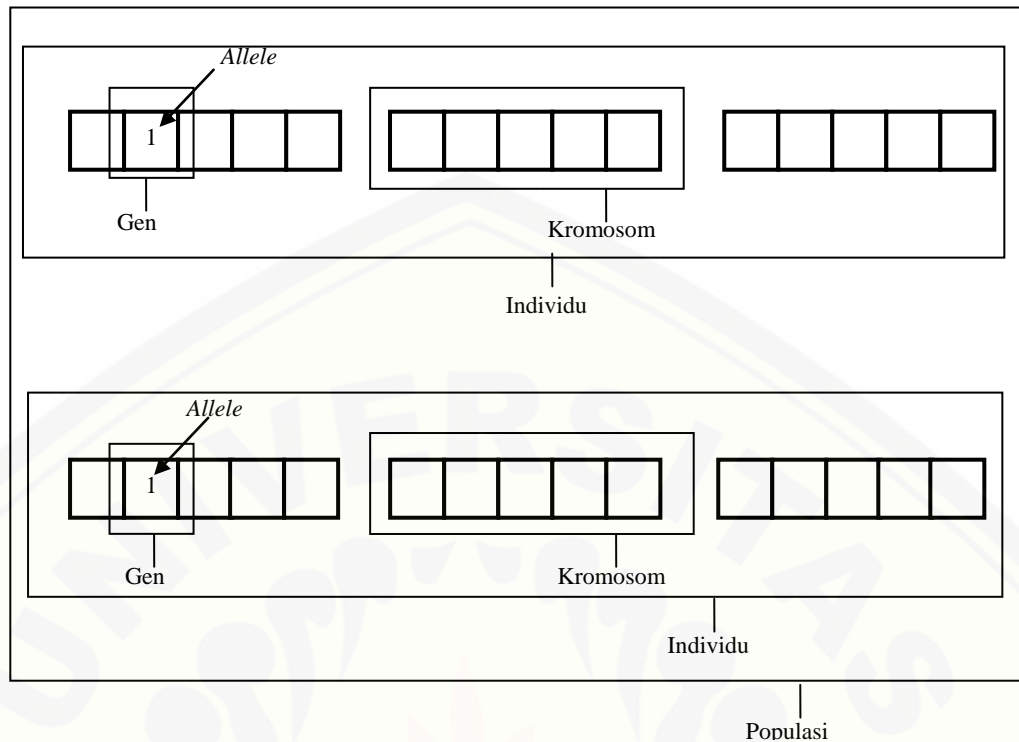
Sebelum algoritma genetika dapat dijalankan, sebaiknya sebuah kode yang sesuai (representatif) untuk persoalan harus dirancang terlebih dahulu. Untuk itu maka titik solusi dalam ruang permasalahan dikodekan dalam bentuk kromosom yang terdiri atas komponen genetik terkecil yaitu gen. Dengan teori evolusi dan teori genetika, di dalam penerapan algoritma genetika akan melibatkan beberapa operator, yaitu :

- a. Operasi evolusi yang melibatkan proses seleksi (*selection*) di dalamnya.
- b. Operasi genetika yang melibatkan operator pindah silang (*crossover*).

Untuk memeriksa hasil optimasi, kita membutuhkan fungsi *fitness*, yang menandakan gambaran hasil (solusi) yang sudah dikodekan. Selama berjalan, induk harus digunakan untuk reproduksi, pindah silang untuk menciptakan keturunan. Jika algoritma genetika didesain secara baik, populasi akan mengalami konvergensi dan akan didapatkan sebuah solusi yang optimum.

2.5.1 Pengertian Individu

Individu menyatakan salah satu solusi yang mungkin. Individu bisa dikatakan sama dengan kromosom, yang merupakan kumpulan gen. Beberapa definisi penting yang perlu diperhatikan dalam mendefinisikan individu untuk membangun penyelesaian permasalahan dengan algoritma genetika diilustrasikan pada Gambar 2.4.



Gambar 2.4 Ilustrasi representasi penyelesaian permasalahan dalam algoritma genetika

- a. *Genotype* (gen), merupakan sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom.
- b. *Allele*, merupakan nilai dari gen.
- c. *Kromosom*, merupakan gabungan gen-gen yang membentuk nilai tertentu.
- d. Individu, menyatakan satu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
- e. Populasi, merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi.
- f. Generasi, menyatakan satu siklus proses evolusi atau satu iterasi didalam algoritma genetika.

2.5.2 Komponen-Komponen Utama Algoritma Genetika

a. Pengkodean

Salah satu masalah yang harus diperhatikan saat akan memulai penggunaan algoritma genetika adalah pengkodean. Pengkodean adalah suatu

teknik untuk menyatakan populasi awal sebagai calon solusi suatu masalah ke dalam suatu kromosom sebagai suatu kunci pokok persoalan ketika menggunakan algoritma genetika. Teknik pengkodean adalah bagaimana mengkodekan gen dari kromosom, dimana gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variabel. Pengkodean sangat tergantung pada masalah yang akan dipecahkan (Hapsari, 2013).

Pengkodean permutasi digunakan pada masalah pengurutan, seperti pada *travelling salesman problem* atau penjadwalan suatu produksi. Dalam pengkodean permutasi, setiap kromosom adalah serangkaian gen yang mewakili angka dalam suatu urutan tertentu. Tabel 2.3 berikut merupakan contoh dari pengkodean permutasi.

Tabel 2.3 Contoh Pengkodean Permutasi

Kromosom A	5	1	7	3	9	2	8	4	6
Kromosom B	2	6	1	5	7	3	9	8	4

b. Membangkitkan Populasi Awal

Membangkitkan populasi awal adalah proses membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Ukuran untuk populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian dilakukan pembangkitan populasi awal. Syarat-syarat yang harus dipenuhi untuk menunjukkan suatu solusi harus benar-benar diperhatikan dalam pembangkitan setiap individunya. Inisialisasi pembangkitan populasi awal dapat diimplementasikan seperti yang ditunjukkan pada Gambar 2.4.

c. Evaluasi Nilai *Fitness*

Fungsi *fitness* adalah suatu fungsi yang digunakan untuk mengevaluasi kelangsungan hidup sebuah kromosom. Dalam seleksi alam, individu yang bernilai *fitness* tinggi akan hidup sedangkan individu yang bernilai *fitness* rendah akan mati (Suyanto, 2005). Fungsi ini membedakan kualitas dari kromosom untuk mengetahui seberapa baik kromosom yang dihasilkan.

Semakin tinggi nilai *fitness* akan semakin besar kemungkinan kromosom tersebut terpilih kegenerasi berikutnya. pada kebanyakan optimasi, fungsi *fitness* dibentuk berdasarkan fungsi objektifnya. Jika masalah optimasinya adalah memaksimumkan maka fungsi *fitness* sama dengan fungsi objektif (Ayuningtyas, 2008). Jika masalah optimasinya adalah meminimumkan, maka fungsi *fitness*-nya didefinisikan menggunakan persamaan (2.2).

$$f_{\pi} = \text{Min}\{f_{\pi-1}, F_{obj}\} \quad (2.2)$$

dimana f_{π} = fungsi *fitness* kromosom π

F_{obj} = nilai objektif dari kromosom π

d. Pindah Silang (*Crossover*)

Pindah silang (*crossover*) adalah operator dari algoritma genetika yang melibatkan dua induk untuk membentuk kromosom baru. Pindah silang menghasilkan titik baru dalam ruang pencarian yang siap untuk diuji. Operasi ini tidak selalu dilakukan pada semua individu yang ada (Budi Santosa dan Paul Willy, 2011).

Prinsip dari *crossover* ini adalah melakukan operasi (pertukaran, aritmatika) pada gen-gen yang bersesuaian dari dua induk untuk menghasilkan individu baru. Proses *crossover* dilakukan pada setiap individu dengan probabilitas *crossover* yang ditentukan.

Secara umum *crossover* dapat dilakukan dengan langkah-langkah sebagai berikut :

- 1) Menentukan jumlah kromosom yang akan di-*crossover*.
- 2) Memilih kromosom-kromosom yang akan di-*crossover*.
- 3) Melakukan *crossover* dengan metode yang telah ditetapkan yaitu dengan menggunakan *crossover* dua poin.

Metode *crossover* yang digunakan pada skripsi ini adalah *Partial-Mapped Crossover (PMX)*, *PMX* diciptakan oleh Goldberg dan Lingle. *PMX* merupakan rumusan modifikasi dari pindah silang dua poin. Pada metode ini, dua posisi penyilangan diseleksi secara random, kedua posisi tersebut tidak boleh sama dan diurutkan naik. Variabel-variabel ditukar antar kromosom pada titik

tersebut untuk menghasilkan anak. *PMX* mempunyai prosedur dan langkah kerja yang diilustrasikan pada Gambar 2.5.

1. Pilih posisi untuk menentukan substring secara acak

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 Induk 1

5 | 4 | 6 | 9 | 2 | 1 | 7 | 8 | 3 Induk 2

2. Tukar substring diantara induk

1 | 2 | 6 | 9 | 2 | 1 | 7 | 8 | 9 *Proto-child 1*

5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 3 *Proto-child 2*

3. Tentukan hubungan *mapping*



4. Menentukan kromosom keturunan mengacu pada hubungan *mapping*

3 | 5 | 6 | 9 | 2 | 1 | 7 | 8 | 4 *Proto-child 1*

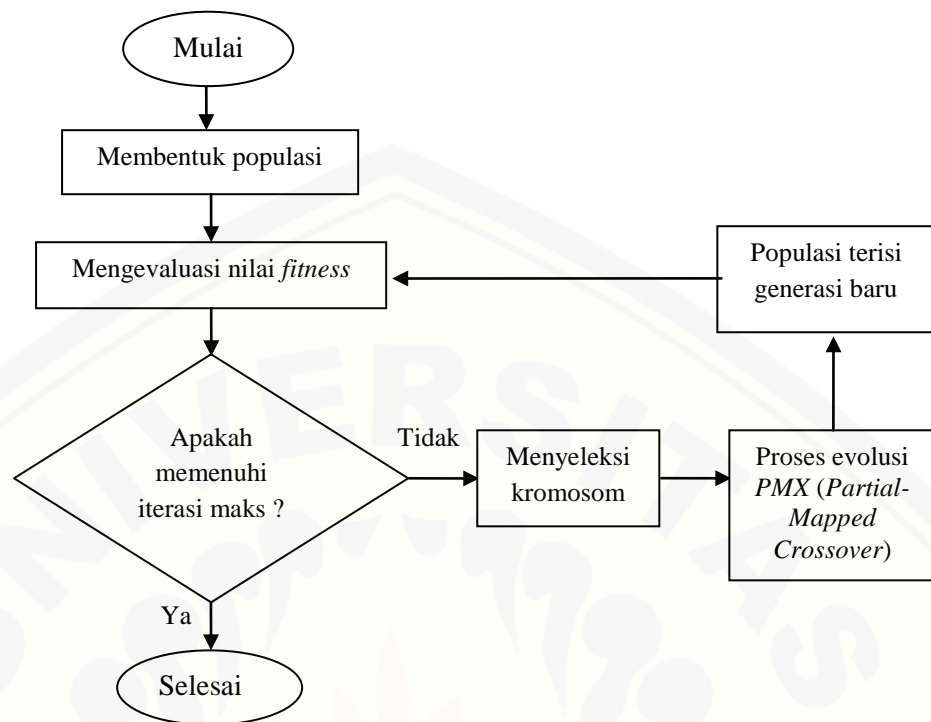
2 | 9 | 3 | 4 | 5 | 6 | 7 | 8 | 1 *Proto-child 2*

Gambar 2.5 Ilustrasi prosedur *PMX*

e. Kriteria Penghentian Generasi

Proses algoritma genetika ini akan berlangsung terus menerus menciptakan generasi baru sampai ditemukannya suatu kondisi berhenti. Kondisi berhenti dalam proses algoritma genetika terjadi apabila jumlah generasi yang diinginkan telah tercapai.

Secara umum prosedur algoritma *genetika* dapat dilihat pada Gambar 2.6.

Gambar 2.6 Skema prosedur algoritma *genetika*

2.6 Algoritma *Simulated Annealing*

Simulated Annealing (SA) mensimulasikan proses *annealing* (pendinginan) pada pembuatan materi yang terdiri dari butir kristal (*glassy*) atau logam. Tujuan dari proses ini adalah menghasilkan struktur kristal yang baik dengan menggunakan energi seminimal mungkin.

Ide SA berasal dari suatu makalah yang dipublikasikan oleh metropolis pada tahun 1953. Jika kita memanaskan suatu materi keras hingga mencair kemudian mendinginkannya, maka struktur dari materi tersebut bergantung pada tingkat pendinginan. Jika materi cair didinginkan secara perlahan, maka akan dihasilkan kristal-kristal yang berkualitas baik. Sebaliknya, jika materi cair didinginkan secara cepat, kristal-kristal yang terbentuk tidak akan sempurna. Algoritma yang diusulkan Metropolis mensimulasikan materi sebagai suatu sistem dari partikel-partikel. Algoritma tersebut mensimulasikan proses pendinginan yang secara bertahap menurunkan suhu sistem hingga konvergen pada keadaan beku dan stabil.

Pada tahun 1983, Kirkpatrick *et al* menggunakan ide dari algoritma Metropolis dan mengaplikasikannya pada permasalahan optimasi. Idanya adalah bagaimana menggunakan *Simulated Annealing* untuk mencari solusi-solusi yang layak dan konvergen pada solusi optimal.

Parameter yang digunakan dalam *Simulated Annealing* sebagai berikut :

a. Keadaan Sistem

Keadaan sistem didefinisikan sebagai solusi yang mungkin. Misalnya, untuk masalah penjadwalan *flowshop* dengan 4 job, sehingga urutan jadwal yang mungkin *J4, J1, J3, J2*.

b. Energi

Energi didefinisikan sebagai seberapa besar fungsi tujuan minimal dari kombinasi keadaan sistem. Dalam minimasi fungsi, misalkan solusi yang sekarang adalah x dan nilai fungsinya $f(x)$, mirip dengan status energi pada sistem termodinamika, energi $E_i = f_i = f(x_i)$ (Santosa dan Willy, 2011). untuk permasalahan penjadwalan *flowshop*, energi didefinisikan sebagai nilai *makespan* dari setiap solusi yang dijadwalkan.

c. Temperatur

Temperatur adalah suatu nilai kendali yang membuat suatu keadaan dapat bergerak atau tidak. Atom-atom akan bergerak bebas pada temperatur yang tinggi dan semakin terbatas gerakannya ketika temperatur turun. Bila temperatur diturunkan secara perlahan dan teratur, atom-atom tersebut akan menghasilkan kristal dengan susunan bagus dengan energi yang minimal. Sebaliknya, jika temperatur diturunkan secara cepat akan menghasilkan *polycrystalline* (kristal yang tidak sempurna) dengan energi yang tidak minimal. untuk permasalahan penjadwalan *flowshop*, temperatur didefinisikan sebagai parameter kontrol.

d. Laju Pendinginan

Dalam masalah penjadwalan *flowshop*, fungsi laju pendinginan dianalogikan seberapa cepat pencapaian solusi akhir dilakukan. Laju pendinginan digunakan dalam proses penurunan temperatur. Penurunan temperatur ditunjukkan dengan persamaan (2.3).

$$T_{baru} = a \times T_{awal} \quad (2.3)$$

dimana T_{awal} = teperatur awal

T_{baru} = teperatur baru

a = faktor reduksi suhu ($a < 1$) (Emanuel dan Aritonang, 2008).

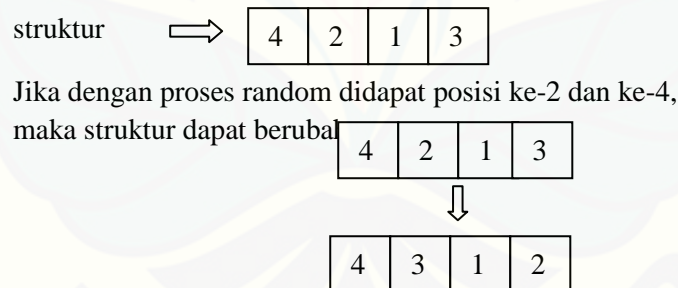
Mekanisme algoritma *SA* pada penjadwalan *flowshop* antara lain sebagai berikut :

a. Membangkitkan solusi awal

Solusi awal pada penjadwalan yaitu membangkitkan jadwal secara random. Pada tahap ini akan ditentukan nilai objektifnya. Hasil yang diperoleh akan disimpan solusi terbaik untuk tahap awal.

b. Mencari solusi baru

Tahap selanjutnya adalah mencari solusi baru dari solusi terbaik yang dihasilkan pada tahap sebelumnya atau sering disebut metode *insertion*. solusi baru dibentuk dengan memilih secara acak satu bagian struktur untuk dipindah ke bagian yang lain. Gambar 2.7 berikut mengilustrasikan metode *insertion*.



Gambar 2.7 Ilustrasi metode *insertion*

c. Mengecek parameter kontrol

Penentuan penurunan parameter kontrol ditentukan untuk memutuskan apakah parameter kontrol perlu diturunkan atau tidak. Apabila parameter kontrol diturunkan, maka lanjut ke langkah d. Jika tidak, maka ulangi langkah b sampai langkah c.

d. Menentukan solusi baru yang terbaik

Pada tahap ini akan terdapat beberapa solusi baru yang diterima, karena itu harus dipilih salah satunya untuk menjadi solusi terbaik. Nantinya solusi terbaik ini akan menjadi solusi awal apabila parameter kontrol diubah.

e. Mencari evaluasi solusi baru

Evaluasi solusi baru digunakan untuk mengevaluasi apakah solusi baru diterima atau tidak. Kriteria evaluasi solusi ditunjukkan dengan persamaan (2.4).

$$\Delta E = E(X_{i+1}) - E(X_i) \quad (2.4)$$

dimana ΔE = selisih nilai objektif

$E(X_{i+1})$ = nilai objektif dari solusi baru

$E(X_i)$ = nilai objektif dari solusi awal

Jika nilai objektif solusi baru lebih kecil dari nilai objektif solusi awal ($\Delta E \leq 0$), maka solusi baru diterima. Tetapi bila solusi baru memiliki nilai objektif lebih besar daripada solusi awal ($\Delta E > 0$), maka solusi baru masih mungkin terpilih dengan probabilitas (Suyanto,2010)

$$P(\Delta E) = e^{-\Delta E/T} > r \quad (2.5)$$

dimana T = parameter kontrol

r = bilangan acak antara 0 dan 1

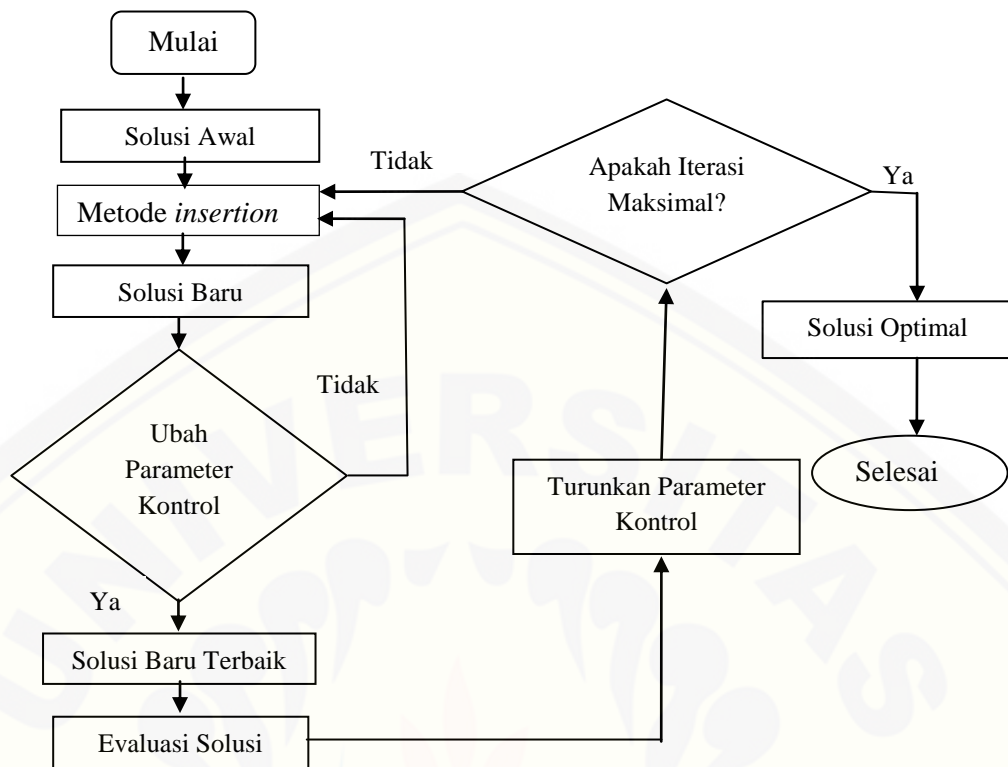
f. Penurunan parameter kontrol

Untuk menurunkan parameter kontrol digunakan persamaan (2.3)

g. Mengecek iterasi maksimal

Penentuan iterasi maksimal ditentukan untuk memutuskan apakah iterasi selesai atau tidak. Jika iterasi maksimal maka selesai, artinya solusi terbaik merupakan solusi yang optimal. Jika iterasi tidak maksimal maka ulangi langkah a sampai langkah g.

Secara umum prosedur algoritma proses penurunan temperatur SA dapat dilihat pada Gambar 2.8.



Gambar 2.8 Skema prosedur algoritma *Simulated Annealing*

Tabel 2.4 dibawah ini mengilustrasikan analogi proses fisika dan algoritma *Simulated Annealing* pada permasalahan optimasi (penjadwalan *flowshop*).

Tabel 2.4 Analogi Proses Fisika dan SA pada Penjadwalan *Flowshop*

Pembeda	Proses Fisika	Penjadwalan <i>Flowshop</i>
Input	Benda padat (baja)	Jumlah mesin dan job
Output	Energi terendah yang dihasilkan	Total <i>makespan</i>
Keadaan Awal	Konfigurasi sistem awal proses yaitu baja yang dicairkan dengan suhu yang tinggi	Susunan solusi awal (posisi awal iterasi)
Prinsip Kerja	Penurunan temperatur atau suhu yang dilakukan secara perlahan (<i>annealing</i>) dimana atom-atom akan mengatur dirinya untuk mencari konfigurasi dengan tingkat energi yang lebih rendah	Penurunan nilai parameter dimana pada tiap penurunannya diharapkan memperoleh jadwal produksi dengan <i>makespan</i> yang minimum
Perubahan Keadaan	Mekanisme untuk merubah konfigurasi interaksi antar molekul	Mekanisme pertukaran jadwal antar titik (Solusi Baru)

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang digunakan dalam tugas akhir ini merupakan data primer dari waktu pemrosesan tiap pekerjaan pada tiap mesin di industri pembuatan kerupuk UD. Samjaya. Industri ini terletak didesa Klatakan, kecamatan Tanggul kabupaten Jember yang memproduksi sepuluh jenis kerupuk. Pembuatan kerupuk dilakukan dengan sembilan proses (mesin), yaitu persiapan bahan, pengadonan, pelembutan, pencetakan, pengukusan, pendinginan, pemotongan (pembentukan), penjemuran dan pengemasan. Secara umum skema proses produksi kerupuk dapat dilihat pada Lampiran 1.

Pengumpulan data dilakukan dalam penelitian kali ini dengan cara menghitung waktu pemrosesan tiap pekerjaan pada tiap mesin dengan bertanya kepada pemilik maupun pegawainya. Data yang diperoleh berupa data waktu proses pembuatan sepuluh jenis kerupuk pada tiap mesin dalam satuan waktu yaitu menit. Data tersebut disajikan dalam bentuk Tabel 3.1.

Tabel 3.1 Data Waktu Pembuatan Kerupuk (menit)

<i>Job</i>	<i>Mesin</i>								
	<i>M1</i>	<i>M2</i>	<i>M3</i>	<i>M4</i>	<i>M5</i>	<i>M6</i>	<i>M7</i>	<i>M8</i>	<i>M9</i>
<i>J1</i>	15	160	15	30	30	1440	150	180	30
<i>J2</i>	15	160	15	30	30	1440	180	360	30
<i>J3</i>	15	160	15	30	30	360	85	180	30
<i>J4</i>	15	60	30	10	3	360	10	360	30
<i>J5</i>	15	60	30	10	3	300	30	180	30
<i>J6</i>	15	60	30	10	3	300	10	360	30
<i>J7</i>	15	60	30	10	3	720	30	180	30
<i>J8</i>	15	45	15	35	30	720	72	360	10
<i>J9</i>	15	45	15	35	30	1440	168	180	15
<i>J10</i>	15	75	36	25	17	1440	45	360	25

Keterangan :

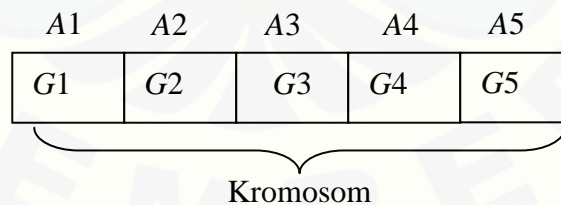
$J1$ = Kerupuk impala panjang	$M1$ = Persiapan bahan
$J2$ = Kerupuk impala pendek	$M2$ = Pengadonan
$J3$ = Kerupuk rajang	$M3$ = Mesin mulen (pelembutan)
$J4$ = Kerupuk mawar 2 udang besar	$M4$ = Pencetakan
$J5$ = Kerupuk mawar 2 udang kecil	$M5$ = Pengukusan
$J6$ = Kerupuk mawar SI besar	$M6$ = Pendinginan
$J7$ = Kerupuk mawar SI kecil	$M7$ = Pemotongan (pembentukan)
$J8$ = Kerupuk kasandra besar	$M8$ = Penjemuran
$J9$ = Kerupuk kasandra kecil	$M9$ = Pengemasan
$J10$ = Kerupuk obar-abir	

3.2 Pengolahan Data dengan Algoritma Genetika

Langkah-langkah yang dilakukan untuk menyelesaikan penjadwalan produksi kerupuk menggunakan algoritma Genetika adalah sebagai berikut :

a. Interpretasi kromosom

Mengkodekan *job* kedalam bentuk kromosom. Setiap kromosom berisi gen, dimana gen adalah *job* dan posisi dari gen (allele) yaitu mesin. Kemungkinan kromosom akan diinterpretasikan pada Gambar 3.1 berikut.



Gambar 3.1 Interpretasi kromosom
dimana $A1, A2, \dots, A5$ = mesin

$G1, G2, \dots, G5$ = *job*

b. Membangkitkan populasi

Populasi awal dibangkitkan 5 kromosom secara random, dimana setiap kromosom berisi 5 gen.

c. Mengevaluasi nilai objektif

Setiap kromosom dalam populasi akan dihitung nilai objektif (*makespan*) dengan menggunakan persamaan (2.1) atau bisa juga menggunakan cara yang ditunjukkan pada Tabel 3.2 berikut.

Tabel 3.3 Pencarian Nilai *Makespan* dengan *i* job dan *j* mesin

M	M1	M2	...	Mj			
J	S_{ij}	E_{ij}	S_{ij}	E_{ij}	...	S_{ij}	E_{ij}
J1	0	$t_{1,1}$	$E_{1,1}$	$S_{1,2} + t_{1,2}$...	$E_{1,j-1}$	$S_{1,j} + t_{1,j}$
J2	$t_{1,1}$	$S_{2,1} + t_{2,1}$	$\max \{E_{1,2}, E_{2,1}\}$	$S_{2,2} + t_{2,2}$...	$\max \{E_{1,j}, E_{2,j-1}\}$	$S_{2,j} + t_{2,j}$
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮
Ji	$t_{j-1,1}$	$S_{i,1} + t_{i,1}$	$\max \{E_{i-1,2}, E_{i,1}\}$	$S_{i,2} + t_{i,2}$...	$\max \{E_{i-1,j}, E_{i,j-1}\}$	$S_{i,j} + t_{i,j}$

Keterangan :

M = mesin

J = job

$t_{i,j}$ = waktu proses job *i* pada mesin *j*

S_{ij} = waktu mulai job *i* pada mesin *j*

E_{ij} = waktu berhenti job *i* pada mesin *j*

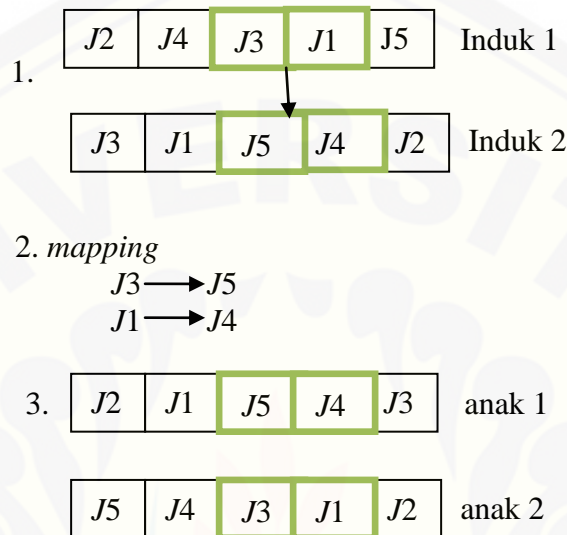
d. Mengevaluasi nilai *fitness*

Pada pengevaluasian nilai *fitness* dilakukan penentuan induk yang dipilih berdasarkan nilai objektif (*makespan*) paling kecil dari populasi yang dibangkitkan.

e. Melakukan proses pindah silang (*crossover*)

Kromosom yang terpilih dari proses seleksi akan menjadi induk untuk melewati proses genetika yaitu dengan pengkodean *crossover* dua poin dengan metode *PMX* yaitu mengambil 2 posisi penyilangan secara random, kedua posisi tersebut tidak

boleh sama dan diurutkan naik. Variabel-variabel ditukar antar kromosom pada titik tersebut untuk menghasilkan anak yang akan diilustrasikan pada Gambar 3.3 berikut.



Gambar 3.3 Pengkodean *crossover two point*

f. Menghitung nilai objektif anak

Menghitung nilai objektif dari anak yang telah dihasilkandari proses *crossover* dengan cara yang sama seperti langkah c.

g. Seleksi

Pada proses seleksi akan dibandingkan nilai objektif yang dihasilkan anak dari proses *crossover* dengan nilai objektif dari populasi awal yang telah dibangkitkan. Jika nilai objektif pada anak lebih kecil dari nilai objektif tiap-tiap kromosom pada populasi awal maka kromosom anak akan menggantikan kromosom dengan nilai objektif yang lebih besar sehingga terbentuk generasi baru.

h. Kriteria proses berhenti

Mengulangi dari langkah b dan langkah selanjutnya sampai proses pencarian berhenti ketika iterasi yang ditentukan tercapai.

3.3 Pengolahan Data dengan Algoritma *Simulated Annealing*

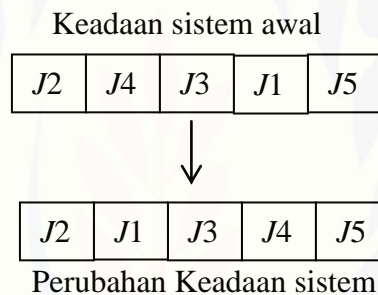
Langkah-langkah yang dilakukan untuk menyelesaikan penjadwalan produksi kerupuk menggunakan algoritma *Simulated Annealing* adalah sebagai berikut :

a. Membentuk solusi awal

Membentuk solusi awal dengan membentuk urutan *job* secara random dan menghitung nilai *makespan*-nya.

b. Membangkitkan solusi baru

Mencari solusi baru dari solusi terbaik yang dihasilkan langkah a dengan menggunakan metode *insertion*, dimana pada pengkodean ini perpindahan memperhatikan urutan *job*. Misal dipindahkan posisi 2 dan 4 akan ditunjukkan pada Gambar 3.4.



Gambar 3.4 Pengkodean dengan metode *insertion*

Pada metode ini ditentukan juga berapa kali pertukaran dilakukan (replikasi) yaitu sebanyak 5 kali. angka replikasi, merupakan angka yang menunjukkan berapa kali *loop*.

c. Menghitung nilai objektif

Setelah didapatkan solusi baru dari proses pada langkah b kemudian dilakukan perhitungan nilai objektif pada masing-masing keadaan sistem dengan cara yang sama dengan langkah c pada langkah perhitungan algoritma Genetika.

d. Menentukan nilai awal temperatur

Penentuan nilai awal temperatur dapat ditentukan jika nilai awal temperatur terlalu tinggi, maka dibutuhkan lebih banyak pengurangan temperatur untuk konvergen. Sebaliknya jika nilai awal temperatur terlalu kecil proses pencarian mungkin

kurang sempurna sehingga ada titik-titik potensial yang bisa menjadi global optimum terlewat.

e. Menurunkan nilai parameter kontrol

Pada penurunan parameter kontrol dilakukan dengan menentukan terlebih dahulu faktor reduksi suhu (α). Apabila faktor reduksi terlalu besar akan memerlukan terlalu banyak langkah komputasi untuk konvergensi. Sebaliknya apabila terlalu kecil bisa berakibat terlalu cepatnya penurunan temperatur sehingga akan banyak titik-titik potensial untuk menjadi solusi global akan terlewat. Untuk menurunkan parameter kontrol digunakan persamaan (2.3)

f. Menentukan solusi baru yang terbaik

Pada tahap penentuan solusi baru terdapat beberapa solusi baru dari proses metode *insertion*, solusi baru dipilih dari salah satu nilai objektif paling kecil dari keadaan system pada pembangkitan awal solusi baru.

g. Mengevaluasi solusi baru

Kriteria evaluasi solusi ditunjukkan pada persamaan (2.8) jika $\Delta E \leq 0$, maka solusi baru diterima. tetapi jika $\Delta E > 0$, maka solusi baru masih mungkin terpilih dengan persamaan (2.9).

h. Kriteria proses berhenti

Proses pencarian berhenti ketika iterasi yang ditentukan tercapai.

3.4 Membuat *flowchart* dari kedua Algoritma

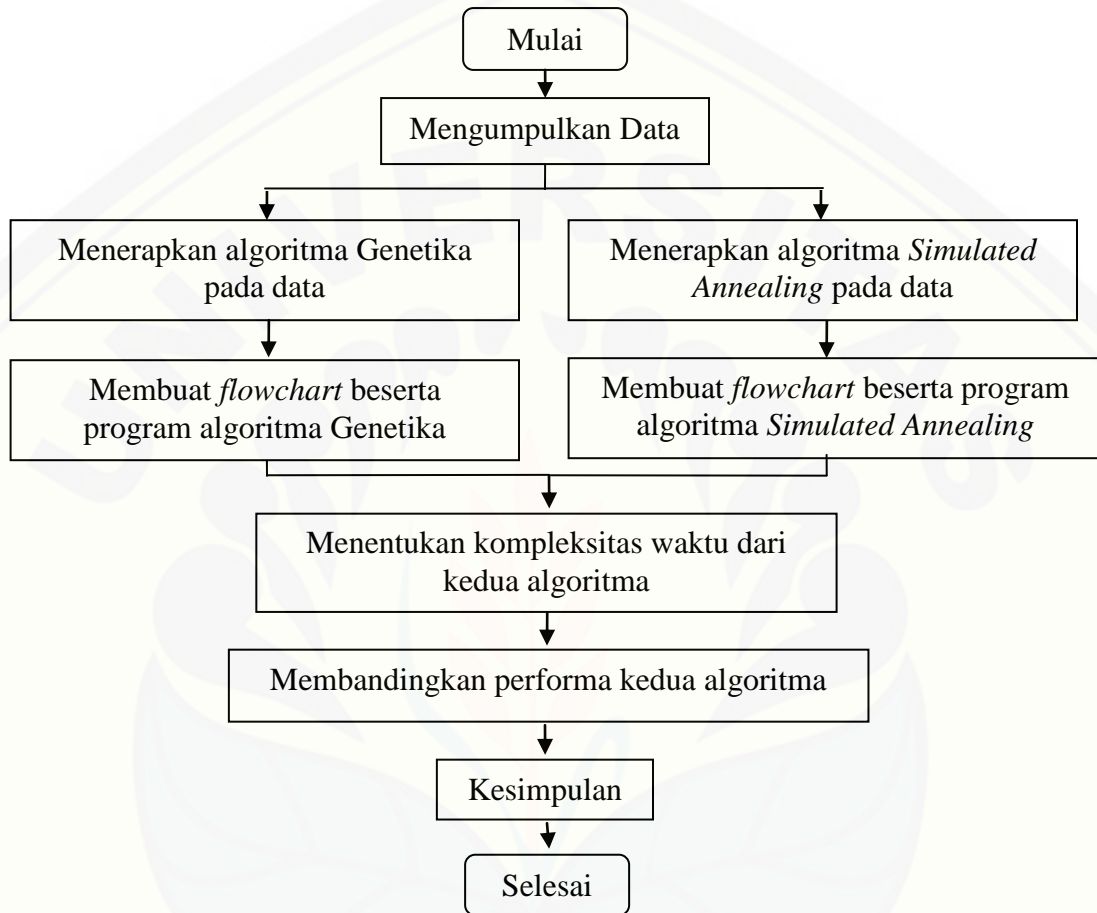
3.5 Menghitung Kompleksitas Waktu dari kedua Algoritma

3.6 Membandingkan performa dari kedua Algoritma

Membandingkan performa dari kedua algoritma diantaranya :

- Membandingkan nilai *makespan* dari kedua algoritma;
- Membandingkan efisiensi algoritma dengan menghitung kompleksitas dari kedua algoritma;
- Membandingkan kecepatan kekonvergenan dari kedua algoritma.

Gambar 3.5 berikut ini merupakan skema dari langkah-langkah yang dilakukan untuk menyelesaikan permasalahan diatas.



Gambar 3.5 Skema langkah-langkah penyelesaian

BAB 4. HASIL DAN PEMBAHASAN

4.1 Penyelesaian Penjadwalan Secara Manual

Untuk mengetahui cara kerja algoritma Genetika dan *Simulated Annealing* secara manual, berikut ini akan diberikan contoh penyelesaian penjadwalan *flowshop* menggunakan sampel data yang kecil berupa lima *job* dan empat mesin, seperti yang tercantum pada Tabel 4.1.

Tabel 4.1 Waktu Proses Produksi dengan 5 *job* dan 4 Mesin (menit)

Job \ Mesin	Mesin 1	Mesin 2	Mesin 3	Mesin 4
Job 1	62	26	38	90
Job 2	98	62	110	98
Job 3	84	38	62	92
Job 4	120	50	158	136
Job 5	128	54	150	172

4.1.2 Penjadwalan Menggunakan Algoritma Genetika

Berikut ini diberikan langkah-langkah penyelesaian penjadwalan secara manual dengan algoritma Genetika.

a. Membangkitkan populasi awal

Populasi awal dibangkitkan secara random. Populasi awal diperoleh dengan mengkodekan masalah penjadwalan ke dalam gen dan kromosom menggunakan pengkodean permutasi. Dalam pembentukan populasi ini ditentukan ukuran populasi sebanyak 5 kromosom contoh :

J4-J1-J2-J3-J5

J1-J3-J4-J2-J5

J1-J2-J3-J4-J5

J2-J3-J5-J4-J1

J3-J1-J5-J4-J2

kriteria optimasi pada perhitungan manual ini sebanyak 2 iterasi.

b. Mengevaluasi nilai objektif

Evaluasi nilai objektif dari populasi awal yang telah dibangkitkan dilakukan dengan cara menghitung nilai *makespan* untuk masing-masing kromosomnya. Perhitungan nilai *makespan* untuk populasi awal ditampilkan pada Tabel 4.2, Tabel 4.3, Tabel 4.4, Tabel 4.5 dan Tabel 4.6.

Tabel 4.2 Perhitungan *Makespan* Kromosom Pertama yang dibangkitkan pada Solusi Awal

	Mesin 1		Mesin 2		Mesin 3		Mesin 4	
	S	E	S	E	S	E	S	E
Job 4	0	110	110	160	160	318	318	454
Job 1	110	172	172	198	318	356	454	544
Job 2	172	270	270	332	356	466	544	642
Job 3	270	354	354	392	466	528	642	734
Job 5	354	482	482	536	536	686	734	906

Tabel 4.3 Perhitungan *Makespan* Kromosom kedua yang dibangkitkan pada Solusi Awal

	Mesin 1		Mesin 2		Mesin 3		Mesin 4	
	S	E	S	E	S	E	S	E
Job 1	0	62	62	88	88	126	126	216
Job 3	62	146	46	184	184	246	246	338
Job 4	146	256	256	306	306	464	464	600
Job 2	256	354	354	416	464	574	600	698
Job 5	354	482	482	536	574	724	724	896

Tabel 4.4 Perhitungan *Makespan* Kromosom ketiga yang dibangkitkan pada Solusi Awal

	Mesin 1		Mesin 2		Mesin 3		Mesin 4	
	S	E	S	E	S	E	S	E
Job 1	0	62	62	88	88	126	126	216
Job 2	62	160	160	222	222	332	332	430
Job 3	160	244	244	282	332	394	430	522
Job 4	244	354	354	404	404	562	562	698
Job 5	354	482	482	536	562	712	712	884

Tabel 4.5 Perhitungan *Makespan* Kromosom keempat yang dibangkitkan pada Solusi Awal

	Mesin 1		Mesin 2		Mesin 3		Mesin 4	
	S	E	S	E	S	E	S	E
Job 2	0	98	98	160	160	270	270	368
Job 3	98	182	182	220	270	332	368	460
Job 5	182	310	310	364	364	514	514	686
Job 4	310	420	420	470	514	672	686	822
Job 1	420	482	482	508	672	710	822	912

Tabel 4.6 Perhitungan *Makespan* Kromosom kelima yang dibangkitkan pada Solusi Awal

	Mesin 1		Mesin 2		Mesin 3		Mesin 4	
	S	E	S	E	S	E	S	E
Job 3	0	84	84	122	122	184	184	276
Job 1	84	146	146	172	184	22	276	366
Job 5	146	274	274	328	328	478	478	650
Job 4	274	384	384	434	478	636	650	786
Job 2	384	482	482	544	636	746	786	884

Keterangan :

S = start atau mulai

E = end atau berhenti

c. Mengevaluasi nilai *fitness*

pada evaluasi nilai *fitness* penentuan induk dapat dilihat pada tabel 4.7 berikut dengan mengambil nilai objektif (*makespan*) paling kecil dari populasi yang telah dibangkitkan yaitu terdapat pada kromosom 3 dan 5

Tabel 4.7 Penentuan Induk

Kromosom (k)	Pengkodean permutasi	Nilai objektif (<i>makespan</i>)	Seleksi
1	4,1,2,3,5	906	-
2	1,3,4,2,5	896	-
3	1,2,3,4,5	884	induk
4	2,3,5,4,1	912	-
5	3,1,5,4,2	884	Induk

d. Melakukan proses pindah silang (*crossover*)

Dari tabel 4.7 didapatkan induk dari kromosom 3 dan kromosom 5 sehingga dilakukan proses *crossover* dengan metode *PMX (Partial-Mapped-Crossover)* dengan cara yang ditunjukkan pada Gambar 3.2.

Tabel 4.8 Kromosom induk 1 dan induk 2

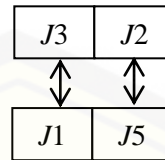
Kromosom (k)	Induk	Urutan <i>job</i>				
		1	2	3	4	5
3	Induk 1	<i>J1</i>	<i>J2</i>	<i>J3</i>	<i>J4</i>	<i>J5</i>
5	Induk 2	<i>J3</i>	<i>J1</i>	<i>J5</i>	<i>J4</i>	<i>J2</i>

- 1) Menukar gen yang berada pada urutan posisi 1 dalam kromosom induk 1 dengan gen yang berada pada urutan posisi 1 dalam kromosom induk 2
- 2) Menukar gen yang berada pada urutan posisi 5 dalam kromosom induk 1 dengan gen yang berada pada urutan posisi 5 dalam kromosom induk 2, yang ditunjukkan pada Tabel 4.9.

Tabel 4.9 Calon anak 1 dan Calon anak 2

Induk	Urutan <i>job</i>				
	1	2	3	4	5
Calon anak 1	<i>J3</i>	<i>J2</i>	<i>J3</i>	<i>J4</i>	<i>J2</i>
Calon anak 2	<i>J1</i>	<i>J1</i>	<i>J5</i>	<i>J4</i>	<i>J5</i>

- 3) Menentukan relasi pemetaan dari gen yang telah ditukar seperti pada Gambar 4.1.



Gambar 4.1 Relasi pemetaan

Dari Gambar 4.1 didapatkan relasi pemetaan sebagai berikut.

$$J3 \leftrightarrow J1 \text{ dan } J2 \leftrightarrow J5$$

- 4) Ubah gen-gen diluar posisi penukaran 1) dan 2) sesuai dengan relasi pemetaan sehingga dihasilkan anak seperti yang ditunjukkan pada Tabel 4.10.

Tabel 4.10 Anak 1 dan Anak 2 dengan Relasi Pemetaan

Kromosom (<i>k</i>)	Anak	Urutan <i>job</i>				
6	Anak 1	<i>J3</i>	<i>J5</i>	<i>J1</i>	<i>J4</i>	<i>J2</i>
7	Anak 2	<i>J1</i>	<i>J3</i>	<i>J2</i>	<i>J4</i>	<i>J5</i>

- e. Menghitung nilai objektif anak

Menghitung nilai objektif dari anak yang didapatkan dengan cara yang sama seperti yang ditunjukkan pada Tabel 4.2 sampai dengan Tabel 4.6 sehingga didapatkan nilai objektif dari anak 1 sebagai kromosom 6 yaitu 912 dan anak 2 sebagai kromosom 7 adalah 896, kemudian kromosom anak tersebut akan menggantikan kromosom yang memiliki nilai objektif besar pada populasi yang dibangkitkan.

- f. Seleksi

Pembentukan generasi baru yang terdiri dari kromosom-kromosom yang dibangkitkan pada solusi awal beserta kromosom anak 2 yang telah menggantikan kromosom 4, generasi baru setelah proses seleksi dapat dilihat pada Tabel 4.11.

Tabel 4.11 Generasi Baru Setelah Seleksi

Kromosom (k)	Pengkodean permutasi	Nilai objektif (<i>makespan</i>)	Urutan <i>job</i>
1	4,1,2,3,5	906	<i>J4-J1-J2-J3-J5</i>
2	1,3,4,2,5	896	<i>J1-J3-J4-J2-J5</i>
3	1,2,3,4,5	884	<i>J1-J2-J3-J4-J5</i>
7	1,3,2,4,5	896	<i>J1-J3-J2-J4-J5</i>
5	3,1,5,4,2	884	<i>J3-J1-J5-J4-J2</i>

- g. Mengulangi langkah a. dan langkah selanjutnya. sampai iterasi yang ditentukan yaitu sebanyak 2 sehingga proses pencarian berhenti. Hasil generasi kedua dapat dilihat pada Tabel 4.12.

Tabel 4.12 Populasi pada Generasi kedua

Kromosom (k)	Pengkodean permutasi	Nilai objektif (<i>makespan</i>)	Urutan <i>job</i>
14	1,4,3,2,5	888	<i>J1-J4-J3-J2-J5</i>
10	3,5,2,4,1	912	<i>J3-J5-J2-J4-J1</i>
11	3,4,1,5,2	898	<i>J3-J4-J1-J5-J2</i>
12	5,2,3,1,4	920	<i>J5-J2-J3-J1-J4</i>
13	1,4,2,3,5	888	<i>J1-J4-J2-J3-J5</i>

- h. Iterasi telah tercapai

Solusi optimal pada algoritma Genetika adalah solusi yang memiliki *makespan* terkecil pada iterasi yang ditentukan. Dari itersi 1 didapatkan hasil *makespan* terkecil yaitu terdapat pada kromosom Tabel 4.12 terdapat kromosom yang memiliki *makespan* terkecil yaitu 884 menit terdapat pada kromosom 3 dan kromosom 5 dapat dilihat pada Tabel 4.11, sedangkan pada iterasi 2 didapatkan hasil *makespan* terkecil yaitu 888 menit terdapat pada kromosom 14 dan kromosom 13 dapat dilihat pada Tabel 4.12.

4.1.2 Penjadwalan Menggunakan Algoritma *Simulated Annealing*

Berikut ini diberikan langkah-langkah penyelesaian penjadwalan secara manual dengan algoritma *Smulated Annealing*.

a. Membentuk solusi awal

Penjadwalan jenis *flowshop* untuk algoritma *Simulated Annealing* membutuhkan sebuah solusi awal sebagai acuan dalam perhitungannya. Penentuan solusi awal yang digunakan dalam algoritma *Simulated Annealing* dilakukan secara random yang memiliki urutan jadwal *J1-J5-J3-J4-J2* dengan nilai *makespan* 892 menit sebagai kondisi awal

b. Membangkitkan solusi baru

Proses pembentukan solusi baru diperoleh menggunakan metode *insertion* yang telah dijelaskan dengan ilustrasi Gambar 2.7. Pada metode ini ditentukan juga berapa kali pertukaran dilakukan (replikasi), yaitu sebanyak 5 kali. Angka replikasi, merupakan angka yang menunjukkan berapa kali *loop* dalam harus dilakukan sebelum menurunkan parameter kontrol. Urutan jadwal yang didapat dari hasil pertukaran yaitu :

J2-J1-J4-J3-J5 dengan *makespan* 878 menit

J1-J2-J4-J3-J5 dengan *makespan* 890 menit

J3-J2-J4-J1-J5 dengan *makespan* 910 menit

J5-J2-J4-J1-J3 dengan *makespan* 920 menit

J5-J2-J3-J1-J4 dengan *makespan* 920 menit

Dari kelima urutan jadwal yang dihasilkan akan dipilih solusi baru yang terbaik sehingga didapatkan urutan jadwal sebagai berikut.

J2-J1-J4-J3-J5 dengan *makespan* 878 menit

Solusi ini disimpan sebagai kondisi baru, kemudian dibandingkan dengan kondisi awal yaitu 892 menit. Kondisi baru yang dihasilkan memiliki waktu yang lebih baik dari kondisi awal sehingga solusi baru diterima.

c. Penurunan nilai parameter kontrol

Nilai parameter awal yang telah ditentukan sebelumnya adalah 125 dengan faktor reduksi sebesar 0,45. Faktor reduksi, merupakan angka yang digunakan untuk menurunkan suhu secara bertahap dan terkendali. Maka nilai parameter

yang sebelumnya 125 diturunkan dengan persamaan (2.3) menjadi 56,25. Pada nilai ini dibangkitkan lagi solusi tetangga dengan cara seperti langkah b. Penurunan nilai parameter kontrol digunakan untuk menunjukkan berapa kali iterasi yang dilakukan. Proses ini berlanjut hingga iterasi maksimal yang ditentukan yaitu sampai iterasi kedua. Hasil dari proses yang dilakukan akan ditampilkan pada Tabel 4.13 berikut.

Tabel 4.13 Hasil Pertukaran untuk Setiap Iterasi

Jumlah iterasi	Parameter kontrol	Solusi baru/ urutan jadwal	Makespan
Iterasi 1	125	<i>J2-J1-J4-J3-J5</i>	878 menit
		<i>J1-J2-J4-J3-J5</i>	890 menit
		<i>J3-J2-J4-J1-J5</i>	910 menit
		<i>J5-J2-J4-J1-J3</i>	920 menit
		<i>J5-J2-J3-J1-J4</i>	920 menit
		Solusi Terpilih	<i>J2-J1-J4-J3-J5</i>
Iterasi 2	56,25	<i>J4-J2-J5-J1-J3</i>	932 menit
		<i>J2-J4-J5-J1-J3</i>	932 menit
		<i>J1-J4-J5-J2-J3</i>	892 menit
		<i>J3-J4-J5-J2-J1</i>	912 menit
		<i>J3-J5-J1-J4-J2</i>	912 menit
		Solusi Terpilih	<i>J1-J4-J5-J2-J3</i>

d. Mengevaluasi solusi baru

Pada nilai parameter kontrol = 125 kondisi baru yang terpilih adalah urutan jadwal dengan nilai *makespan* 878 menit. Solusi ini dibandingkan dengan kondisi awal yaitu urutan jadwal dengan nilai *makespan* 892 menit. Kondisi baru yang dihasilkan lebih baik dari kondisi awal sehingga kondisi awal diterima menjadi kondisi baru yaitu dengan urutan jadwal *J2-J1-J4-J3-J5* dengan nilai *makespan* 878 menit.

Pada nilai parameter kontrol = 56,25 kondisi baru yang terpilih adalah urutan jadwal dengan nilai *makespan* 892 menit. Solusi ini dibandingkan dengan kondisi awal yaitu urutan jadwal dengan nilai *makespan* 878 menit. Kondisi baru yang dihasilkan tidak lebih baik dari kondisi awal sehingga

perlu dilakukan evaluasi solusi baru. Pada metode *Simulated Annealing* solusi yang lebih buruk dapat diterima dengan probabilitas tertentu untuk terhindar dari minimum lokal. Dengan menggunakan persamaan (2.4) didapatkan selisih nilai objektif (ΔE) = 14 menit, kemudian membangkitkan bilangan random 0,745 sehingga dari persamaan (2.5) didapatkan $P(\Delta E) = 0,78$, didapatkan hasil $r < P(\Delta E)$ maka perubahan kondisi baru menjadi kondisi awal diperbolehkan .

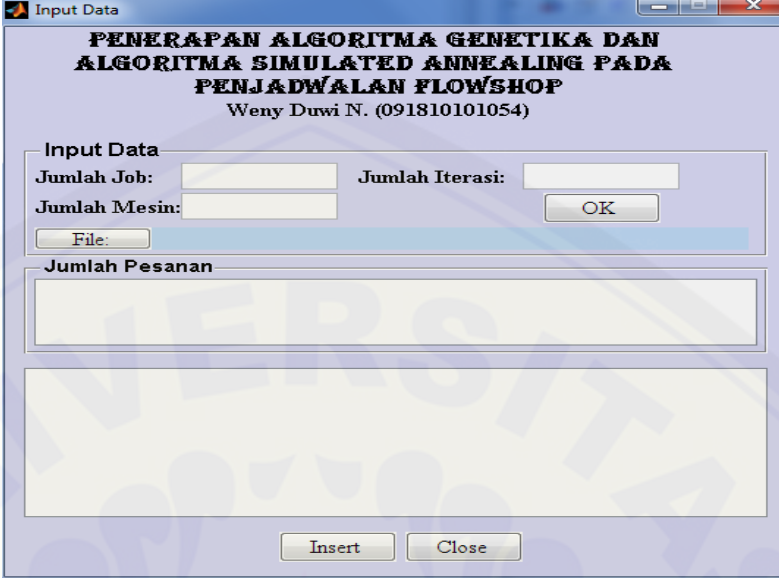
e. Mengecek iterasi maksimal

Iterasi yang diinginkan telah tercapai yaitu sejumlah 2 iterasi sehingga iterasi telah terpenuhi. Dari keseluruhan urutan jadwal yang didapat maka dipilih satu iterasi terakhir yang mempunyai nilai *makespan* terkecil yaitu dengan urutan jadwal *J1-J4-J5-J2-J3* dengan nilai *makespan* 892 menit.

4.2 Penyelesaian dengan Menggunakan Program

Program penjadwalan *flowshop* menggunakan algoritma Genetika dan algoritma *Simulated Annealing* dibuat dengan bantuan program *Matlab*. *Matlab* adalah bahasa pemrograman tingkat tinggi yang dikhususkan untuk komputasi teknis. Bahasa ini mengintegrasikan kemampuan komputasi, visualisasi dan pemrograman dalam sebuah lingkungan yang mudah untuk digunakan.

Programan penjadwalan *flowshop* menggunakan algoritma Genetika dan *Simulated Annealing* diberi nama GA_dan_SA. Semua fungsi yang dibuat kemudian dipanggil melalui fungsi utama yaitu RESET. Tampilan program ditunjukkan pada Gambar 4.2, pada Gambar 4.2 terdapat beberapa kolom yang ditampilkan, diantaranya yaitu :

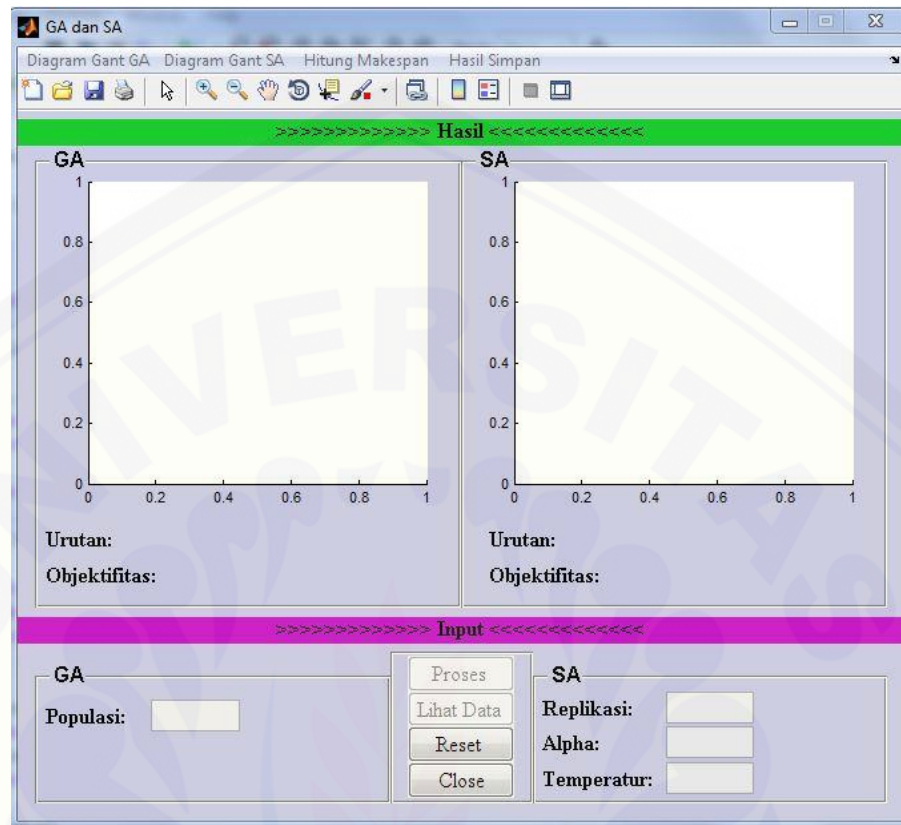


The screenshot shows a software window titled "Input Data" with a blue border. The title bar includes standard Windows window controls. The main content area has a light blue background and contains the following elements:

- Title:** "PENERAPAN ALGORITMA GENETIKA DAN ALGORITMA SIMULATED ANNEALING PADA PENJADWALAN FLOWSHOP" in bold black text.
- Author:** "Weny Duwi N. (091810101054)" in smaller black text.
- Input Data Section:** A light blue rectangular box containing:
 - Labels "Jumlah Job:" and "Jumlah Iterasi:" followed by empty text input fields.
 - Label "Jumlah Mesin:" followed by an empty text input field.
 - A "File:" label followed by a file selection button.
 - An "OK" button positioned to the right of the "Jumlah Iterasi:" field.
- Jumlah Pesanan Section:** A light blue rectangular box containing two empty text input fields for manual entry.
- Buttons:** "Insert" and "Close" buttons are located at the bottom center of the window.

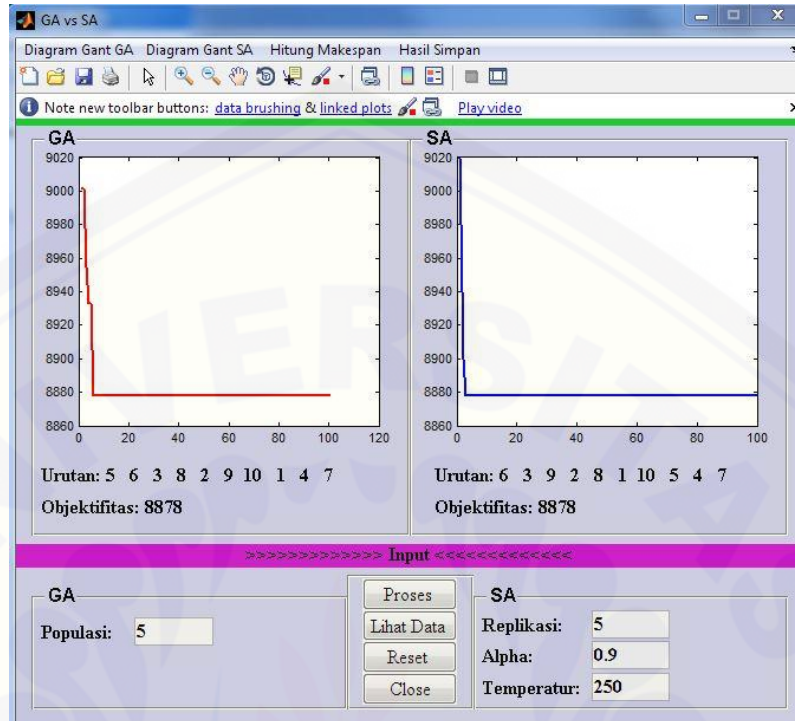
Gambar 4.2 Tampilan program RESET

- a. *Input data*, yaitu tampilan awal program untuk memulai meng-*input* jumlah *job*, jumlah mesin dan jumlah iterasi jika ingin meng-*input* data secara manual, namun data juga dapat diambil dari file yang telah tersimpan dalam bentuk txt dengan mengklik tombol File maka data akan muncul secara otomatis mengisi kolom jumlah *job* dan juga jumlah mesin. Kolom dibawah Jumlah Pesanan mendefinisikan berapa banyak pesanan dari setiap *job* yang ada, dimana kolom tersebut dapat diedit secara manual.
- b. *Insert*, yaitu tombol untuk menuju konvergensi dari kedua algoritma yang akan ditunjukkan dari tampilan program pada Gambar 4.3 Konvergensi GA dan SA, terdapat kolom yang perlu di-*input*-kan suatu nilai parameter-parameter yang dibutuhkan kedua algoritma tersebut diantaranya yaitu kolom replikasi, alpha dan temperature untuk algoritma *Simulated Annealing* kemudian populasi untuk algoritma Genetika.

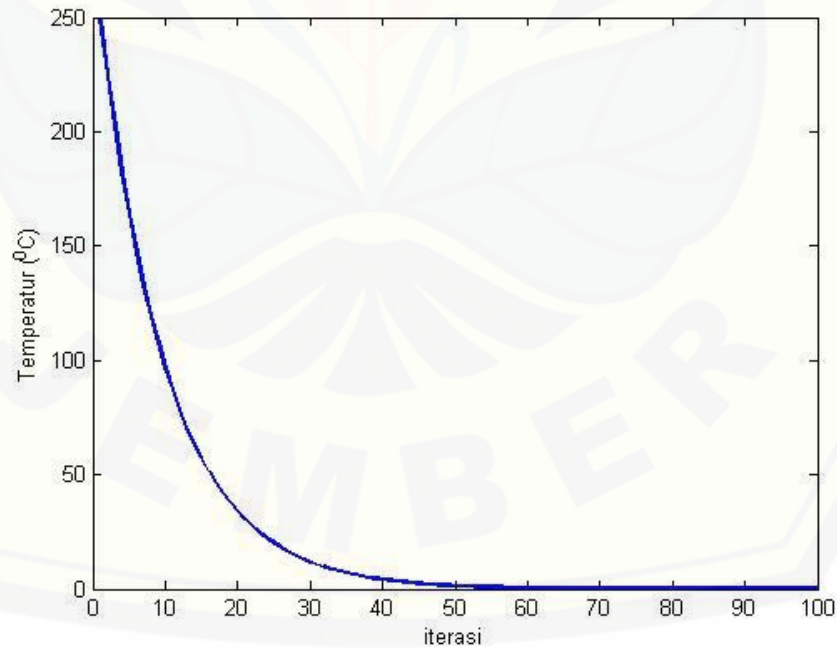


Gambar 4.3 Konvergensi GA dan SA

- c. Proses, yaitu tombol untuk mengetahui tingkat konvergensi dan hasil optimal urutan job dan *makespan* terbaik yang didapat dari kedua algoritma beserta penurunan suhu (algoritma *Simulated Annealing*) yang akan ditunjukkan pada Gambar 4.4 dan Gambar 4.5 setelah meng-*input* parameter-parameter pada masing-masing kolom bagian kedua algoritma tersebut. Hal ini berarti, dari data yang telah di-*input*-kan akan diperoleh jadwal dari algoritma Genetika yaitu $J5 - J2 - J1 - J7 - J6 - J8 - J9 - J10 - J3 - J4$ dengan nilai *makespan* 9038 menit, sedangkan untuk algoritma *Simulated Annealing* yaitu $J4 - J2 - J6 - J9 - J7 - J10 - J8 - J3 - J1 - J5$ dengan nilai *makespan* 8878 menit.



Gambar 4.4 Tingkat konvergensi algoritma Genetika dan *Simulated Annealing*

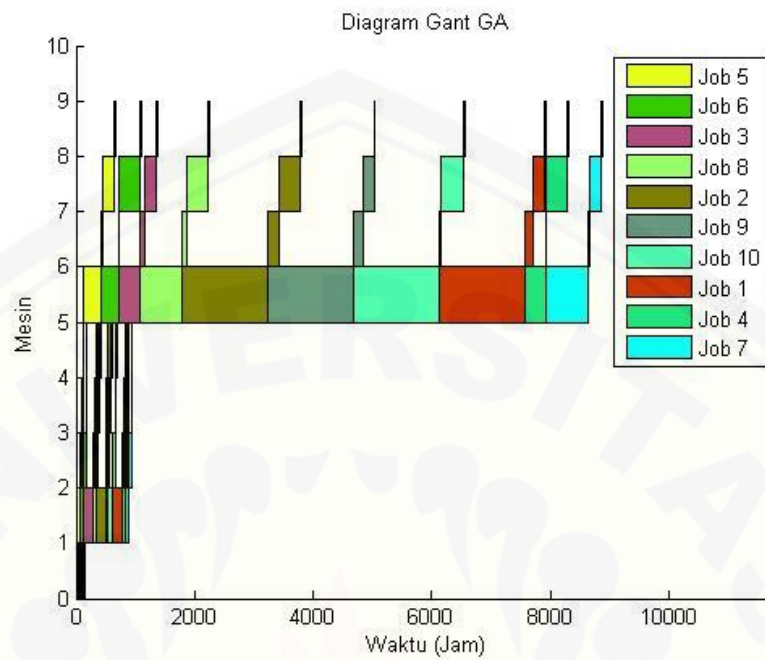


Gambar 4.5 Penurunan temperatur pada algoritma *Simulated Annealing*

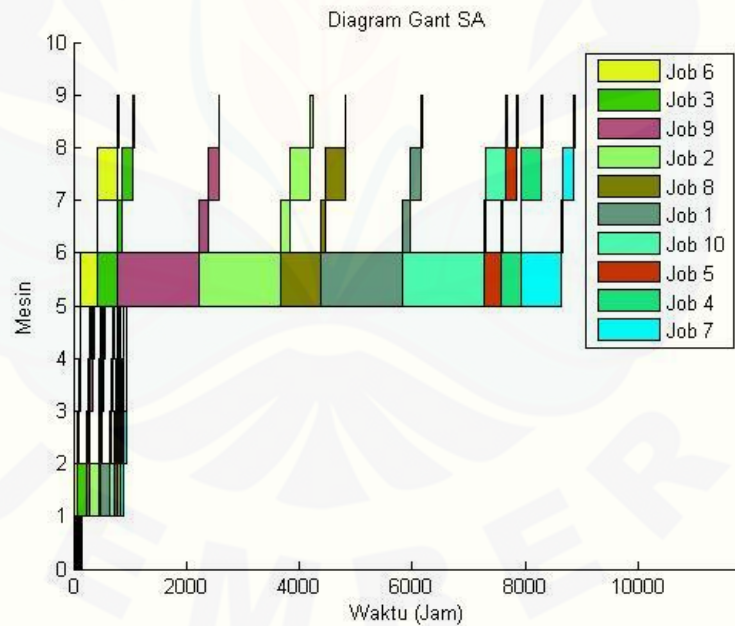
- d. Lihat data, yaitu tombol untuk mengecek data inputan yang akan diproses untuk mendapatkan tingkat konvergensi dari algoritma genetika dan *Simulated Annealing*.
- e. Reset, yaitu tombol untuk mengulang kembali meng-*input* data dan pesanan yang ditunjukkan pada Gambar 4.2.
- f. Menu-menu pada Gambar 4.4 terdiri dari Hitung *makespan* yaitu untuk perhitungan manual nilai *makespan* dari urutan *job* yang ditentukan ditunjukkan pada Gambar 4.6, Diagram Gant GA dan Diagram Gant SA yaitu untuk mengetahui urutan *job – job* yang tersedia dikerjakan di mesin-mesin yang ada berupa diagram yang ditunjukkan pada Gambar 4.7 dan 4.8, Urutan mesin dari masing-masing algoritma digambarkan dengan sumbu horisontal dan sumbu vertikal menggambarkan pergerakan waktu. Perbedaan warna yang digunakan pada *gant chart* bertujuan untuk memudahkan pengguna untuk membedakan waktu yang diperlukan oleh tiap *job* ketika diproses dalam urutan mesin yang ada.

	S	E	S	E	S	E	S	E
Job 4	118	478	478	488	488	848	848	878
Job 2	478	1918	1918	2098	2098	2458	2458	2488
Job 7	1918	2638	2638	2668	2668	2848	2848	2878
Job 1	2638	4078	4078	4228	4228	4408	4408	4438
Job 5	4078	4378	4378	4408	4408	4588	4588	4618
Job 6	4378	4678	4678	4688	4688	5048	5048	5078
Job 3	4678	5038	5038	5123	5123	5303	5303	5333
Job 8	5038	5758	5758	5830	5830	6190	6190	6200
Job 9	5758	7198	7198	7366	7366	7546	7546	7561
Job 10	7198	8638	8638	8683	8683	9043	9043	9068

Gambar 4.6 Hasil perhitungan manual nilai *makespan*



Gambar 4.7 Diagram *gant* hasil akhir dari algoritma Genetika



Gambar 4.8 Diagram *gant* hasil akhir dari algoritma *Simulated Annealing*

Jika ingin mengetahui solusi yang dihasilkan akan selalu sama dalam setiap kali pengujian, maka harus dilakukan pengujian minimal sebanyak 14 kali dengan jumlah iterasi pengujian sebanyak 100 iterasi dan 5 kali pengulangan. Hal

ini bertujuan untuk menguji seberapa baik solusi yang dihasilkan oleh masing-masing algoritma apabila dikerjakan dalam jumlah iterasi yang sama. Setelah semua parameter-parameter yang dibutuhkan masing-masing algoritma di-*input*-kan dalam program output GUI Konvergensi GA_dan_SA kemudian di jalankan sebanyak 14 kali pengujian, sehingga diperoleh output yang dirangkum pada Gambar 4.11 kecepatan konvergensi dari dua algoritma . Pengujian pada algoritma Genetika terlihat bahwa nilai *makespan* yang berbeda-beda nilainya pada urutan jadwal yang tidak sama untuk setiap kali proses namun pada tingkat konvergensinya lebih cepat, sedangkan pengujian pada algoritma *Simulated Annealing* terlihat bahwa nilai *makespan* sama pada urutan jadwal yang berbeda untuk setiap kali proses namun tingkat konvergensinya lebih lambat. Pada algoritma Genetika urutan jadwal terbaik di peroleh pada proses ke-3, 5, 9, dan 14 dengan salah satu urutan jadwalnya $J5 - J4 - J1 - J6 - J10 - J8 - J3 - J2 - J9 - J7$ dan nilai *makespan*-nya adalah 8878 menit. Sedangkan pada algoritma *Simulated Annealing* urutan jadwal terbaik diperoleh untuk setiap kali proses dengan salah satu urutan jadwal $J4 - J2 - J6 - J9 - J7 - J10 - J8 - J3 - J1 - J5$ dan nilai *makespan*-nya yaitu 8878 menit. Dari 14 kali proses tersebut dapat dilihat hasilnya bahwa algoritma algoritma Genetika memiliki tingkat konvergensi lebih cepat dibanding dengan *Simulated Annealing* sehingga algoritma Genetika lebih efektif apabila diterapkan pada penjadwalan produksi kerupuk Samjaya, karena dapat mengurangi waktu operasional mesin dalam proses produksi dengan jumlah pesanan masing-masing 1 kwintal untuk setiap *job*.

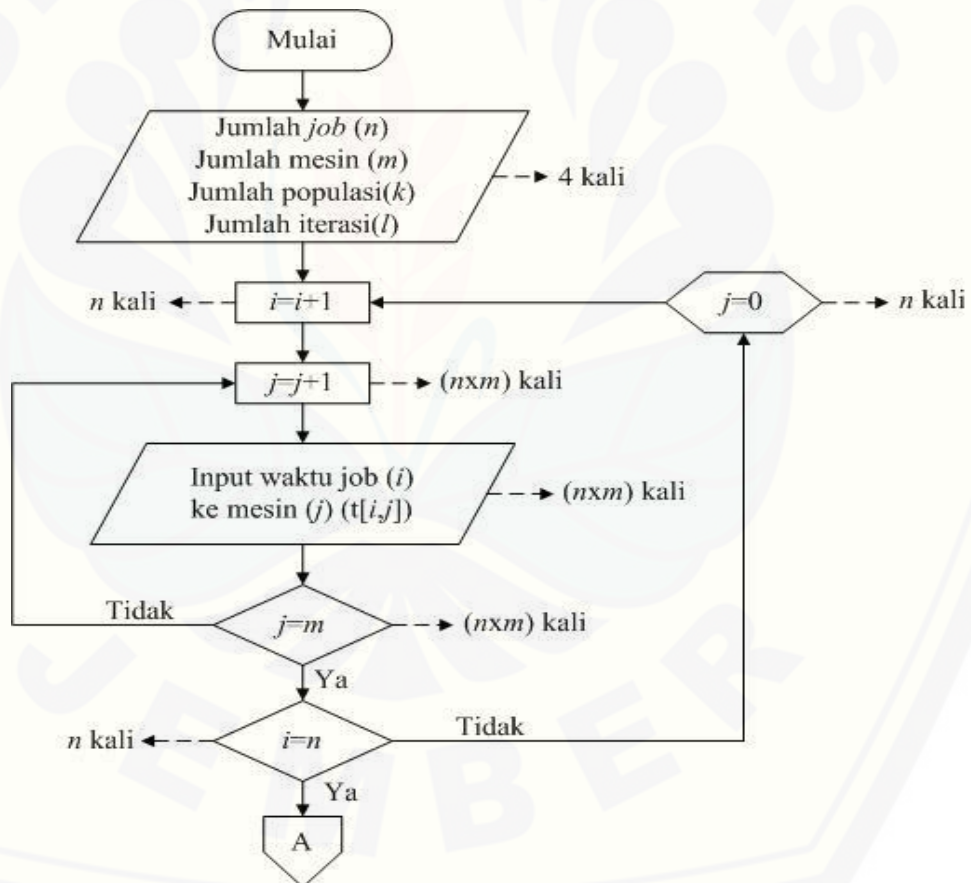
Perbedaan solusi yang didapatkan oleh kedua algoritma tersebut dikarenakan bahwa kedua algoritma merupakan algoritma dari metode *heuristik*, dimana untuk setiap kali proses pencarian solusi dilakukan secara random sehingga tidak dapat dijamin bahwa hasil dari tiap solusi selalu sama dan terbaik. Namun, diharapkan dapat mendekati optimal global dan dapat diterima solusinya.

4.3 Perbandingan Algoritma Genetika dan Algoritma *Simulated Annealing*

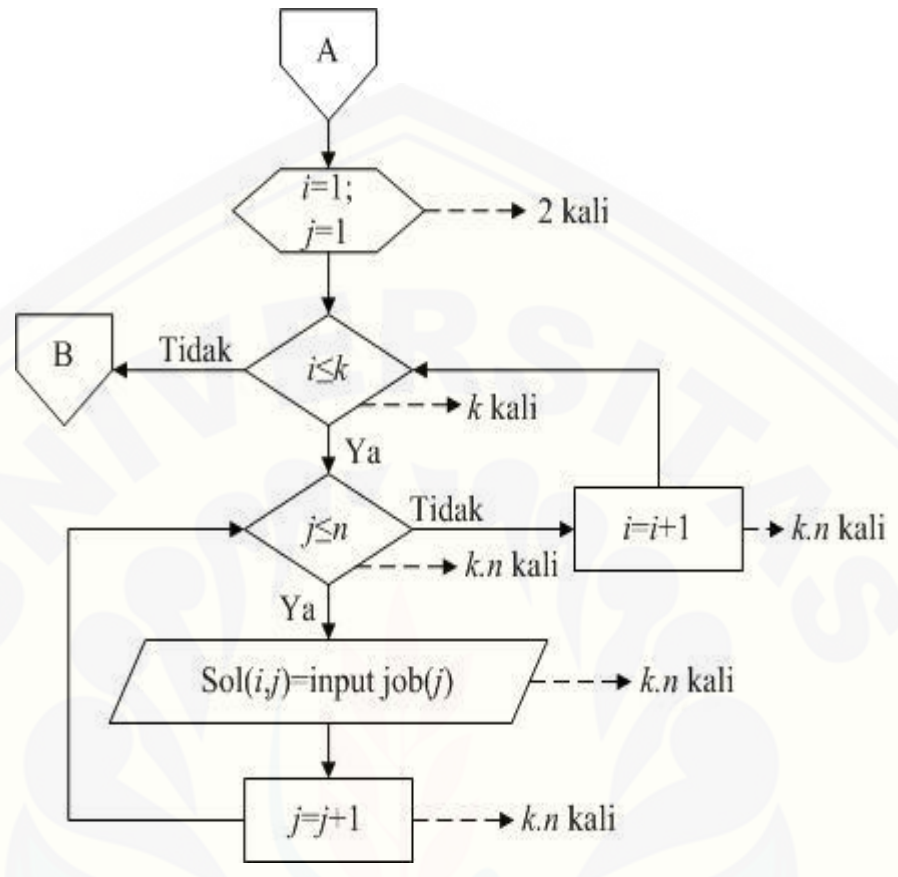
4.3.1 Perhitungan kompleksitas waktu

a. Kompleksitas waktu solusi awal

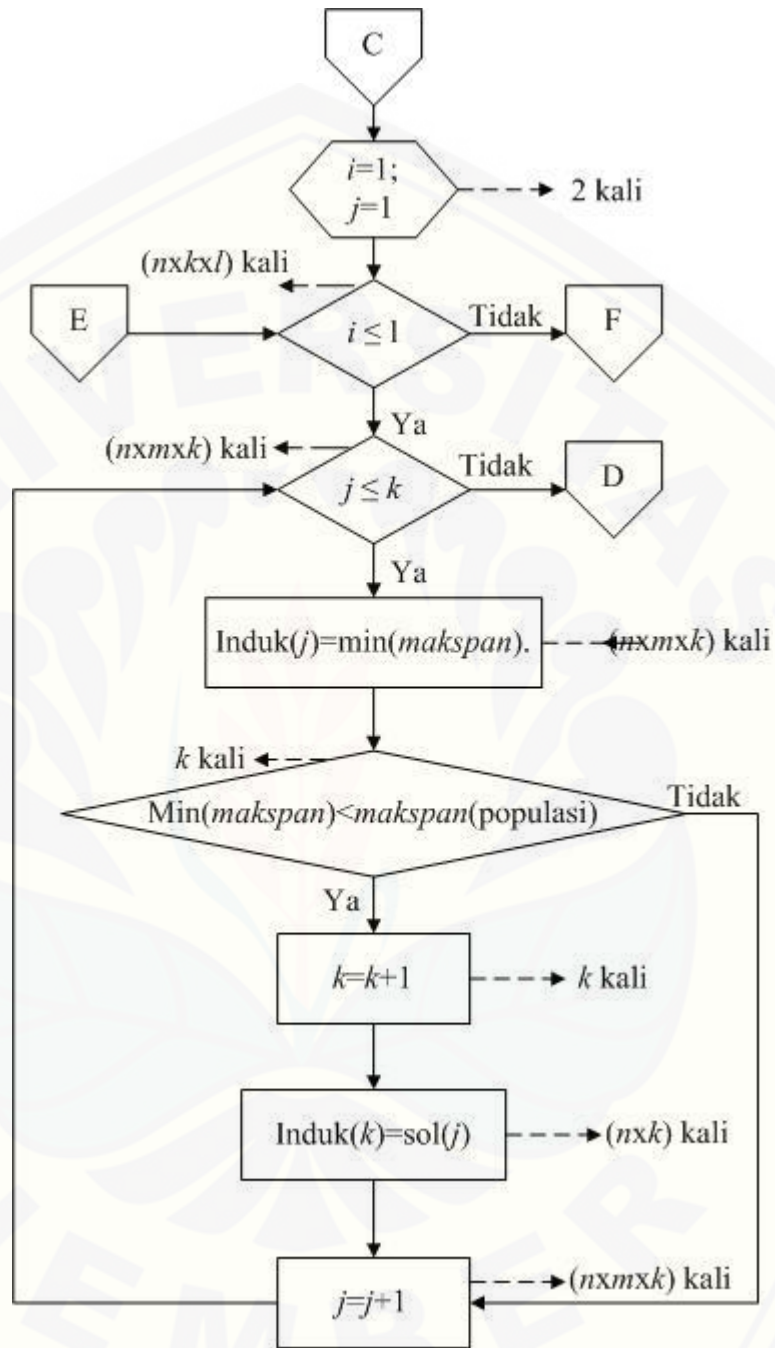
Untuk menentukan kemangkusan algoritma, penulis menyertakan perhitungan kompleksitas waktu algoritma dengan memperhatikan tahapan komputasi masing-masing algoritma pada *flowchart*. Jika kedua algoritma sama-sama membutuhkan solusi awal sebagai acuan untuk perhitungan selanjutnya, maka pada Gambar 4.9 akan menampilkan tahapan perhitungan kompleksitas algoritma Genetika.



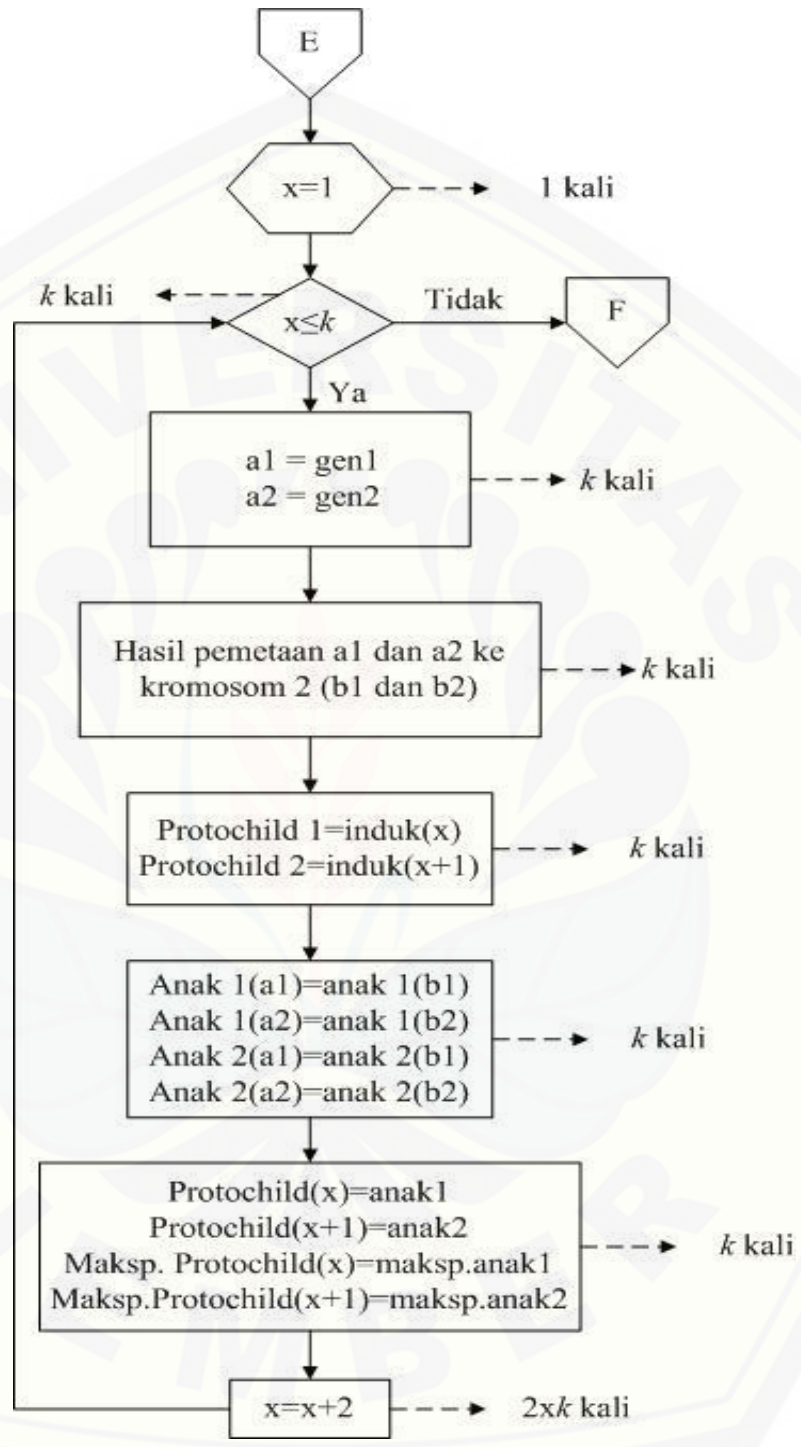
Jumlah perhitungannya adalah $T_{AG1} = 4 + 3n + 3nm$



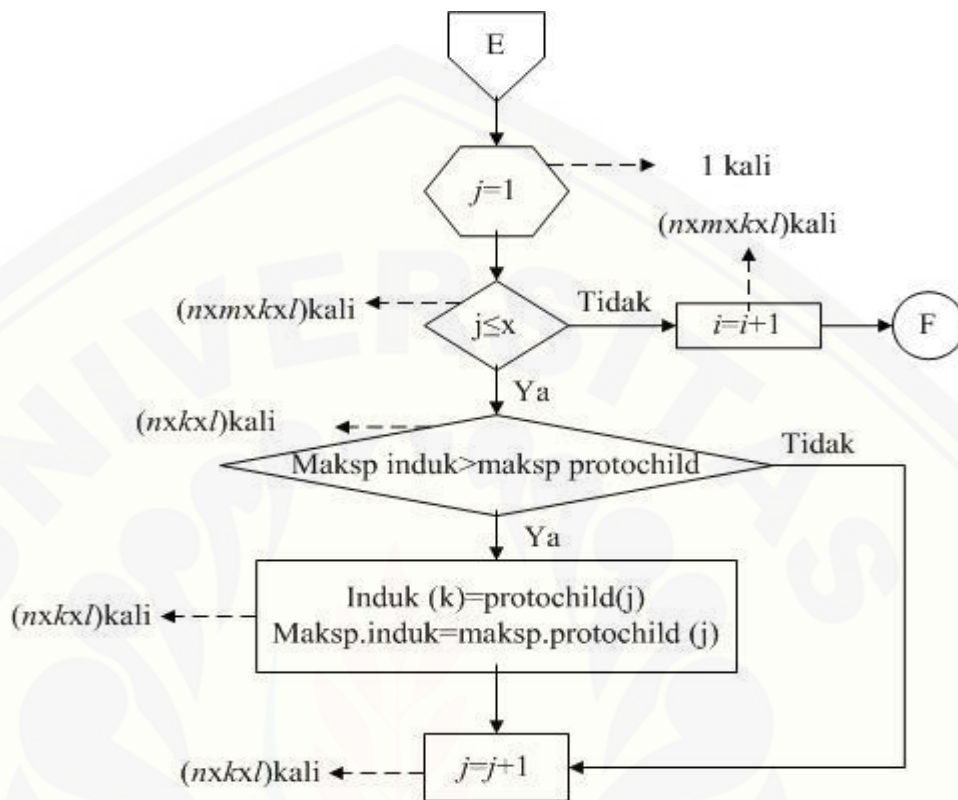
Jumlah perhitungannya adalah $T_{AG2} = 2 + k + 4kn$



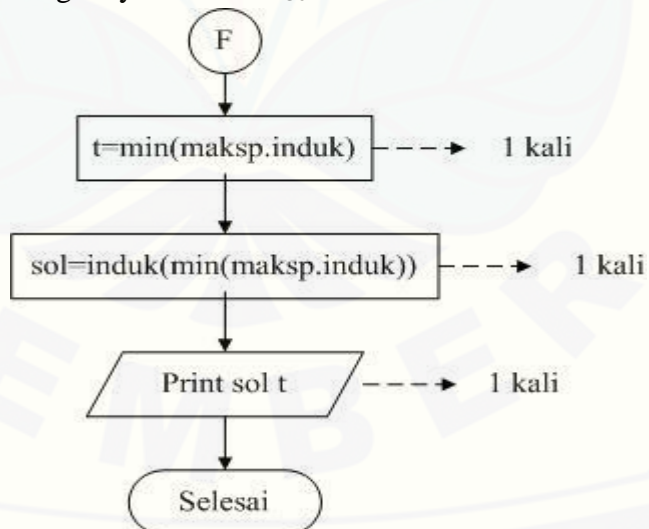
Jumlah perhitungannya adalah $T_{AG4} = 2+1+kl+5klmn$



Jumlah perhitungannya adalah $T_{AG5} = 1+8k$



Jumlah perhitungannya adalah $T_{AG6} = 1 + 2klnm + 3kln$



Jumlah perhitungannya adalah $T_{AG7} = 3$

4.9 Gambar flowchart algoritma Genetika

Berikut perhitungan kompleksitas waktu dari tahapan perhitungan untuk algoritma Genetika.

$$\begin{aligned}
 T(n, m, k, l) &= T_{AG1} + T_{AG2} + T_{AG3} + T_{AG4} + T_{AG5} + T_{AG6} + T_{AG7} \\
 &= (4 + 3n + 3nm) + (2 + k + 4kn) + (1 + 4k + 5kn + \\
 &\quad 7knm) + (2 + l + kl + klmn) + (1 + 8k) + (1 + 2klmn + \\
 &\quad 3kln) + 3 \\
 &= (4 + 2 + 1 + 2 + 1 + 1 + 3) + (3)n + (3)nm + (1 + 4 + 8)k + \\
 &\quad (4 + 5)kn + (7knm) + (1)l + (1)kl + (1 + 2)klmn + (3)kln \\
 &= 14 + 3n + 3nm + 13k + 9kn + 7knm + l + kl + 3klmn + \\
 &\quad 3kln
 \end{aligned}$$

Berikut perhitungan kompleksitas waktu dari tahapan perhitungan algoritma Genetika dengan satu kali iterasi (l) dan lima jumlah populasi (k)

$$\begin{aligned}
 T(n, m) &= 14 + 3n + 3nm + 13.5 + 9.5n + 7.5nm + 1 + 5.1 + 3.5.1nm + \\
 &\quad 3.5.1.n
 \end{aligned}$$

$$T(n, m) = 14 + 3n + 3nm + 65 + 45n + 35nm + 1 + 5 + 15nm + 15n$$

$$T(n, m) = 85 + 63n + 53nm$$

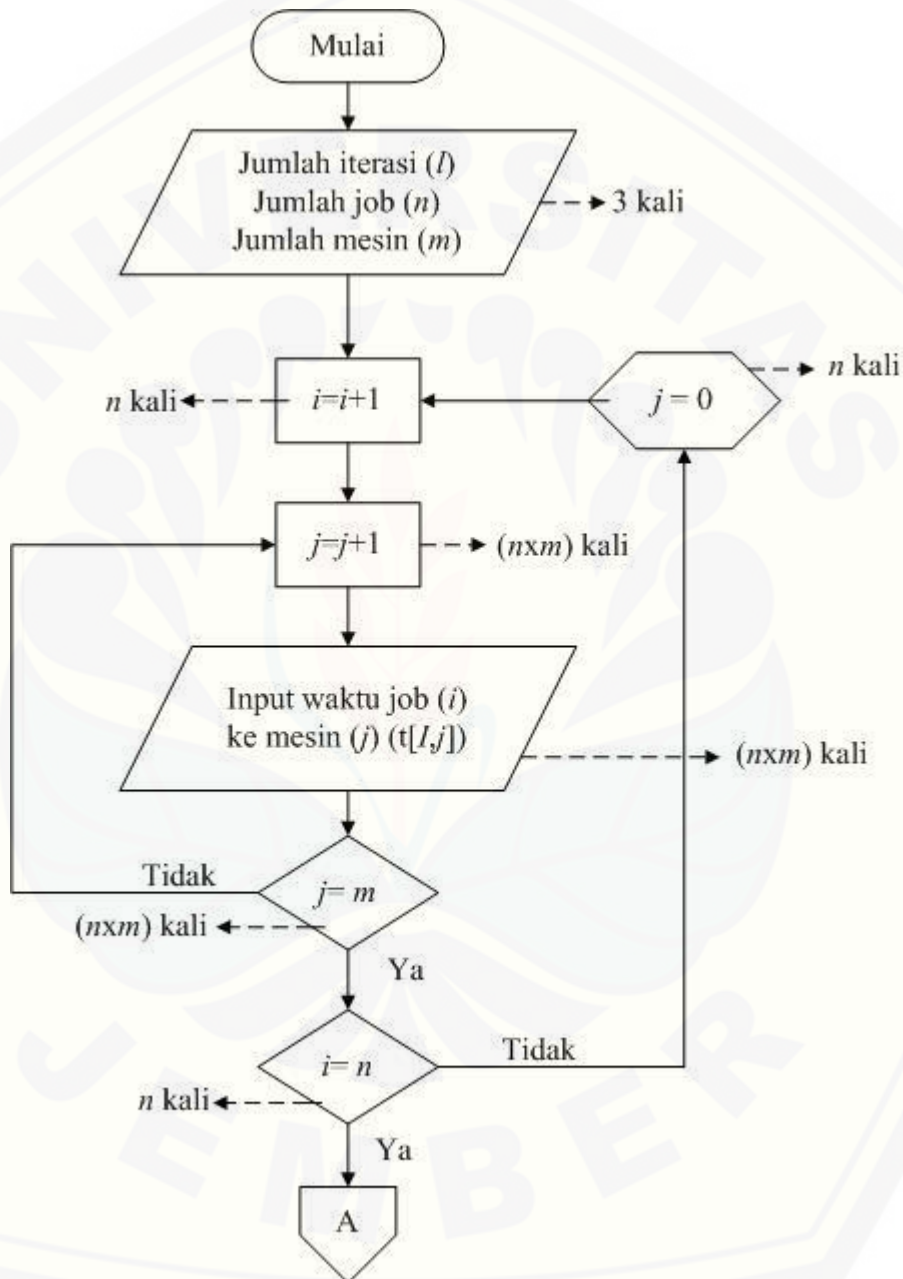
Karena

$$85 + 63n + 53nm \leq 85nm + 63nm + 53nm = 201nm$$

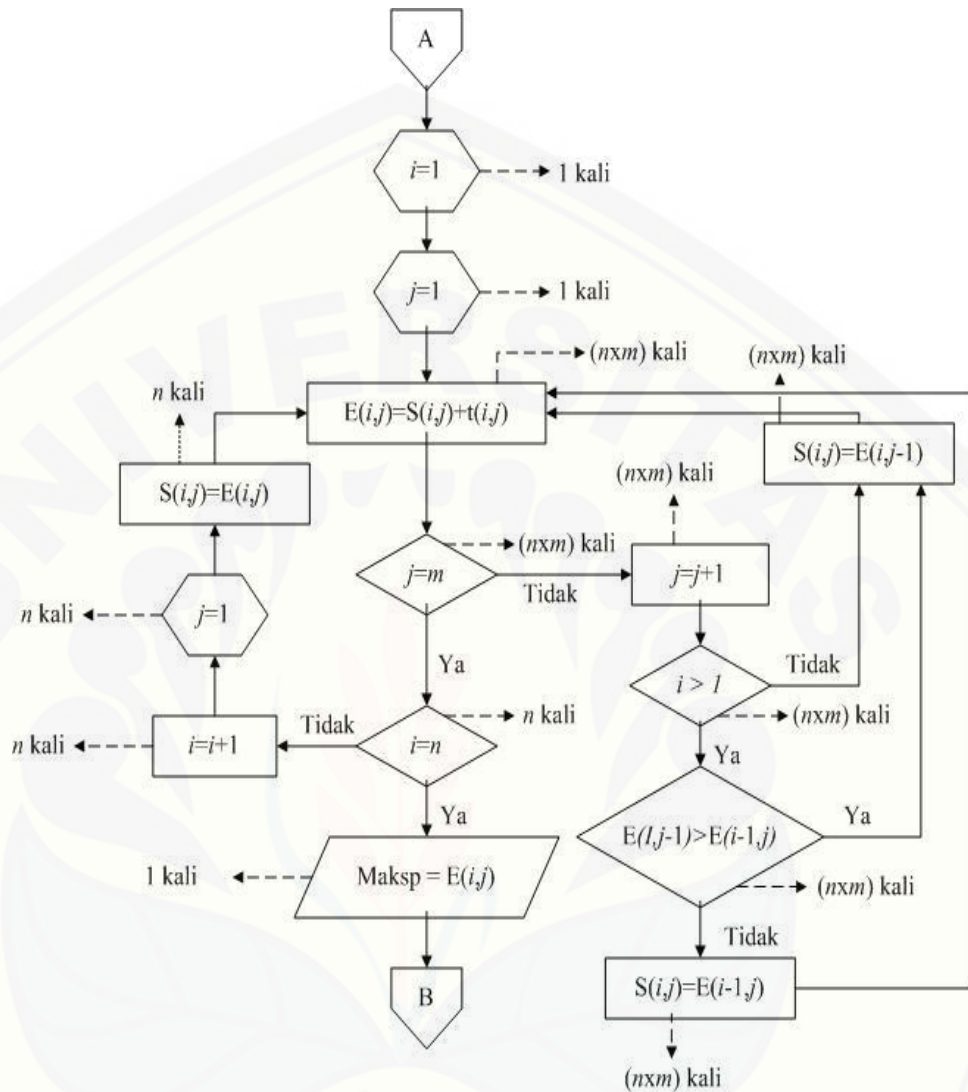
Maka kompleksitas algoritma Genetika adalah $O(nm)$

b. Kompleksitas Waktu Algoritma *Simulated Annealing*

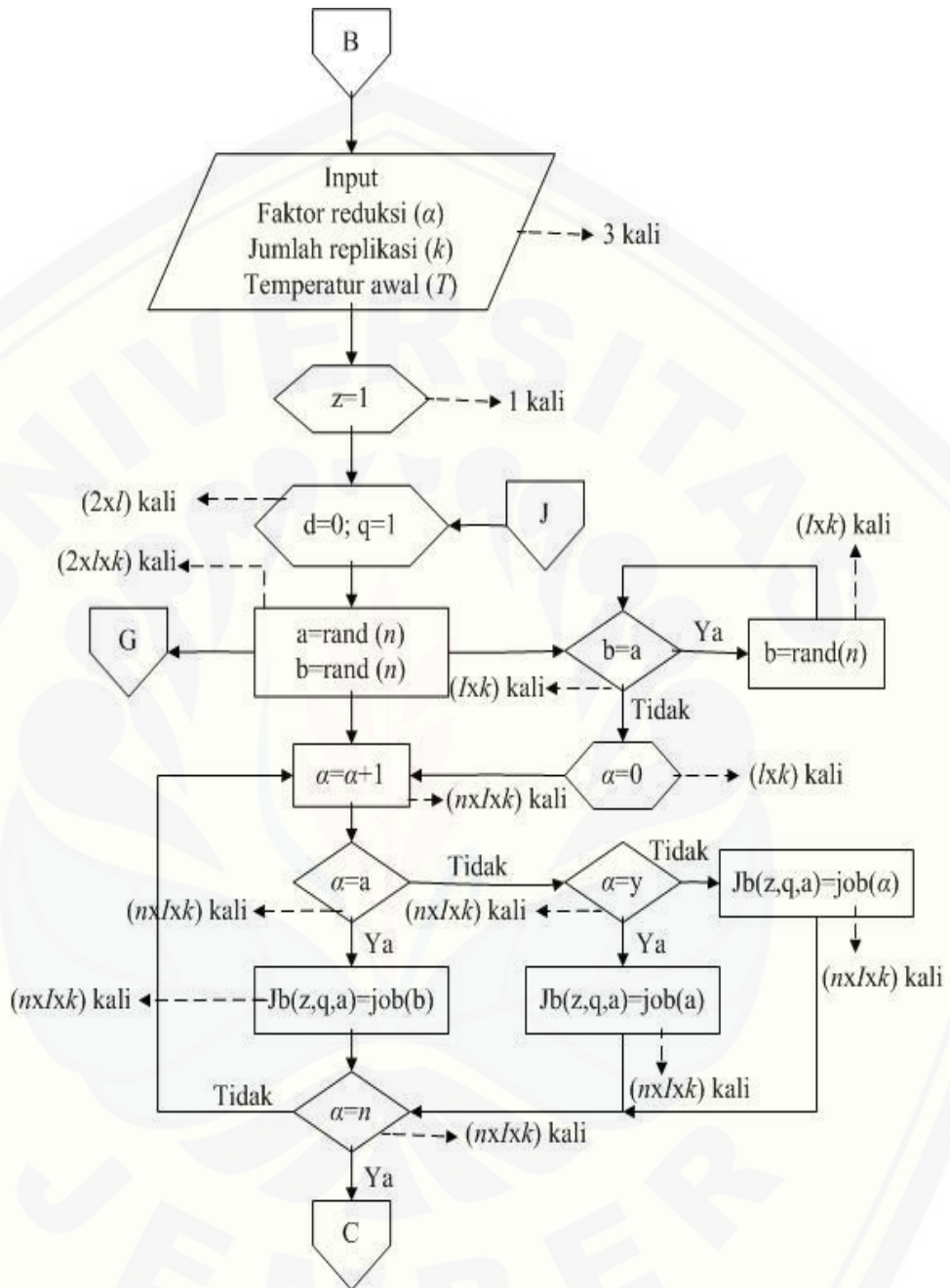
Gambar 4.10 menampilkan tahapan perhitungan untuk algoritma *Simulated Annealing*



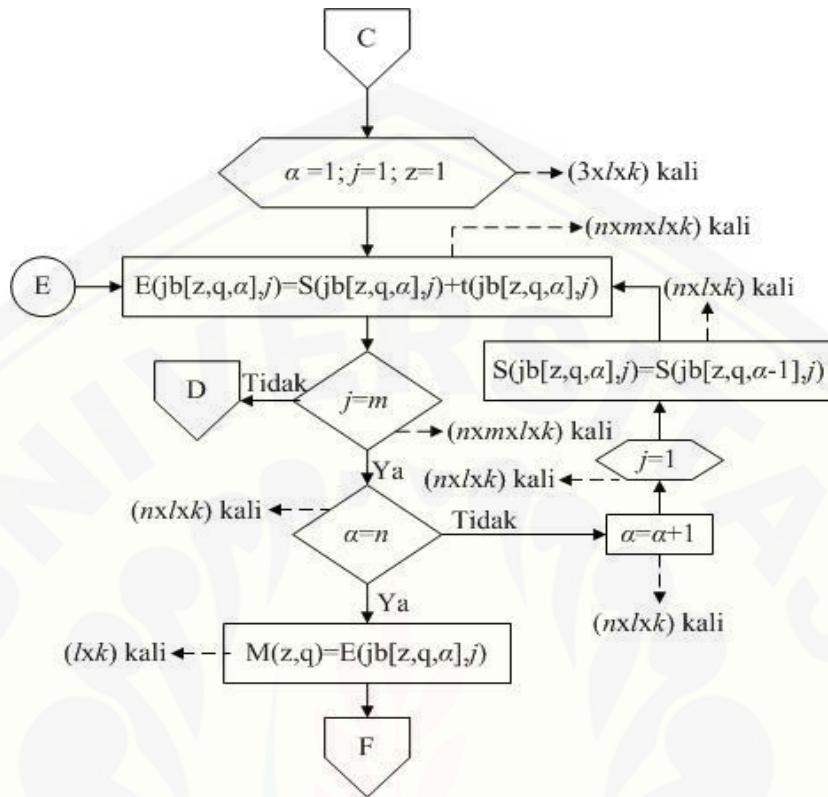
Jumlah perhitungannya adalah $T_{SA1} = 3 + 3n + 3nm$



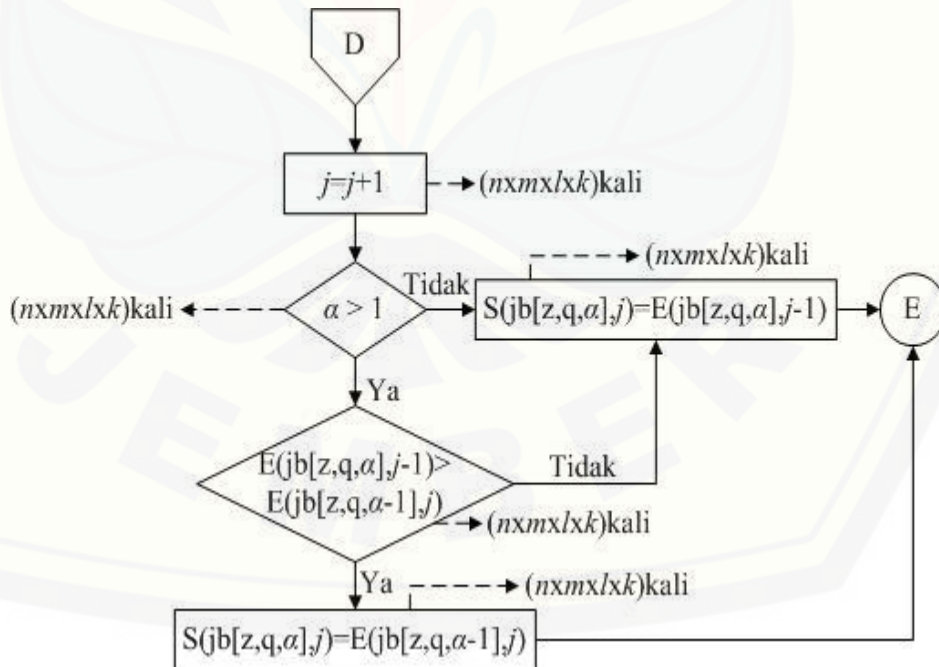
Jumlah perhitungannya adalah $T_{SA2} = 3 + 4n + 7nm$



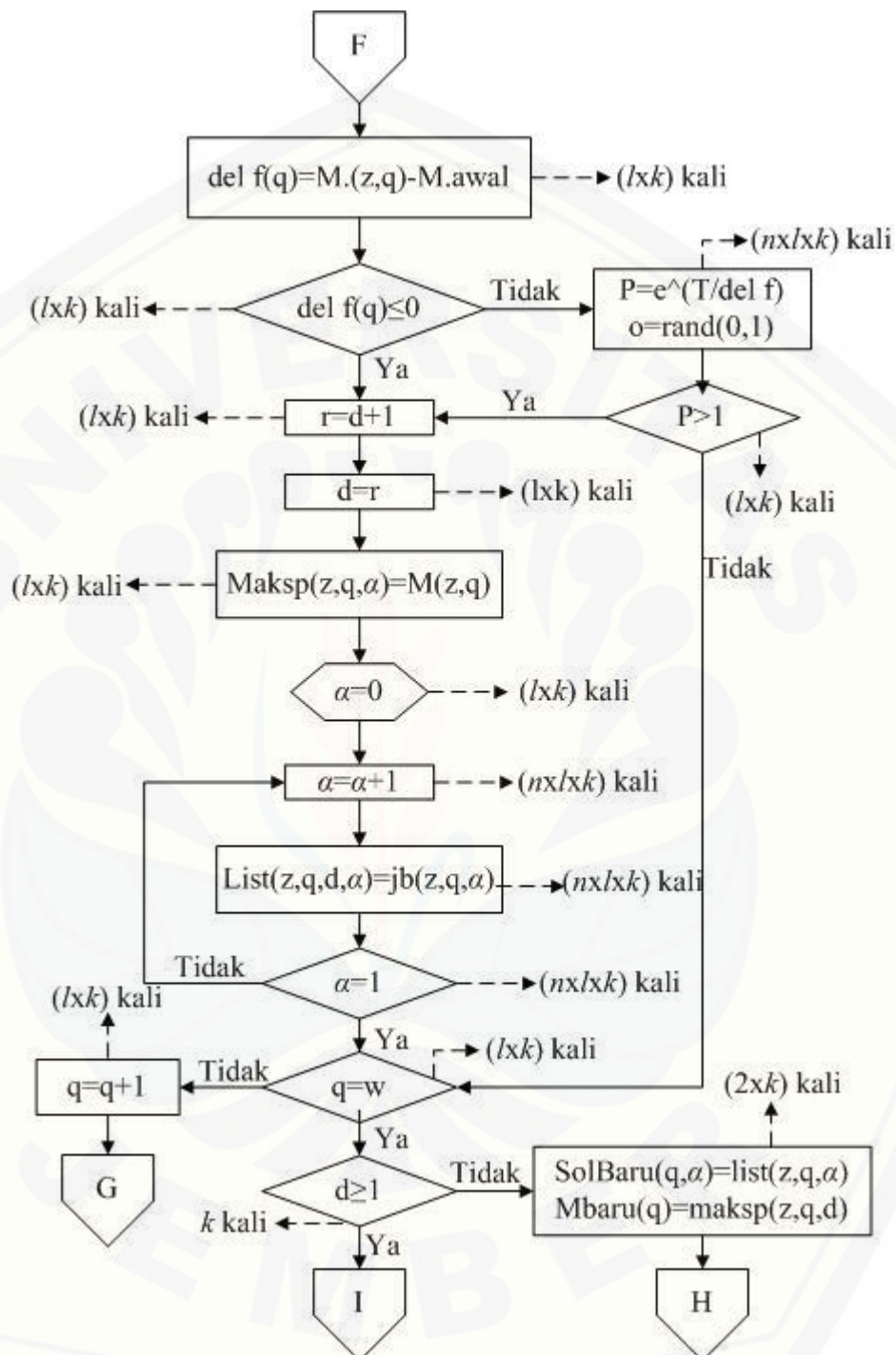
Jumlah perhitungannya adalah $T_{SA3} = 4 + 2l + 5lk + 7lkn$



Jumlah perhitungannya adalah $T_{SA4} = 4lk + 4lkn + 2lknm$



Jumlah perhitungannya adalah $T_{SA5} = 5lknm$



Gambar 4.10 flowchart algoritma Simulated Annealing

Jumlah perhitungannya adalah $T_{SA6} = 3k + 9kl + 4kln$

Berikut perhitungan kompleksitas waktu dari tahapan perhitungan untuk algoritma *Simulated Annealing*.

$$\begin{aligned}
 T(n, m, k, l) &= T_{SA1} + T_{SA2} + T_{SA3} + T_{SA4} + T_{SA5} + T_{SA6} \\
 &= (3 + 3n + 3nm) + (3 + 4n + 7nm) + (4 + 4l + 5kl + 7kln) \\
 &\quad + (4kl + 4kln + 2klnm) + 5klnm + (3k + 9lk + 4kln) \\
 &= (3 + 3 + 4) + (3 + 4)n + (3 + 7)nm + (4)l + (5 + 4 + 9)kl \\
 &\quad (7 + 4 + 4)kln + (2 + 5)klnm + (3)k \\
 &= 10 + 7n + 10nm + 4l + 18kl + 15kln + 7klnm + 3k
 \end{aligned}$$

Berikut perhitungan kompleksitas waktu dari tahapan perhitungan algoritma Genetika dengan satu kali iterasi (l) dan lima jumlah replikasi (k)

$$T(n, m) = 10 + 7n + 10nm + 4.1 + 18.5.1 + 15.5.1n + 7.5.1nm + 3.5$$

$$T(n, m) = 10 + 7n + 10nm + 4 + 90 + 65n + 35nm + 15$$

$$T(n, m) = 119 + 72n + 45nm$$

Karena

$$119 + 72n + 45nm \leq 119nm + 72nm + 45nm = 236nm$$

Maka kompleksitas algoritma *Simulated Annealing* adalah $O(nm)$

Apabila ditinjau dari perhitungan kompleksitas waktu yang dihasilkan, algoritma Genetika lebih bagus dari algoritma *Simulated Annealing*. Karena perbedaannya hanya terdapat pada jumlah koefisien saja, sedangkan ordernya sama maka perbedaan tersebut dapat diabaikan. Kedua algoritma memiliki kompleksitas waktu yang sama yaitu $O(nm)$. Artinya, proses perhitungan menggunakan algoritma Genetika membutuhkan waktu yang sama jika dibandingkan dengan proses perhitungan menggunakan algoritma *Simulated Annealing*. Dengan kata lain menurut kompleksitas waktu yang diperoleh dapat dikatakan algoritma Genetika dan algoritma *Simulated Annealing* mempunyai tingkat efisiensi yang sama.

4.3.2 Kecepatan konvergensi

Untuk mengetahui tingkat kecepatan konvergensi dari perubahan iterasi pada masing-masing algoritma dalam 14 kali percobaan dapat dilihat pada Gambar 4.11.

GA			
Urutan Jadwal	Waktu	Makespan	Iterasi Konvergen
J4 J9 J8 J3 J2 J10 J6 J5 J1 J7	0.14063	8878	3
J4 J3 J6 J2 J1 J5 J9 J10 J8 J7	0.10587	8878	6
J5 J1 J6 J3 J9 J10 J2 J4 J8 J7	0.10061	8878	2
J4 J2 J8 J9 J3 J5 J1 J6 J10 J7	0.12268	8878	4
J5 J8 J4 J6 J3 J9 J2 J10 J1 J7	0.10261	8878	7
J4 J6 J9 J5 J8 J3 J10 J1 J2 J7	0.11711	8878	3
J6 J5 J3 J2 J1 J4 J9 J10 J8 J7	0.098356	8878	10
J6 J3 J8 J4 J9 J10 J2 J1 J5 J7	0.10059	8878	6
J6 J10 J3 J1 J4 J9 J2 J8 J7 J5	0.098012	8878	9
J6 J3 J1 J5 J9 J4 J8 J10 J2 J7	0.1061	8878	4
J6 J4 J2 J3 J9 J10 J5 J1 J8 J7	0.099341	8878	3
J5 J10 J1 J4 J9 J6 J3 J8 J2 J7	0.093047	8878	5
J4 J8 J3 J2 J5 J9 J10 J6 J1 J7	0.10024	8878	2
J5 J6 J3 J8 J2 J9 J10 J1 J4 J7	0.10582	8878	5

SA			
Urutan Jadwal	Waktu	Makespan	Iterasi Konvergen
J6 J4 J1 J8 J9 J2 J3 J5 J10 J7	0.085565	8878	7
J7 J8 J9 J2 J6 J3 J10 J4 J1 J5	0.067898	8878	1
J4 J9 J6 J8 J1 J10 J2 J3 J7 J5	0.068224	8878	9
J4 J2 J10 J6 J1 J8 J7 J9 J3 J5	0.067513	8878	8
J5 J1 J10 J4 J8 J3 J9 J6 J2 J7	0.065897	8878	2
J6 J3 J2 J5 J9 J4 J8 J1 J10 J7	0.067001	8878	1
J5 J9 J10 J8 J4 J3 J1 J6 J2 J7	0.066756	8878	16
J5 J9 J3 J1 J8 J2 J6 J10 J4 J7	0.066548	8878	16
J5 J9 J4 J2 J3 J10 J6 J8 J1 J7	0.066602	8878	11
J6 J5 J3 J1 J4 J8 J9 J10 J2 J7	0.066992	8878	41
J7 J4 J3 J6 J8 J10 J2 J9 J1 J5	0.066034	8878	9
J4 J6 J9 J3 J10 J8 J2 J1 J5 J7	0.066359	8878	7
J5 J4 J1 J10 J2 J6 J9 J3 J8 J7	0.066363	8878	1
J6 J3 J9 J2 J8 J1 J10 J5 J4 J7	0.06645	8878	2

Gambar 4.11 Tingkat kecepatan konvergensi algoritma

Dari Gambar 4.11 terlihat bahwa algoritma Genetika tingkat konvergensinya relatif stabil pada setiap 14 kali percobaan sedangkan pada algoritma *Simulated Annealing* tingkat konvergensinya tidak konsisten pada setiap 14 kali percobaan. Namun kestabilan tingkat konvergensi pada suatu algoritma tidak bisa menjamin bahwa algoritma tersebut adalah algoritma terbaik.

BAB 5. PENUTUP

5.1 Kesimpulan

Dari penelitian yang telah dilakukan, dapat disimpulkan beberapa hal berikut.

- a. Penjadwalan produksi kerupuk UD.Samjaya berdasarkan algoritma Genetika dengan mengambil iterasi 100 dari empat belas kali percobaan menghasilkan *makespan* terbaik sebesar 8878 menit dengan urutan jadwal jenis kerupuk mawar 2 udang kecil – kerupuk mawar 2 udang besar – kerupuk impala panjang – kerupuk mawar SI besar – kerupuk obar abir – kerupuk kasandra besar – kerupuk Rajang – kerupuk impala pendek – kerupuk kasandra kecil – kerupuk mawar SI kecil. Sedangkan perhitungan algoritma *Simulated Annealing* untuk penjadwalan produksi kerupuk UD.Samjaya dengan iterasi 100 dari empat belas kali percobaan menghasilkan *makespan* terbaik 8878 menit dengan urutan jadwal jenis kerupuk mawar 2 udang kecil – kerupuk kasandra kecil – kerupuk impala pendek – kerupuk Rajang – kerupuk mawar 2 udang besar – kerupuk impala panjang – kerupuk mawar SI besar – kerupuk obar abir – kerupuk kasandra besar – kerupuk mawar SI kecil;
- b. Berdasarkan nilai *makespan* yang diperoleh, algoritma Genetika dan algoritma *Simulated Annealing* mempunyai tingkat efektifitas yang sama untuk diterapkan pada penjadwalan produksi kerupuk UD. Samjaya;
- c. Dari empat belas kali percobaan dalam 100 iterasi dapat disimpulkan bahwa algoritma Genetika lebih cepat konvergen daripada algoritma *Simulated Annealing*;

- d. Berdasarkan kompleksitas waktu yang diperoleh dapat dikatakan algoritma Genetika dan algoritma *Simulated Annealing* mempunyai tingkat efisiensi yang sama karena membutuhkan waktu komputasi sebesar $O(nm)$.

5.2 Saran

Permasalahan optimasi penjadwalan *flowshop* menggunakan algoritma Genetika dan *Simulated Annealing* ini masih bisa dikembangkan, misalnya membandingkan proses pencarian solusinya dengan algoritma optimasi lainnya. Selain itu, dapat juga dikembangkan dengan mencari solusi baru dengan metode-metode yang lain.

DAFTAR PUSTAKA

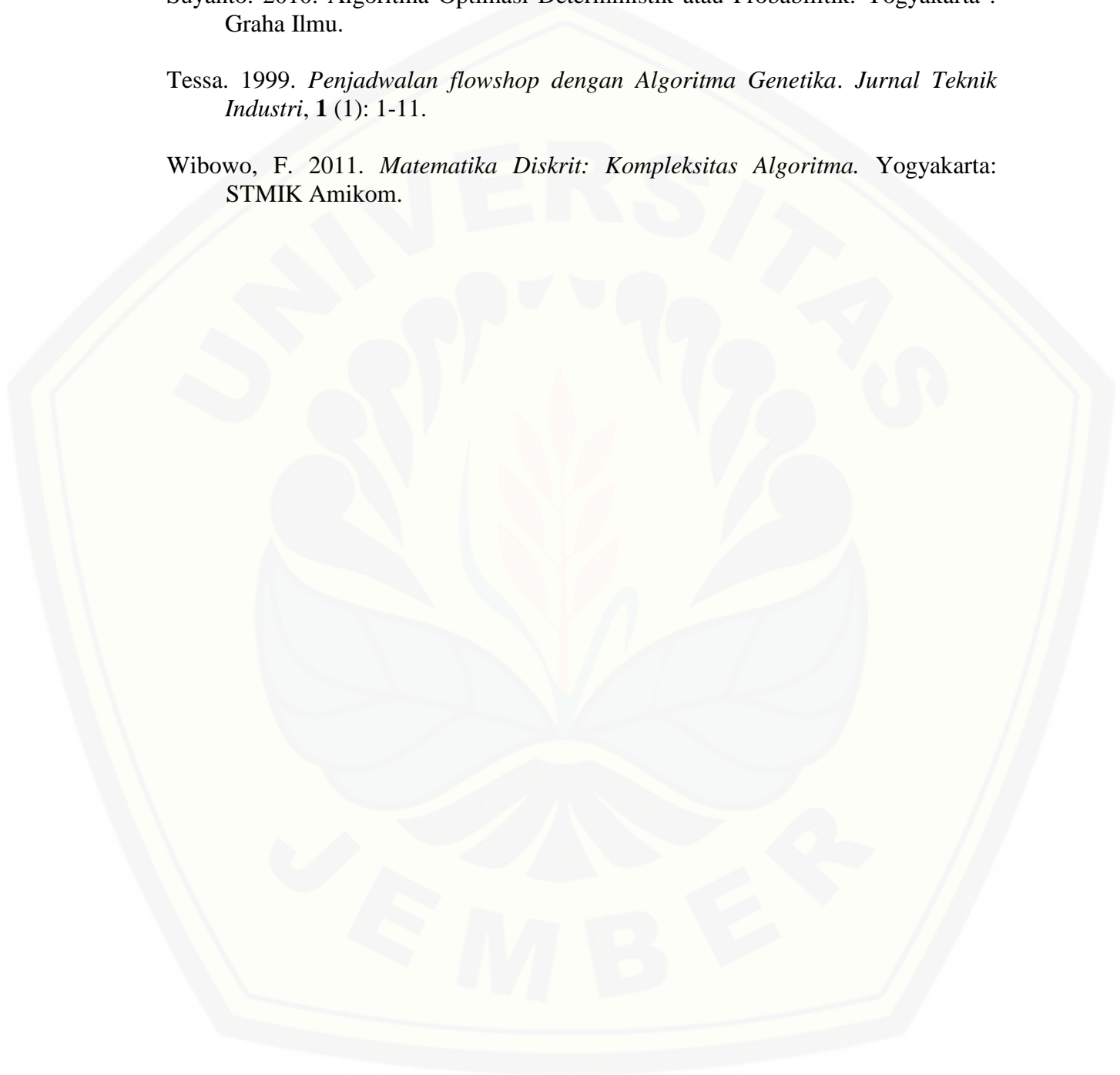
- Andriyanti, N. 2014. *Penerapan Algoritma Genetika dan Tabu Search untuk Penjadwalan Produksi karung plastik di PT. Forindo Prima Perkasa*. Tidak dipublikasikan. Skripsi. Jember : Jurusan Matematika Fakultas MIPA Universitas Jember.
- Ayuningtiyas, D. S. 2008. *Aplikasi Algoritma Genetika pada Masalah Penugasan*. Tidak dipublikasikan. Skripsi. Jember : Jurusan Matematika Fakultas MIPA Universitas Jember.
- Emanuel, A. W. dan Aritonang, A. F. 2008. *Aplikasi Desktop Pencarian Rute Jalan dengan Algoritma Simulated Annealling*. *Jurnal Informatika*. **4(2)**: 93-103.
- Ginting, R. 2009. *Penjadwalan Mesin*. Yogyakarta : Graha Ilmu.
- Hapsari, V. P. 2013. *Penerapan Algoritma Genetika pada Optimasi Keuntungan Pengiriman Barang Pecah Belah dalam Sebuah Kontainer*. Tidak dipublikasikan. Skripsi. Jember : Jurusan Matematika Fakultas MIPA Universitas Jember.
- Hartanto, E. 2007. *Usulan Perbaikan Sistem Penjadwalan Produksi n Job m Machine pada Perusahaan PT Polinayaguna Perkasa*. Skripsi. Jakarta : Jurusan Teknik Industri Universitas Bina Nusantara.
- Hasad, A. 2011. *Algoritma Optimasi dan Aplikasinya*. <http://andihasad.Files.wordpress.com/2011/11/algoritma-optimasi-dan-aplikasinya.pdf> [5 Desember 2012]
- Mariani. 2009. *Perbandingan Metode Hamid, Simulated Annealing dan Metode Heuristik FRB pada Penjadwalan Flowshop dengan Kriteria Makespan*. Surabaya : Universitas Kristen Petra.
- Nugraha, D, W. 2012. *Penerapan Kompleksitas Waktu Algoritma Prim Untuk Menghitung Kemampuan Komputer Dalam Melaksanakan Perintah*. *Jurnal Ilmiah Foristek* . **2 (2)**: 195-207.
- Santosa, B. dan Willy P. 2011. *Metode Metaheuristik Konsep dan Implementasi*. Surabaya : Guna Widya.

Suyanto. 2005. *Algoritma Genetika dalam MATLAB*. Yogyakarta : Andi.

Suyanto. 2010. *Algoritma Optimasi Deterministik atau Probabilitik*. Yogyakarta : Graha Ilmu.

Tessa. 1999. *Penjadwalan flowshop dengan Algoritma Genetika*. *Jurnal Teknik Industri*, **1** (1): 1-11.

Wibowo, F. 2011. *Matematika Diskrit: Kompleksitas Algoritma*. Yogyakarta: STMIK Amikom.



Lampiran A. Gambar proses produksi UD. Samjaya



1. Persiapan bahan



2. Pengadonan



3. Pelembutan dengan mesin mulen



4. Pencetakan



5. Pengukusan



6. Pendinginan



7. Pemotongan mesin dan pemotongan manual



8. Penjemuran matahari dan penjemuran kipas



9. Pengemasan

Lampiran B. Tampilan *insert* data dengan jumlah pesanan yang ditambahkan

a. Tampilan *insert* data penelitian yang telah tersimpan dalam bentuk txt.

File: D:\file2 lecture\lecture\TA\wheny\program\data waktu

Jumlah Pesanan

Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job
1	5	1	2	1	1	

	Mesin 1	Mesin 2	Mesin 3	Mesin 4	Mesin 5	Me:
Job 1	15	160	15	30	30	
Job 2	15	160	15	30	30	
Job 3	15	160	15	30	30	
Job 4	15	60	30	10	3	
Job 5	15	60	30	10	3	

Insert Close

b. Tampilan data inputan dari data lain (manual)

File:

Jumlah Pesanan

Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job
1	5	1	2	1	1	

	Mesin 1	Mesin 2	Mesin 3	Mesin 4	Mesin 5	Me:
Job 1	0	0	0	0	0	
Job 2	0	0	0	0	0	
Job 3	0	0	0	0	0	
Job 4	0	0	0	0	0	
Job 5	0	0	0	0	0	

Insert Close