



UNIVERSITAS INDONESIA

**OPTIMASI DI JALUR *LOW PRESSURE DIE CASTING*
PERUSAHAAN OTOMOTIF DENGAN METODE
*ALGORITMA DIFFERENTIAL EVOLUTION***

SKRIPSI

**HENDRIK KURNIAWAN SAPUTRA
0706201084**

**FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INDUSTRI
DEPOK
DESEMBER 2009**



UNIVERSITAS INDONESIA

**OPTIMASI DI JALUR *LOW PRESSURE DIE CASTING*
PERUSAHAAN OTOMOTIF DENGAN METODE
*ALGORITMA DIFFERENTIAL EVOLUTION***

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana teknik

**HENDRIK KURNIAWAN SAPUTRA
0706201084**

**FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INDUSTRI
DEPOK
DESEMBER 2009**

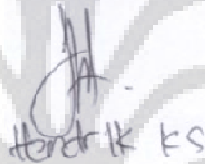
HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar**

Nama : HENDRIK KURNIAWAN SAPUTRA

Npm : 0706201084

Tanda Tangan :



Hendrik KS

Tanggal : 31 Desember 2009

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Hendrik Kurniawan Saputra
NPM : 0706201084
Program Studi : Teknik Industri
Judul Skripsi : Optimasi di Jalur *Low Pressure Die Casting*
Perusahaan Otomotif dengan Metode Algoritma
Differential Evolution

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana pada Program Studi Teknik Industri, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Ir. Amar Rachman, MEIM
Penguji : Arian Dhini, ST, MT
Penguji : Dr. Ir. T. Yuri M. Zagloel, MEngSc
Penguji : Ir. M. Dachyar, MSc



Ditetapkan di : Jakarta

Tanggal : 31 Desember 2009

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepada Allah SWT karena atas rahmat, dan ridho-Nya akhirnya penyusunan skripsi ini dapat diselesaikan. Penulis menyadari bahwa skripsi ini tidak akan dapat dibuat tanpa bantuan dan bimbingan dari berbagai pihak. Karena itu, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

- Ir. Amar Rachman, MEIM, selaku dosen pembimbing yang telah banyak memberi bantuan, masukan dan bimbingan yang berharga bagi penulis.
- Bapak Jojok Purnomo di PT AHM yang telah memberikan kesempatan kepada penulis untuk mengumpulkan data untuk penelitian ini dan memberikan jawaban atas pertanyaan-pertanyaan yang sering penulis tanyakan.
- Bapak Agam Suwala dan Ibu Marcella Iskandar di PT AHM yang telah memberikan perizinan pada saat pelaksanaan skripsi ini.
- Keluarga, atas curahan kasih sayang , dukungan, dan doa yang diberikan.
- Teman-teman penulis, khususnya rekan-rekan TIUI 2007 yang telah memberikan dukungan, semangat, serta kebersamaan selama ini.
- Pihak-pihak lain yang juga telah membantu penyelesaian skripsi ini namun tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan. Penulis berharap skripsi ini dapat memberikan manfaat bagi semua pihak yang membacanya.

Jakarta, 31 Desember 2009

Penulis

**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI
KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Hendrik Kurniawan Saputra
NPM : 0706201084
Departemen : Teknik Industri
Fakultas : Teknik
Jenis karya : Skripsi


demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Optimasi di Jalur *Low Pressure Die Casting* Perusahaan Otomotif dengan Metode Algoritma *Differential Evolution*

berserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Jakarta
Pada tanggal : 31 Desember 2009
Yang menyatakan


(Hendrik K S)

ABSTRAK

Nama : Hendrik Kurniawan Saputra
Program Studi : Teknik Industri
Judul : Optimasi di Jalur *Low Pressure Die Casting* Perusahaan Otomotif dengan Metode Algoritma *Differential Evolution*

Penelitian ini membahas masalah penjadwalan *job shop* pada suatu perusahaan otomotif. Pada sistem ini akan dihasilkan sejumlah produk dalam beberapa jenis dengan rute yang dapat berbeda satu sama lain. Penjadwalan produksi merupakan suatu permasalahan yang kompleks sehingga dibutuhkan metode yang tepat untuk mendapatkan solusi yang optimal untuk masalah ini. Metode penelitian yang digunakan adalah salah satu dari metode meta-heuristik, yaitu algoritma *differential evolution* (DE). Prinsip algoritma DE sesuai dengan analogi evolusi biologi, yaitu terdiri dari proses inialisasi populasi, proses mutasi, proses pindah silang, dan proses seleksi. Algoritma ini memiliki beberapa keunggulan, yaitu konsepnya sederhana, mudah diaplikasikan, cepat dalam menghasilkan solusi, dan tangguh. Fungsi tujuan dari permasalahan ini adalah meminimumkan *makespan*.

Penjadwalan yang diperoleh melalui algoritma *differential evolution* menghasilkan *makespan* sebesar 1.207.624,4 detik, sedangkan jadwal perusahaan menghasilkan 1.253.272,8 detik. Jadi, usulan jadwal menghasilkan penurunan *makespan* sebesar 3,64% dibandingkan jadwal perusahaan.

Kata kunci :
Penjadwalan, *Job Shop*, *Makespan*, Algoritma *Differential Evolution*

ABSTRACT

Name : Hendrik Kurniawan Saputra
Study Program : Industrial Engineering
Title : Optimizing at Low Pressure Die Casting Line in The Automotive Company using Differential Evolution Algorithm

This research presents job shop scheduling at a automotive company. This system yields large amount of different products with some different manufacture processes. Production scheduling is a complex problem so that appropriated method to produces the optimal solution of it is needed. Method of this research is one of meta-heuristic algorithms, differential evolution (DE) algorithm. The principle of DE algorithm is based on analogy of biological evolution that consists of population initiation process, mutation process, crossover process, and selection process. This algorithm has some strengths because of its simply structure, ease to use, speed, and robustness. The objective function in this problem is to minimize makespan.

The schedule that is obtained from differential evolution algorithm produces makespan of 1.207.624,4 seconds, meanwhile the schedule of company produces 1.253.272,8 seconds. Thus, new schedule produces reduction of makespan about 3.64% compared with schedule of company.

Key words :
Scheduling, Job Shop, Makespan, Differential Evolution Algorithm

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PERNYATAAN ORISINALITAS	ii
HALAMAN PERSETUJUAN	iii
KATA PENGANTAR.....	iv
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI	v
ABSTRAK	vi
ABSTRACT	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR.....	x
DAFTAR TABEL	xi
DAFTAR LAMPIRAN.....	xii
1. PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Diagram Keterkaitan Masalah	2
1.3 Perumusan Masalah	3
1.4 Tujuan Penelitian	3
1.5 Batasan Masalah	3
1.6 Metodologi Penelitian	4
1.7 Sistematika Penulisan	6
2. DASAR TEORI.....	7
2.1 Penjadwalan Produksi	7
2.1.1 Pengertian Penjadwalan Produksi.....	7
2.1.2 Tujuan Penjadwalan Produksi.....	7
2.1.3 Klasifikasi Penjadwalan Produksi.....	8
2.1.4 Istilah dalam Penjadwalan Produksi	8
2.1.5 Karakteristik dan Kendala Proses	9
2.1.6 Fungsi Tujuan dan Pengukuran Performa Penjadwalan Produksi	10
2.2 Penjadwalan <i>Job Shop</i>	10
2.3 Metode Penyelesaian Masalah Penjadwalan Produksi	11
2.3.1 Tipe Heuristik Klasik.....	11
2.3.2 Tipe Heuristik Modern (Meta-Heuristik).....	12
2.4 Algoritma <i>Differential Evolution</i> (DE).....	14
2.4.1 Sejarah Algoritma DE.....	14
2.4.2 Konsep Dasar	16
2.4.3 Tahapan Pengerjaan	17
2.4.3.1 Inisialisasi	19
2.4.3.2 Evaluasi Populasi Awal	20
2.4.3.3 Mutasi	20
2.4.3.4 Pindah Silang	20
2.4.3.5 Evaluasi Vektor <i>Trial</i>	21
2.4.3.6 Seleksi	21
2.4.3.7 Terminasi	22
2.4.4 Penerapan Algoritma DE pada Masalah Penjadwalan <i>Job Shop</i>	22
2.4.4.1 Elemen Dasar Algoritma DE	23
2.4.4.2 Prosedur Operasi Pencarian	24

2.5 <i>Design of Experiments</i> (DOE)	27
2.5.1 Tujuan <i>Design of Experiments</i>	27
2.5.2 Tipe Percobaan.....	27
2.5.3 Prinsip Dasar	29
3. PENGUMPULAN DATA.....	30
3.1 Profil Perusahaan	30
3.2 Pengumpulan Data Penelitian.....	30
3.2.1 Data Jam Kerja di PT AHM.....	30
3.2.2 Data Pesanan Part	31
3.2.3 Data Rute dan Waktu Proses.....	32
3.2.4 Jadwal Produksi di Jalur <i>LPDC</i> PT AHM Periode Oktober 2009.....	34
4. PENGOLAHAN DATA DAN ANALISIS	38
4.1 Penyusunan Algoritma.....	38
4.1.1 Langkah-Langkah Penyusunan Algoritma	38
4.1.2 Verifikasi dan Validasi Program.....	41
4.1.2.1 Hasil <i>Run</i> Program	43
4.1.2.2 Hasil Perhitungan Manual.....	43
4.2 Input	54
4.2.1 Input Data.....	54
4.2.2 Input Parameter.....	54
4.3 Pengolahan Data dan Hasil	56
4.3.1 Hasil Penjadwalan Jalur <i>LPDC</i> PT AHM.....	56
4.3.2 Hasil Penjadwalan dengan Algoritma DE	56
4.4 Analisis	61
4.4.1 Analisis Skenario Parameter.....	61
4.4.2 Analisis Waktu Komputasi (Waktu <i>Run</i> Program).....	67
4.4.3 Analisis Hasil	67
5. KESIMPULAN	69
DAFTAR REFERENSI.....	70

DAFTAR GAMBAR

Gambar 1.1 Diagram Keterkaitan Permasalahan.....	3
Gambar 1.2 Diagram Alir Metodologi Penelitian.....	5
Gambar 2.1 Contoh Rute Penjadwalan <i>Job Shop</i>	11
Gambar 2.2 Diagram Alir Pengerjaan Algoritma DE.....	17
Gambar 2.3 Representasi Proses <i>Differential Evolution</i>	18
Gambar 2.4 Proses Pindah Silang.....	21
Gambar 2.5 Diagram Alir Algoritma DE untuk <i>Job Shop Sequencing Problem</i>	26
Gambar 3.1 Rute Proses Produksi di Jalur <i>LPDC</i>	33
Gambar 4.1 Rute Operasi Data <i>Dummy</i>	42
Gambar 4.2 Gant Chart Individu 1 Populasi Target	44
Gambar 4.3 Gant Chart Individu 2 Populasi Target	45
Gambar 4.4 Gant Chart Individu 3 Populasi Target	45
Gambar 4.5 Gant Chart Individu 4 Populasi Target	46
Gambar 4.6 Gant Chart Individu 5 Populasi Target	46
Gambar 4.7 Gant Chart Individu 1 Populasi <i>Trial</i>	50
Gambar 4.8 Gant Chart Individu 2 Populasi <i>Trial</i>	50
Gambar 4.9 Gant Chart Individu 3 Populasi <i>Trial</i>	51
Gambar 4.10 Gant Chart Individu 4 Populasi <i>Trial</i>	51
Gambar 4.11 Gant Chart Individu 5 Populasi <i>Trial</i>	52
Gambar 4.12 Tampilan Hasil <i>Run</i> 1.....	58
Gambar 4.13 Tampilan Hasil <i>Run</i> 2.....	59
Gambar 4.14 Tampilan Hasil <i>Run</i> 3.....	60
Gambar 4.15 Interaksi antara Dua Faktor.....	63
Gambar 4.16 Grafik Faktor Utama untuk <i>Makespan</i>	63

DAFTAR TABEL

Tabel 2.1	Proses Pindah Silang.....	21
Tabel 2.2	Proses Seleksi	22
Tabel 3.1	Waktu Kerja PT AHM.....	31
Tabel 3.2	Data Pesanan Periode Oktober 2009.....	32
Tabel 3.3	Jumlah dan Alokasi Mesin setiap Rute Proses	32
Tabel 3.4	Waktu Proses Produksi	33
Tabel 3.5	Penjadwalan di Jalur <i>LPDC</i> Lengkap (Periode Oktober 2009).....	34
Tabel 4.1	Data <i>dummy</i> untuk Validasi	42
Tabel 4.2	Parameter yang Digunakan dalam Validasi.....	42
Tabel 4.3	Populasi Target (Hasil <i>Run</i> Program).....	43
Tabel 4.4	Permutasi Populasi Target	44
Tabel 4.5	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 1 Populasi Target	44
Tabel 4.6	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 2 Populasi Target	45
Tabel 4.7	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 3 Populasi Target	45
Tabel 4.8	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 4 Populasi Target	46
Tabel 4.9	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 5 Populasi Target	46
Tabel 4.10	Vektor Target.....	47
Tabel 4.11	Vektor Acak 1 (Hasil <i>Run</i> Program).....	47
Tabel 4.12	Vektor Acak 2 (Hasil <i>Run</i> Program).....	48
Tabel 4.13	Populasi Mutan	48
Tabel 4.14	Populasi <i>Trial</i>	49
Tabel 4.15	Permutasi Populasi <i>Trial</i>	49
Tabel 4.16	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 1 Populasi <i>Trial</i>	50
Tabel 4.17	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 2 Populasi <i>Trial</i>	50
Tabel 4.18	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 3 Populasi <i>Trial</i>	51
Tabel 4.19	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 4 Populasi <i>Trial</i>	51
Tabel 4.20	Perhitungan Waktu Produksi setiap <i>Job</i> Individu 5 Populasi <i>Trial</i>	52
Tabel 4.21	Perbandingan antara <i>Makespan</i> Populasi Target dan Populasi <i>Trial</i>	52
Tabel 4.22	Populasi Target Baru	53
Tabel 4.23	Permutasi Populasi Target Baru.....	53
Tabel 4.24	Hasil Terbaik Perhitungan Data <i>Dummy</i>	54
Tabel 4.25	Skenario untuk DOE.....	55
Tabel 4.26	Kombinasi Parameter Terbaik	56
Tabel 4.27	Hasil Perhitungan Jadwal PT AHM.....	56
Tabel 4.28	Hasil <i>Run</i> 1.....	57
Tabel 4.29	Hasil <i>Run</i> 2.....	57
Tabel 4.30	Hasil <i>Run</i> 3.....	57
Tabel 4.31	ANOVA untuk <i>Makespan</i>	61

DAFTAR LAMPIRAN

Lampiran 1 *Script M-File* Program untuk Perhitungan Jadwal PT AHM

Lampiran 2 *Script M-File Program* untuk Pencarian Solusi Jadwal

Lampiran 3 Hasil *Design of Experiments*



BAB 1

PENDAHULUAN

1.1 Latar Belakang

Dengan semakin berkembangnya dunia industri di bidang manufaktur, maka setiap perusahaan saling bersaing untuk cepat dalam merespon permintaan konsumen agar mampu meningkatkan *market share*. Dengan demikian, penjadwalan yang efektif dan efisien berperan penting dalam *modern competitive* dan *marketplace*. Pada bidang manufaktur, penjadwalan merupakan suatu permasalahan dimana adanya beberapa tujuan dan sumber daya yang harus dialokasikan untuk mendapatkan tujuan perusahaan, seperti memaksimalkan utilisasi dari pekerja dan mesin, meminimumkan waktu yang dibutuhkan untuk menyelesaikan keseluruhan proses yang telah dijadwalkan (*makespan*).

PT Astra Honda Motor (PT AHM) merupakan industri manufaktur yang bergerak dibidang otomotif yang memproduksi beberapa jenis sepeda motor. Proses produksi sepeda motor yang dilakukan terdiri dari beberapa tahapan proses produksi seperti : *casting line*, *machining line*, *assembly engine line*, *assembly unit line* dan *shipping*. Untuk jalur *casting* dan *machining*, proses produksi yang dilakukan selain untuk menghasilkan unit sepeda motor, tetapi juga di alokasikan untuk memproduksi *spare part* sepeda motor dalam memenuhi pelayanan *after market* berdasarkan bagian pemasaran (*rem part*).

Salah satu proses produksi sepeda motor adalah proses *casting* yang memproduksi *left crank case*, *right crank case*, *cylinder comp*, *cylinder head* dan *piston*. Proses *casting* terbagi menjadi dua tempat produksi, yaitu: jalur *die casting* dan jalur *low pressure die casting (lpdc)*. Pada jalur *lpdc*, produk yang dihasilkan berupa *cylinder head* dan *piston* sepeda motor yang diproses dengan menggunakan beberapa buah mesin. Masing-masing produk tersebut umumnya melalui beberapa tahapan proses. Beberapa tahapan proses tersebut dikerjakan pada mesin yang berbeda. Dengan demikian, maka dibutuhkan suatu penjadwalan produksi yang optimal agar kombinasi proses produksi dari setiap produk tersebut berjalan efektif dan efisien, sehingga target produksi yang diharapkan dapat tercapai.

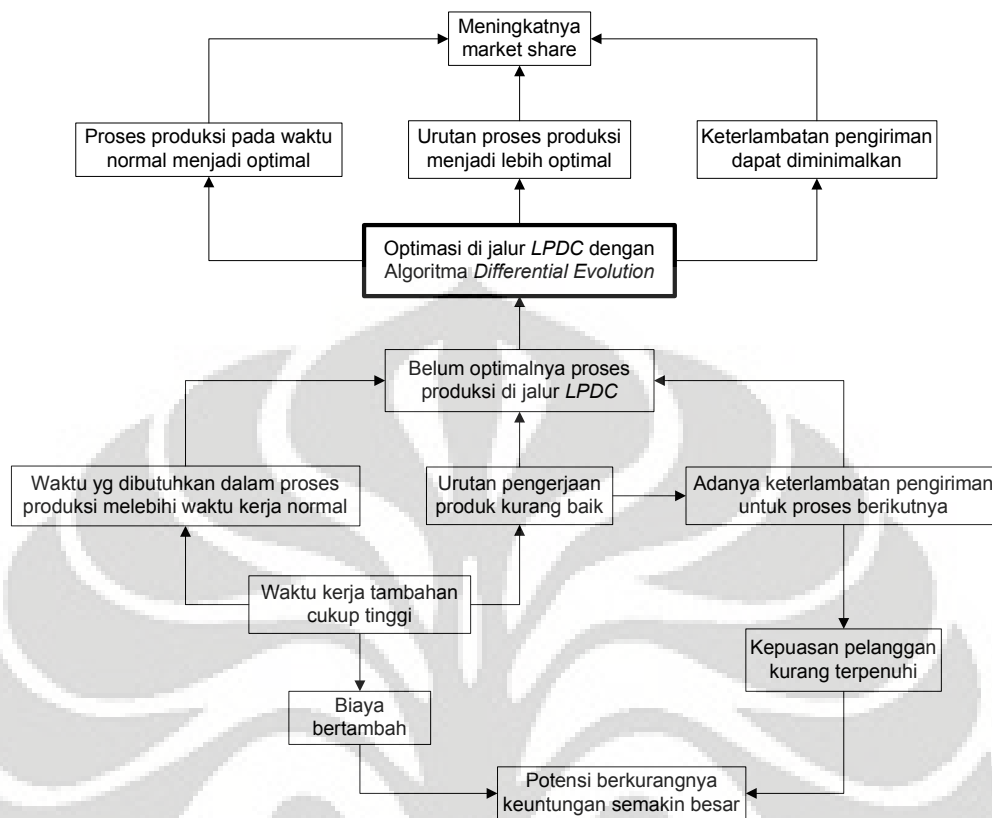
Permasalahan penjadwalan merupakan suatu permasalahan *combinatorial optimization* kompleks yang dikategorikan sebagai permasalahan *nondeterministic polynomial hard (NP-hard)*, yaitu permasalahan dimana waktu penyelesaian pencarian solusinya akan meningkat secara eksponensial seiring dengan semakin besarnya ukuran permasalahan. Hingga kini, algoritma yang bersifat eksak (contohnya *branch and bound* dan *linear programming*) belum tentu dapat menghasilkan solusi yang mendekati optimal dan juga memiliki waktu perhitungan yang lama. Begitu juga dengan metode heuristik klasik seperti *priority dispatch rule*, walaupun membutuhkan waktu perhitungan yang lebih singkat, tetapi solusi yang didapat belum tentu mencapai optimal. Oleh karena itu, solusi praktis yaitu dengan menggunakan pengembangan heuristik klasik yang disebut meta-heuristik untuk mendapatkan solusi yang lebih mendekati optimal.

Pada penelitian ini akan dicari solusi untuk menyelesaikan permasalahan penjadwalan produksi dengan menggunakan metode Algoritma *Differential Evolution*, yang merupakan metode *meta-heuristik* terbaru. Metode ini merupakan versi pengembangan dari Algoritma Genetika. Prinsipnya adalah berdasarkan analogi evolusi biologi, yang terdiri dari proses penginisialisasian populasi, proses mutasi, proses penyilangan, dan proses penyeleksian. Dari penelitian yang telah dilakukan, metode ini lebih cepat dalam perhitungan dan solusi yang didapatkan lebih mendekati optimal dibandingkan metode optimasi global lainnya¹. Metode baru yang hanya memerlukan sedikit variabel kontrol ini tangguh, mudah digunakan, dan sangat cocok untuk perhitungan paralel.

1.2 Diagram Keterkaitan Masalah

Untuk melihat gambaran sistematis yang lebih menyeluruh, maka disusun suatu diagram keterkaitan masalah seperti pada gambar 1.1

¹ Rainer Storn and Kenneth Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", in *Journal of Global Optimization* 11, Kluwer Academic Publishers, Netherlands, 1997, p.341.



Gambar 1.1 Diagram Keterkaitan Masalah

1.3 Perumusan Masalah

Pokok permasalahan yang akan dibahas pada penelitian ini adalah mengenai perlunya perancangan suatu sistem penjadwalan produksi yang efisien di PT AHM, khususnya pada jalur *Low Pressure Die Casting*.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah memperoleh suatu sistem penjadwalan produksi (*job shop*) yang lebih baik (lebih mendekati optimal) dengan meminimumkan *makespan* melalui penerapan *Algoritma Differential Evolution*.

1.5 Batasan Masalah

Untuk mendapatkan hasil penelitian yang spesifik dan terarah, maka ruang lingkup permasalahan dari penelitian ini adalah sebagai berikut :

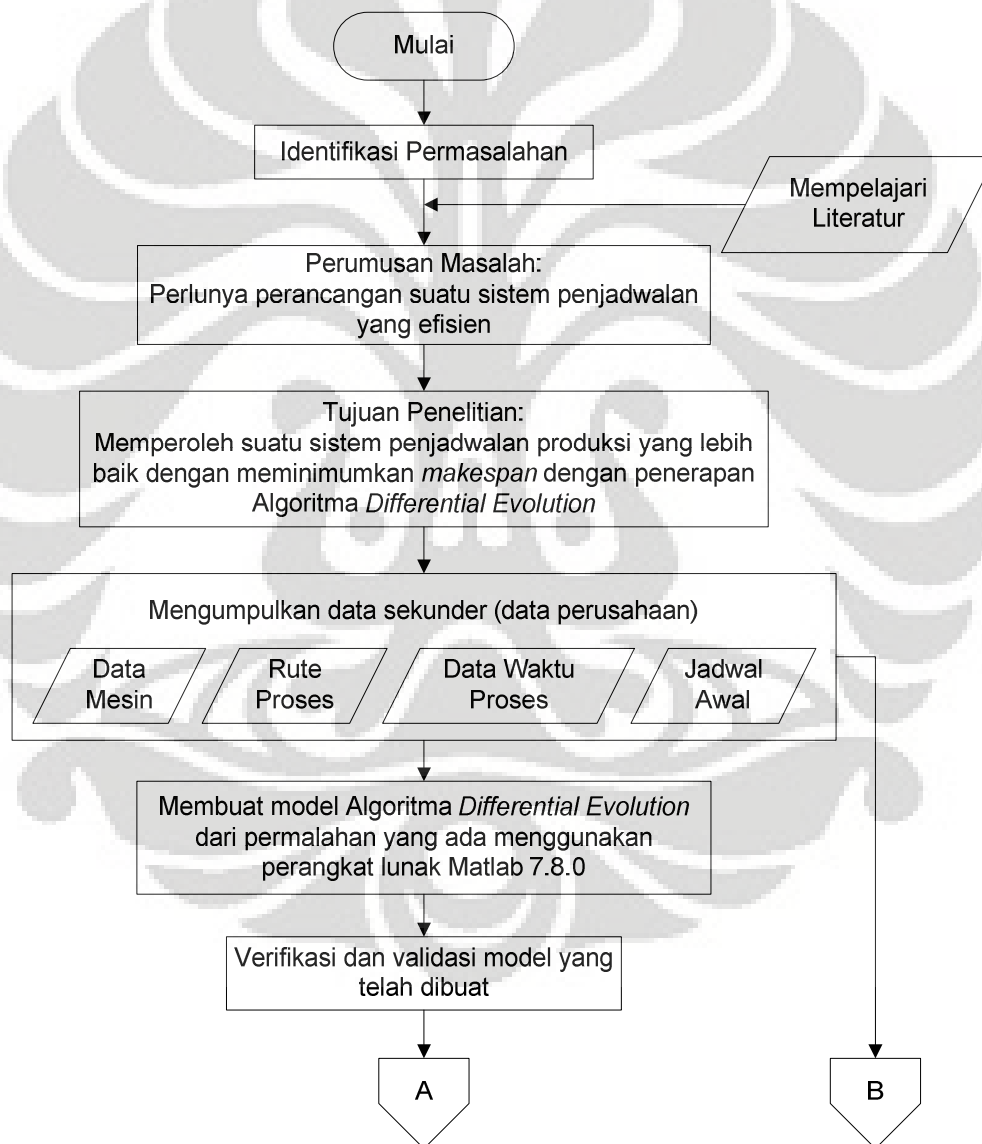
1. Penelitian ini menggunakan studi kasus pada jalur *Low Pressure Die Casting* di PT AHM yang jenis produksinya merupakan penjadwalan *Job Shop*.
2. Data-data untuk penelitian ini menggunakan data rencana produksi yang diambil pada bulan oktober 2009.
3. Data besarnya waktu *set-up* dan perpindahan semua material sudah termasuk ke dalam waktu proses produksi yang bersangkutan.
4. Penelitian ini tidak memperhitungkan faktor kendala peralatan dan kendala pekerja.
5. Kondisi mesin produksi diasumsikan berjalan dengan normal, mengabaikan terjadinya *repair* ataupun *breakdown* ditengah-tengah waktu proses.
6. Kondisi bahan baku merupakan komponen standar yang diasumsikan memenuhi syarat dan dalam kondisi *ready stock*
7. Fungsi tujuan yang ingin diperoleh dalam penelitian ini yaitu meminimumkan *makespan* (waktu penyelesaian keseluruhan proses).

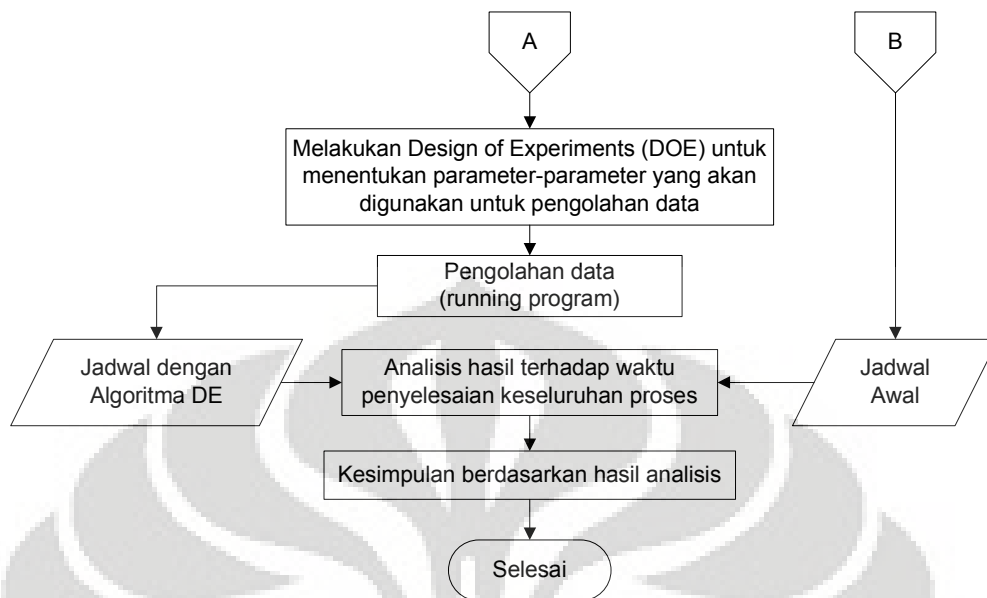
1.6 Metodologi Penelitian

Berikut adalah langkah-langkah metodologi yang digunakan dalam penelitian ini, sebagaimana tergambar pada diagram alir dari metodologi penelitian (gambar 1.2) dengan penjelasan sebagai berikut :

- 1 Melakukan identifikasi permasalahan di perusahaan.
- 2 Mengumpulkan dan menyusun studi literatur yang berkaitan dengan masalah yang telah diidentifikasi.
- 3 Merumuskan masalah, yaitu perlunya perancangan suatu sistem penjadwalan produksi yang efisien.
- 4 Menentukan tujuan, yaitu memperoleh suatu sistem penjadwalan yang lebih baik (meminimumkan *makespan*).
- 5 Mengidentifikasi data yang dibutuhkan dan selanjutnya mengumpulkan data sekunder perusahaan.
- 6 Membuat model matematis dari permasalahan melalui metode Algoritma *Differential Evolution* dengan menggunakan perangkat lunak Matlab 7.8.0.
- 7 Melakukan validasi dan verifikasi terhadap program yang telah dibuat.

- 8 Melakukan *Design of Experiments* (DOE) untuk menentukan parameter-parameter yang akan digunakan untuk pengolahan data.
- 9 Melakukan pengolahan data.
- 10 Membandingkan dan menganalisis solusi jadwal menggunakan algoritma *differential evolution* dengan jadwal yang lama, dimana faktor pembanding yaitu *makespan*.
- 11 Menarik kesimpulan berdasarkan hasil analisis tersebut.





Gambar 1.2 Diagram Alir Metodologi Penelitian

1.7 Sistematika Penulisan

Penulisan laporan penelitian ini dibagi menjadi lima bab.

Bab 1 merupakan bab pendahuluan, menjelaskan mengenai latar belakang permasalahan, diagram yang menggambarkan keterkaitan masalah, perumusan masalah, tujuan penelitian yang ingin dicapai, batasan masalah yang dilakukan, metodologi penelitian yang dilakukan oleh penulis, serta sistematika penulisan.

Bab 2 yang merupakan bab landasan teori berisikan mengenai teori-teori yang berkaitan dengan penjadwalan produksi dan algoritma *differential evolution*.

Bab 3 merupakan bab pengumpulan data, menjelaskan mengenai data yang diambil oleh penulis selama penelitian yang akan dijadikan input dalam pengolahan data yang dilakukan pada tahap selanjutnya.

Bab 4 merupakan pengolahan data dan analisis hasil yang diperoleh. Dalam bab ini terdapat pengembangan program komputer untuk mendapatkan fungsi tujuan dari penelitian, hasil pelaksanaan program, dan analisis hasil program tersebut.

Bab 5 merupakan kesimpulan yang diambil berdasarkan hasil penelitian dan analisisnya.

BAB 2 DASAR TEORI

2.1 Penjadwalan Produksi

Penjadwalan produksi merupakan tahap implementasi dari perencanaan produksi pada lingkungan *shopfloor*. Kelancaran dan aktivitas proses produksi ditentukan oleh penjadwalan produksi yang dibuat. Tujuan yang ingin dicapai adalah memproduksi produk sesuai kebutuhan pada waktu yang telah ditentukan dan jumlah yang diinginkan dengan menggunakan sumber daya manufaktur yang ada secara efisien.

2.1.1 Pengertian Penjadwalan Produksi

Penjadwalan produksi secara umum didefinisikan sebagai penetapan waktu dari penggunaan peralatan, fasilitas, dan aktivitas manusia dalam sebuah organisasi². Penjadwalan produksi mencakup tahapan *loading*, *sequencing*, dan *detailed scheduling*. Pada tahap *loading*, setiap *job* ditentukan prosesnya, kemudian beban (*load*) setiap mesin ditentukan melalui pekerjaan yang harus diproses, dan ditentukan urutan pengerjaan *job* yang dikenal dengan sebutan *sequencing*. Dari urutan tersebut diatur waktu mulai dan selesainya pekerjaan melalui penjadwalan secara mendetail.

2.1.2 Tujuan Penjadwalan Produksi

Penjadwalan memiliki beberapa tujuan. Namun tujuan tersebut dapat saling berkontradiksi. Oleh karena itu, upaya pengoptimasian penjadwalan sangat diperlukan. Adapun tujuan penjadwalan produksi antara lain³:

- Memenuhi waktu pesanan.
- Meminimumkan waktu *set-up*, waktu *work in process*, dan *idle time*.
- Menghasilkan tingkat kegunaan mesin atau pekerja yang tinggi.
- Menetapkan informasi pekerjaan yang cepat.
- meminimumkan biaya produksi dan tenaga kerja.

² Everett E. Adam, JR and Ronald J. Ebert, *Production and Operations Management*, Prentice hall, New Jersey, 1992

³ Steven Nahmias, *Production and Operation Analysis*, Mcgraw-Hill, New York, 1997, hal. 401

2.1.3 Klasifikasi Penjadwalan Produksi

Penjadwalan produksi menurut Pinedo dan Chao (1999) dibagi menjadi beberapa kriteria yaitu:

1. Berdasarkan mesin yang dipergunakan dalam proses:
 - penjadwalan pada mesin tunggal (*single machine shop*).
 - penjadwalan pada mesin jamak.
2. Berdasarkan pola kedatangan *job*:
 - penjadwalan statis, dimana *job* datang bersamaan dan siap dikerjakan pada mesin yang tidak bekerja.
 - penjadwalan dinamis, dimana kedatangan *job* tidak menentu.
3. Berdasarkan lingkungan penjadwalan:
 - *Flow Shop*
Tiap *job* atau pesanan memiliki rute pengerjaan (*routing*) yang sama. Aliran bisa bersifat diskrit, kontinu, maupun semikontinu.
 - *Job Shop*
Setiap *job* atau pesanan memiliki rute pengerjaan yang berbeda-beda, sesuai permintaan konsumen (*complex routing*). Karena kompleksnya aliran, maka penjadwalan pun sangat kompleks. Aliran bersifat diskrit, dan *part* tidak bersifat multiguna (*part* yang mungkin menjadi WIP pada *job* yang satu tidak bisa digunakan pada *job* yang lain).
 - *Assembly Line*
Hampir serupa dengan *flow shop*, akan tetapi proses hanya meliputi bagian perakitan dengan volume yang tinggi dan karakteristik produk yang sedikit. Tidak ditemui *buffer inventory*, kecuali pada bagian awal lini perakitan.

2.1.4 Istilah dalam Penjadwalan Produksi

Berikut istilah-istilah beserta notasinya yang digunakan dalam penjadwalan (Pinedo dan Chao, 1999):

- Setiap *Job* $i \in \{1, 2, \dots, n\}$ yang akan dijadwalkan pada j mesin $\{j=1, 2, \dots, m\}$. Proses pengerjaan *job* i pada mesin j disebut dengan operasi O_{ij} .

- Waktu proses (*processing time*), p_{ij} , yaitu lamanya waktu yang harus dihabiskan *job i* di mesin *j* untuk memproses operasi O_{ij} .
- Waktu tenggat (*due date*), d_i , adalah batas waktu penyelesaian *job i* yang telah ditentukan. Apabila penyelesaian *job* diluar waktu ini, maka akan dikenakan penalti pada *job* tersebut.
- Waktu siap (*release date*), r_i , adalah waktu ketika *job i* masuk ke sistem, yaitu waktu paling awal *job i* bisa mulai diproses. Biasanya $r_i = 0$.
- Waktu mulai (*start time*), s_{ij} , adalah waktu mulai diprosesnya *job i* di mesin *j*.
- Waktu penyelesaian (*completion time*), C_{ij} , adalah waktu penyelesaian pemrosesan *job i* pada mesin *j*.
- *Makespan* biasanya dilambangkan dengan C_{max} , yaitu waktu pengerjaan seluruh *job*.

2.1.5 Karakteristik dan Kendala Proses

Kendala penjadwalan produksi menurut Pinedo dan Chao (1999), yaitu:

- Kendala *precedence*
Kendala ini terjadi ketika suatu *job* baru dapat mulai diproses setelah satu atau sekumpulan *job* lainnya telah selesai diproses
- Kendala biaya dan waktu *setup* yang bergantung pada urutan *job* (*sequence-dependent*)
- *Preemption*
Preemption berarti jika proses produksi sedang berlangsung, maka dapat dihentikan dan digantikan dengan mengerjakan *job* yang baru datang. Keadaan ini biasanya dikarenakan *job* yang berprioritas rendah dapat disela prosesnya oleh *job* yang berprioritas tinggi.
- Kendala mesin dan pekerja
Dalam lingkungan mesin paralel, karakteristik mesin yang digunakan harus sama. Jika tidak sama, maka akan mengganggu proses produksi. Selain itu, umur mesin juga mempengaruhi kapasitas produksi yang dihasilkan. Sedangkan kendala pekerja berkaitan dengan penjadwalan jam kerja operator.

2.1.6 Fungsi Tujuan dan Pengukuran Performa Penjadwalan Produksi

Tujuan yang biasa digunakan untuk menilai performa penjadwalan yang dibuat adalah sebagai berikut

- Meminimumkan *flow time* dan *makespan*.
- Memaksimumkan utilisasi (minimasi waktu mesin dan pekerja yang menganggur).
- Meminimumkan *inventory* dan *WIP (work in process)*.
- Meminimumkan keterlambatan baik *earliness* maupun *tardiness*.
- Meminimumkan jumlah *job* yang terlambat (*number of tardy job*).
- Meminimumkan total biaya penalti atas keterlambatan.

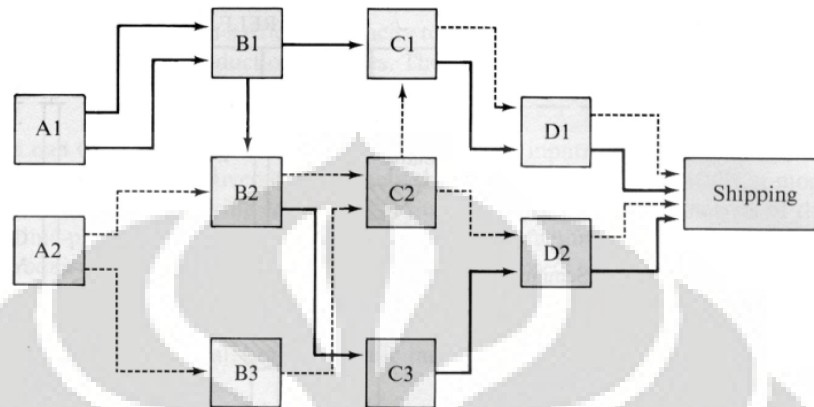
2.2 Penjadwalan *Job Shop*

Job shop adalah suatu lingkungan manufaktur dimana *job-job* yang datang memiliki rute pengerjaan atau operasi yang seringkali tidak sama. Bentuk sederhana dari model ini mengasumsikan bahwa setiap *job* hanya melewati satu jenis mesin sebanyak satu kali dalam rutenya pada proses tersebut. Namun ada juga model lainnya dimana setiap *job* diperbolehkan untuk melewati mesin sejenis lebih dari satu kali pada rutenya. Model ini disebut juga *job shop* dengan *recirculation* (pengulangan).

Karakteristik penjadwalan *job shop* dapat dijabarkan sebagai berikut:

- Ada sejumlah m mesin dan sejumlah n *job*.
- Setiap *job* terdiri dari satu rantai urutan yang dapat berbeda satu sama lain.
- Setiap operasi dalam *job* diproses oleh salah satu mesin yang ada dengan waktu proses yang diasumsikan tetap.
- Setiap proses operasi dapat melewati satu jenis mesin lebih dari satu kali.
- Tidak ada *preemption* (penundaan satu *job* oleh *job* lain).
- Permasalahan penjadwalan untuk model *job shop* merupakan salah satu permasalahan optimasi kombinatorial yang kompleks sehingga disebut *NP-hard* (*NP* merupakan singkatan dari *nondeterministic polynomial*).

Bentuk permasalahan penjadwalan model *job shop* dapat digambarkan dalam bentuk seperti di bawah ini:



Gambar 2.1 Contoh Rute Penjadwalan *Job Shop*

2.3 Metode Penyelesaian Masalah Penjadwalan Produksi

Masalah penjadwalan produksi dapat diselesaikan dengan menggunakan metode heuristik yang terdiri dari 2 jenis, yaitu:

2.3.1 Tipe Heuristik Klasik

Algoritma ini menyusun satu per satu solusi dari masalah penjadwalan. Mulai dari nol, algoritma-algoritma ini memilih mesin-mesin atau *job-job* atau operasi-operasi mana yang harus dijadwalkan terlebih dahulu. Algoritma heuristik klasik yang sering digunakan untuk menyelesaikan penjadwalan *job shop* yaitu *priority dispatch rule*.

Priority dispatch rule adalah suatu aturan penjadwalan yang mengatur *job* mana pada suatu antrian *job* pada suatu mesin yang harus diproses terlebih dahulu berdasarkan prioritas-prioritas tertentu. Jadi, pada saat suatu mesin telah selesai memproses satu *job*, maka berdasarkan *priority dispatch rule* dipilih satu *job* yang memiliki prioritas tertinggi untuk selanjutnya diproses pada mesin tersebut. Berikut ini adalah beberapa aturan yang merupakan *basic priority dispatch rules*, yaitu⁴:

⁴ Everett E. Adam, JR and Ronald J. Ebert, *Op.Cit.*, p.421

- *First Come First Serve (FCFS)*
Menurut aturan ini, urutan penjadwalan dilakukan berdasarkan waktu kedatangan *job* atau pesanan pelanggan. Jadi, *job* yang pertama kali datang, akan dikerjakan terlebih dahulu dan begitu seterusnya untuk *job-job* berikutnya.
- *Earliest Due Date (EDD)*
Menurut aturan ini, urutan penjadwalan dilakukan berdasarkan pada *due date* setiap *job*. Aturan ini mengabaikan waktu kedatangan dan total waktu proses setiap *job*. Artinya, *job* yang memiliki *due date* yang paling awal di antara *job-job* lainnya dipilih sebagai *job* yang memiliki prioritas paling tinggi untuk diproses pada sebuah mesin. Aturan ini cenderung digunakan untuk meminimumkan maksimum *lateness* pada *job-job* yang ada dalam antrian.
- *Minimum Slack First (MS)*
Menurut aturan ini, *job* diurutkan berdasarkan waktu *slack* yang paling kecil. Pada saat sebuah mesin selesai memroses suatu *job*, maka kemudian dihitung waktu *slack* yang tersisa ($d_i - p_i - t$, 0) dari tiap-tiap *job* yang ada dalam antrian, dimana t adalah waktu sekarang. *Job* yang memiliki waktu *slack* yang paling kecil kemudian dipilih sebagai *job* yang memiliki prioritas paling tinggi untuk diproses selanjutnya. Aturan ini digunakan untuk meminimumkan fungsi tujuan yang berkaitan dengan *due date*, yaitu *lateness* dan *tardiness*.
- *Shortest Processing Time First (SPT)*
Menurut aturan ini, *job* diurutkan berdasarkan pada lamanya waktu proses tiap *job*. Jadi, *job* yang memiliki waktu proses paling singkat akan diproses terlebih dahulu dan kemudian dilanjutkan oleh *job-job* lainnya sampai pada *job* yang paling lama waktu prosesnya. Aturan ini berguna untuk penyeimbangan beban kerja antar mesin yang disusun secara paralel.

2.3.2 Tipe Heuristik Modern (Meta-Heuristik)

Algoritma heuristik modern atau yang lebih dikenal dengan meta-heuristik memecahkan masalah penjadwalan produksi dengan melakukan perbaikan mulai dengan satu atau lebih solusi awal. Solusi awal ini dapat dihasilkan secara acak, dapat pula dihasilkan berdasarkan heuristik tertentu. Empat algoritma meta-

heuristik yang dapat digunakan dalam memecahkan masalah penjadwalan *job shop* yaitu:

- *Simulated Annealing*
Ide dasar *Simulated Annealing* terbentuk dari pemrosesan logam. *Annealing* (memanaskan kemudian mendinginkan) dalam pemrosesan logam ini adalah suatu proses bagaimana membuat bentuk cair berangsur-angsur menjadi bentuk yang lebih padat seiring dengan penurunan temperatur. *Simulated annealing* biasanya digunakan untuk penyelesaian masalah yang mana perubahan keadaan dari suatu kondisi ke kondisi yang lainnya membutuhkan ruang yang sangat luas.
- *Tabu Search*
Tabu search merupakan metode optimasi yang menggunakan *short-term memory* untuk menjaga agar proses pencarian tidak terjebak pada nilai *optima local*. Metode ini menggunakan *tabu list* untuk menyimpan sekumpulan solusi yang baru saja dievaluasi. Selama proses optimasi, pada setiap iterasi, solusi yang akan dievaluasi akan dicocokkan terlebih dahulu dengan isi *tabu list* untuk melihat apakah solusi tersebut sudah ada pada *tabu list*. Apabila sudah ada, maka solusi tersebut tidak akan dievaluasi lagi. Keadaan ini terus berulang sampai tidak ditemukan lagi solusi yang tidak terdapat dalam *tabu list*. Pada metode *tabu search*, solusi baru dipilih jika solusi tersebut yang merupakan anggota bagian himpunan solusi tetangga merupakan solusi dengan fungsi tujuan paling baik jika dibandingkan dengan solusi-solusi lainnya dalam himpunan solusi tetangga tersebut. Tetangga (*neighbour*) dari suatu solusi adalah solusi-solusi lain yang dapat diperoleh dari solusi tersebut dengan cara memodifikasinya berdasarkan aturan-aturan tertentu yang dikenal dengan nama *neighborhood functions*.
- Algoritma Genetika
Algoritma Genetika dimodelkan berdasar proses alami, yaitu model seleksi alam oleh Darwin, sedemikian hingga kualitas individu akan sangat kompatibel dengan lingkungannya (dalam hal ini kendala permasalahan). Algoritma genetika memberikan suatu alternatif untuk proses penentuan nilai parameter dengan meniru cara reproduksi genetika. Teknik pencarian

dilakukan sekaligus atas sejumlah solusi yang mungkin yang disebut dengan populasi. Setiap individu adalah satu buah solusi unik dan populasi adalah satu himpunan solusi pada setiap tahapan iterasi. Algoritma genetika bekerja untuk mencari struktur individu berkualitas tinggi yang terdapat dalam populasi.

- **Algoritma *Differential Evolution***

Differential Evolution Algorithm (Algoritma Evolusi Diferensial) merupakan metode metaheuristik akhir. Metode ini terbilang cukup baru, merupakan versi pengembangan dari Algoritma Genetika. Prinsipnya adalah berdasarkan analogi evolusi biologi, yang terdiri dari proses penginisialisasian populasi, proses mutasi, proses penyilangan, dan proses penyeleksian. Keunggulan algoritma ini adalah berstruktur sederhana, mudah dalam pengimplementasian, cepat dalam mencapai solusi, dan bersifat tangguh (memiliki standar deviasi yang kecil).

2.4 Algoritma *Differential Evolution* (DE)

2.4.1 Sejarah Algoritma DE

Dalam matematika dan komputasi, algoritma merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan catatan untuk setiap masalah, ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria, dalam hal ini berbeda dengan heuristik. Algoritma sering mempunyai langkah pengulangan (iterasi) atau memerlukan keputusan sampai tugasnya selesai⁵.

Ada ratusan algoritma yang ada, akan tetapi Wolpert dan Macready menunjukkan bahwa belum ada algoritma superior yang dapat menyelesaikan semua permasalahan. Selama empat dekade, belum ada riset yang dapat memberikan solusi algoritma terbaik, karena dalam praktiknya masih banyak kendala, seperti fungsi objektif yang *non-differentiable*, *non-continuous*, non-linear, multi-dimensi, memiliki banyak *constraint* dan *stochasticity*⁶. Sampai akhirnya pada tahun 1995, Storn dan Price menawarkan pilihan baru yaitu

⁵ <http://www.id.wikipedia.org/wiki/Algoritma>, (accessed 5 November 2009)

⁶ Rasmus K. Ursem, "Differential Evolution Made Easy", *Technical Report* no. 1, 2005

algoritma *differential evolution* (DE), yang telah melalui serangkaian tes dan menjadi kandidat terkuat sebagai algoritma terbaik⁷. DE merupakan pengembangan dari *genetic algorithm* (GA), yaitu teknik pencarian solusi yang menirukan konsep evolusi natural melalui operasi reproduksi, pindah silang, dan mutasi. Perbedaan utama antara DE dan GA adalah pada skema mutasi DE yang *self adaptive* dan pada proses seleksi dimana semua solusi pada DE memiliki kesempatan yang sama untuk terpilih sebagai induk (vektor target)⁸.

Untuk pertama kalinya DE dijelaskan oleh Price dan Storn di ICSI *technical report* pada tahun 1995⁹. Satu tahun kemudian, DE sukses didemonstrasikan di kontes internasional pertama mengenai evolution optimasi yang diadakan bersamaan dengan IEEE (*International Conference on Evolutionary Computation*) dan berhasil memenangkan tempat ketiga. Terinspirasi dari hasil tersebut, Price dan Storn menulis sebuah artikel untuk jurnal Dr. Dobbs ("*Differential Evolution: A simple evolution strategy for fast optimization*") yang diterbitkan pada April 1997 dan selanjutnya mereka menerbitkan artikel lagi untuk *Journal of Global Optimization* ("*Differential Evolution: A simple and efficient Heuristik for Global Optimization over Continuous Space*"). Artikel-artikel ini memperkenalkan algoritma DE ke publik internasional dan mendemonstrasikan keuntungan DE dibandingkan metode heuristik lainnya (metode yang didasarkan pada penilaian dan pengalaman tetapi tidak dapat menjamin menghasilkan solusi optimal matematik).

Pada tahun 1997, Storn dan Price telah membuktikan bahwa DE lebih akurat dan lebih efisien dibandingkan *simulated annealing* (metode berbasis probabilitas dan statistik) dan algoritma genetika (GA). Pada tahun 2004, pernyataan ini diperkuat lagi oleh Lampinen dan Storn, bahkan diperoleh bukti baru bahwa DE lebih baik dibandingkan dengan program *evolutionary* (EA) lain sekalipun. Keunggulan DE dibandingkan algoritma yang lain adalah strukturnya

⁷ B.V. Babu and Rakesh Angira, "Optimization of Thermal Cracker Operation using Differential Evolution", in *Proceedings of International Symposium and 54th Annual Session of II*, 2001

⁸ Dervis Karaboga and Selcuk Okdem, "A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm", *Turk J. Elec Engin.*, vol. 12, no.1, 2004, p.53

⁹ http://www.en.wikipedia.org/wiki/Differential_evolution, (accessed 5 November 2009)

yang sederhana, mudah untuk diaplikasikan, cepat, dan tangguh¹⁰. Hingga kini, DE telah sukses diaplikasikan dalam berbagai bidang.

2.4.2 Konsep Dasar

Algoritma DE merupakan algoritma optimasi global yang efisien, yang didasarkan pada prinsip evolusi. DE merupakan bagian dari algoritma *evolutionary* dan memiliki beberapa keunggulan dibandingkan dengan metode optimasi klasik yang lain, antara lain:

- Memiliki populasi yang berisi calon-calon penyelesaian.
- Merupakan metode *non-deterministic* yang menghasilkan solusi-solusi berbeda meskipun model awalnya tidak diubah, karena pemakaian acak *sampling*.
- Menggabungkan elemen-elemen dari solusi-solusi yang telah ada untuk menciptakan solusi baru dengan mewarisi ciri-ciri yang dimiliki oleh setiap orang tua.

Hampir sama dengan algoritma evolusi lainnya, DE menggunakan individu (vektor) sebagai representasi solusi kandidat. Teknik pencariannya dilakukan sekaligus atas sejumlah solusi yang dikenal dengan istilah populasi. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi dari individu-individu melalui iterasi yang disebut generasi. Setiap individu didefinisikan sebagai faktor berdimensi-D ($X \in R^D$) yang merupakan anggota populasi pada generasi ke-G. Populasi dinotasikan sebagai $P(G)=\{X_1, X_2, \dots, X_{NP}\}$.

Pada setiap generasi, individu akan melalui proses evaluasi dengan menggunakan alat ukur yaitu nilai fungsi objektif (*OBF/Objective Function*) yang akan menunjukkan kualitas populasi tersebut. Populasi generasi yang baru akan dibentuk dengan cara mengevaluasi OBF dari vektor induk dan anak (vektor *trial*). Vektor *trial* terbentuk dari gabungan individu generasi sekarang yang

¹⁰ Srikanta Routroy and Rambabu Kodali, "Differential Evolution Algorithm for Supply Chain Inventory Planning", in *Journal of Manufacturing Technology Management*, vol. 16, no.1, 2005, p.12.

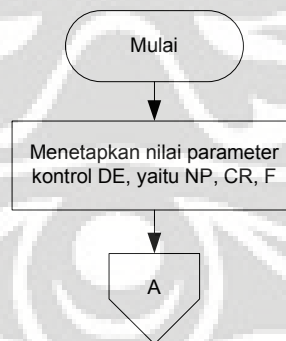
bertindak sebagai vektor target yang telah mengalami proses mutasi dan pindah silang.

Algoritma ini mengeksploitasi populasi solusi potensial yang menyelidiki ruang pencarian dengan mekanisme proses mutasi, pindah silang, dan penyeleksian. Mutasi merupakan proses utama yang memberikan penekanan pada perbedaan sepasang individu acak saluran populasi¹¹. Pindah silang akan menampilkan rekombinasi linear antara individu hasil mutasi (vektor mutasi) dengan orang tua (vektor target) dan menghasilkan satu anak (vektor *trial*).

Proses penyeleksian antara kedua vektor ini bersifat deterministik (yang terbaik diantara keduanya akan menjadi populasi untuk generasi berikutnya) yaitu dengan cara membandingkan fungsi objektif antara kedua individu tersebut. Dalam proses evaluasi nantinya, sistem akan menolak individu yang lain, sehingga ukuran populasi akan tetap konstan dan tidak berubah selama masa pencarian¹². Setelah melalui beberapa generasi dan mencapai kriteria terminasi, maka algoritma ini akan konvergen ke kromosom terbaik yang menjadi solusi penyelesaian.

2.4.3 Tahapan Pengerjaan

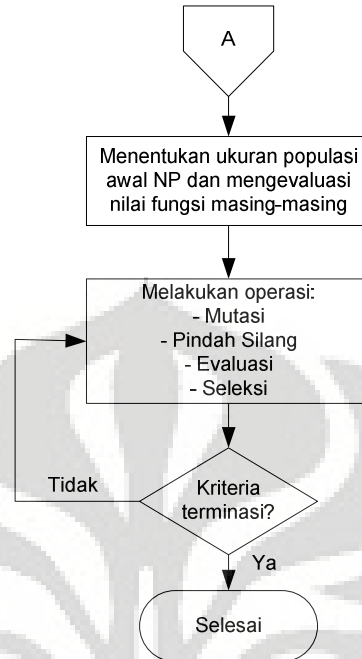
Tahapan pengerjaan algoritma DE dapat dilihat pada gambar 2.2 dan 2.3, yaitu sebagai berikut¹³:



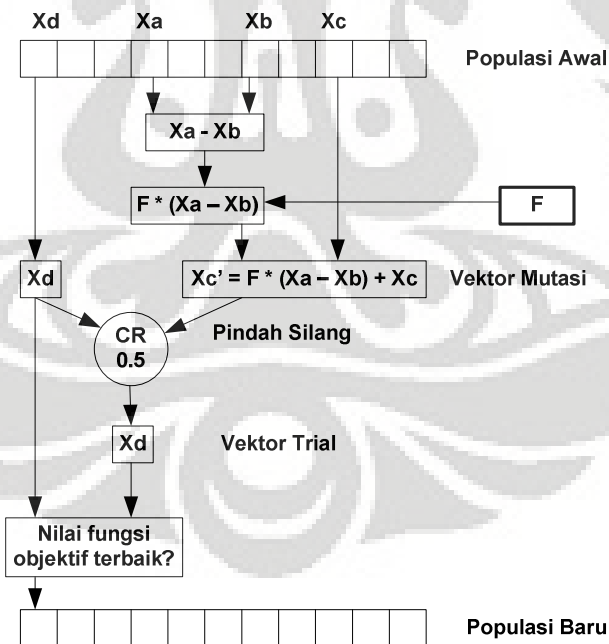
¹¹ Dervis Karaboga and Selcuk Okdem, *Op.Cit*, p.54

¹² I.L. Lopez Cruz, L.G. Van Willigenburg and G. Van Straten, Parameter Control Strategy in Differential Evolution Algorithm for Optimal Control, in *Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing*, May 21-24, 2001, Cancun, Mexico, pp. 211-216.

¹³ Neal M., et. al., "Applying Differential Evolution to a Whole-Farm Model to Assist Optimal Strategic Decision Making", 2006



Gambar 2.2 Diagram Alir Pengerjaan Algoritma DE



Gambar 2.3 Representasi Proses *Differential Evolution*

2.4.3.1 Inisialisasi

Tahap ini meliputi penerapan parameter kontrol dan populasi awal. Tujuan penetapan parameter kontrol adalah untuk menemukan solusi yang dapat diterima melalui sejumlah evaluasi fungsi dan nantinya akan berdampak pada performa DE (efektifitas, efisiensi, dan ketangguhan). Ada tiga parameter kontrol pada DE, yaitu ukuran populasi, parameter kontrol pindah silang, dan parameter kontrol mutasi.

Ukuran populasi (NP) merupakan jumlah saluran populasi dalam satu generasi, dan nilainya tidak akan berubah selama proses pencarian. Namun, jika pencarian mengalami *stuck* maka NP dapat dinaikkan. Pada umumnya $NP = 10 \times d$, dimana d adalah ukuran dimensi. Dimensi merupakan input parameter yang nilainya akan berubah-ubah selama proses pencarian solusi optimal. Populasi awal (berisikan individu sejumlah NP) yang diinisialisasikan, merupakan solusi awal yang dapat diperoleh dari metode heuristik maupun diperoleh secara acak.

Parameter kontrol mutasi (F) merupakan faktor konstan dan *real* yang akan mengendalikan operasi mutasi, nilainya berada pada kisaran $[0,2]$. Faktor F yang berada pada kisaran $[0,4,1]$ dinilai efektif. Nilai F yang lebih besar dari 1 akan membawa DE mencari solusi di luar jangkauan yang layak. Dan nilai F yang kurang dari 0.4 juga tidak efektif karena akan membawa DE mencari solusi yang mendekati area vektor target. Jika berada pada solusi *stuck*, selain dengan menaikkan NP, dapat juga dengan menaikkan faktor F. DE lebih sensitif terhadap pemilihan faktor F dibandingkan pemilihan CR.

Parameter kontrol pindah silang (CR) merupakan faktor pengendali operasi pindah silang, berada pada kisaran $[0,1]$. Faktor CR berperan sebagai *fine tuning element* (element penentu) pada saat operasi pindah silang. Faktor CR akan memberikan aturan berapa banyak rata-rata gen yang bertalian dari vektor mutasi dikopi ke keturunan. Nilai CR yang tinggi (contohnya satu) akan mempercepat terjadinya konvergensi. Terkadang untuk beberapa permasalahan, nilai CR perlu diturunkan agar DE lebih tangguh.

2.4.3.2 Evaluasi Populasi Awal

Dari populasi yang ada, dilakukan evaluasi untuk menyesuaikan nilai parameter individu terhadap nilai fungsi objektifnya. Kemudian akan dipilih 4 vektor secara acak, dimana vektor pertama akan menjadi vektor target, selisih nilai vektor kedua dan ketiga akan menjadi vektor selisih, dan vektor keempat sebagai pembentuk vektor mutasi.

2.4.3.3 Mutasi

Mutasi adalah proses pertukaran sejumlah gen dalam satu individu dengan menukar nilai karakter pada gen-gen tersebut dengan kebalikannya. Mutasi dilakukan untuk menjaga agar tidak terciptanya konvergensi prematur (solusi yang tidak optimal). Biasanya proses mutasi ini melibatkan beberapa individu (umumnya tiga). Proses ini diformulasikan dengan rumus:

$$X4' = F \times (X2 - X3) + X4 \quad (2.1)$$

Dimana:

$X4'$ = Vektor Mutasi

F = Parameter kontrol mutasi

$X2, X3, X4$ = Vektor yang dipilih secara acak

2.4.3.4 Pindah Silang

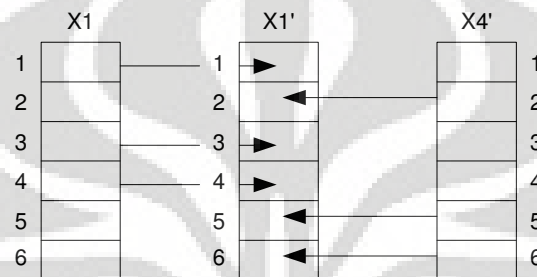
Pindah silang atau kawin silang bertujuan untuk menambah keanekaragaman gen dalam populasi dengan penyilangan antar gen yang diperoleh dari produksi sebelumnya. Vektor mutasi $X4'$ dikawinkan dengan vektor target $X1$ menggunakan operasi pindah silang untuk menghasilkan vektor *trial*. Gen *trial* individual diwariskan dari $X4'$ dan $X1$ yang ditentukan melalui nilai faktor pindah silang (CR), seperti terlihat pada tabel 2.1. dan gambar 2.4. Proses pindah silang akan dibantu dengan matriks baru yang nilainya diperoleh secara acak. Untuk kasus pada gambar 2.4, matriks baru merupakan matriks 6x1. Jika nilai baris matriks acak lebih besar dibandingkan dengan nilai CR, maka baris pada vektor target akan menjadi baris pada vektor *trial*, dan sebaliknya.

Tabel 2.1 Proses Pindah Silang

```

for i = 1 : 6
  if rand (0,1) > CR
    X1' (i,1) = X1 (l,1)
  else
    X1 (i,1) = X4' (l,1)
  end
end
end

```

**Gambar 2.4** Proses Pindah Silang

2.4.3.5 Evaluasi Vektor Trial

Vektor *trial* akan dievaluasi, untuk menyesuaikan nilai parameter individu terhadap nilai fungsi objektifnya.

2.4.3.6 Seleksi

Penyeleksian dilakukan untuk memilih individu manakah yang akan menjadi saluran populasi generasi berikutnya (vektor target atau vektor *trial*). Vektor *trial* dapat menggantikan vektor target individual jika dan hanya jika nilai fungsi objektifnya lebih baik daripada nilai fungsi objektif vektor target. Pada tabel 2.2 dapat terlihat bahwa jika nilai vektor *trial* lebih besar dibandingkan dengan vektor target, maka vektor *trial* akan menggantikan vektor target pada generasi sekarang dan akan menjadi vektor baru untuk generasi berikutnya¹⁴.

¹⁴ Efren Mezura-Montes, Jesús Velásquez-Reyes, and Carlos A. Coello Coello, "Promising Infeasibility and Multiple Offspring Incorporated to Differential Evolution for Constrained Optimization", *GECCO'05*, Washington, DC, 2005.

2.4.3.7 Terminasi

Proses pencarian akan berhenti jika telah dicapai kriteria terminasi. Namun, bila kriteria berhenti (terminasi) belum terpenuhi maka akan dibentuk lagi generasi baru dengan mengulangi langkah-langkah sebelumnya. Beberapa kriteria berhenti yang sering digunakan antara lain: generasi/iterasi tertentu, waktu pencarian maksimum, dan nilai OBF terbaik tidak lagi berubah.

Tabel 2.2 Proses Seleksi

<pre> if OBF vektor target >= OBF vektor trial vektor target = vektor target else vektor target = vektor trial end </pre>
<pre> if OBF vektor target >= OBF vektor trial X selanjutnya = vektor target else X selanjutnya = vektor trial end </pre>

DE memiliki beberapa varian¹⁵, dinotasikan dalam $DE/x/y/z$, dimana x mendefinisikan vektor/individu yang akan dimutasi, bisa *random* ataupun *best vector*; y mendefinisikan jumlah *difference vector* yang digunakan; dan z mendefinisikan skema pindah silang yakni *binomial* atau *exponential*. Varian yang sering digunakan adalah $DE/rand/1/bin$ ¹⁶. Varian ini dianggap sebagai versi dasar dari DE.

2.4.4 Penerapan Algoritma DE pada Masalah Penjadwalan *Job Shop*

Algoritma DE yang akan diterapkan pada permasalahan ini menggunakan varian $DE/rand/1/bin$ yang diperkenalkan Storn dan Price¹⁷. Adapun *pseudo code* algoritma DE yang akan digunakan adalah sebagai berikut:

¹⁵ Nasimul Noman and Hitoshi Iba, "Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization", *GECCO'05*, Washington, DC, USA, 2005

¹⁶ Hui-Yuan Fan, Jouni Lampinen, and Yeshayahou Levy, "An Easy-to-Implement Differential Evolution Approach for Multi-Objective Optimization", in *International Journal for Computer-Aided Engineering and Software*, 2006, vol. 23, no. 2, p.126

¹⁷ Tasgetiren et al, "Particle Swarm Optimization and Differential Evolution Algorithm for Job Shop Scheduling Problem", in *Proceedings of the 4th International Symposium on Intelligent Manufacturing System (IMS2004)*, Sakarya, Turkey, 2004, p.6

```

Inisialisasi parameter
Inisialisasi populasi target
Melakukan operasi permutasi
Evaluasi
Do{
    Membentuk populasi mutan
    Membentuk populasi trial
    Melakukan operasi permutasi
    Mengevaluasi populasi trial
    Proses Penyeleksian
}While (Berhenti)

```

2.4.4.1 Elemen Dasar Algoritma DE

Sebelum memulai proses pencarian solusi optimal pada permasalahan penjadwalan *job shop* dengan metode algoritma *Differential Evolution*, diperkenalkan terlebih dahulu elemen-elemen dasar algoritma DE yang akan digunakan. Elemen-elemen dasar tersebut adalah sebagai berikut¹⁸:

- Individu target: X_i^t adalah individu ke- i anggota populasi pada generasi ke- t , dan diuraikan sebagai $X_i^t = |X_{1i}^t, X_{2i}^t, X_{3i}^t, \dots, X_{ni}^t|$ dimana X_{ij}^t merupakan nilai dimensi individu ke- i terhadap dimensi ke- j ($j=1,2,\dots,n$).
- Individu mutan: V_i^t adalah individu ke- i anggota populasi pada generasi ke- t , dan diuraikan sebagai $V_i^t = |V_{1i}^t, V_{2i}^t, V_{3i}^t, \dots, V_{ni}^t|$ dimana V_{ij}^t merupakan nilai dimensi individu ke- i terhadap dimensi ke- j ($j=1,2,\dots,n$).
- Individu *trial*: U_i^t adalah individu ke- i anggota populasi pada generasi ke- t , dan diuraikan sebagai $U_i^t = |U_{1i}^t, U_{2i}^t, U_{3i}^t, \dots, U_{ni}^t|$ dimana U_{ij}^t merupakan nilai dimensi individu ke- i terhadap dimensi ke- j ($j=1,2,\dots,n$).
- Populasi target: P^t adalah kumpulan individu X_i^t sejumlah NP dalam populasi target pada generasi ke- t .

¹⁸ *Ibid...*p.7

- Populasi mutan: V^t adalah kumpulan individu V_i^t sejumlah NP dalam populasi mutan pada generasi ke-t.
- Populasi *trial*: U^t adalah kumpulan individu U_i^t sejumlah NP dalam populasi *trial* pada generasi ke-t.
- Operasi permutasi: π_i^t operasi permutasi *job* terhadap individu X_i^t . Diuraikan sebagai $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, \dots, \pi_{in}^t]$, dimana π_{ij}^t merupakan penugasan operasi ke-j individu ke-i, pada iterasi ke-t.
- Konstanta mutasi: $F \in (0,2)$
- Konstanta pindah silang: $CR \in (0,1)$
- Fungsi objektif: Dalam suatu masalah minimasi, fungsi objektif dilambangkan dengan $f_i(\pi_i^t \leftarrow X_i^t)$. Dalam penelitian ini, fungsi objektif yang digunakan adalah $f_i(\pi_i^t \leftarrow X_i^t) = \min \sum_{j=1}^n (\beta_j T_j)$ dimana β_j adalah penalti atas satu unit tardiness dan T_j adalah total unit tardiness yang terjadi, sedangkan untuk *earliness* tidak dikenakan penalti (penalti dianggap bernilai nol).
- Kriteria terminasi: adalah kondisi dimana program akan berhenti berjalan, bisa dalam bentuk jumlah maksimum generasi atau waktu maksimum CPU bekerja.

2.4.4.2 Prosedur Operasi Pencarian

Untuk melakukan proses pencarian solusi optimal pada permasalahan penjadwalan *job shop* dengan metode algoritma DE, dilakukan beberapa tahap atau prosedur. Prosedur ditunjukkan dalam diagram alir yang tertera pada gambar 2.5. Prosedur tersebut dijabarkan sebagai berikut¹⁹:

1. Tahap inisialisasi

- Menetapkan $t = 0$, CR, F, dan NP

Dimensi : jumlah *job*

- Membangun individu awal sebanyak NP, yakni $\{X_i^t, i = 1, 2, \dots, NP\}$,

dimana $X_i^0 = [X_{i1}^0, X_{i2}^0, \dots, X_{i,mm}^0]$; $X_{ik}^0 = X \text{ min} + (X \text{ max} - X \text{ min}) \cdot xr$

¹⁹ *Ibid.*, p.8

$X_{min} = -1$; $X_{max} = 1$; $r =$ bilangan random (0-1)

- Melakukan operasi permutasi $\pi_i^0 = [\pi_{i1}^0, \pi_{i2}^0, \dots, \pi_{in}^0]$ untuk setiap X_i^0 dengan mengaplikasikan aturan *Smallest Position Value* (SPV).
- Mengevaluasi setiap individu i dalam populasi dengan menggunakan fungsi objektif $f_i^0(\pi_i^0 \leftarrow X_i^0)$ ($i = 1, 2, \dots, NP$), untuk memilih individu target.

2. Meng-update generasi $t=t+1$

3. Membentuk populasi mutan

Untuk setiap individu target, akan dicari individu mutan $V_i^{t+1} = [v_{i1}^{t+1}, v_{i2}^{t+1}, \dots, v_{i,nn}^{t+1}]$ yang diperoleh melalui operasi $V_i^{t+1} = X_{ai}^t + F(X_{bi}^t - X_{ci}^t)$, ($a_i \neq b_i \neq c_i$)

4. Membentuk populasi *trial*

Individu *trial* $U_i^{t+1} = [u_{i1}^{t+1}, u_{i2}^{t+1}, \dots, u_{i,nn}^{t+1}]$ diperoleh melalui operasi:

$$u_{ik}^{t+1} = \begin{cases} v_{ik}^{t+1}, & \text{if } r_{ik}^{t+1} \leq CR \\ x_{ik}^t, & \text{otherwise} \end{cases}$$

CR adalah konstanta pindah silang pada *range* (0,1), dan r_{ik}^{t+1} adalah bilangan acak *uniform* antara 0 sampai 1.

5. Melakukan operasi permutasi *job*

Operasi permutasi *job* dilakukan dengan menerapkan aturan SPV untuk melakukan operasi permutasi $\phi_i^t = [\phi_{i1}^t, \phi_{i2}^t, \dots, \phi_{i,nn}^t]$ ($i = 1, 2, \dots, NP$).

6. Mengevaluasi populasi *trial*

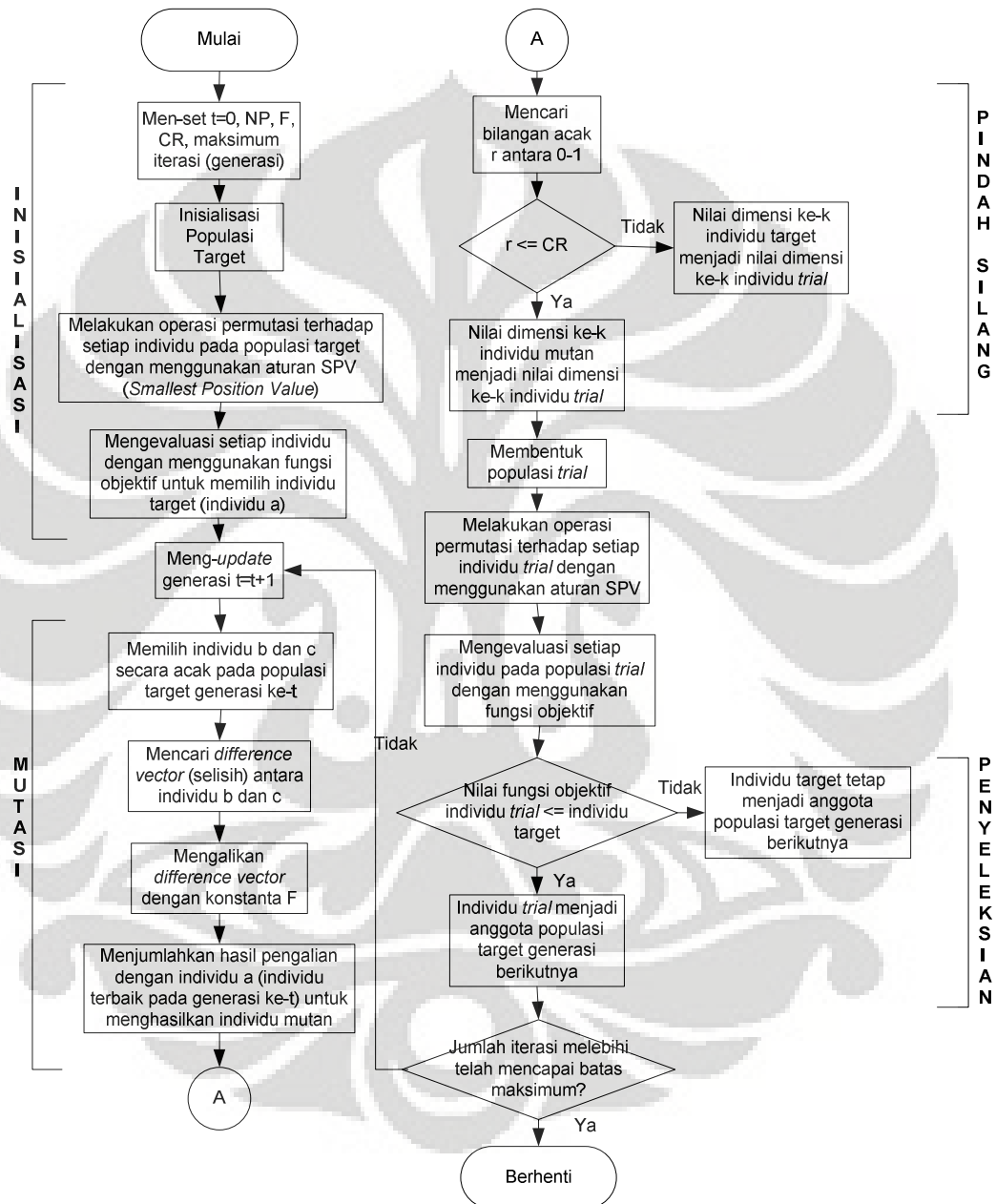
Evaluasi populasi *trial* menggunakan fungsi objektif $f_i^{t+1}(\pi_i^{t+1} \leftarrow U_i^{t+1})$ ($i = 1, 2, \dots, NP$)

7. Melakukan penyeleksian

Nilai fungsi objektif individu *trial* U_i^{t+1} akan dibandingkan dengan individu target generasi sebelumnya, X_i^t , untuk menentukan apakah individu *trial* tersebut layak menjadi anggota populasi target generasi berikutnya atau tidak.

8. Menghentikan operasi pencarian

Jika jumlah iterasi sudah mencapai jumlah iterasi yang telah ditentukan, maka algoritma dihentikan, namun jika belum maka kembali ke tahap 2.



Gambar 2.5 Diagram Alir Algoritma DE untuk *Job Shop Sequencing Problem*

2.5 *Design of Experiments (DOE)*

Percobaan ialah sebuah pengujian atau rangkaian pengujian yang mana perubahan yang berguna dilakukan pada variabel input dari sebuah proses atau sistem sehingga dapat diamati alasan untuk merubah respon dari proses. DOE merupakan suatu alat statistik yang penting dalam dunia industri untuk peningkatan performa/kinerja dari sebuah proses manufaktur. DOE juga dapat menjadi sebuah aplikasi dalam pengembangan proses baru. DOE juga dikaitkan dengan proses memanipulasi faktor-faktor yang terkendali agar tidak hanya menentukan efek faktor tersebut pada output yang diharapkan tetapi juga menemukan kombinasi dari faktor-faktor yang ada agar dihasilkannya output yang maksimum.

2.5.1 Tujuan *Design of Experiments*

Tujuan dari dilakukannya percobaan adalah²⁰:

1. Menentukan variabel paling berpengaruh pada output y .
2. Menentukan nilai optimal variabel x agar dicapai nilai y yang ideal.
3. Menentukan nilai optimal variabel x agar variansi nilai y minimum.
4. Menentukan nilai optimal variabel x agar pengaruh dari faktor yang tidak dapat dikendalikan z_1, z_2, \dots, z_q minimum.

2.5.2 Tipe Percobaan

Ada empat tipe percobaan yaitu *trial and error*, *one factor at a time*, *full factorial*, *fractional factorial*²¹.

1. ***Trial and Error Experiments***

Trial and Error Experiments merupakan percobaan memanipulasi satu faktor tanpa memperhatikan faktor lainnya. Kelemahan dari metode ini adalah kurang akuratnya hasil yang diperoleh, memakan biaya yang tinggi, waktu yang lama, dan tidak efisien.

²⁰ Douglas C. Montgomery. *Design and Analysis of Experiments*, John Wiley & Sons, New York, 1996, p.2.

²¹ Mark A. Fryman, *Quality and Process Improvement*, USA, 2002, p.320.

2. *One Factor at a Time Experiments*

Perbedaan metode ini dengan percobaan *trial and error* yaitu:

- a. ada rangkaian faktor yang diamati, tetapi hanya satu faktor saja yang diubah pada setiap melakukan percobaan.
- b. hanya satu faktor yang diubah-ubah sementara faktor lainnya dianggap konstan.

Percobaan ini merupakan perkembangan dari metode *trial and error* yang membentuk pendekatan metode percobaan yang sistematis. Kelemahan dari percobaan ini adalah bahwa hasil yang diharapkan kadang tidak tercapai, memakan waktu yang lama, tidak efisien, dan dapat memberikan kesimpulan yang salah dalam suatu percobaan.

3. *Full Factorial*

Percobaan *full factorial* berbeda dengan dua percobaan sebelumnya di mana setiap kombinasi faktor diuji pada level yang berbeda-beda. Metode ini akan memiliki beberapa keuntungan dibandingkan dua metode sebelumnya, sebab kesimpulan yang didapat akan lebih akurat karena setiap kombinasi faktor diujicobakan. Akan tetapi, kelemahan dari metode ini adalah waktu yang diperlukan serta biaya yang dikeluarkan akan besar dengan menjalankan semua kombinasi faktor. Jumlah percobaan/*treatment* yang harus dicoba akan bertambah besar secara signifikan apabila jumlah faktor bertambah.

Uji Hipotesis dalam *Factorial Design*

Misalkan ada 2 faktor yaitu A dan B, maka uji hipotesis yang terjadi yaitu:

- Melihat apakah ada pengaruh dari faktor A:

H_0 : $\tau_1 = \tau_2 = \dots = \tau_a = 0$ (tidak ada pengaruh yang signifikan dari faktor A)

H_1 : $\tau_i \neq 0$ (ada pengaruh yang signifikan dari faktor A)

- Melihat apakah ada pengaruh dari faktor B:

H_0 : $\beta_1 = \beta_2 = \dots = \beta_b = 0$ (tidak ada pengaruh yang signifikan dari faktor B)

H_1 : $\beta_j \neq 0$ (ada pengaruh yang signifikan dari faktor B)

- Melihat interaksi antara faktor A dan B dilakukan dengan:

H_0 : $(\tau\beta)_{ij} = 0$ (tidak ada interaksi yang signifikan antara faktor A dan B)

H_1 : $(\tau\beta)_{ij} \neq 0$ (ada interaksi yang signifikan dari faktor B)

4. *Fractional Factorial*

Banyaknya jumlah percobaan yang harus dilakukan pada *full factorial*, membuat metode tersebut tidak selalu bisa diterapkan pada semua eksperimen/percobaan, terlebih dengan adanya keterbatasan waktu dalam melakukan percobaan. Oleh karena itu, ada metode yang disebut *fractional factorial*. Metode ini akan menjalankan percobaan hanya sebagian/seporsi dari setiap kombinasi yang mungkin. Percobaan *fractional factorial* merupakan salah satu metode yang paling banyak digunakan dalam pengembangan produk dan peningkatan proses. Penggunaan utama dari metode ini adalah untuk *screening experiments* (menyeleksi kombinasi percobaan).

2.5.3 Prinsip Dasar

Menurut Montgomery (1996), ada tiga prinsip dasar dalam melakukan perancangan percobaan adalah *replication*, *blocking*, dan *randomization*.

1. *Replication* (Replikasi)

Replikasi memiliki 2 manfaat penting yaitu memudahkan *experimenter* untuk mendapatkan estimasi kesalahan dari percobaan yang dilakukan dan membantu *experimenter* mendapatkan perkiraan pengaruh efek dari faktor yang digunakan dalam percobaan lebih akurat.

2. *Randomization* (Randomisasi)

Randomisasi berarti urutan percobaan yang akan diuji dilakukan secara acak untuk menghindari terjadinya efek luar yang mempengaruhi hasil percobaan sehingga percobaan tidak valid/bias. Apabila kita tidak melakukan randomisasi, maka ada kemungkinan percobaan tersebut bisa dipengaruhi oleh faktor lingkungan, kelelahan operator, dan kelainan material yang digunakan, dan lain-lain.

3. *Blocking*

Blocking adalah suatu teknik yang digunakan untuk meningkatkan keakuratan dari percobaan. Dengan memblok, kita membagi percobaan ke dalam kelompok atau grup. Sistem blok diberlakukan karena ada kemungkinan terjadinya perbedaan nilai akhir yang cukup jauh apabila percobaan tersebut tidak dikelompokkan.

BAB 3

PENGUMPULAN DATA

3.1 Profil Perusahaan

PT Astra Honda Motor (PT AHM) merupakan salah satu industri manufaktur yang bergerak dibidang otomotif yang ada di Indonesia. PT AHM didirikan pada tahun 2001 dengan memproduksi berbagai jenis sepeda motor. Sejak berdirinya perusahaan, banyak perubahan yang dialami hingga saat ini. Dengan semakin berkembangnya industri-industri manufaktur yang ada di Indonesia, tentunya merupakan suatu tantangan tersendiri bagi PT AHM dalam menjalankan proses produksinya, agar produk yang dihasilkan dapat bersaing di pasaran. Dengan jumlah permintaan yang cukup tinggi untuk saat ini, ditambah total target produksi di tahun 2009 sebanyak 2.600.000 unit, maka perlu dilakukan perbaikan-perbaikan pada proses produksi yang sudah ada, supaya dapat meningkatkan produktivitas dan efisiensi.

3.2 Pengumpulan Data Penelitian

Data yang digunakan dalam penelitian ini merupakan data sekunder yang diperoleh dari data perusahaan PT AHM. Data yang digunakan untuk penelitian penjadwalan *Job Shop* di PT AHM berupa:

1. Data jam kerja di PT AHM.
2. Data pesanan part, yaitu berupa jenis-jenis *cylinder head* dan *piston* yang dipesan serta jumlah pesanan pada periode Oktober 2009.
3. Data rute proses operasi yang harus dilalui oleh tiap *item* yang akan diproduksi dan data waktu proses tiap rute yang dibutuhkan untuk mengerjakan tiap *item* tersebut.
4. Data jumlah tiap mesin yang harus dilalui untuk rute-rute proses operasi.

3.2.1 Data Jam Kerja di PT AHM

PT AHM beroperasi mulai dari hari senin sampai dengan hari jum'at. Setiap harinya PT AHM memberlakukan pembagian waktu kerja menjadi tiga *shift* yaitu *shift* 1 (pagi), *shift* 2 (sore) dan *shift* 3 (malam). Untuk *shift* 1 dimulai

pada pukul 07.00 WIB hingga pukul 16.00 WIB dengan total waktu kerja sebanyak +/- 8 jam. Sedangkan untuk *shift 2* dimulai pada pukul 16.00 WIB hingga pukul 24.00 WIB dengan total waktu kerja sebanyak +/- 7 jam. Dan untuk *shift 3* dimulai pada pukul 00.00 WIB hingga pukul 07.00 WIB dengan total waktu kerja sebanyak +/- 6 jam. Untuk lebih jelasnya dapat dilihat pada tabel 3.1 berikut.

Tabel 3.1 Waktu Kerja PT AHM

	<i>Shift 1</i>		<i>Shift 2</i>		<i>Shift 3</i>	
Waktu Kerja (detik)	07.00 - 16.00	32400	16.00 - 24.00	28800	00.00 - 07.00	25200
Waktu Kerja Nett (detik)						
Aktifitas	<i>Shift 1</i>		<i>Shift 2</i>		<i>Shift 3</i>	
- P5M	07.00 - 07.05	300	16.00 - 16.05	300	00.00 - 00.05	300
- Istirahat 1	09.30 - 09.40	600	17.55 - 18.10	900	01.55 - 02.10	900
- Istirahat Makan	12.00 - 12.40	2400	19.25 - 20.00	2100	03.00 - 03.45	2100
- Istirahat 2	14.20 - 14.30	600	22.20 - 22.30	600	04.20 - 04.30	600
- 5K2S	15.55 - 16.00	300	23.55 - 24.00	300	06.55 - 07.00	300
Sub Total (detik)		4200		4200		4200
Waktu Kerja Nett (detik)		28200		24600		21000
Total Waktu Kerja Nett (detik)	73800					

3.2.2 Data Pesanan Part

Barang jadi diproduksi dalam 2 jenis yaitu *in-house* dan *out-house* (dikerjakan oleh subkontraktor). Di *in-house* dikerjakan oleh 3 *stage* yaitu *casting line*, *machining line*, dan *assembly engine line*. Setiap *stage* melakukan penjadwalan sendiri-sendiri. Untuk penelitian ini, data pesanan yang akan dibahas hanya untuk *stage* di jalur *low pressure die casting* dan hanya pada part *cylinder head* dan *piston*. Setiap paket pesanan berbeda jenis dan jumlah pesannya. Tabel 3.2 merupakan data pesanan di jalur *low pressure die casting* PT AHM untuk jenis part *cylinder head* dan *piston* Proses pengerjaan untuk pesanan pada periode Oktober 2009 dimulai dari tanggal 1 Oktober 2009. Total *cylinder head* dan *piston* yang harus diproduksi pada periode tersebut adalah 24220 buah.

Tabel 3.2 Data Pesanan Periode Oktober 2009

No	Part Name	Pesanan Part		
		Pcs	Pcs/lot	Lot
1	Cyl Head KVLV	3970	200	20
2	Cyl Head KWWF	1830	200	9
3	Cyl Head KEHT	1340	150	9
4	Cyl Head KCJT	1160	150	8
5	Piston KVLV	3250	400	8
6	Piston KWWF	1810	300	6
7	Piston KVYP	3180	400	8
8	Piston KWCA	1720	300	6
9	Cyl Head KVBS	2460	200	13
10	Cyl Head KVYP	2510	200	13
11	Cyl Head KWCA	990	150	7
Total		24220	2650	107

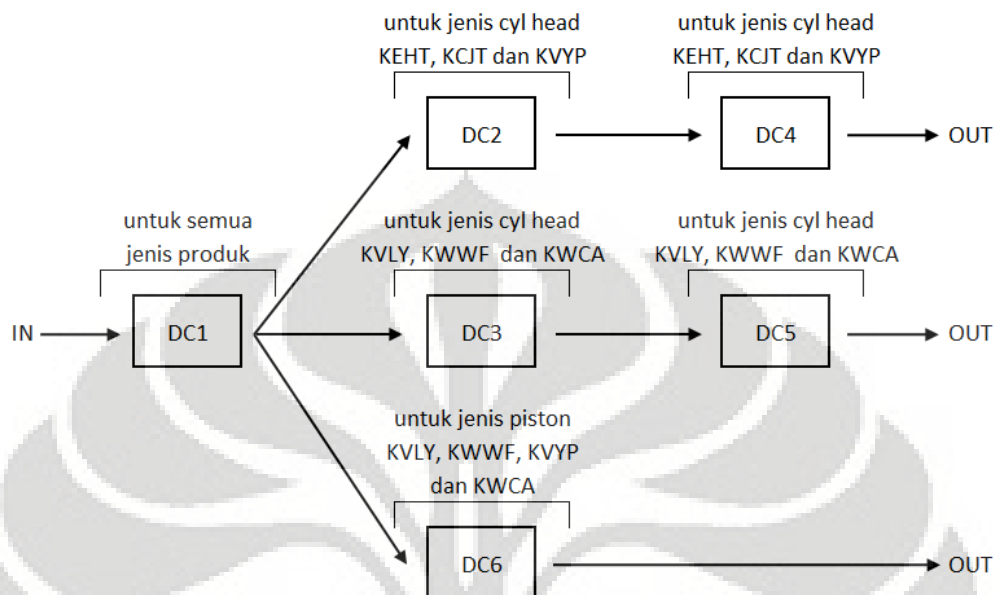
3.2.3 Data Rute dan Waktu Proses

Rute proses *cylinder head* dan *piston* merupakan rute *job shop* karena tidak semua jenis produk melewati rute yang sama. Pengerjaan setiap *cylinder head* terdiri dari 3 proses pemesinan dan setiap *piston* terdiri dari 2 proses pemesinan, setelah melalui proses pemesinan tersebut maka akan dilakukan proses trimming untuk menghilangkan *scrap* dari part kemudian dilakukan proses *shot blasting* untuk membuat *visual part* seragam/*homogen*, setelah proses di jalur *lpdc* selesai maka *cylinder head* dan *piston* akan dikirim ke proses *machining*. Tabel 3.3 merupakan urutan proses produksi yang harus dilalui untuk memproduksi *cylinder head* dan *piston*, jumlah mesin atau peralatan untuk setiap proses dan alokasinya.

Tabel 3.3 Jumlah dan Alokasi Mesin Setiap Rute Proses

Kode Proses	Nama Proses	Jumlah Mesin	Alokasi
DC1	Melting	1	Semua Jenis Produk
DC2	LPDC	1	Untuk jenis cyl Head KEHT, KCJT dan KVYP
DC3	LPDC	1	Untuk jenis cyl Head KVLV, KWWF dan KWCA
DC4	chipping & cutting	1	Untuk jenis cyl Head KEHT, KCJT dan KVYP
DC5	chipping & cutting	1	Untuk jenis cyl Head KVLV, KWWF dan KWCA
DC6	LPDC	1	Untuk jenis piston KVLV, KWWF, KVYP dan KWCA

Untuk lebih jelas mengenai urutan rute proses *cylinder head* dan *piston* dapat dilihat pada gambar 3.1 berikut ini :



Gambar 3.1 Rute Proses Produksi di Jalur LPDC

Tabel 3.4 merupakan waktu proses *cylinder head* dan *piston* untuk setiap proses. Waktu proses pada suatu rute yang bernilai nol menunjukkan bahwa *cylinder head* dan *piston* yang bersangkutan tidak melewati rute tersebut (sesuai gambar 3.1).

Tabel 3.4 Waktu Proses Produksi

No	Part Name	Waktu Proses / Lot (detik)					
		DC1	DC2	DC3	DC4	DC5	DC6
1	Cyl Head KVLY	6521.2	0	29500.6	0	16380.6	0
2	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
3	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
4	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0
5	Piston KVLY	5196.6	0	0	0	0	23242.2
6	Piston KWWF	5196.6	0	0	0	0	23242.2
7	Piston KVYP	5099.1	0	0	0	0	33729.7
8	Piston KWCA	4831.8	0	0	0	0	24183.8
9	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
10	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
11	Cyl Head KWCA	4265.6	0	17342.8	0	11322.8	0

3.2.4 Jadwal Produksi di Jalur LPDC PT AHM Periode Oktober 2009

Dalam membuat jadwal produksi, jumlah pesanan untuk setiap jenis *cylinder head* dan *piston* dipecah lagi menjadi satuan *job*. Sebagai contoh, jika *cylinder head* KVLVY dipesan sebanyak 20 lot, dikarenakan setiap mesin hanya bisa mengerjakan satu operasi, sehingga susunan mesin disebut seri (tidak ada yang paralel), maka pesanan akan diproduksi tidak sekaligus 20 lot, tetapi dipecah menjadi 20 *job*. Jadi, ada 107 *job* yang akan dijadwalkan oleh PT AHM pada periode bulan Oktober 2009. *Job-job* tersebut akan melewati 6 mesin (rute proses). Selanjutnya, 107 pesanan tersebut akan dibuat urutan pekerjaannya. Berikut ini urutan pengerjaan setiap *job* di jalur LPDC oleh PT AHM.

Tabel 3.5 Penjadwalan di jalur LPDC Lengkap (Periode Oktober 2009)

No	Part Name	Waktu Proses / Lot (detik)					
		DC1	DC2	DC3	DC4	DC5	DC6
1	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
2	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
3	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
4	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
5	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
6	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
7	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
8	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
9	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
10	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
11	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
12	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
13	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
14	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
15	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
16	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
17	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
18	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
19	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
20	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
21	Cyl Head KVLVY	6521.2	0	29500.6	0	16380.6	0
22	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
23	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
24	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0

Tabel 3.5 (sambungan)

No	Part Name	Waktu Proses / Lot (detik)					
		DC1	DC2	DC3	DC4	DC5	DC6
25	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
26	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
27	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
28	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
29	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
30	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
31	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
32	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
33	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
34	Piston KVLV	5196.6	0	0	0	0	23242.2
35	Piston KVLV	5196.6	0	0	0	0	23242.2
36	Piston KVLV	5196.6	0	0	0	0	23242.2
37	Piston KVLV	5196.6	0	0	0	0	23242.2
38	Piston KWWF	5196.6	0	0	0	0	23242.2
39	Piston KWWF	5196.6	0	0	0	0	23242.2
40	Piston KWWF	5196.6	0	0	0	0	23242.2
41	Piston KWWF	5196.6	0	0	0	0	23242.2
42	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0
43	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0
44	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0
45	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0
46	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0
47	Cyl Head KWCA	4265.6	0	17342.8	0	11322.8	0
48	Cyl Head KWCA	4265.6	0	17342.8	0	11322.8	0
49	Cyl Head KWCA	4265.6	0	17342.8	0	11322.8	0
50	Cyl Head KWCA	4265.6	0	17342.8	0	11322.8	0
51	Cyl Head KWCA	4265.6	0	17342.8	0	11322.8	0
52	Piston KVYP	5099.1	0	0	0	0	33729.7
53	Piston KVYP	5099.1	0	0	0	0	33729.7
54	Piston KVYP	5099.1	0	0	0	0	33729.7
55	Piston KVYP	5099.1	0	0	0	0	33729.7
56	Piston KWCA	4831.8	0	0	0	0	24183.8
57	Piston KWCA	4831.8	0	0	0	0	24183.8
58	Piston KWCA	4831.8	0	0	0	0	24183.8
59	Piston KWCA	4831.8	0	0	0	0	24183.8
60	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
61	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
62	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
63	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0

Tabel 3.5 (sambungan)

No	Part Name	Waktu Proses / Lot (detik)					
		DC1	DC2	DC3	DC4	DC5	DC6
64	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
65	Cyl Head KVBS	5205.0	28462.5	0	13142.5	0	0
66	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
67	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
68	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
69	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
70	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
71	Cyl Head KVYP	5205.0	28462.5	0	13142.5	0	0
72	Cyl Head KVLV	6521.2	0	29500.6	0	16380.6	0
73	Cyl Head KVLV	6521.2	0	29500.6	0	16380.6	0
74	Cyl Head KVLV	6521.2	0	29500.6	0	16380.6	0
75	Cyl Head KVLV	6521.2	0	29500.6	0	16380.6	0
76	Cyl Head KVLV	6521.2	0	29500.6	0	16380.6	0
77	Cyl Head KVLV	6521.2	0	29500.6	0	16380.6	0
78	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
79	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
80	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
81	Cyl Head KEHT	5308.4	26435.3	0	12994.3	0	0
82	Cyl Head KWCA	4265.6	0	17342.8	0	11322.8	0
83	Cyl Head KWCA	4265.6	0	17342.8	0	11322.8	0
84	Piston KVLV	5196.6	0	0	0	0	23242.2
85	Piston KVLV	5196.6	0	0	0	0	23242.2
86	Piston KVLV	5196.6	0	0	0	0	23242.2
87	Piston KVLV	5196.6	0	0	0	0	23242.2
88	Piston KVYP	5099.1	0	0	0	0	33729.7
89	Piston KVYP	5099.1	0	0	0	0	33729.7
90	Piston KVYP	5099.1	0	0	0	0	33729.7
91	Piston KVYP	5099.1	0	0	0	0	33729.7
92	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
93	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
94	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
95	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
96	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
97	Piston KWWF	5196.6	0	0	0	0	23242.2
98	Piston KWWF	5196.6	0	0	0	0	23242.2
99	Piston KWCA	4831.8	0	0	0	0	24183.8
100	Piston KWCA	4831.8	0	0	0	0	24183.8
101	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
102	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0

Tabel 3.5 (sambungan)

No	Part Name	Waktu Proses / Lot (detik)					
		DC1	DC2	DC3	DC4	DC5	DC6
103	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
104	Cyl Head KWWF	6521.2	0	29500.6	0	16380.6	0
105	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0
106	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0
107	Cyl Head KCJT	5308.4	26435.3	0	12994.3	0	0



BAB 4

PENGOLAHAN DATA DAN ANALISIS

4.1 Penyusunan Algoritma

Data penelitian ini diolah dengan menggunakan bahasa pemrograman Matlab. Bahasa pemrograman ini dipilih karena banyak memberi kemudahan untuk mengimplementasikan algoritma-algoritma yang banyak menggunakan operasi matriks.

Matlab adalah suatu bahasa pemrograman tingkat tinggi yang diperuntukkan untuk komputasi teknis. Matlab mengintegrasikan aspek komputasi, visualisasi, dan pemrograman dalam suatu lingkungan yang mudah dilakukan²³. Dalam memvisualisasikan sebuah objek, Matlab memiliki kemampuan merotasi obyek tanpa mengubah programnya. Konstruksi penyelesaian komputasi teknis dengan Matlab dapat dilakukan lebih cepat dibandingkan dengan bahasa pemrograman tradisional, seperti C, C++, dan Fortran. Matlab menyediakan fungsi-fungsi matematis untuk aljabar linear, statistik, optimasi, dan lainnya. Selain itu, Matlab juga menyediakan fitur-fitur dokumentasi dan integrasi algoritma berbasis Matlab dengan bahasa dan aplikasi lain, seperti C, C++, Fortran, Java, COM, dan Microsoft Excel. Bahasa Matlab memudahkan operasi-operasi vektor dan matriks yang merupakan dasar bagi permasalahan di bidang teknik dan ilmiah²⁴.

4.1.1 Langkah-Langkah Penyusunan Algoritma

Diagram alir penyusunan program algoritma *Differential Evolution* (DE) untuk menyelesaikan penjadwalan *job shop* pada PT AHM dapat dilihat pada gambar 2.5 dan program yang telah dibuat dapat dilihat pada lampiran 2. Prosedur penyusunan program tersebut dapat diuraikan sebagai berikut :

1. Inisialisasi populasi awal
 - Melakukan penyetelan pada generasi ke-0.

²³ Budi Santosa, *Matlab untuk Statistika & Teknik Optimasi - Aplikasi untuk Rekayasa & Bisnis*, Graha Ilmu, Yogyakarta, 2008, p.1.

²⁴ <http://www.mathworks.com/products/matlab/description1.html>, (accessed 5 November 2009)

Pada awal perhitungan sebelum generasi (iterasi) dimulai, dilakukan input parameter-parameter, yaitu ukuran populasi (NP), operator mutasi (F), dan operator pindah silang (CR). Ketiga parameter ini ditambah dengan jumlah iterasi yang akan digunakan ditentukan sesuai dengan hasil *Design of Experiments* untuk pemilihan parameter-parameter yang akan dibahas lebih lanjut. Parameter-parameter yang digunakan dengan mempertimbangkan hasil *DOE* tersebut yaitu: NP = 1070; F = 0,6; CR = 0,5; dan jumlah iterasi = 500.

- Menentukan populasi awal (populasi target awal)

Ukuran populasi menyatakan jumlah individu dalam populasi. Satu individu dinyatakan dengan satu kolom. Tiap individu memiliki 107 gen/dimensi (jumlah *job*/pesanan), dinyatakan dengan jumlah baris. Populasi awal dibentuk dengan setiap dimensi untuk setiap individu dicari secara acak dengan rumus:

$\text{batas_bawah} + (\text{batas_atas} - \text{batas_bawah}) \times \text{bilangan acak}$

Nilai-nilai input dari rumus tersebut yaitu -1 (batas bawah), 1 (batas atas), dan bilangan acak antara 0 dan 1. Karena populasi terdiri dari 1070 individu (nilai NP) dan setiap individu terdiri dari 107 dimensi, maka populasi awal merupakan matriks berukuran 107 x 1070. Pada algoritma DE ini, definisi individu dan vektor adalah sama. Setiap individu awal pada populasi awal ini diurutkan menjadi vektor permutasi sehingga dapat mempresentasikan urutan *job*.

- Menentukan vektor permutasi (urutan)

Nilai dimensi pertama hingga ke-107 setiap individu awal memiliki nilai yang berbeda-beda. Untuk setiap individu awal ini dilakukan pengurutan nilai dimensi dari yang terkecil hingga yang terbesar. Pengurutan tersebut akan menghasilkan vektor berdimensi 107 dengan nilai setiap dimensinya berupa indeks hasil pengurutan. Indeks-indeks tiap individu inilah yang nantinya akan menjadi urutan-urutan pengerjaan 107 pesanan sesuai data penelitian. Sebagai contoh, jika ada tiga *job* yang dikerjakan dan terdapat tiga dimensi pada suatu individu, yaitu 0,46; -0,74; dan 0,98; yang berturut-turut terdapat pada dimensi ke-1, dimensi ke-2, dan dimensi ke-3,

maka nilai dimensi tersebut pada vektor permutasi pada individu itu berturut-turut adalah 2, 1, dan 3. Artinya, urutan pengerjaan *job* yaitu 2-1-3. Proses ini berlaku untuk seluruh individu.

- Mengevaluasi setiap individu

Setelah mendapatkan vektor-vektor urutan *job* setiap individu, kemudian dibuat fungsi objektifnya (*makespan*). Pengevaluasian dilakukan dengan cara menghubungkan vektor tersebut dengan data waktu operasi sehingga fungsi objektif dapat dihitung. Pengevaluasian ini dimaksudkan untuk mengetahui individu pada generasi (iterasi) awal yang memiliki *makespan* yang selanjutnya akan diturunkan pada generasi selanjutnya.

- Memperbaharui generasi (iterasi)

Populasi individu pada iterasi awal akan berevolusi membentuk populasi individu iterasi baru. Individu-individu mengalami evolusi melalui serangkaian proses, yang dimulai dengan proses mutasi, proses pindah silang, dan proses penyeleksian. Jika generasi awal disimbolkan sebagai $t = 0$, maka iterasi baru disimbolkan sebagai $t = t + 1$.

2. Proses Mutasi

Proses mutasi merupakan proses utama dari algoritma DE karena merupakan langkah pertama individu dalam berevolusi. Sesuai nama algoritma (Differential Evolution), proses ini menekankan perbedaan nilai atau selisih dari dua vektor acak (vektor acak 1 dan vektor acak 2) yang memunculkan *difference vector*. Selanjutnya, *difference vector* tersebut akan dikalikan dengan operasi mutasi (F), yang hasilnya kemudian dijumlahkan dengan vektor target. Proses mutasi dianggap sangat penting sehingga penentuan parameter untuk mutasi pun sangat menentukan hasil yang diperoleh.

3. Proses pindah silang

Proses pindah silang merupakan rekombinasi antara individu target dengan individu mutan yang menghasilkan individu *trial*. Nilai dimensi individu *trial* ini sebagian berasal dari individu target dan sebagian lagi berasal dari individu mutan, dengan mempertimbangkan operator pindah silang (CR) dan bilangan acak. Jika bilangan acak r (antara 0 sampai 1) yang dihasilkan lebih kecil atau sama nilainya dengan CR maka yang berpeluang menjadi nilai dimensi ke- k

individu *trial* adalah nilai dimensi ke-k individu mutan, begitu pun sebaliknya. Nilai CR berada pada rentang 0 sampai 1.

4. Proses penyeleksian

Penyeleksian dilakukan antara individu target dan individu *trial*. Proses tersebut bertujuan untuk menentukan individu yang layak menjadi anggota pada generasi berikutnya. Proses penyeleksian dilakukan dengan membandingkan nilai fungsi objektif individu target dengan individu *trial*. Individu yang memiliki nilai *makespan* yang lebih kecil akan menjadi individu anggota populasi dari generasi berikutnya. Dengan adanya proses penyeleksian ini, maka populasi individu generasi berikutnya akan menjadi lebih baik.

5. Terminasi

Pada penelitian ini, kriteria terminasi yang digunakan adalah jumlah iterasi (generasi). Proses pembentukan iterasi baru akan terus berulang sampai jumlah iterasi yang telah ditentukan tercapai. Jumlah iterasi yang ditentukan adalah 500 iterasi berdasarkan percobaan yang telah dilakukan. Jadi, jika iterasi sudah mencapai 500 kali, maka program komputer secara otomatis berhenti melakukan perhitungan. Penentuan jumlah iterasi juga dipengaruhi oleh lamanya waktu perhitungan. Jumlah iterasi yang sangat besar memiliki kemungkinan untuk mencapai hasil yang optimal, tetapi waktu perhitungan yang dibutuhkan akan sangat lama.

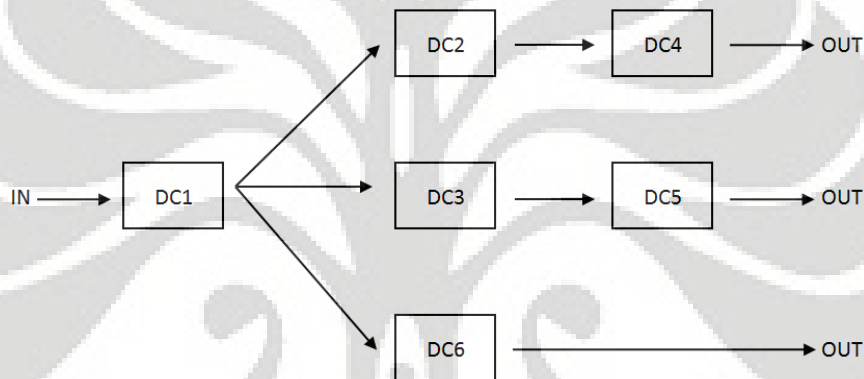
4.1.2 Verifikasi dan Validasi Program

Sebelum data penelitian diolah ke dalam program yang telah disusun, verifikasi dan validasi program harus dilakukan. Model program dikatakan telah terverifikasi apabila telah berjalan sesuai konseptual model, dimana ada perubahan output, yaitu minimum *makespan*. Untuk bagian-bagian selanjutnya, *makespan* yang dimaksud adalah *makespan* yang paling minimum. Artinya, *makespan* hanya dihitung jika waktu penyelesaian lebih cepat daripada waktu seharusnya diselesaikan.

Selanjutnya, dilakukan validasi program. Validasi terhadap program dilakukan dengan memasukkan data *dummy*. Hasil run program dengan data

dummy kemudian dibandingkan dengan perhitungan manual. Jika hasil keduanya sama, maka program telah tervalidasi.

Data *dummy* yang digunakan adalah 5 buah *job* dan setiap *job* terdiri dari 3 proses pemesinan untuk *cylinder head* dan 2 proses pemesinan untuk *piston*. Gambaran mengenai rute tersebut dapat dilihat pada gambar 4.1. Waktu operasi setiap rute untuk setiap jenis dapat dilihat pada tabel 4.1. Waktu operasi yang bernilai nol menunjukkan bahwa rute tersebut tidak dilewati oleh jenis yang bersangkutan. Jumlah operasi dan jumlah mesin ini sesuai dengan data penelitian. Parameter algoritma DE untuk validasi ini juga ditentukan dan dapat dilihat pada tabel 4.2.



Gambar 4.1 Rute Operasi Data *Dummy*

Tabel 4.1 Data *dummy* untuk Validasi

No	Part Name	Waktu proses/lot (detik)					
		DC1	DC2	DC3	DC4	DC5	DC6
1	Cyl Head KVLV	500	0	3000	0	1500	0
2	Cyl Head KEHT	500	2500	0	1500	0	0
3	Piston KVLV	500	0	0	0	0	2500
4	Piston KVYP	500	0	0	0	0	3000
5	Cyl Head KVBS	500	3000	0	1500	0	0

Tabel 4.2 Parameter yang Digunakan dalam Validasi

Parameter	Nilai
Ukuran Populasi	5
Jumlah Iterasi	1
Operasi Mutasi	0.5
Operator Pindah Silang	0.8

4.1.2.1 Hasil *Run* Program

Hasil *run* program menunjukkan urutan pengerjaan terbaik yaitu 5-4-1-2-3 dengan minimum *makespan* yaitu 7500 detik.

4.1.2.2 Hasil Perhitungan Manual

Langkah-langkah perhitungan manual untuk validasi program adalah sebagai berikut:

1. Melihat populasi target yang didapat dari program.

Populasi target, yang merupakan populasi awal, berisi bilangan acak antara -1 dan 1 sesuai dengan rumus yang telah dibuat pada program. Matriks populasi target untuk data *dummy* ini berukuran 5 x 5 (ukuran populasi x jumlah *job*). Matriks populasi target ini adalah matriks yang memiliki populasi sebanyak 5 individu atau 5 vektor. Jadi, 1 kolom merepresentasikan 1 individu. Tiap kolom memiliki 5 baris, berarti 1 individu terdiri dari 5 gen (*job*).

Tabel 4.3 Populasi Target (Hasil *Run* Program)

Individu				
1	2	3	4	5
-0.30615	0.29059	-0.75508	-0.62285	-0.39178
-0.70409	-0.38776	-0.73915	0.36699	-0.58823
-0.65946	-0.41843	-0.54242	0.08297	-0.14549
-1.05920	-0.13897	-0.78615	0.04281	-1.15151
-0.15278	0.10055	-0.81393	0.88363	-0.30155

2. Melakukan pengurutan (permutasi) pada setiap individu dari populasi target.

Urutan setiap individu didapatkan dengan cara mengurutkan bilangan acak setiap kolom pada tabel 4.3. dari yang terkecil hingga terbesar. Sebagai contoh, pada kolom pertama, urutan bilangan acak dari yang terkecil hingga terbesar adalah -1,05920; -0,70409; -0,65946; -0,30615; dan -0,15278. Bilangan-bilangan tersebut berturut-turut terdapat pada kolom ke 4, 2, 3, 1, dan 5. Jadi, urutan pengerjaan *job* pada individu (kolom) tersebut adalah 4-2-3-1-5. Setiap individu memiliki urutan *job* masing-masing. Tabel 4.4 menyatakan permutasi (urutan pengerjaan *job*) populasi target untuk setiap individu.

Tabel 4.4 Permutasi Populasi Target

Individu				
1	2	3	4	5
4	3	5	1	4
2	2	4	4	2
3	4	1	3	1
1	5	2	2	5
5	1	3	5	3

3. Setelah didapat urutan *job* tersebut, maka untuk setiap individu dicari fungsi objektifnya, yaitu nilai *makespan*.

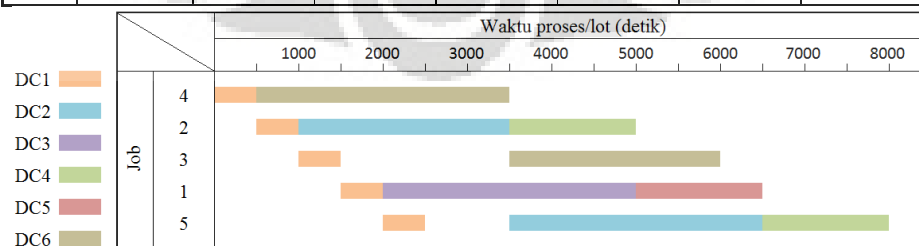
Prosesnya adalah sebagai berikut:

- Mengeset waktu mula-mula adalah sama dengan nol.
- Untuk setiap individu, sesuai urutan *job* masing-masing, dicari waktu penyelesaian untuk setiap *job* dengan terus menambahkan waktu proses di setiap proses yang bersangkutan dengan waktu penyelesaian *predecessor*.
- Kemudian waktu penyelesaian setiap *job* akan dicari nilai *makespan* yang terkecil.

Tabel-tabel di bawah ini menunjukkan proses perhitungan *makespan* untuk setiap individu dalam populasi.

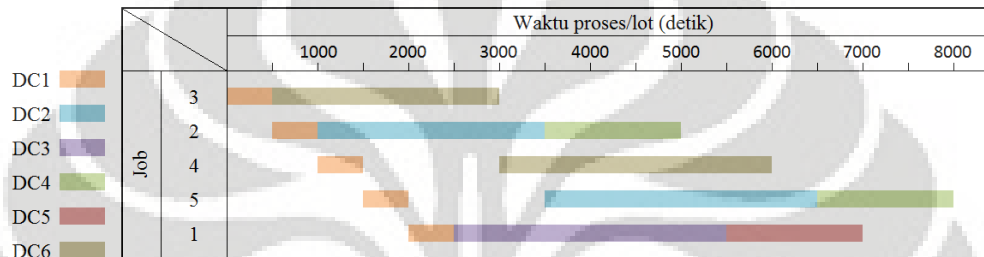
Tabel 4.5 Perhitungan Waktu Produksi setiap *Job* Individu 1 Populasi Target

Job	Waktu proses/lot (detik)						<i>Makespan</i>
	DC1	DC2	DC3	DC4	DC5	DC6	
4	500	0	0	0	0	3000	8000
2	500	2500	0	1500	0	0	
3	500	0	0	0	0	2500	
1	500	0	3000	0	1500	0	
5	500	3000	0	1500	0	0	

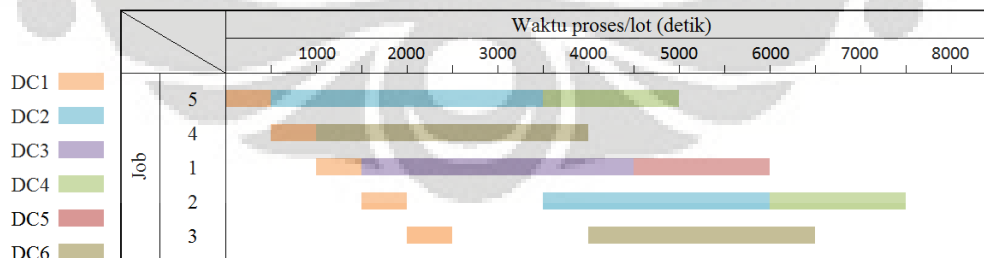
**Gambar 4.2** Gant Chart Individu 1 Populasi Target

Tabel 4.6 Perhitungan Waktu Produksi setiap *Job* Individu 2 Populasi Target

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
3	500	0	0	0	0	2500	8000
2	500	2500	0	1500	0	0	
4	500	0	0	0	0	3000	
5	500	3000	0	1500	0	0	
1	500	0	3000	0	1500	0	

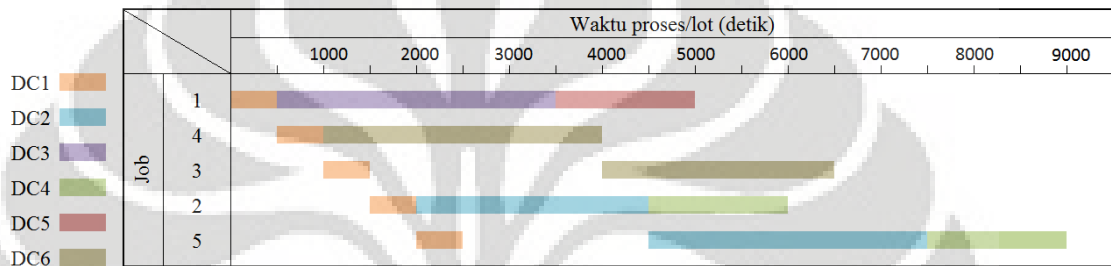
**Gambar 4.3** Gant Chart Individu 2 Populasi Target**Tabel 4.7** Perhitungan Waktu Produksi setiap *Job* Individu 3 Populasi Target

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
5	500	3000	0	1500	0	0	7500
4	500	0	0	0	0	3000	
1	500	0	3000	0	1500	0	
2	500	2500	0	1500	0	0	
3	500	0	0	0	0	2500	

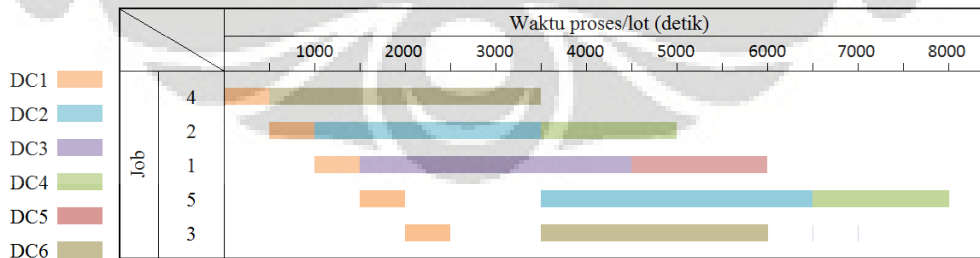
**Gambar 4.4** Gant Chart Individu 3 Populasi Target

Tabel 4.8 Perhitungan Waktu Produksi setiap *Job* Individu 4 Populasi Target

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
1	500	0	3000	0	1500	0	9000
4	500	0	0	0	0	3000	
3	500	0	0	0	0	2500	
2	500	2500	0	1500	0	0	
5	500	3000	0	1500	0	0	

**Gambar 4.5** Gant Chart Individu 4 Populasi Target**Tabel 4.9** Perhitungan Waktu Produksi setiap *Job* Individu 5 Populasi Target

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
4	500	0	0	0	0	3000	8000
2	500	2500	0	1500	0	0	
1	500	0	3000	0	1500	0	
5	500	3000	0	1500	0	0	
3	500	0	0	0	0	2500	

**Gambar 4.6** Gant Chart Individu 5 Populasi Target

4. Setelah didapat nilai *makespan* seluruh individu, maka dipilih individu yang memiliki *makespan* yang terkecil, yaitu individu 3. Individu 3 memiliki nilai *makespan* sebesar 7500 detik dengan urutan pengerjaan *job* 5-4-1-2-3. Jadi, individu 3 pada populasi target akan menjadi vektor target untuk tahap selanjutnya.
5. Lalu dibuatlah populasi mutan dengan input vektor target dan dua vektor acak. Penjelasannya adalah sebagai berikut:
 - Seperti telah dibahas sebelumnya, vektor target didapat dari individu populasi target yang memiliki *makespan* terkecil, yaitu individu 3. Semua kolom (individu) pada vektor target berisi bilangan-bilangan yang sama, yaitu individu 3 pada populasi target.

Tabel 4.10 Vektor Target

Individu				
1	2	3	4	5
-0.75508	-0.75508	-0.75508	-0.75508	-0.75508
-0.73915	-0.73915	-0.73915	-0.73915	-0.73915
-0.54242	-0.54242	-0.54242	-0.54242	-0.54242
-0.78615	-0.78615	-0.78615	-0.78615	-0.78615
-0.81393	-0.81393	-0.81393	-0.81393	-0.81393

- Kemudian dipilih dua vektor acak yang setiap kolomnya (individu) berasal dari populasi target dengan suatu aturan yaitu antara vektor target, vektor acak 1, dan vektor acak 2 pada kolom tertentu tidak boleh sama. Sebagai contoh, pada kolom pertama vektor target, vektor acak 1, dan vektor acak 2, angkanya berturut-turut berasal dari populasi target pada kolom 3,1, dan 2. Ketiga kolom tersebut berbeda, tidak boleh ada yang sama.

Tabel 4.11 Vektor Acak 1 (Hasil *Run* Program)

Individu				
1	2	3	4	5
0.27500	-0.45160	-0.62285	0.29059	0.27500
0.43712	0.13527	0.36699	-0.38776	0.43712
-0.15112	-0.94498	0.08297	-0.41843	-0.15112
-0.50328	0.22743	0.04281	-0.13897	-0.50328
0.87197	-0.15278	0.88363	0.10055	0.87197

Tabel 4.12 Vektor Acak 2 (Hasil *Run* Program)

Individu				
1	2	3	4	5
-0.62285	0.27500	0.27500	0.27500	-0.45160
0.36699	0.43712	0.43712	0.43712	0.13527
0.08297	-0.15112	-0.15112	-0.15112	-0.94498
0.04281	-0.50328	-0.50328	-0.50328	0.22743
0.88363	0.87197	0.87197	0.87197	-0.15278

- Setelah didapat vektor target, vektor acak 1, dan vektor acak 2, kemudian terbentuk populasi mutan (tabel 4.13). Populasi ini didapatkan dengan menggunakan rumus:

populasi mutan = vektor target + (vektor acak 1 – vektor acak 2)*konstanta mutasi (F).

Rumus ini berlaku untuk setiap gen. Jadi, sebagai contoh, pada baris pertama kolom pertama populasi mutan, yaitu 0,30615 didapatkan dari $(-0.75508) + (0.27500 - (-0.62285)) * 0,5$. Angka-angka $-0,75505$; $0,27500$; dan $-0,62285$ secara berurutan adalah angka-angka di baris pertama kolom pertama dari vektor target, vektor acak 1, dan vektor acak 2. Sedangkan $0,5$ adalah nilai dari operator mutasi (konstanta mutasi).

Tabel 4.13 Populasi Mutan

Individu				
1	2	3	4	5
-0.30615	-1.11838	-1.20401	-0.74728	-0.39178
-0.70409	-0.89007	-0.77421	-1.15159	-0.58823
-0.65946	-0.93935	-0.42538	-0.67608	-0.14549
-1.05920	-0.42079	-0.51310	-0.60400	-1.15151
-0.81975	-1.32630	-0.80810	-1.19964	-0.30155

- Setelah populasi mutan didapat, maka dibentuklah populasi *trial*. Setiap gen individu populasi *trial* ini berasal dari gen individu populasi target atau gen individu populasi mutan. Untuk memilih antara gen individu populasi target atau populasi mutan, digunakan bilangan acak antara 0 dan 1, dan operator pindah silang (CR) sebesar 0,8. Jika bilangan acak yang keluar bernilai kurang

dari atau sama dengan 0,8; maka nilai suatu gen pada populasi *trial* berasal dari gen yang bersangkutan dari populasi mutan. Begitu pun sebaliknya, jika lebih dari 0,8; maka berasal dari populasi target. Tabel 4.14 menunjukkan populasi *trial*. Gen-gen yang bercetak tebal pada tabel tersebut adalah gen-gen yang berasal dari populasi mutan.

Tabel 4.14 Populasi *Trial*

Individu				
1	2	3	4	5
-0.30615	0.29059	-1.20401	-0.74728	-0.39178
-0.70409	-0.89007	-0.77421	0.36699	-0.58823
-0.65946	-0.93935	-0.42538	-0.67608	-0.14549
-1.05920	-0.42079	-0.51310	-0.60400	-1.15151
-0.15278	0.10055	-0.80810	0.88363	-0.30155

7. Melakukan pengurutan (permutasi) pada setiap individu dari populasi *trial*. Caranya sama dengan pengurutan pada setiap individu dari populasi target. Permutasi populasi *trial* dapat dilihat pada tabel 4.15.

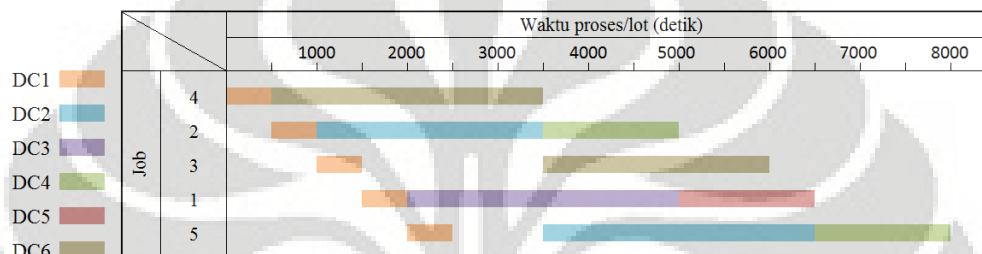
Tabel 4.15 Permutasi Populasi *Trial*

Individu				
1	2	3	4	5
4	3	1	1	4
2	2	5	3	2
3	4	2	4	1
1	5	4	2	5
5	1	3	5	3

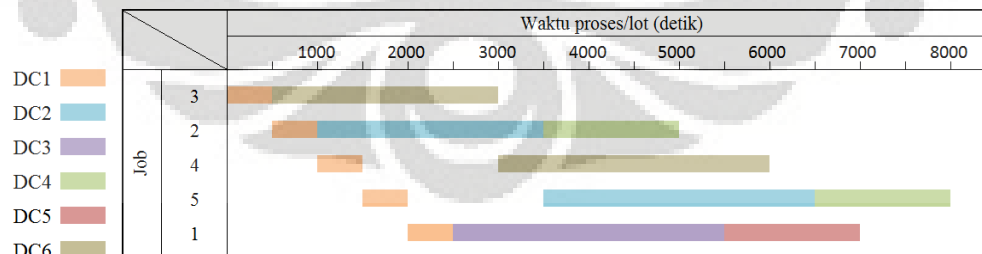
8. Setelah didapatkan urutan *job* dari setiap individu populasi *trial*, maka dari setiap individu tadi, dicari *makespan* masing-masing (tabel 4.16 sampai tabel 4.20).

Tabel 4.16 Perhitungan Waktu Produksi setiap *Job* Individu 1 Populasi *Trial*

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
4	500	0	0	0	0	3000	8000
2	500	2500	0	1500	0	0	
3	500	0	0	0	0	2500	
1	500	0	3000	0	1500	0	
5	500	3000	0	1500	0	0	

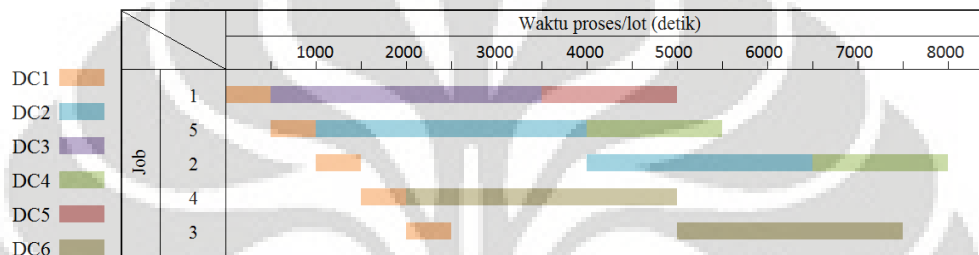
**Gambar 4.7** Gant Chart Individu 1 Populasi *Trial***Tabel 4.17** Perhitungan Waktu Produksi setiap *Job* Individu 2 Populasi *Trial*

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
3	500	0	0	0	0	2500	8000
2	500	2500	0	1500	0	0	
4	500	0	0	0	0	3000	
5	500	3000	0	1500	0	0	
1	500	0	3000	0	1500	0	

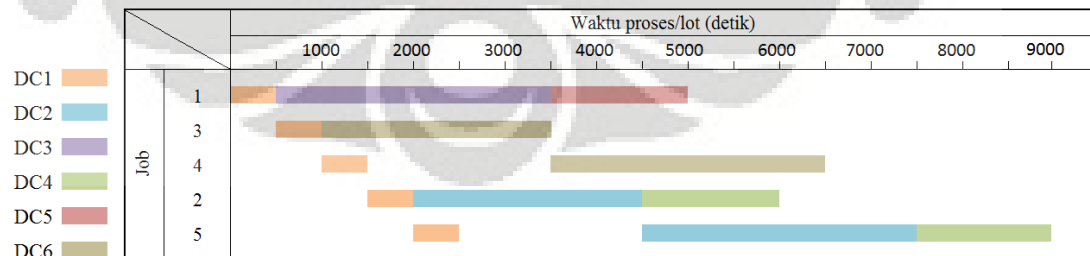
**Gambar 4.8** Gant Chart Individu 2 Populasi *Trial*

Tabel 4.18 Perhitungan Waktu Produksi setiap *Job* Individu 3 Populasi *Trial*

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
1	500	0	3000	0	1500	0	8000
5	500	3000	0	1500	0	0	
2	500	2500	0	1500	0	0	
4	500	0	0	0	0	3000	
3	500	0	0	0	0	2500	

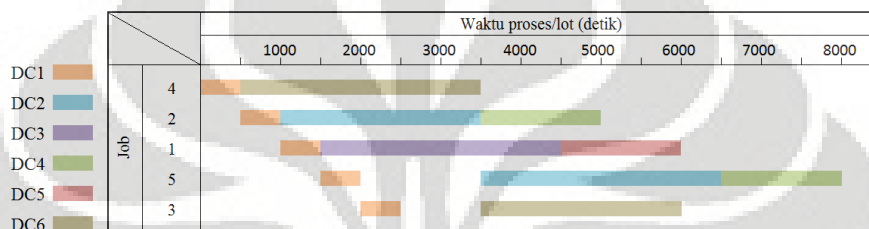
**Gambar 4.9** Gant Chart Individu 3 Populasi *Trial***Tabel 4.19** Perhitungan Waktu Produksi setiap *Job* Individu 4 Populasi *Trial*

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
1	500	0	3000	0	1500	0	9000
3	500	0	0	0	0	2500	
4	500	0	0	0	0	3000	
2	500	2500	0	1500	0	0	
5	500	3000	0	1500	0	0	

**Gambar 4.10** Gant Chart Individu 4 Populasi *Trial*

Tabel 4.20 Perhitungan Waktu Produksi setiap *Job* Individu 5 Populasi *Trial*

Job	Waktu proses/lot (detik)						Makespan
	DC1	DC2	DC3	DC4	DC5	DC6	
4	500	0	0	0	0	3000	8000
2	500	2500	0	1500	0	0	
1	500	0	3000	0	1500	0	
5	500	3000	0	1500	0	0	
3	500	0	0	0	0	2500	

**Gambar 4.11** Gant Chart Individu 5 Populasi *Trial*

9. Kemudian dilakukan tahap seleksi. Setelah didapat nilai *makespan* setiap individu pada populasi *trial*, nilai-nilai tersebut dibandingkan dengan nilai *makespan* setiap individu populasi target. Perbandingan tersebut dapat dilihat pada tabel 4.21. Sebagai contoh, nilai *makespan* individu 1 populasi target dibandingkan dengan nilai *makespan* individu 1 populasi *trial*. Jika individu 1 populasi target memiliki nilai *makespan* yang lebih kecil dari individu 1 populasi *trial*, maka nilai populasi target, *makespan*, dan urutan (permutasi) untuk iterasi selanjutnya (iterasi 2) berturut-turut diambil dari individu 1 populasi target, *makespan* individu 1 populasi target, dan permutasi individu 1 populasi target, begitupun sebaliknya jika nilai *makespan* target lebih besar atau sama dengan nilai *makespan* populasi *trial*.

Tabel 4.21 Perbandingan antara *Makespan* Populasi Target dan Populasi *Trial*

Total <i>Makespan</i> Populasi Target (detik)	Total <i>Makespan</i> Populasi <i>Trial</i> (detik)	Asal Populasi Target Iterasi Selanjutnya
8000	8000	Populasi <i>Trial</i>
8000	8000	Populasi <i>Trial</i>
7500	8000	Populasi Target
9000	9000	Populasi <i>Trial</i>
8000	8000	Populasi <i>Trial</i>

10. Dari langkah sebelumnya, populasi target iterasi 2 terbentuk. Dari populasi target ini, proses-proses mulai dari pembentukan vektor target akan berulang sampai terbentuk populasi target kembali untuk iterasi 3. Tabel 4.22 menunjukkan populasi target baru dimana populasi ini akan membentuk vektor target, dan seterusnya pada iterasi 2. Tabel 4.23 menunjukkan permutasi (urutan pengerjaan *job*) dari populasi target baru.

Tabel 4.22 Populasi Target Baru

Individu				
1	2	3	4	5
-0.30615	0.29059	-0.75508	-0.74728	-0.39178
-0.70409	-0.89007	-0.73915	0.36699	-0.58823
-0.65946	-0.93935	-0.54242	-0.67608	-0.14549
-1.05920	-0.42079	-0.78615	-0.60400	-1.15151
-0.15278	0.10055	-0.81393	0.88363	-0.30155

Tabel 4.23 Permutasi Populasi Target Baru

Individu				
1	2	3	4	5
4	3	5	1	4
2	2	4	3	2
3	4	1	4	1
1	5	2	2	5
5	1	3	5	3

11. Karena proses validasi ini hanya menggunakan satu iterasi, jadi program berakhir hanya sampai pada tahap seleksi . Jadi, dari setiap perbandingan nilai total *makespan* terkecil setiap gen antara populasi target dan populasi *trial*, langsung diambil gen yang memiliki total *makespan* terkecil. Nilai inilah yang menjadi solusi penyelesaian data *dummy* untuk validasi ini. Untuk lebih jelasnya dapat dilihat pada tabel 4.24.

Tabel 4.24 Hasil Terbaik Perhitungan Data *Dummy*

Total <i>Makespan</i> Populasi Target (detik)	Total <i>Makespan</i> Populasi <i>Trial</i> (detik)	Total Minimum <i>Makespan</i> Populasi Target Baru (detik)
8000	8000	8000
8000	8000	8000
7500	8000	7500
9000	9000	9000
8000	8000	8000

Dari tabel terlihat bahwa dari setiap perbandingan total *makespan* populasi target dan populasi *trial*, nilai total *makespan* yang terkecil adalah 7500 detik, nilai tersebut terletak pada individu 3 pada nilai-nilai total *makespan* populasi target yang baru. Jadi, sesuai tabel 4.23, urutan pengerjaan *job* adalah 5 – 4 – 1 – 2 – 3 (individu 3 pada permutasi populasi target baru).

12. Maka didapatkan nilai *makespan* terbaik, yaitu 7500 detik dengan urutan pengerjaan *job* 5 – 4 – 1 – 2 – 3.
13. Nilai *makespan* yang dihasilkan dari perhitungan manual sama dengan hasil keluaran program, maka program telah tervalidasi.

4.2 Input

4.2.1 Input Data

Input data yang dimasukkan pada program yaitu penjabaran lengkap penjadwalan jalur *low pressure die casting* PT AHM yang ada pada tabel 3.6, yaitu rute dan waktu proses untuk 107 *job*. Data ini akan dibuat ke dalam suatu matriks yang berukuran 107 x 6, artinya terdapat 107 *job* (baris) dan 6 rute (kolom). Untuk kolom 1 hingga kolom 6 berisi waktu produksi (detik) setiap operasi pada setiap *job*.

4.2.2 Input Parameter

Untuk menentukan nilai parameter-parameter algoritma DE yang digunakan untuk pengolahan data, terlebih dahulu dilakukan *Design of Experiments* (DOE) terhadap empat parameter untuk melihat kombinasi terbaik dari keempat parameter tersebut. Empat parameter tersebut adalah ukuran populasi, jumlah generasi (iterasi), operator mutasi, dan operator pindah silang. Untuk setiap

percobaan didapatkan *makespan* sebagai output. DOE dilakukan untuk mendapatkan hasil yang maksimum dalam mendapatkan minimum *makespan*. Skenario DOE ini akan diuji untuk 4 faktor yang akan berfungsi sebagai faktor dan setiap faktor terdiri dari 3 level, yaitu level rendah, sedang, dan tinggi. DOE yang dilakukan berjenis *full factorial design*. Jadi, ada 81 (3^4) kombinasi yang mungkin untuk dilakukan percobaan. Setiap kombinasi akan diulang sebanyak 5 kali untuk melihat keakuratan data yang dihasilkan. Jadi, total percobaan yaitu $81 \times 5 = 405$ percobaan.

Nilai-nilai setiap level untuk setiap faktor untuk DOE dapat dilihat pada tabel 4.25 Percobaan ini sengaja dirancang untuk ukuran populasi dan jumlah iterasi dengan nilai yang kecil (berkisar dari 15 sampai 45) agar waktu komputasi selama percobaan tidak terlalu lama.

Tabel 4.25 Skenario untuk DOE

Faktor	Level 1 (rendah)	Level 2 (sedang)	Level 3 (tinggi)
Ukuran Populasi (NP)	15	30	45
Jumlah Iterasi (JI)	15	30	45
Operator Mutasi (F)	0.4	0.6	0.8
Operator Pindah Silang (CR)	0.5	0.7	0.9

Skenario DOE untuk parameter diuji dengan menggunakan program MINITAB 15 dengan $\alpha = 5\%$. Setelah dilakukan percobaan sebanyak 405 kali dengan output *makespan* (hasil percobaan ini dapat dilihat pada lampiran 3). Dengan melihat hasil skenario parameter yang telah dilakukan, maka dilakukan analisis yang akan dibahas pada bagian analisis skenario parameter, dan dibuat suatu kombinasi parameter yang dianggap merupakan kombinasi terbaik sesuai hasil skenario, terdapat pada tabel 4.26 Kombinasi parameter inilah yang menjadi input parameter untuk program algoritma DE dan diharapkan dapat memberikan kualitas solusi terbaik untuk masalah penjadwalan ini dengan fungsi tujuan meminimumkan *makespan*.

Tabel 4.26 Kombinasi Parameter Terbaik

Faktor	Nilai
Ukuran Populasi (NP)	1070
Jumlah Iterasi (JI)	500
Operator Mutasi (F)	0.6
Operator Pindah Silang (CR)	0.5

4.3 Pengolahan Data dan Hasil

Program untuk pengolahan data penelitian ini dijalankan dengan spesifikasi komputer, yaitu Intel (R) Core(TM)2 Duo CPU T6600 @ 2.20GHz 2.20GHz, 2.00 GB of RAM. *Script M File* program untuk perhitungan jadwal produksi jalur LPDC PT AHM dapat dilihat pada lampiran 1. dan *Script M File* program untuk mendapatkan solusi (usulan) jadwal dapat dilihat pada lampiran 2.

4.3.1 Hasil Penjadwalan Jalur LPDC PT AHM

Tabel 4.27 Hasil Perhitungan Jadwal Jalur LPDC PT AHM

Hasil Penjadwalan Produksi di Jalur LPDC PT AHM										
Urutan Rute Proses Pengerjaan	1	2	3	4	5	6	7	8	9	10
	11	12	13	14	15	16	17	18	19	20
	21	22	23	24	25	26	27	28	29	30
	31	32	33	34	35	36	37	38	39	40
	41	42	43	44	45	46	47	48	49	50
	51	52	53	54	55	56	57	58	59	60
	61	62	63	64	65	66	67	68	69	70
	71	72	73	74	75	76	77	78	79	80
	81	82	83	84	85	86	87	88	89	90
	91	92	93	94	95	96	97	98	99	100
	101	102	103	104	105	106	107	-	-	-
<i>makespan</i> (detik)	1253272.8									

4.3.2 Hasil Penjadwalan dengan Algoritma DE

Setelah parameter-parameter dari permasalahan penjadwalan ini ditentukan, maka program yang telah dibuat di-*run* sebanyak tiga kali. Hasil yang terbaik yaitu hasil *run* yang memiliki nilai *makespan* terkecil. Tabel 4.28 sampai Tabel 4.30 berikut ini menunjukkan hasil *run* sebanyak tiga kali.

Tabel 4.28 Hasil Run 1

Hasil Run 1														
Urutan Rute Proses Pengerjaan	67	19	85	81	29	47	45	78	8	39	56	89	106	
	97	101	49	105	65	99	31	32	12	77	17	3	79	
	91	44	35	73	55	21	30	43	69	104	107	102	41	
	46	74	33	15	62	70	34	50	87	23	36	72	10	
	95	98	80	9	27	13	48	1	11	96	84	63	6	
	26	54	83	59	25	2	5	75	16	18	64	60	4	
	7	38	76	20	82	88	37	14	52	90	66	94	61	
	57	53	100	92	103	28	24	51	68	93	22	58	71	
	86	40	42	-	-	-	-	-	-	-	-	-	-	-
	<i>makespan</i> (detik)	1207624.4												
Waktu Komputasi (detik)	538.8130													

Tabel 4.29 Hasil Run 2

Hasil Run 2														
Urutan Rute Proses Pengerjaan	27	61	8	30	15	81	45	69	80	20	42	105	93	
	38	79	54	32	90	6	53	106	74	41	100	83	5	
	37	33	103	39	16	65	49	72	31	48	2	46	44	
	75	35	7	89	84	91	94	101	102	86	107	21	78	
	18	47	24	62	34	56	104	19	63	96	67	52	28	
	85	26	76	12	97	11	60	51	1	88	55	64	23	
	40	66	98	3	87	29	70	99	36	82	10	77	14	
	13	59	92	22	71	9	68	25	4	95	73	50	57	
	17	58	43	-	-	-	-	-	-	-	-	-	-	-
	<i>makespan</i> (detik)	1207624.4												
Waktu Komputasi (detik)	544.4701													

Tabel 4.30 Hasil Run 3

Hasil Run 3														
Urutan Rute Proses Pengerjaan	64	45	4	107	88	16	66	79	43	30	82	10	41	
	90	96	33	13	75	100	5	29	72	6	42	34	54	
	15	1	19	85	95	47	46	80	17	26	49	55	44	
	97	78	38	18	105	62	98	57	73	21	51	102	93	
	58	32	12	74	22	27	83	53	31	103	106	94	63	
	92	67	71	56	52	7	39	3	89	50	25	104	2	
	99	36	61	14	28	8	101	59	91	86	48	23	40	
	69	11	37	20	24	60	84	9	77	68	35	65	87	
	70	76	81	-	-	-	-	-	-	-	-	-	-	-
	<i>makespan</i> (detik)	1207624.4												
Waktu Komputasi (detik)	572.4354													

Urutan terbaik =

Columns 1 through 13

67 19 85 81 29 47 45 78 8 39 56 89 106

Columns 14 through 26

97 101 49 105 65 99 31 32 12 77 17 3 79

Columns 27 through 39

91 44 35 73 55 21 30 43 69 104 107 102 41

Columns 40 through 52

46 74 33 15 62 70 34 50 87 23 36 72 10

Columns 53 through 65

95 98 80 9 27 13 48 1 11 96 84 63 6

Columns 66 through 78

26 54 83 59 25 2 5 75 16 18 64 60 4

Columns 79 through 91

7 38 76 20 82 88 37 14 52 90 66 94 61

Columns 92 through 104

57 53 100 92 103 28 24 51 68 93 22 58 71

Columns 105 through 107

86 40 42

Nilai makespan awal = 1.2076e+006

Waktu Komputasi : 528.8310 detik

Gambar 4.12 Tampilan Hasil *Run 1*

Urutan terbaik =

Columns 1 through 13

27 61 8 30 15 81 45 69 80 20 42 105 93

Columns 14 through 26

38 79 54 32 90 6 53 106 74 41 100 83 5

Columns 27 through 39

37 33 103 39 16 65 49 72 31 48 2 46 44

Columns 40 through 52

75 35 7 89 84 91 94 101 102 86 107 21 78

Columns 53 through 65

18 47 24 62 34 56 104 19 63 96 67 52 28

Columns 66 through 78

85 26 76 12 97 11 60 51 1 88 55 64 23

Columns 79 through 91

40 66 98 3 87 29 70 99 36 82 10 77 14

Columns 92 through 104

13 59 92 22 71 9 68 25 4 95 73 50 57

Columns 105 through 107

17 58 43

Nilai makespan awal = 1.2076e+006

Waktu Komputasi : 544.4701 detik

Gambar 4.13 Tampilan Hasil *Run 2*

Urutan terbaik =

Columns 1 through 13

64 45 4 107 88 16 66 79 43 30 82 10 41

Columns 14 through 26

90 96 33 13 75 100 5 29 72 6 42 34 54

Columns 27 through 39

15 1 19 85 95 47 46 80 17 26 49 55 44

Columns 40 through 52

97 78 38 18 105 62 98 57 73 21 51 102 93

Columns 53 through 65

58 32 12 74 22 27 83 53 31 103 106 94 63

Columns 66 through 78

92 67 71 56 52 7 39 3 89 50 25 104 2

Columns 79 through 91

99 36 61 14 28 8 101 59 91 86 48 23 40

Columns 92 through 104

69 11 37 20 24 60 84 9 77 68 35 65 87

Columns 105 through 107

70 76 81

Nilai makespan awal = 1.2076e+006

Waktu Komputasi : 572.4354 detik

Gambar 4.14 Tampilan Hasil *Run 3*

4.4 Analisis

4.4.1. Analisis Skenario Parameter

ANOVA (*Analysis of Variance*) adalah salah satu teknik yang memungkinkan untuk menguji perbedaan pengaruh faktor dari sampel yang diambil. Dari tabel ANOVA, dapat diketahui faktor-faktor yang berpengaruh dari model terhadap output yang diamati melalui indikator *p-value*. Tingkat kepercayaan yang digunakan adalah 95% sehingga $\alpha = 5\%$ (0,05). Jika nilai *p-value* pada suatu faktor $\leq 0,05$; maka faktor yang bersangkutan berpengaruh signifikan terhadap output yang diamati secara statistik atau tidak menerima hipotesis nol (H_0). Sedangkan jika *p-value* $> 0,05$; maka faktor tersebut tidak memiliki pengaruh terhadap output yang diamati atau menerima hipotesis nol.

Melalui ANOVA yang terdapat pada tabel 4.31, akan dijelaskan pengaruh setiap faktor dan interaksi antarfaktor terhadap output, yaitu perubahan *makespan*.

Tabel 4.31 ANOVA untuk *Makespan*

Source	DF	Seq SS	Adj SS	Adj MS	F	P
NP	2	2996941553	2996941553	1498470777	80.85	0.000
JI	2	480813750	480813750	240406875	12.97	0.000
F	2	3718426533	3718426533	1859213267	100.31	0.000
CR	2	15050397839	15050397839	7525198920	406.02	0.000
NP*JI	4	237921616	237921616	59480404	3.21	0.013
NP*F	4	1060189433	1060189433	265047358	14.30	0.000
NP*CR	4	1559449393	1559449393	389862348	21.03	0.000
JI*F	4	526785689	526785689	131696422	7.11	0.000
JI*CR	4	554202603	554202603	138550651	7.48	0.000
F*CR	4	1498680032	1498680032	374670008	20.22	0.000
NP*JI*F	8	162986642	162986642	20373330	1.10	0.363
NP*JI*CR	8	80418678	80418678	10052335	0.54	0.824
NP*F*CR	8	681508935	681508935	85188617	4.60	0.000
JI*F*CR	8	556037865	556037865	69504733	3.75	0.000
NP*JI*F*CR	16	651877325	651877325	40742333	2.20	0.005
Error	324	6005017042	6005017042	18534003		
Total	404	35821654928				

Penjelasan faktor di atas adalah sebagai berikut:

- NP (ukuran populasi), JI (jumlah iterasi), F (operator mutasi), dan CR (operator pindah silang) memiliki nilai *p-value* $\leq 0,05$. Artinya keempat faktor ini memiliki pengaruh yang signifikan terhadap perubahan output (*makespan*). Selain dari *p-value*, nilai F_{ANOVA} (nilai F dari ANOVA) juga dapat digunakan sebagai indikator seberapa besar pengaruh suatu faktor terhadap output

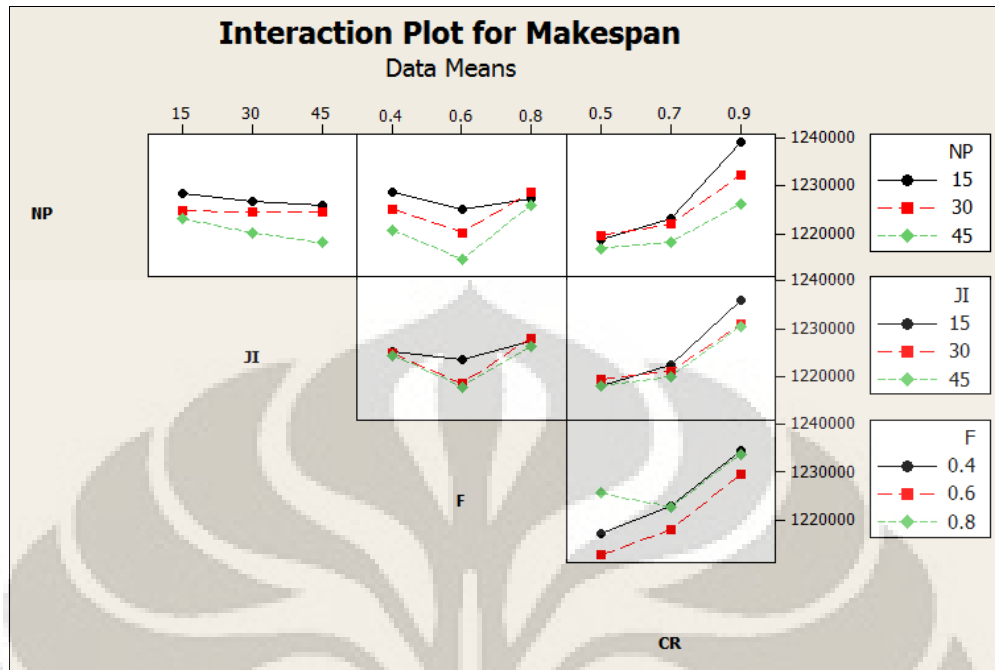
dibandingkan dengan faktor lainnya. Semakin besar nilai F_{ANOVA} , maka semakin besar pengaruh faktor yang bersangkutan terhadap output, dan sebaliknya. Jadi, jika dilihat dari nilai F_{ANOVA} , maka faktor yang paling berpengaruh dari keempat faktor adalah operator pindah silang, kemudian secara berurutan diikuti oleh operator mutasi, ukuran populasi, dan jumlah iterasi.

- Interaksi antara NP – JI, NP – F, NP – CR, JI – F, JI – CR, dan F – CR berpengaruh signifikan terhadap output.
- Interaksi antara NP – JI – F dan NP – JI – CR tidak berpengaruh terhadap output.
- Interaksi antara NP – F – CR dan JI – F – CR berpengaruh signifikan terhadap output.
- Interaksi antara NP – JI – F – CR berpengaruh signifikan terhadap output.

Jika ANOVA menunjukkan interaksi antar faktor, gambar 4.4 menunjukkan secara detail bagaimana interaksi antara 2 faktor pada setiap level yang ada. Selain itu, dari gambar ini, dapat dilihat kombinasi terbaik 2 faktor yang menghasilkan output terbaik. Sedangkan untuk interaksi antara 3 faktor atau lebih tidak dapat ditunjukkan dalam gambar.

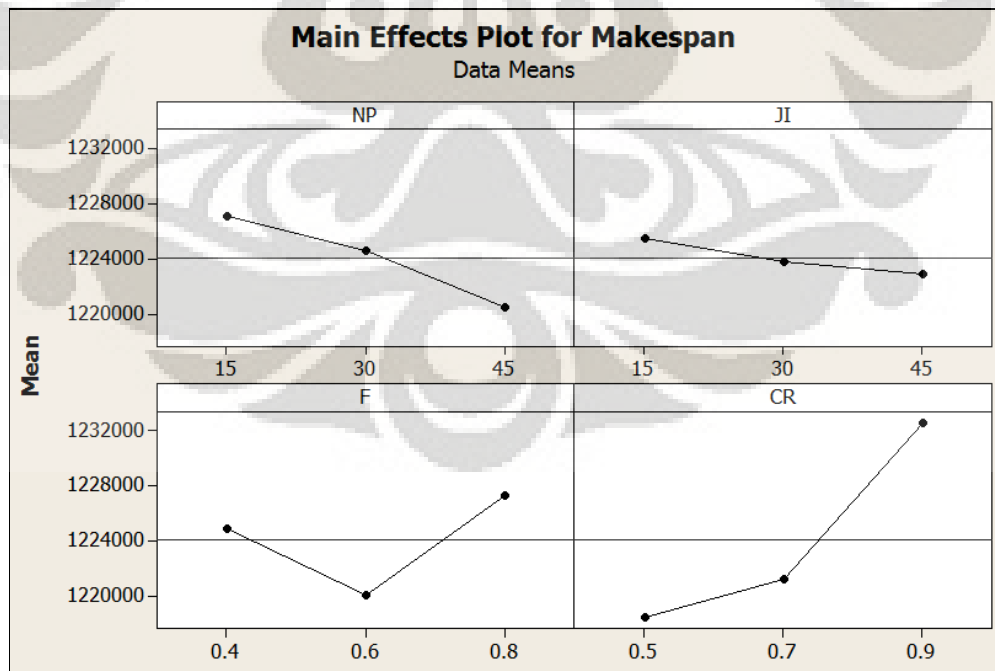
Penjelasan dari gambar 4.5 adalah sebagai berikut:

- Interaksi antara NP – JI berpengaruh terhadap output tetapi tidak signifikan. Artinya jika salah satu parameter diubah, sedangkan parameter yang lainnya tidak berubah, hasilnya tidak akan berubah secara signifikan. Hal ini terlihat pada gambar, yaitu kemiringan garis yang bersangkutan hampir mendatar.
- Interaksi antara NP – F, NP – CR, JI – F, JI – CR, dan F – CR berpengaruh signifikan terhadap output. Jadi, jika salah satu parameter diubah dan parameter lainnya diubah akan menunjukkan nilai output berbeda.



Gambar 4.15 Interaksi antara Dua Faktor

Gambar 4.6 adalah *main effect plot*, yang menunjukkan efek utama dari setiap level pada setiap faktor terhadap output (*makespan*).



Gambar 4.16 Grafik Faktor Utama untuk *makespan*

Berikut adalah penjelasan gambar 4.6 :

- Ukuran populasi (NP)

Faktor ini memiliki gradien (kemiringan) garis yang negatif, artinya semakin besar jumlah ukuran populasi, maka *makespan* yang dihasilkan akan semakin kecil, dan sebaliknya. Maka, untuk memperkecil *makespan*, ukuran populasi yang digunakan adalah yang bernilai besar.

- Jumlah Iterasi (JI)

Dari gambar terlihat untuk faktor ini memiliki gradien (kemiringan) garis yang negatif hampir mendatar, artinya jumlah iterasi berpengaruh terhadap output (*makespan*) tetapi tidak signifikan. Jika semakin besar jumlah iterasi, maka *makespan* yang dihasilkan tidak akan berubah secara signifikan. Maka, untuk memperkecil *makespan*, jumlah iterasi yang digunakan adalah yang bernilai relatif kecil.

- Operator Mutasi (F)

Dari gambar terlihat nilai output (*makespan*) terkecil didapatkan ketika menggunakan parameter F berlevel sedang, yaitu 0,6. Oleh karena itu, untuk memperkecil *makespan*, nilai F yang digunakan adalah yang bernilai sedang (nilai tengah antara kisaran parameter F yang efektif).

- Operator Pindah Silang (CR)

Gradien garis untuk faktor ini adalah positif, artinya semakin kecil nilai CR, maka *makespan* yang dihasilkan juga akan semakin kecil. Maka, untuk memperkecil *makespan*, parameter CR yang digunakan adalah yang bernilai kecil.

Untuk ukuran populasi memiliki pengaruh signifikan terhadap output (*makespan*) sedangkan jumlah iterasi berpengaruh terhadap output (*makespan*) tetapi tidak signifikan. Jika ukuran populasi dan jumlah iterasi bertambah besar, maka hasil yang diperoleh akan semakin baik. Tetapi, karena keduanya juga berpengaruh besar terhadap pertambahan waktu komputasi (waktu *run* program), maka penggunaan nilai jumlah iterasi dan ukuran populasi yang sama-sama besar akan membuat waktu komputasi akan sangat lama. Dari hasil ANOVA (tabel 4.31) terlihat bahwa interaksi antara ukuran populasi dan jumlah iterasi berpengaruh terhadap output tetapi tidak signifikan sehingga jika salah satunya sudah bernilai besar, maka sudah cukup untuk mendapatkan output yang

diharapkan. Dari hasil ANOVA pula, terlihat bahwa ukuran populasi lebih berpengaruh terhadap output dibandingkan dengan jumlah iterasi sehingga nilai parameter yang akan diperbesar secara signifikan adalah ukuran populasi. Oleh karena itu, interaksi antara keduanya juga dilihat.

Jadi, dapat disimpulkan bahwa kombinasi parameter yang tepat untuk permasalahan penjadwalan *job-shop* ini adalah:

1. Ukuran populasi yang besar

Setelah melewati satu kali proses mutasi, pindah silang, dan seleksi, maka satu iterasi telah selesai. Proses ini akan terus berulang sampai jumlah iterasi yang telah ditentukan tercapai. Semakin besar ukuran populasi akan menyebabkan tingginya variasi individu pada populasi pada proses mutasi dan pindah silang sehingga tidak terjadi konvergensi, tetapi jika ukuran populasi terlalu besar akan menyebabkan waktu komputasi terlalu lama.

Karena ukuran populasi memiliki pengaruh yang sangat signifikan, maka jumlah iterasi ditetapkan dengan nilai yang besar dengan mempertimbangkan output (*makespan*) dan waktu komputasi. Untuk menentukan ukuran populasi, dilakukan lima kali percobaan, yaitu untuk ukuran populasi sebesar 535, 1070, 1605, 2140 dan 2675. Dari kelima percobaan tersebut, terlihat bahwa ukuran populasi di atas 1070 tidak memberikan peningkatan solusi yang lebih baik secara signifikan dan waktu komputasi yang dibutuhkan juga lebih lama. Dengan ukuran populasi itu pula, waktu komputasi yang dibutuhkan tidak terlalu lama, yaitu kurang dari 10 menit. Oleh karena itu, ukuran populasi yang digunakan adalah 1070.

2. Jumlah iterasi yang relatif kecil

Meskipun antara ukuran populasi dan jumlah iterasi, parameter yang akan lebih diperbesar adalah ukuran populasi, akan tetapi sesuai hasil DOE, parameter ini masih memiliki hubungan dengan parameter lainnya sehingga nilai untuk parameter ini akan tetap ditingkatkan walaupun peningkatannya kecil. Dari hasil ANOVA, terlihat bahwa interaksi antara $JI - F - CR$ berpengaruh signifikan terhadap output. Artinya, penentuan jumlah iterasi ditentukan pula oleh dua parameter lainnya (operator mutasi dan operator pindah silang). Jumlah iterasi yang terlalu kecil kemungkinan urutan yang

dihasilkan dan diuji akan semakin kecil sehingga solusi kurang mendekati optimal. Sedangkan jumlah iterasi yang terlalu besar akan menyebabkan waktu komputasi terlalu lama. Oleh karena itu, dipilih jumlah iterasi yang relatif kecil (tidak besar dan tidak terlalu kecil) yaitu 500 iterasi.

3. Nilai operator mutasi (F) yang sedang

Berdasarkan hasil DOE, nilai F yang terbaik untuk masalah penjadwalan ini adalah yang bernilai sedang. Hal ini sesuai dengan teori. Nilai F yang bernilai terlalu kecil akan menyebabkan DE mencari solusi yang mendekati area vektor target. Perpindahan vektor target akan sangat kecil bahkan dapat terkesan tidak mengalami perpindahan sehingga dapat mengakibatkan kondisi *stuck*. Akibatnya, solusi yang didapatkan akan jauh dari optimal karena DE lambat bergerak dalam mencari solusi yang lebih optimal. Sedangkan jika nilai F terlalu besar (lebih besar dari 1), perubahan vektor target akan sangat besar. Karena semua gen vektor target sama-sama mengalami perubahan yang besar pada populasi mutan, maka jika sebagian besar gen setiap individu pada populasi mutan menjadi gen pada populasi *trial* (jika nilai CR terlalu besar), maka urutan (permutasi) yang dihasilkan pada populasi *trial* kemungkinan besar tidak akan terlalu berubah secara signifikan dibandingkan dengan urutan populasi target. Hal ini juga menyebabkan DE lambat untuk mencari hasil yang optimal. Nilai F dan CR saling berhubungan, sesuai dengan hasil DOE, yaitu interaksi antara F dan CR berpengaruh signifikan terhadap output (nilai $p\text{-value} \leq 0,05$). Berdasarkan hasil DOE, nilai F yang digunakan adalah 0,6.

4. Nilai operator pindah silang (CR) yang besar

Berdasarkan hasil DOE, nilai CR yang terbaik adalah nilai CR yang terkecil (0,5). Hal ini juga sesuai dengan teori, yaitu nilai CR yang tinggi akan mempercepat terjadinya konvergensi sehingga nilai CR kadang perlu diturunkan agar DE lebih tangguh. Nilai CR yang besar akan menyebabkan peluang munculnya bilangan acak yang lebih kecil atau sama dengan nilai CR akan semakin besar sehingga sebagian besar nilai dimensi (gen) dari individu *trial* berasal dari individu mutan. Hal ini tidak baik karena ada kemungkinan populasi mutan hanya memiliki pergerakan yang kecil dari populasi target sehingga jika sebagian besar gen menjadi gen pada populasi *trial*, urutan

(permutasi) yang dihasilkan populasi *trial* tidak akan berubah signifikan dibandingkan populasi target. Hal ini akan menyebabkan pencarian solusi yang lebih baik akan lambat. Sesuai hasil DOE, nilai CR yang digunakan adalah 0,5.

4.4.2 Analisis Waktu Komputasi (Waktu *Run*)

Waktu komputasi dipengaruhi oleh ukuran populasi dan jumlah iterasi. Semakin besar ukuran populasi dan jumlah iterasi, maka waktu komputasi akan semakin lama. Waktu komputasi menjadi batasan penyelesaian masalah ini. Karena semua *job* berjumlah 107, maka jumlah kemungkinan urutan pengerjaan *job* adalah $107!$ ($107 \times 106 \times 105 \times \dots \times 2 \times 1$). Untuk menjangkau semua kemungkinan urutan ini, ukuran populasi dan/atau jumlah iterasi yang dibutuhkan akan sangat besar sehingga waktu komputasi akan sangat lama. Oleh karena itu, pemilihan ukuran populasi dan jumlah iterasi juga mempertimbangkan lamanya waktu komputasi.

4.4.3 Analisis Hasil

Dari ketiga hasil run program, nilai *makespan* ketiga hasil *run* program adalah sama 1.207.624,4 detik, tetapi urutan pengerjaan *jobnya* yang berbeda. Hal ini terjadi karena pada proses inisialisasi awal, setiap individu awal memiliki nilai yang berbeda-beda. Untuk setiap individu awal ini dilakukan pengurutan nilai dimensi dari yang terkecil hingga yang terbesar. Pengurutan tersebut akan menghasilkan vektor berdimensi 107 dengan nilai setiap dimensinya berupa indeks hasil pengurutan.

Setelah mendapatkan vektor-vektor urutan *job* setiap individu, kemudian dibuat fungsi objektifnya (minimum *makespan*). Pengevaluasian dilakukan dengan cara menghubungkan vektor tersebut dengan data waktu operasi sehingga fungsi objektif dapat dihitung. Pengevaluasian ini dimaksudkan untuk mengetahui individu pada generasi (iterasi) awal yang memiliki *makespan* terkecil yang selanjutnya, individu pada generasi (iterasi) awal inilah yang kemudian akan menjadi individu target.

Pada proses mutasi akan diperoleh individu mutan yang memiliki nilai *makespan* terkecil dengan urutan pengerjaan *job* yang membentuk individu mutan. Pada tahap proses pindah silang akan didapatkan individu *trial* yang berasal dari rekombinasi antara individu target dengan individu mutan. Selanjutnya pada tahap proses seleksi, akan dilakukan penyeleksian antara individu target dengan individu *trial*. Proses ini bertujuan untuk menentukan individu yang layak menjadi anggota generasi selanjutnya. Individu yang mempunyai nilai *makespan* terkecil akan menjadi individu anggota generasi selanjutnya.

Dari data *run* program yang telah dilakukan dapat diketahui bahwa nilai minimum *makespan* dari generasi (iterasi) awal sampai generasi terakhir yaitu sebanyak 500 generasi, semua nilai *makespan* yang diambil sebagai solusi adalah nilai *makespan* yang memiliki nilai paling kecil (optimal). Oleh karena fungsi tujuan adalah meminimumkan *makespan*, urutan pengerjaan *job* yang diperoleh adalah urutan pengerjaan *job* yang memiliki nilai *makespan* terkecil juga.

Oleh karena fungsi tujuan dari penyelesaian masalah pada penelitian ini adalah meminimumkan *makespan*, nilai *makespan* pada jadwal jalur *Low Pressure Die Casting* PT AHM adalah 1.253.272,8 detik, sedangkan usulan jadwal yang dihasilkan menghasilkan nilai *makespan* sebesar 1.207.624,4 detik. Jika dibandingkan, nilai *makespan* pada usulan jadwal mengalami penurunan sebesar 3,64% dari jadwal PT AHM.

BAB 5 KESIMPULAN

Berdasarkan hasil analisis dari permasalahan penjadwalan *job shop* pada jalur *low pressure die casting* PT Astra Honda Motor (PT AHM) menggunakan algoritma *differential evolution* dengan bantuan bahasa pemrograman MATLAB 7.8.0, diperoleh kesimpulan sebagai berikut :

1. Jadwal di jalur *Low Pressure Die Casting* PT AHM memiliki nilai *makespan* sebesar 1.253.272,8 detik.
2. Usulan jadwal hasil program dengan metode algoritma *differential evolution* memiliki nilai *makespan* sebesar 1.207.624,4 detik.
3. Nilai *makespan* untuk usulan jadwal memiliki penurunan dibanding jadwal PT AHM yaitu sebesar 3,64%.
4. Waktu *run* program untuk memperoleh usulan jadwal yaitu adalah 551,9062 detik.

Beberapa hal yang dapat dilakukan sebagai lanjutan penelitian ini adalah :

1. Melakukan koreksi fungsi tujuan. Koreksi fungsi tujuan berupa penambahan atau penggantian fungsi tujuan pada penelitian ini, dapat berupa total *tardiness/lateness*, atau fungsi tujuan lainnya.
2. Menerapkan parameter kontrol algoritma *differential evolution* yang berbeda, sehingga dapat dibandingkan dengan hasil penelitian ini.
3. Menggunakan penelitian ini sebagai bagian dari penelitian yang lebih luas, misalkan *flowshop* atau *assembly line*.

DAFTAR REFERENSI

- Adam, Everett E. & Ebert, Ronald J. (1992). *Production and Operations Management*. New Jersey: Prentice hall.
- Babu, B.V. & Angira, Rakesh. (2001) Optimization of Thermal Cracker Operation using Differential Evolution. *Proceedings of International Symposium and 54th Annual Session of II*.
- Baker, Kenneth R. (2001). *Elements of Sequencing and Scheduling*. Hanover: Author.
- Fan, Hui-Yuan., Lampinen, Jouni., & Levy, Yeshayahou. (2006). An Easy-to-Implement Differential Evolution Approach for Multi-Objective Optimization. *International Journal for Computer-Aided Engineering and Software*, vol. 23, no. 2, 124-138.
- Friman, Mark A. (2002). *Quality and Process Improvement*. USA.
- Karaboga, Dervis & Okdem, Selcuk. (2004). A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm. *Turk J. Elec Engin*, vol. 12, no. 1, 53-60.
- Lopez Cruz, Willigenburg, L.G. Van, I.L., & Straten, G. Van. (2001). Parameter Control Strategy in Differential Evolution Algorithm for Optimal Control. *Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing*, Cancun, Mexico, 211-216.
- Mezura-Montes, Efren., Velásquez-Reyes, Jesús., & Coello Coello, Carlos A. (2005). Promising Infeasibility and Multiple Offspring Incorporated to Differential Evolution for Constrained Optimization. *GECCO'05*, Washington, DC.
- Montgomery Douglas C. (1996). *Design and Analysis of Experiments*. New York: John Wiley & Sons.
- Nahmias, Steven. (1997). *Production and Operation Analysis*. New York: McGraw-Hill.
- Neal, M., et. al. (2006). Applying Differential Evolution to a Whole-Farm Model to Assist Optimal Strategic Decision Making.
- Nearchou, Andreas C. (2008). A Differential Evolution for Common Due Date Early/Tardy Job Scheduling Problem. *Computers and Operations Research*, vol. 35, 1329-1343.

- Nearchou, Andreas C. & Omirou, Sotiris L. (2006). Differential Evolution for Sequencing and Scheduling Optimization. *J Heuristics, Springer Science+Business Media*, 12, 395-411.
- Noman, Nasimul & Iba, Hitoshi. (2005). Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization. *GECCO'05*, Washington, DC.
- Price, K.V. (1999). An Introduction to Differential Evolution. In: *Corne D, Dorigo M, Glover F (eds.), New ideas in optimization*. McGraw-Hill, London, 79-108.
- Routroy, Srikanta & Kodali, Rambabu (2005). Differential Evolution Algorithm for Supply Chain Inventory Planning. *Journal of Manufacturing Technology Management*, vol. 16, no. 1, 7-17.
- Santosa, Budi. (2008). *Matlab untuk Statistika & Teknik Optimasi - Aplikasi untuk Rekayasa & Bisnis*. Yogyakarta: Graha Ilmu.
- Storn, Rainer & Price, Kenneth. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11, 341-359.
- Tasgetiren, M. Fatih, et al. (2004). Particle Swarm Optimization and Differential Evolution Algorithm for Job Shop Scheduling Problem. *Proceedings of the 4th International Symposium on Intelligent Manufacturing System (IMS2004)*, Sakarya, Turkey, 1-18.
- Ursen, Rasmus K. (2005). Differential Evolution Made Easy. *Technical Report*, No.1.
- <http://www.id.wikipedia.org/wiki/Algoritma>
- http://www.en.wikipedia.org/wiki/Differential_evolution
- <http://www.mathworks.com/products/matlab/description1.html>

Lampiran 1: *Script M-File* Program untuk Perhitungan Jadwal PT AHM

```
clc;
clear;
tic;

jumlah_pesanan = 107;
jumlah_rute = 6;

%-----Data waktu proses-----
%-----Kolom 1-6 adalah waktu proses tiap rute-----
data2 = [6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
```

Lampiran 1: *Script M-File* Program untuk Perhitungan Jadwal PT AHM (lanjutan)

5196.6	0	0	0	0	23242.2
5196.6	0	0	0	0	23242.2
5196.6	0	0	0	0	23242.2
5196.6	0	0	0	0	23242.2
5196.6	0	0	0	0	23242.2
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
4265.6	0	17342.8	0	11322.8	0
4265.6	0	17342.8	0	11322.8	0
4265.6	0	17342.8	0	11322.8	0
4265.6	0	17342.8	0	11322.8	0
4265.6	0	17342.8	0	11322.8	0
5099.1	0	0	0	0	33729.7
5099.1	0	0	0	0	33729.7
5099.1	0	0	0	0	33729.7
5099.1	0	0	0	0	33729.7
4831.8	0	0	0	0	24183.8
4831.8	0	0	0	0	24183.8
4831.8	0	0	0	0	24183.8
4831.8	0	0	0	0	24183.8
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
6521.2	0	29500.6	0	16380.6	0
6521.2	0	29500.6	0	16380.6	0
6521.2	0	29500.6	0	16380.6	0
6521.2	0	29500.6	0	16380.6	0
6521.2	0	29500.6	0	16380.6	0
6521.2	0	29500.6	0	16380.6	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0

Lampiran 1: *Script M-File* Program untuk Perhitungan Jadwal PT AHM (lanjutan)

```

4265.6  0  17342.8  0  11322.8  0
4265.6  0  17342.8  0  11322.8  0
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
4831.8  0  0  0  0  24183.8
4831.8  0  0  0  0  24183.8
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0];
N = size(data2);
wp = data2(1:N,1:6);
w = zeros(jumlah_pesanan,6);
for a = 1 :jumlah_pesanan
for b = 1 :jumlah_rute
if a == 1 && b == 1
w(a,b)=wp(a,b);
end
if a == 1 && b == 2
w(a,b)=w(a,b-1)+wp(a,b);
end
if a == 1 && b >= 3 && b <= 5
w(a,b)=w(a,b-2)+wp(a,b);
end
if a == 1 && b == 6
w(a,b)=w(a,b-5)+wp(a,b);
end
if a > 1 && b == 1
w(a,b)=w(a-1,b)+wp(a,b);

```

Lampiran 1: *Script M-File* Program untuk Perhitungan Jadwal PT AHM (lanjutan)

```
end
if a > 1 && b == 2
    w(a,b)=max(w(a,b-1),w(a-1,b))+wp(a,b);
end
if a > 1 && b >= 3 && b <= 5
    w(a,b)=max(w(a,b-2),w(a-1,b))+wp(a,b);
end
if a > 1 && b ==6
    w(a,b)=max(w(a,b-5),w(a-1,b))+wp(a,b);
end
end
end
w1=zeros(jumlah_pesanan, 1);
for a = 1 : jumlah_pesanan
    if wp(a,4)==0 && wp(a,6)==0
        w1(a)=w(a,5);
    end
    if wp(a,5)==0 && wp(a,6)==0
        w1(a)=w(a,4);
    end
    if wp(a,4)==0 && wp(a,5)==0
        w1(a)=w(a,6);
    end
end
index_rute8=zeros(jumlah_pesanan,1);
wp1=zeros(jumlah_pesanan,1);
for a = 1 : jumlah_pesanan
    [wp1(a),index_rute8(a)]=min(w1);
    w1(index_rute8(a))=9999999999;
end
waktu_total=max(wp1);
%-----Solusi Akhir-----
disp('Makespan =');
disp(waktu_total);
```


Lampiran 2: *Script M-File* Program untuk Pencarian Solusi Jadwal

```
clc;
clear;
tic;

jumlah_pesanan = 107;
jumlah_rute = 6;
jumlah_dimensi = jumlah_pesanan;
ukuran_populasi = 1070;
jumlah_iterasi = 500;
F = 0.6; %operator mutasi
CR = 0.5; %operator pindah silang
batas_bawah = -1;
batas_atas = 1;

%-----Data waktu proses-----
%-----Kolom 1-6 adalah waktu proses tiap rute-----
data2 = [6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5205.0  28462.5  0  13142.5  0  0
5308.4  26435.3  0  12994.3  0  0
```

Lampiran 2: *Script M-File* Program untuk Pencarian Solusi Jadwal (lanjutan)

5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5196.6	0	0	0	23242.2	
5196.6	0	0	0	23242.2	
5196.6	0	0	0	23242.2	
5196.6	0	0	0	23242.2	
5196.6	0	0	0	23242.2	
5196.6	0	0	0	23242.2	
5196.6	0	0	0	23242.2	
5196.6	0	0	0	23242.2	
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
5308.4	26435.3	0	12994.3	0	0
4265.6	0	17342.8	0	11322.8	0
4265.6	0	17342.8	0	11322.8	0
4265.6	0	17342.8	0	11322.8	0
4265.6	0	17342.8	0	11322.8	0
4265.6	0	17342.8	0	11322.8	0
5099.1	0	0	0	33729.7	
5099.1	0	0	0	33729.7	
5099.1	0	0	0	33729.7	
5099.1	0	0	0	33729.7	
4831.8	0	0	0	24183.8	
4831.8	0	0	0	24183.8	
4831.8	0	0	0	24183.8	
4831.8	0	0	0	24183.8	
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
5205.0	28462.5	0	13142.5	0	0
6521.2	0	29500.6	0	16380.6	0
6521.2	0	29500.6	0	16380.6	0
6521.2	0	29500.6	0	16380.6	0

Lampiran 2: *Script M-File* Program untuk Pencarian Solusi Jadwal (lanjutan)

```

6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
4265.6  0  17342.8  0  11322.8  0
4265.6  0  17342.8  0  11322.8  0
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5196.6  0  0  0  0  23242.2
5196.6  0  0  0  0  23242.2
4831.8  0  0  0  0  24183.8
4831.8  0  0  0  0  24183.8
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
6521.2  0  29500.6  0  16380.6  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0
5308.4  26435.3  0  12994.3  0  0];

```

```

N=size(data2);
wp=data2(1:N,1:6);
%-----Membentuk populasi target-----
populasi_target=batas_bawah+(batas_atas-
batas_bawah)*rand(jumlah_pesanan,ukuran_populasi);
%-----Mencari urutan pengerjaan populasi target-----
for a = 1 : ukuran_populasi;
    [hasil,urutan_awal(:,a)] = sort(populasi_target(:,a));
end
%-----Menghitung total makespan populasi target-----
for a = 1 : ukuran_populasi
    w = zeros(jumlah_pesanan,6);

```

Lampiran 2: *Script M-File* Program untuk Pencarian Solusi Jadwal (lanjutan)

```
for b = 1 : jumlah_pesanan
for c = 1 : jumlah_rute
    if b==1 && c==1
        w(b,c)=wp(urutan_awal(b,a),c);
    end
    if b==1 && c==2
        w(b,c)=w(b,c-1)+wp(urutan_awal(b,a),c);
    end
    if b==1 && c>=3 && c<=5
        w(b,c)=w(b,c-2)+wp(urutan_awal(b,a),c);
    end
    if b==1 && c==6
        w(b,c)=w(b,c-5)+wp(urutan_awal(b,a),c);
    end
    if b>1 && c==1
        w(b,c)=w(b-1,c)+wp(urutan_awal(b,a),c);
    end
    if b>1 && c==2
        w(b,c)=max(w(b,c-1),w(b-1,c))+wp(urutan_awal(b,a),c);
    end
    if b>1 && c>=3 && c<=5
        w(b,c)=max(w(b,c-2),w(b-1,c))+wp(urutan_awal(b,a),c);
    end
    if b>1 && c==6
        w(b,c)=max(w(b,c-5),w(b-1,c))+wp(urutan_awal(b,a),c);
    end
end
end
w1=zeros(jumlah_pesanan, 1);
for b = 1 : jumlah_pesanan
    if wp(urutan_awal(b,a),4)==0 && wp(urutan_awal(b,a),6)==0
        w1(b)=w(b,5);
    end
    if wp(urutan_awal(b,a),5)==0 && wp(urutan_awal(b,a),6)==0
        w1(b)=w(b,4);
    end
    if wp(urutan_awal(b,a),4)==0 && wp(urutan_awal(b,a),5)==0
        w1(b)=w(b,6);
    end
end
index_rute8=zeros(jumlah_pesanan,1);
wp1=zeros(jumlah_pesanan,1);
for b = 1 : jumlah_pesanan
    [wp1(b),index_rute8(b)]=min(w1);
    w1(index_rute8(b))=999999999;
end
```

Lampiran 2: *Script M-File* Program untuk Pencarian Solusi Jadwal (lanjutan)

```
end
nilai_makespan_awal(a)=max(wp1);
end
[nilai_makespan_min_awal,index_vektor_target]=min(nilai_makespan_awal);
urutan=urutan_awal(:,index_vektor_target);
%-----Hasil awal sebelum iterasi-----
disp('Urutan=');
disp(urutan);
disp('Nilai makespan awal=');
disp(nilai_makespan_min_awal);
%Memulai iterasi-----
proses = 0;
jumlah_maksimum_iterasi = 0;
while (proses==0)
    jumlah_maksimum_iterasi=jumlah_maksimum_iterasi+1;
    if jumlah_maksimum_iterasi==jumlah_iterasi
        proses=1;
    end
    %-----Mencari vektor target-----
    [nilai_makespan_min_awal,index_vektor_target]=min(nilai_makespan_awal);
    vektor_target=populasi_target(:,index_vektor_target);
    for a = 2 :ukuran_populasi
        vektor_target(:,a)=vektor_target(:,a-1);
    end
    %-----Mencari populasi mutan-----
    %mencari 2 vektor acak-----
    for a = 1 : ukuran_populasi
        index_vektor_acak1 = 1;
        index_vektor_acak2 = 1;
    while(index_vektor_acak1==index_vektor_acak2)||
(index_vektor_acak1==index_
vektor_target)||
(index_vektor_acak2==index_vektor_target)
        index_vektor_acak1=randint(1,1,ukuran_populasi)+1;
        index_vektor_acak2=randint(1,1,ukuran_populasi)+1;
    end
        vektor_acak1(:,a)=populasi_target(:,index_vektor_acak1);
        vektor_acak2(:,a)=populasi_target(:,index_vektor_acak2);
    end
    %-----Membentuk populasi mutan-----
    populasi_mutan=(vektor_acak1-vektor_acak2)*F+vektor_target;
    %-----Membentuk populasi trial-----
    for a = 1 : jumlah_pesanan
    for b = 1 : ukuran_populasi
        if (rand())<=CR)
            populasi_trial(a,b)=populasi_mutan(a,b);
        else
```

Lampiran 2: *Script M-File* Program untuk Pencarian Solusi Jadwal (lanjutan)

```
        populasi_trial(a,b)=populasi_target(a,b);
    end
end
end
%-----Mencari urutan pengerjaan populasi trial-----
for a = 1 : ukuran_populasi;
    [hasil,urutan_trial(:,a)]=sort(populasi_trial(:,a));
end
%-----Menghitung total makespan populasi trial-----
nilai_makespan_anak = [];
for a = 1 : ukuran_populasi
    w=zeros(jumlah_pesanan,6);
    for b = 1 : jumlah_pesanan
        for c = 1 : jumlah_rute
            if b==1 && c==1
                w(b,c)=wp(urutan_trial(b,a),c);
            end
            if b==1 && c==2
                w(b,c)=w(b,c-1)+wp(urutan_trial(b,a),c);
            end
            if b==1 && c>=3 && c<=5
                w(b,c)=w(b,c-2)+wp(urutan_trial(b,a),c);
            end
            if b==1 && c==6
                w(b,c)=w(b,c-5)+wp(urutan_trial(b,a),c);
            end
            if b>1 && c==1
                w(b,c)=w(b-1,c)+wp(urutan_trial(b,a),c);
            end
            if b>1 && c==2
                w(b,c)=max(w(b,c-1),w(b-1,c))+wp(urutan_trial(b,a),c);
            end
            if b>1 && c>=3 && c<=5
                w(b,c)=max(w(b,c-2),w(b-1,c))+wp(urutan_trial(b,a),c);
            end
            if b>1 && c==6
                w(b,c)=max(w(b,c-5),w(b-1,c))+wp(urutan_trial(b,a),c);
            end
        end
    end
end
end
w1=zeros(jumlah_pesanan,1);
for b = 1 : jumlah_pesanan
    if wp(urutan_trial(b,a),4)==0 && wp(urutan_trial(b,a),6)==0
        w1(b)=w(b,5);
    end
end
```

Lampiran 2: *Script M-File* Program untuk Pencarian Solusi Jadwal (lanjutan)

```
if wp(urutan_trial(b,a),5)==0 && wp(urutan_trial(b,a),6)==0
    w1(b)=w(b,4);
end
if wp(urutan_trial(b,a),4)==0 && wp(urutan_trial(b,a),5)==0
    w1(b)=w(b,6);
end
end
index_rute8=zeros(jumlah_pesanan,1);
wp1=zeros(jumlah_pesanan,1);
for b = 1 : jumlah_pesanan
    [wp1(b),index_rute8(b)]=min(w1);
    w1(index_rute8(b))=999999999;
end
nilai_makespan_anak(a)=max(wp1);
end
%-----Memilih antara populasi target awal atau trial-----
for a = 1 : ukuran_populasi
    if nilai_makespan_anak(a)<nilai_makespan_awal(a)
        populasi_target(:,a)=populasi_trial(:,a);
        urutan_awal(:,a)=urutan_trial(:,a);
        nilai_makespan_awal(a)=nilai_makespan_anak(a);
    end
end
end
[nilai_makespan_akhir,index_vektor_target_akhir]=min(nilai_makespan_awal);
urutan_akhir=urutan_awal(:,index_vektor_target_akhir);
%----Solusi akhir----
disp('Urutan terbaik =');
disp(urutan_akhir);
disp('Makespan = ');
disp(nilai_makespan_akhir);
disp(strcat('Waktu Komputasi : ',num2str(toc),'detik'));
```

Lampiran 3: Hasil *Design of Experiment*

No	Parameter				Hasil (<i>makespan</i>)				
	NP	JI	F	CR	<i>Run 1</i>	<i>Run 2</i>	<i>Run 3</i>	<i>Run 4</i>	<i>Run 5</i>
1	15	15	0.4	0.5	1227520.20	1219070.90	1219125.60	1220918.40	1219961.50
2	15	15	0.4	0.7	1227188.00	1218372.70	1225498.60	1227759.70	1226500.00
3	15	15	0.4	0.9	1247793.60	1245621.30	1230556.10	1249871.20	1244389.90
4	15	15	0.6	0.5	1213227.90	1212604.40	1212969.20	1221500.40	1213526.40
5	15	15	0.6	0.7	1226750.20	1227336.20	1225362.30	1227268.60	1220770.20
6	15	15	0.6	0.9	1230257.20	1244183.50	1246275.10	1249013.20	1244309.30
7	15	15	0.8	0.5	1218343.90	1218692.70	1214427.10	1218977.40	1219392.90
8	15	15	0.8	0.7	1226636.40	1227321.90	1225184.00	1226873.20	1210555.10
9	15	15	0.8	0.9	1245744.00	1231311.60	1244423.00	1244102.40	1243423.00
10	15	30	0.4	0.5	1217756.20	1218017.60	1218514.60	1222107.90	1219244.70
11	15	30	0.4	0.7	1226583.30	1226451.30	1224932.40	1226830.80	1215886.40
12	15	30	0.4	0.9	1245508.10	1240438.90	1243737.60	1231238.10	1242963.10
13	15	30	0.6	0.5	1212924.40	1212559.60	1220771.90	1212999.10	1212871.70
14	15	30	0.6	0.7	1218076.60	1231518.50	1218783.60	1218559.40	1219342.20
15	15	30	0.6	0.9	1222788.80	1240238.00	1242151.60	1237832.80	1240157.40
16	15	30	0.8	0.5	1226518.60	1226011.60	1224589.50	1226144.90	1216899.90
17	15	30	0.8	0.7	1225876.90	1225914.10	1211577.00	1226115.00	1225863.40
18	15	30	0.8	0.9	1232719.20	1239054.00	1240489.60	1237445.20	1238792.60
19	15	45	0.4	0.5	1225846.50	1225750.20	1223800.60	1222421.10	1225595.10
20	15	45	0.4	0.7	1225869.30	1216557.70	1224174.00	1225765.90	1225782.90
21	15	45	0.4	0.9	1233538.70	1236657.80	1237417.70	1237319.80	1237684.60
22	15	45	0.6	0.5	1212456.20	1223214.20	1212707.80	1212723.50	1212038.20
23	15	45	0.6	0.7	1217436.20	1217904.40	1223382.60	1217652.80	1218654.20
24	15	45	0.6	0.9	1242413.00	1235942.50	1236521.20	1234001.00	1237542.70
25	15	45	0.8	0.5	1217396.90	1217801.00	1218068.30	1217622.60	1224322.20
26	15	45	0.8	0.7	1217264.70	1225311.90	1223510.30	1225110.50	1225035.80
27	15	45	0.8	0.9	1237571.50	1235696.80	1232346.10	1236650.20	1236433.60
28	30	15	0.4	0.5	1217345.30	1217560.80	1218023.50	1217288.00	1212865.50
29	30	15	0.4	0.7	1224716.90	1212035.50	1223159.70	1224692.90	1224687.00
30	30	15	0.4	0.9	1236901.80	1235473.80	1235403.50	1224538.80	1235897.70
31	30	15	0.6	0.5	1211411.80	1212141.60	1212246.20	1212587.50	1225966.80
32	30	15	0.6	0.7	1217338.20	1216980.50	1217979.10	1226351.80	1217546.20
33	30	15	0.6	0.9	1236851.60	1235086.10	1225080.60	1233709.20	1235891.80
34	30	15	0.8	0.5	1224632.30	1217092.50	1223116.70	1223927.20	1224642.20
35	30	15	0.8	0.7	1235189.60	1224544.70	1223106.50	1223809.20	1224090.70

Lampiran 3: Hasil *Design of Experiment* (lanjutan)

No	Parameter				Hasil (<i>makespan</i>)				
	NP	JI	F	CR	<i>Run 1</i>	<i>Run 2</i>	<i>Run 3</i>	<i>Run 4</i>	<i>Run 5</i>
36	30	15	0.8	0.9	1236753.60	1234178.00	1234065.90	1232808.60	1242859.70
37	30	30	0.4	0.5	1217078.00	1216259.00	1217234.80	1217086.60	1216890.20
38	30	30	0.4	0.7	1234848.70	1223859.40	1222892.20	1223658.50	1223607.80
39	30	30	0.4	0.9	1236566.90	1232738.70	1234051.30	1225907.90	1235005.30
40	30	30	0.6	0.5	1211362.80	1211890.00	1211993.40	1212068.10	1211452.00
41	30	30	0.6	0.7	1225087.90	1216973.40	1217419.30	1217190.00	1216989.10
42	30	30	0.6	0.9	1224292.00	1235325.60	1222534.80	1223243.00	1223054.00
43	30	30	0.8	0.5	1236285.40	1232666.10	1237118.90	1232287.10	1234104.40
44	30	30	0.8	0.7	1223756.00	1223232.80	1222431.40	1218165.80	1223019.20
45	30	30	0.8	0.9	1236262.50	1232019.80	1232636.20	1232227.00	1225602.40
46	30	45	0.4	0.5	1225780.10	1215289.70	1216721.80	1215648.30	1216236.20
47	30	45	0.4	0.7	1223362.40	1222676.80	1235733.70	1222997.60	1222952.80
48	30	45	0.4	0.9	1233923.50	1231735.10	1232532.80	1231733.00	1244325.50
49	30	45	0.6	0.5	1210660.20	1218121.00	1211954.90	1211783.90	1210950.40
50	30	45	0.6	0.7	1216847.90	1216155.60	1216870.00	1226173.70	1216825.20
51	30	45	0.6	0.9	1222056.30	1222378.00	1222272.90	1222849.40	1247609.60
52	30	45	0.8	0.5	1233107.30	1231489.70	1232488.00	1212397.20	1232435.30
53	30	45	0.8	0.7	1220999.00	1222353.60	1218201.60	1222825.00	1222199.90
54	30	45	0.8	0.9	1231557.00	1243098.40	1232384.60	1231208.20	1232168.00
55	45	15	0.4	0.5	1218299.50	1215171.60	1215509.10	1215230.50	1215523.30
56	45	15	0.4	0.7	1220948.30	1221952.90	1220622.40	1222712.90	1213049.80
57	45	15	0.4	0.9	1230701.10	1230962.50	1225928.30	1231163.40	1231133.30
58	45	15	0.6	0.5	1218904.20	1211309.70	1211590.10	1210424.70	1210772.30
59	45	15	0.6	0.7	1215454.60	1214717.30	1214592.30	1218411.20	1214507.70
60	45	15	0.6	0.9	1230695.20	1248990.80	1231214.10	1231060.00	1230849.30
61	45	15	0.8	0.5	1243035.40	1221544.80	1220165.50	1222632.80	1221165.50
62	45	15	0.8	0.7	1220666.80	1240022.10	1220062.10	1222363.80	1220993.10
63	45	15	0.8	0.9	1230277.20	1230098.80	1224441.30	1230129.00	1230798.60
64	45	30	0.4	0.5	1214249.00	1212821.00	1213176.10	1213823.60	1213936.60
65	45	30	0.4	0.7	1220344.80	1219830.60	1219496.30	1222283.20	1220895.60
66	45	30	0.4	0.9	1230232.40	1229741.30	1230543.10	1230031.50	1230134.90
67	45	30	0.6	0.5	1210140.00	1209447.70	1210036.50	1210071.20	1210242.70
68	45	30	0.6	0.7	1215112.50	1214527.90	1213982.10	1214397.20	1214471.90
69	45	30	0.6	0.9	1214293.80	1213102.50	1213697.50	1214145.60	1213958.90
70	45	30	0.8	0.5	1229474.00	1229688.60	1228958.50	1229987.00	1229764.20

Lampiran 3: Hasil *Design of Experiment* (lanjutan)

No	Parameter				Hasil (<i>makespan</i>)				
	NP	JI	F	CR	<i>Run 1</i>	<i>Run 2</i>	<i>Run 3</i>	<i>Run 4</i>	<i>Run 5</i>
71	45	30	0.8	0.7	1220017.30	1219593.80	1219445.60	1222027.30	1220815.00
72	45	30	0.8	0.9	1229370.60	1229154.00	1228527.20	1229912.40	1229622.20
73	45	45	0.4	0.5	1208865.70	1208447.70	1208762.30	1208910.50	1208969.00
74	45	45	0.4	0.7	1218737.50	1219490.40	1219342.20	1221835.10	1220355.10
75	45	45	0.4	0.9	1228721.10	1229011.30	1226184.00	1229538.80	1228907.80
76	45	45	0.6	0.5	1208299.50	1208196.10	1208163.50	1208575.60	1208447.70
77	45	45	0.6	0.7	1209203.50	1209245.10	1209378.70	1209029.60	1209013.90
78	45	45	0.6	0.9	1213392.70	1212737.70	1213072.60	1213496.10	1213831.20
79	45	45	0.8	0.5	1228205.60	1228587.80	1225652.30	1228706.90	1228804.40
80	45	45	0.8	0.7	1218662.80	1219080.80	1219258.90	1221701.80	1220064.90
81	45	45	0.8	0.9	1228074.80	1227760.20	1225602.00	1228194.20	1227291.40

