OPTIMASI PENJADWALAN *JOB SHOP* DENGAN METODE ALGORITMA *DIFFERENTIAL EVOLUTION* UNTUK MEMINIMUMKAN TOTAL BIAYA KETERLAMBATAN PENYELESAIAN PESANAN DI PT X

SKRIPSI

LINA ASTUTI
0404070387

UNIVERSITAS INDONESIA
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INDUSTRI
DEPOK
JULI 2008

OPTIMASI PENJADWALAN *JOB SHOP* DENGAN METODE ALGORITMA *DIFFERENTIAL EVOLUTION* UNTUK MEMINIMUMKAN TOTAL BIAYA KETERLAMBATAN PENYELESAIAN PESANAN DI PT X

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana teknik

LINA ASTUTI
0404070387

UNIVERSITAS INDONESIA
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INDUSTRI
DEPOK
JULI 2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

OPTIMASI PENJADWALAN *JOB SHOP* DENGAN METODE ALGORITMA *DIFFERENTIAL EVOLUTION* UNTUK MEMINIMUMKAN TOTAL BIAYA KETERLAMBATAN PENYELESAIAN PESANAN DI PT X

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Studi Teknik Industri Departemen Teknik Industri Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 18 Juli 2008

(Lina Astuti)
NPM 0404070387

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Lina Astuti
NPM	:	0404070387
Program Studi	:	Teknik Industri
Judul Skripsi	:	Optimasi Penjadwalan *Job Shop* dengan Metode Algoritma *Differential Evolution* untuk Meminimumkan Total Biaya Keterlambatan Penyelesaian Pesanan di PT X

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	: Ir. Yadrifil, M.Sc	( )
Penguji	: Ir. Isti Surjandari, MT, MA, Ph.D	( )
Penguji	: Ir. Akhmad Hidayatno, MBT	( )

Ditetapkan di : Depok

Tanggal : 18 Juli 2008

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.
- Segenap karyawan PT. X yang telah memberikan kesempatan kepada penulis untuk mengumpulkan data untuk penelitian ini dan memberikan jawaban atas pertanyaan-pertanyaan yang sering penulis tanyakan.
- Keluarga, atas curahan kasih sayang , dukungan, dan doa yang diberikan.
- Teman-teman penulis, khususnya rekan-rekan TIUI 2004 yang telah memberikan dukungan, semangat, serta kebersamaan selama empat tahun 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.

Depok, 18 Juli 2008

Penulis

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

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

Nama : Lina Astuti
NPM : 0404070387
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 Penjadwalan *Job Shop* dengan Metode Algoritma *Differential Evolution* untuk Meminimumkan Total Biaya Keterlambatan Penyelesaian Pesanan Di PT X

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Nonekslusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, 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 : Depok
Pada tanggal : 18 Juli 2008
Yang menyatakan

(Lina Astuti)

RIWAYAT HIDUP PENULIS

Nama : Lina Astuti
Tempat, Tanggal Lahir : Jakarta, 6 Oktober 1986
Alamat : Jl. Apus 1A No. 19 Rt 12/06
Kel. Kota Bambu Selatan, Kec. Palmerah
Jakarta Barat 11420
Pendidikan :

a.	SD	:	SDN Parapat Tangerang (1992 - 1995)
			SDN 1 Banjarbaru (1995 – 1998)
b.	SLTP	:	SLTPN 1 Banjarbaru (1998)
			SLTPN 61 Jakarta (1998 – 2001)
c.	SMU	:	SMUN 16 Jakarta (2001 – 2004)
d.	S-1	:	Departemen Teknik Industri,
			Fakultas Teknik Universitas Indonesia (2004 – 2008)

ABSTRAK

Nama : Lina Astuti
Program Studi : Teknik Industri
Judul : Optimasi Penjadwalan *Job Shop* dengan Metode Algoritma *Differential Evolution* untuk Meminimumkan Total Biaya Keterlambatan Penyelesaian Pesanan di PT X

Penelitian ini membahas masalah penjadwalan *job shop* pada suatu perusahaan. 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 inisialisasi 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 ialah meminimumkan total biaya keterlambatan seluruh *job*.

Penjadwalan yang diperoleh melalui algoritma *differential evolution* menghasilkan total biaya keterlambatan seluruh *job* sebesar 28395 menit, sedangkan jadwal perusahaan menghasilkan 33190 menit. Jadi, usulan jadwal menghasilkan penurunan total biaya keterlambatan sebesar 14,45% dibandingkan jadwal perusahaan. Selain itu; jumlah *job* yang terlambat, total keterlambatan, dan total waktu penyelesaian seluruh *job* juga mengalami penurunan; yaitu secara berurutan sebesar 11,11%; 11,47%; dan 0,1%.

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

ABSTRACT

Name : Lina Astuti
Study Program : Industrial Engineering
Title : Optimizing Job Shop Scheduling using Differential Evolution Algorithm for Minimizing Total of Tardiness Costs in PT X

This research presents job shop scheduling at a 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 metaheuristic 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 total of tardiness costs of all jobs.

The schedule that is obtained from differential evolution algorithm produces total of tardiness costs of 28395 minutes, meanwhile the schedule of company produces 33190 minutes. Thus, new schedule produces reduction of total of tardiness costs about 14.45% compared with schedule of company. Moreover, the number of tardy jobs, total of tardiness, and makespan also show reduction about 11.11%, 11.47%, and 0.1% respectively.

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

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

1. PENDAHULUAN

1.1. Latar Belakang

Setiap perusahaan berusaha untuk cepat dalam merespon permintaan konsumen agar mampu meningkatkan *market share*. Dengan demikian, penjadwalan yang efektif dan efisisen berperan penting dalam *modern competitive marketplace*. Pada bidang manufaktur, penjadwalan merupakan suatu permasalahan dimana adanya beberapa tujuan dan sumber daya yang harus dialokasikan untuk mendapatkan tujuan perusahaan, seperti memaksimumkan utilisasi dari pekerja dan mesin, meminimumkan waktu yang dibutuhkan untuk menyelesaikan keseluruhan proses yang telah dijadwalkan (*makespan*), atau meminimumkan keterlambatan setiap *job* yang datang.

Penjadwalan yang optimal sangatlah diperlukan dalam sistem produksi di perusahaan. Apabila di dalam suatu perusahaan terdapat suatu penjadwalan yang optimal, maka penggunaan waktu, tenaga, produksi, dan keuntungan yang dimiliki juga akan lebih optimal. Selain itu, dengan adanya penjadwalan yang optimal, maka suatu perusahaan dapat menjaga citra baik pada pangsa pasarnya dengan menggunakan ketepatan waktu dalam pemenuhan bagi konsumennya.

Suatu perusahaan yang sistem produksinya bersifat *make-to-orders* harus memenuhi pesanan konsumen sesuai waktu yang diminta (*due date*). Adakalanya perusahaan akan terkena biaya penalti jika pemenuhan pesanan tidak sesuai waktu yang diminta. Di lain pihak, ketersediaan sumber daya, seperti mesin, pekerja, dan peralatan yang dimiliki perusahaan berjumlah terbatas. Jika keterbatasan tersebut tidak diatur dengan baik, maka perusahaan akan lebih memiliki risiko terhadap terjadinya keterlambatan pesanan tersebut.

PT X adalah perusahaan yang memproduksi (merakit) alat-alat berat. Beberapa komponen dari alat-alat berat tersebut juga diproduksi di perusahaan tersebut. Proses produksi di PT X memakan waktu yang cukup lama, sedangkan untuk setiap periode tertentu ada sejumlah pesanan yang harus terpenuhi. Hal ini menuntut perusahaan agar merancang sebuah sistem penjadwalan yang efektif dan efisien agar seluruh permintaan dapat dipenuhi tepat waktu.

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 lainnya[1]. Metode baru yang hanya memerlukan sedikit variabel kontrol ini tangguh, mudah digunakan, dan sangat cocok untuk perhitungan paralel.

1.2.Diagram Keterkaitan Masalah

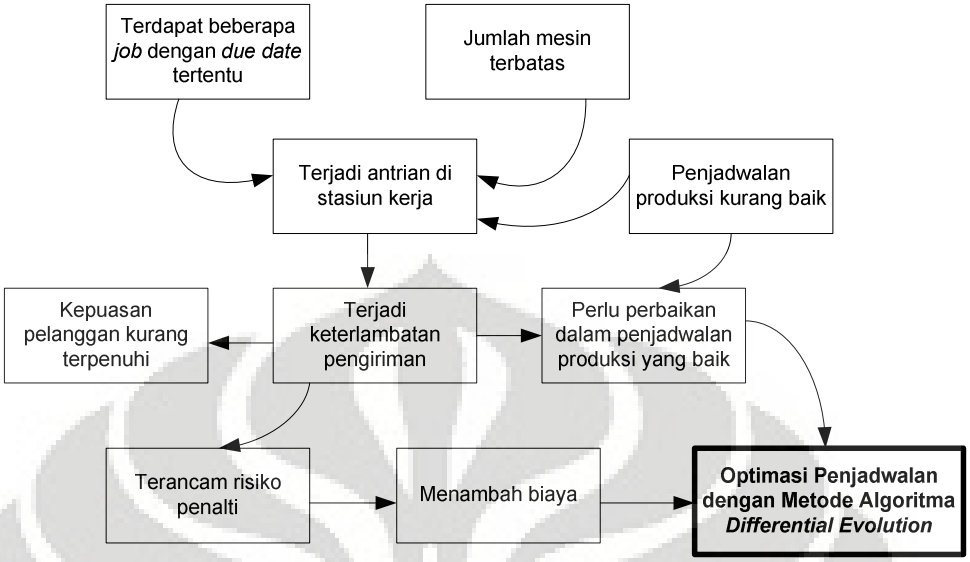
Untuk melihat gambaran sistematik yang lebih menyeluruh, maka disusun suatu diagram keterkaitan masalah seperti pada gambar 1.1.

1.3. Perumusan Masalah

Pokok permasalahan yang akan dibahas adalah perlunya perancangan suatu sistem penjadwalan yang efisien.

---

[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

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah memperoleh suatu sistem penjadwalan yang lebih baik (lebih mendekati optimal) dengan meminimumkan total biaya keterlambatan penyelesaian seluruh pekerjaan 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:

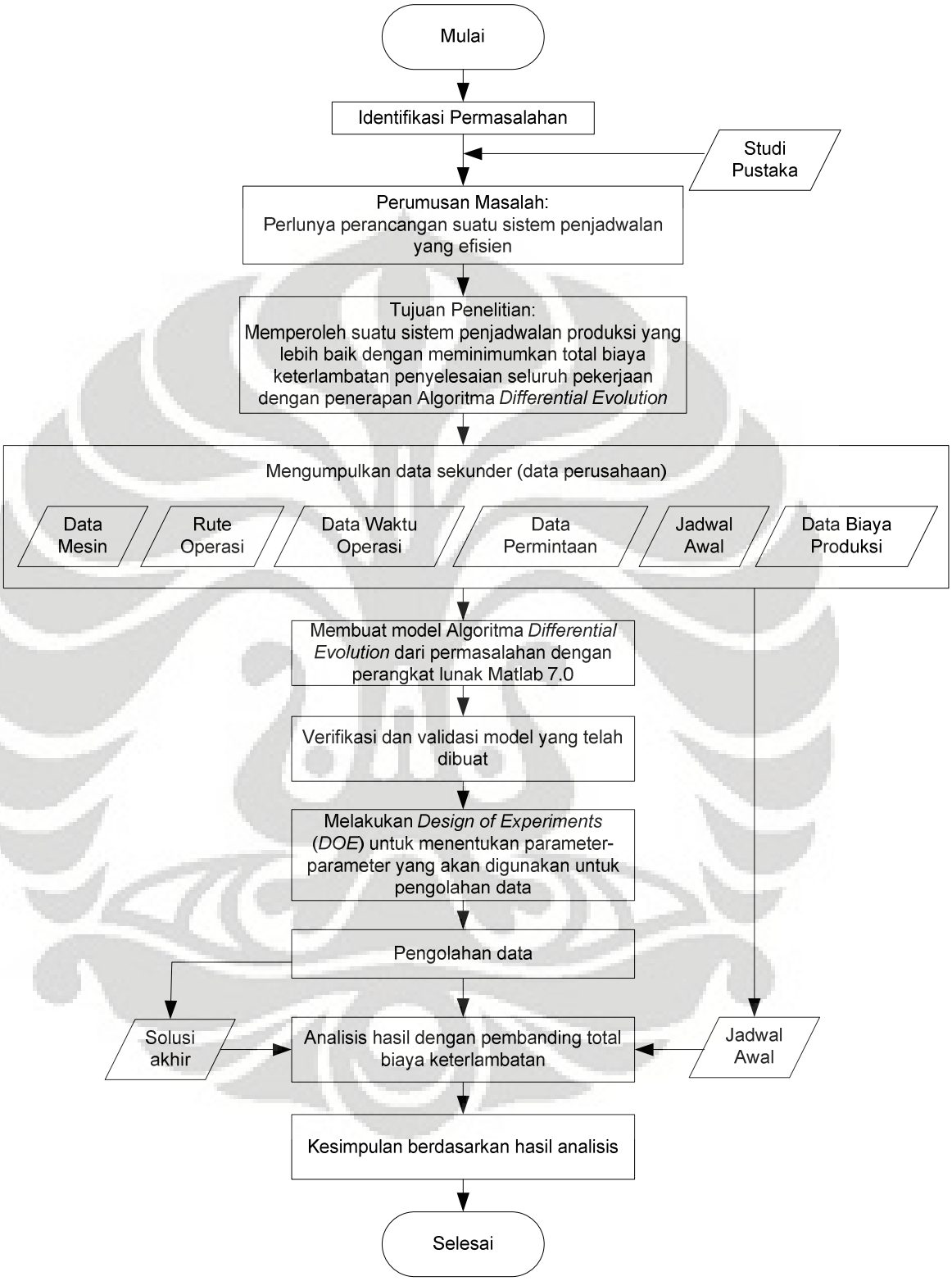
1. Rute proses produksi yang akan dibahas adalah rute proses produksi *Revo Frame* karena komponen inilah yang harus tiba terlebih dahulu di bagian perakitan untuk memulai perakitan *Hydraulic Excavator*.
2. Data besarnya waktu *set-up* dan perpindahan semua material sudah termasuk ke dalam waktu proses produksi yang bersangkutan.
3. Satu mesin hanya dapat memproses satu pekerjaan .
4. Penjadwalan bersifat statis dan *non-preemptive,* artinya semua order diterima di awal periode penjadwalan dan penjadwalan dilakukan di awal periode
5. Penelitian hanya memperhitungkan ketersediaan mesin.

6. Kondisi mesin produksi diasumsikan berjalan dengan normal, mengabaikan terjadinya *breakdown*.
7. Fungsi tujuan yang ingin diperoleh yaitu meminimumkan total biaya keterlambatan seluruh pekerjaan.
8. Biaya penalti atas keterlambatan setiap *job* diperoleh dari perbandingan total biaya produksi semua jenis *hydraulic excavato*r, yaitu produk yang akan dirakit dimana *Revo Frame* adalah salah satu komponennya.

1.6. Metodologi Penelitian

Berikut adalah langkah-langkah metodologi yang digunakan dalam penelitian ini, sebagaimana tergambarkan pada diagram alir dari metodologi penelitian (gambar 1.2.):

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 yang efisien.
4 Menentukan tujuan, yaitu memperoleh suatu sistem penjadwalan yang lebih baik (meminimumkan total biaya keterlambatan seluruh *job*).
5 Mengidentifikasi data yang dibutuhkan dan selanjutnya mengumpulkan data sekunder perusahaan.
6 Membuat model matematis dari permasalahan lalu dilakukan pembuatan program dengan perangkat lunak Matlab 7.0.
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 Membandingkan dan menganalisis solusi jadwal yang baru dengan jadwal yang lama, dimana faktor pembanding yaitu total biaya keterlambatan.
10 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.

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 organisasi[2]. 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 lain [3]:

- 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* $\{i=1,2,\ldots,n\}$ yang akan dijadwalkan pada *j* mesin $\{j=1,2,\ldots,m\}$. 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.*
- Keterlambatan (*lateness*), $L_i = C_i - d_i$, adalah selisih antara waktu penyelesaian *job i* dengan waktu tenggatnya. *Lateness* baru dapat dihitung setelah *job i* selesai menjalani semua proses, dan dapat bernilai negatif, nol, atau positif.
- Keterlambatan positif (*tardiness*), $T_i = \max(L_i, 0)$, adalah besarnya keterlambatan penyelesaian *job i*.
- Keterlambatan negatif (*earliness*), $T_i = \min(L_i, 0)$, adalah besarnya keterlambatan penyelesaian *job i*.

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*

  Preempion 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*

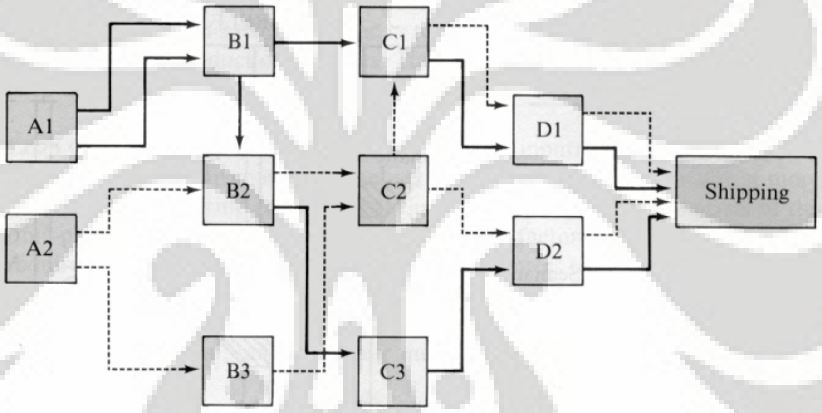
Job shop adalah suatu lingkungan manufaktur dimana *job-job* yang datang memiliki rute pengerjaan atau operasi yang seringkali tidak sama. Bentuk sederhana dari model ini mengasumsikan bahwa setiap *job* hanya melewati satu jenis mesin sebanyak satu kali dalam rutenya pada proses tersebut. Namun ada 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 sepeti di bawah ini:



Gambar 2.1. Contoh Rute Penjadwalan *Job Shop*

2.3. Penalti *Earliness* dan *Tardiness* dalam Penjadwalan Produksi

Seperti telah dibahas sebelumnya dalam istilah penjadwalan produksi, *earliness* dan *tardiness* adalah dua jenis *lateness* (keterlambatan). Lateness dapat didefinisikan dengan persamaan:

$$L_j = C_j - d_j \qquad (2.1)$$

dengan L_j adalah *lateness* tiap *job* (j = 1,2,…,n), C_j adalah total waktu penyelesaian tiap *job*, dan d_j adalah *due date* (batas waktu penyelesaian tiap *job*).

Earliness adalah keterlambatan negatif (nilai seberapa lebih awal waktu penyelesaian suatu *job* dibandingkan *due date* dari job tersebut) dan *tardiness* adalah keterlambatan positif (nilai seberapa lebih lambat waktu penyelesaian *job* dibandingkan due date dari *job* tersebut). Jika Ej dan Tj adalah representasi dari

earliness dan *tardiness* dari *job* j, maka *earliness* dan *tardiness* dapat didefinisikan sebagai berkut[4]:

$$E_j = \max\{0, d_j - C_j\} = (d_j - C_j)^+ \quad (2.2)$$

$$T_j = \max\{0, C_j - d_j\} = (C_j - d_j)^+ \quad (2.3)$$

Setiap job memiliki sebuah unit penalti *earliness* $\alpha_j > 0$ dan sebuah unit penalti atas *tardiness* $\beta_j > 0$. Dengan asumsi bahwa fungsi penalti adalah linear, maka fungsi objektif dasar masalah *earliness* dan *tardiness* untuk suatu jadwal S dapat ditulis dengan f(S), dimana:

$$f(S) = \sum_{j=1}^{n} \left[\alpha_j (d_j - C_j)^+ + \beta_j (C_j - d_j)^+\right] \quad (2.4)$$

Sesuai dengan definisi sebelumnya,

$$f(S) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j) \quad (2.5)$$

Penelitian untuk minimasi persamaan di atas pernah dilakukan (Nearchou, 2008) untuk mendorong penyelesaian setiap *job* dalam waktu yang sedekat mungkin dengan *due date*.

2.4. Metode Penyelesaian Masalah Penjadwalan Produksi

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

2.4.1. 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 memroses satu *job*, maka berdasarkan *priority dispatch rule* dipilih satu *job* yang memiliki prioritas tertinggi untuk selanjutnya diproses pada mesin tersebut.

[4] Kenneth R. Baker. *Elements of Sequencing and Scheduling*, 2001, p.5-1

Berikut ini adalah beberapa aturan yang merupakan *basic priority dispatch rules*, yaitu[5]:

- *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 (di – pi – 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.

[5] Everett E. Adam, JR and Ronald J. Ebert, *Op.Cit*, p.421

2.4.2. 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 dievalusi. Selama proses optimasi, pada setiap iterasi, solusi yang akan dievalusi 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 dievalusi 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 penginilisasian 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).

2.5. Algoritma *Differential Evolution* (DE)

2.5.1. 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 selesai[6].

[6] http://www.id.wikipedia.org/wiki/Algoritma, (last updated 4 May 2008, accessed 5 June 2008)

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*[7]. Sampai akhirnya pada tahun 1995, Storn dan Price menawarkan pilihan baru yaitu algoritma *differential evolution* (DE), yang telah melalui serangkaian tes dan menjadi kandidat terkuat sebagai algoritma terbaik[8]. 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)[9].

Untuk pertama kalinya DE dijelaskan oleh Price dan Storn di ICSI *technical report* pada tahun 1995[10]. 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

---

[7] Rasmus K. Ursen, "Differential Evolution Made Easy", *Technical Report* no. 1, 2005

[8] 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

[9] 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

[10] http://www.en.wikipedia.org/wiki/Differential_evolution, (last updated 8 March 2008, accessed 5 June 2008)

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 efisisen dibandingkan *simulated annealing* (metode berbasiskan 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 yang sederhana, mudah untuk diaplikasikan, cepat, dan tangguh[11]. Hingga kini, DE telah sukses diaplikasikan dalam berbagai bidang.

2.5.2.Konsep Dasar

Algoritma DE merupakan algoritma optimasi global yang efisisen, 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

[11] 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.

anggota populasi pada generasi ke-G. Populasi dinotasikan sebagai $P(G)=\{X_1, X_2,\ldots, X_{NP}\}$.

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 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 populasi[12]. 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 pencarian[13]. Setelah melalui beberapa generasi dan mencapai kriteria terminasi, maka algoritma ini akan konvergen ke kromosom terbaik yang menjadi solusi penyelesaian.

2.5.3. Tahapan Pengerjaan

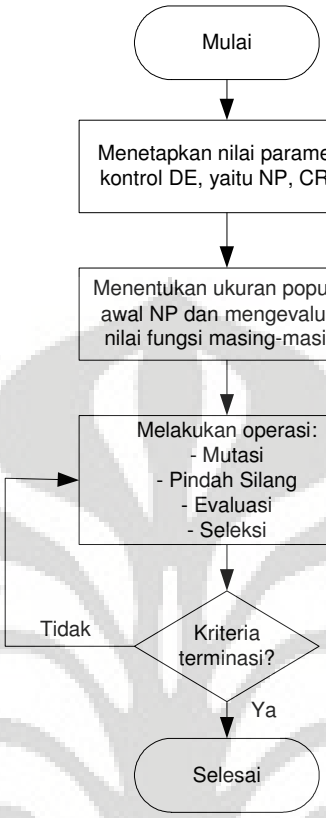
Tahapan pengerjaan algoritma DE dapat dilihat pada gambar 2.2. dan 2.3., yaitu sebagai berikut[14]:

---

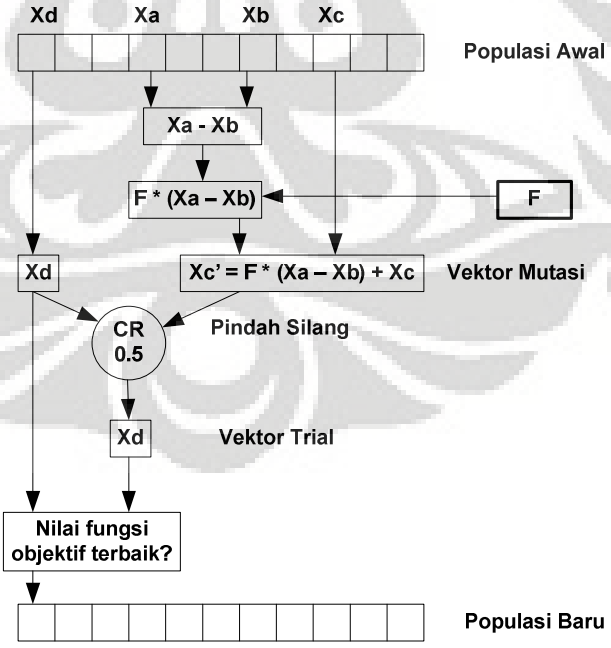
[12] Dervis Karaboga and Selcuk Okdem, *Op.Cit*, p.54

[13] 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.

[14] 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*

2.5.3.1. 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, efisisensi, 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 x 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.

2.5.3.2. 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.

2.5.3.3. 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:

$$X4' = F \times (X2 - X3) + X4 \quad (2.6)$$

Dimana:

X4' = Vektor Mutasi

F = Parameter kontrol mutasi

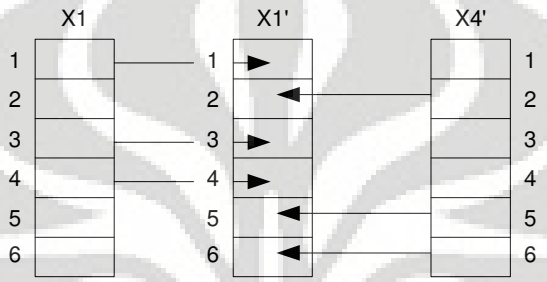
X2, X3, X4 = Vektor yang dipilih secara acak

2.5.3.4. 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 (I,1)
        else
                X1 (i,1) = X4' (I,1)
        end
end
```



Gambar 2.4. Proses Pindah Silang

2.5.3.5. Evaluasi Vektor Trial

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

2.5.3.6. 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 berikutnya[15].

[15] 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.

2.5.3.7.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

```
if OBF vektor target >= OBF vektor trial
        vektor target = vektor target
else
        vektor target = vektor trial
end
```

```
if OBF vektor target >= OBF vektor trial
        X selanjutnya = vektor target
else
        X selanjutnya = vektor trial
end
```

DE memiliki beberapa varian[16], 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[17]. Varian ini dianggap sebagai versi dasar dari DE.

2.5.4. 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 Price[18]. Adapun *pseudo code* algoritma DE yang akan digunakan adalah sebagai berikut:

[16] Nasimul Noman and Hitoshi Iba, "Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization", *GECCO'05*, Washington, DC, USA, 2005
[17] 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
[18] Tasgetiren et al, "Particle Swarm Optimization and Differential Evolution Algorithm for Job Shop Scheduling Problem", in *Proceedings of thr 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)

2.5.4.1. 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 berikut[19]:

- Individu target: X_i^t adalah individu ke-i anggota populasi pada generasi ke-t, dan diuraikan sebagai $X_i^t = | X_1^t, X_2^t, X_3^t, ..., X_n^t |$ dimana X_{ij}^t merupakan nilai dimensi individu ke-i terhadap dimensi ke-j (j=1,2,…,n).
- Individu mutan: V_i^t adalah individu ke-i anggota populasi pada generasi ke-t, dan diuraikan sebagai $V_i^t = | V_1^t, V_2^t, V_3^t, ..., V_n^t |$ dimana V_{ij}^t merupakan nilai dimensi individu ke-i terhadap dimensi ke-j (j=1,2,…,n).
- Individu *trial*: U_i^t adalah individu ke-i anggota populasi pada generasi ke-t, dan diuraikan sebagai $U_i^t = | U_1^t, U_2^t, U_3^t, ..., U_n^t |$ dimana U_{ij}^t merupakan nilai dimensi individu ke-i terhadap dimensi ke-j (j=1,2,…,n).
- Populasi target: P^t adalah kumpulan individu X_i^t sejumlah NP dalam populasi target pada generasi ke-t.

[19] *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 $\pi_i^t = [\pi_{i1}^t, \pi_{12}^t, ..., \pi_{i,n}^t]$, dimana π_{ij}^t merupakan penugasan operasi ke-j individu ke-i, pada iterasi ke-t.
- Konstanta mutasi: F∈ (0,2)
- Konstanta pindah silang: CR∈ (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.

2.5.4.2. 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 berikut[20]:

1. Tahap inisialisasi
   - Menetapkan t = 0, CR, F, dan NP
     Dimensi : jumlah *job*
   - Membangun individu awal sebanyak NP, yakni $\{X_i^t, i = 1,2,...,NP\}$, dimana $X_i^0 = [X_{i1}^0, X_{i2}^0, .., X_{i,nm}^0]$; $X_{ik}^0 = X\min + (X\max - X\min)xr$

[20] *Ibid.*, p.8

Xmin= -1; Xmax= 1; r = bilangan random (0-1)

- Melakukan operasi permutasi $\pi_i^0 = [\pi_{i1}^0, \pi_{12}^0, ..., \pi_{i,n}^0]$ 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,…,NP), untuk memilih individu target.

2. Meng-*update* generasi t=t+1
3. Membentuk populasi mutan

   Untuk setiap individu target, akan dicari individu mutan $V_i^{t+1} = [v_{i1}^{t+1}, v_{i2}^{t+1}, .., v_{i,nm}^{t+1}]$ 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 $U_i^{t+1} = [u_{i1}^{t+1}, u_{i2}^{t+1}, .., u_{i,nm}^{t+1}]$ diperoleh melalui operasi:

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

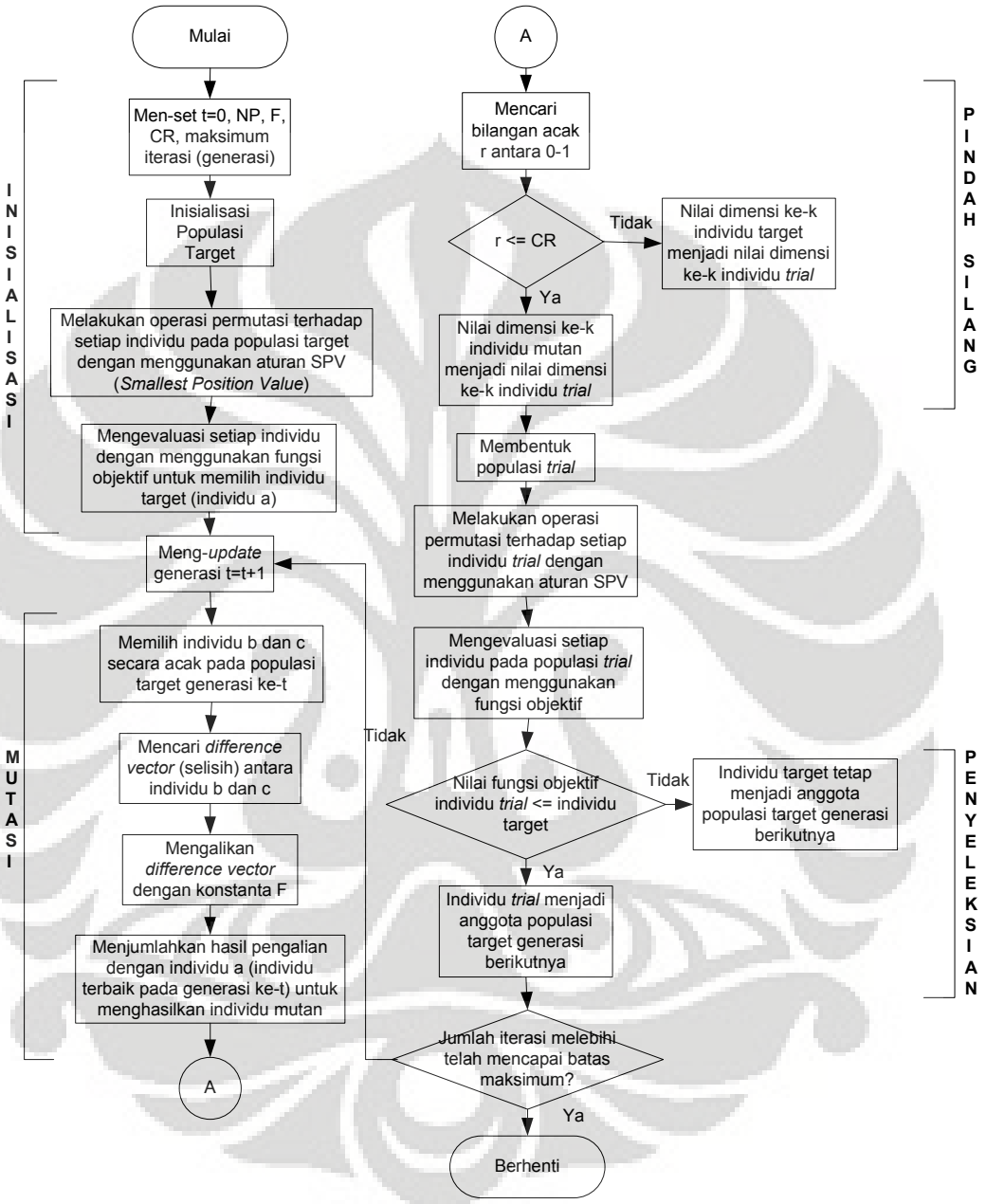
   CR adalah konstanta pindah silang pada *range* (0,1), dan r_{ik}^{t+1} adalah bilangan acak *uniform* antara 0 sampai 1.
5. Melakukan operasi permutasi *job*

   Operasi permutasi job dilakukan dengan menerapkan aturan SPV untuk melakukan operasi permutasi $\varphi_i^t = [\varphi_{i1}^t, \varphi_{i2}^t, .., \varphi_{i,nm}^t]$ (i = 1,2,…,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,…,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*

2.6. *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.

2.6.1. Tujuan *Design of Experiments*

Tujuan dari dilakukannya percobaan adalah[21]:

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 $z_1, z_2, ..., z_q$ minimum.

2.6.2. Tipe Percobaan

Ada empat tipe percobaan yaitu *trial and error*, *one factor at a time*, *full factorial*, *fractional factorial*[22].

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.

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.

---

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

[22] Mark A. Fryman, *Quality and Process Improvement,* USA, 2002, p.320.

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 hipótesis yang terjadi yaitu:

- Melihat apakah ada pengaruh dari faktor A:

H_0 : $\tau_1 = \tau_2 = ... = \tau_a = 0$ (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 : $\beta_1 = \beta_2 = ... = \beta_b = 0$ (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).

2.6.3. 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.

3. PENGUMPULAN DATA

3.1. Profil Perusahaan

PT X berdiri pada tahun 1982 sebagai perusahaan yang memproduksi dan merakit serangkaian alat-alat berat dan komponen-komponen terkait yang sebagian besar digunakan di bidang konstruksi, kehutanan, dan sektor-sektor infrastruktur lainnya. Pada awalnya kegiatan perusahaan hanya merakit alat-alat berat tetapi kemudian berkembang menjadi industri alat berat terpadu pertama di Indonesia dengan fasilitas produksi yang mencakup pabrik rangka dan komponen serta pabrik cor sejak awal dasawarsa 90-an.

Dengan kapasitas produksi yang telah ditingkatkan kini PT X telah mampu memproduksi *Hydraulic Excavator, Bulldozer, Motor Grader, dan Off-highway Dump Truck* yang banyak digunakan dalam kegiatan penambangan besar. Selain itu PT X juga telah mampu memproduksi *fabricated components* dan *steel cast components* untuk konsumsi dalam negeri maupun ekspor.

3.2. Pengumpulan Data Penelitian

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

1. Data jam kerja di PT X
2. Data pesanan, yaitu berupa jenis-jenis *Revo Frame* yang dipesan, jumlah yang dipesan, dan *due date* pesanan pada periode Januari sampai Februari 2008.
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
5. Data total biaya produksi produk *Hidraulic Excavator*, yaitu produk yang akan dirakit dimana Revo Frame merupakan salah satu komponen dari produk tersebut.

3.2.1. Data Pesanan dan Jam Kerja

Pesanan diterima kurang lebih 3 bulan sebelum *due date*-nya. Keadaan ini diharuskan sehingga PPIC dapat terlebih dahulu menyusun MPS, MRP dan rencana pemesanan barang. Apabila PPIC merasa bahwa permintaan dapat dipenuhi sesuai *due date*-nya, maka PPIC akan menyusun *Baan Process Planned MRP Order Lead Time Simulation*, yang merupakan jadwal kapan material harus diantar, kapan fabrikasi harus mulai produksi, kapan *assembly* harus mulai produksi, dan lain-lain. Dalam *Baan Process* juga disertai dengan *lead time* setiap operasi.

Barang jadi diproduksi dalam 2 jenis yaitu *in-house* dan *out-house* (dikerjakan oleh subkontraktor). Di *in-house* dikerjakan oleh 3 *stage* yaitu *foundry*, fabrikasi, dan perakitan/*assembly*. Setiap *stage* melakukan penjadwalan sendiri-sendiri, sesuai dengan *lead time* pengiriman ke *stage* selanjutnya. Proses ini merupakan rangkaian *shop floor control/SFC* yang terdiri dari *loading, sequencing,* dan *detailed scheduling.* Jadi, misalkan *stage* fabrikasi harus menyelesaikan pekerjaannya dan mengirimkannya ke *stage assembly* berdasarkan *lead time* dari PPIC, maka fabrikasi akan menyusun urutan penjadwalan *job* yang masuk ke dalam setiap mesin.

Untuk penelitian ini, data pesanan yang akan dibahas hanya untuk *stage* fabrikasi dan hanya pada komponen *Revo Frame.* Komponen ini diambil menjadi bahan penelitian ini karena komponen inilah yang harus pertama kali tiba di *stage assembly* untuk membuat produk jadi yang disebut *Hydraulic Excavator.* Jadi, ketika PPIC selesai membuat *Baan Process*, maka akan disusun *Production Order* (PO) yang berisi *due date* setiap *job* pada *stage* fabrikasi ke *stage assembly*. Setiap paket pesanan berbeda jenis, jumlah pesanannya, dan *due datenya*. *Due date* dihitung mulai dari PPIC memberikan PO ke bagian fabrikasi sampai jadwal barang dikirimkan ke bagian *assembly*. Dalam 1 minggu terdapat 5 hari kerja dimana 1 hari memiliki 2 shift dengan rincian waktu pada tabel 3.1.

Tabel 3.2. merupakan data pesanan untuk komponen *Revo Frame* pada *excavator* sesuai *due date*-nya. Proses pengerjaan untuk paket pesanan pada periode Januari sampai Februari 2008 dimulai dari tanggal 24 Januari 2008. Total *Revo Frame* yang harus diproduksi pada periode tersebut adalah 90 buah.

Tabel 3.1. Waktu Kerja PT X

Shift	Waktu Kerja	Durasi (menit)	Total Jam Kerja (menit)
Pagi	07.00 - 09.50	170	460
	10.00 - 12.00	120	
	13.00 - 14.50	110	
	15.00 - 16.00	60	
Malam	19.30 - 22.00	150	420
	22.30 - 01.00	150	
	01.10 - 03.10	120	
Total			880

Tabel 3.2. Data Pesanan Periode Januari – Februari 2008

Jenis *Revo Frame*	Januari 2008								Februari 2008											
	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12
PC100F-6			L													1			3	1
PC200-7			I				3	6	4			2								
PC300-7		4	B		2	4														
PC300-8			U			2	1						2							
PC300LC-8			R										1	3		3				
PC400LC-7													1	1						4

Jenis *Revo Frame*	Februari 2008										Total
	13	14	15	16	17	18	19	20	21	22	
PC100F-6			1			2	2				10
PC200-7											15
PC300-7	2	5	2							3	22
PC300-8									4		9
PC300LC-8							1	2	6	5	21
PC400LC-7	4						2	1			13
								Total			90

3.2.2. Rute dan Waktu Operasi

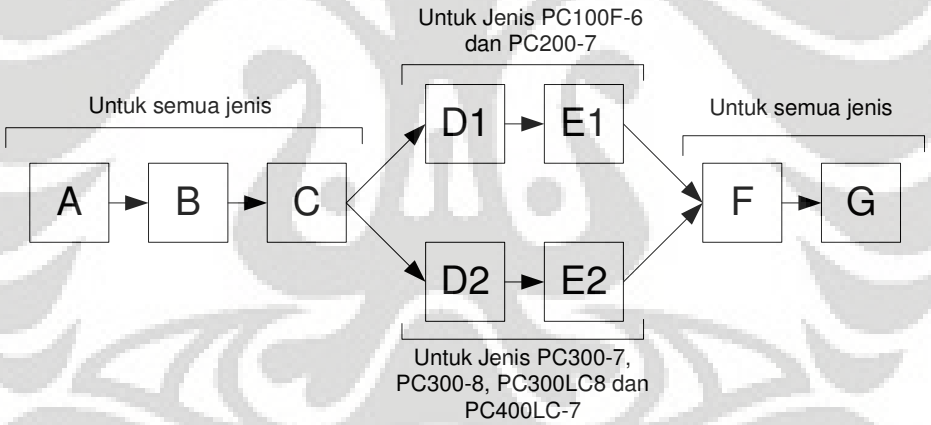
Rute proses operasi *Revo Frame* merupakan rute *job shop* karena tidak semua barang melewati rute yang sama. Pengerjaan *Revo Frame* terdiri dari 7 proses yang dimulai dengan *Straightening Press* (STP) dan diakhiri dengan proses *painting* (pengecatan). Setelah komponen selesai dicat, maka komponen tersebut siap untuk dikirim ke bagian perakitan. Tabel 3.3. merupakan urutan proses

produksi yang harus dilalui untuk memproduksi *Revo Frame*, jumlah mesin atau peralatan untuk tiap proses, dan alokasinya.

Tabel 3.3. Jumlah dan Alokasi Mesin setiap Rute Operasi

Kode Proses	Nama Proses	Jumlah Mesin/	Alokasi
A	*Straightening Press*	1	semua jenis
B	*Machining*	1	semua jenis
C	*Washing*	1	semua jenis
D	*Tack Welding*	2	1 mesin untuk PC100 dan PC200
			1 mesin untuk semua jenis PC300 dan PC400
E	*Semi Automated Welding*	2	1 mesin untuk PC100 dan PC200
			1 mesin untuk semua jenis PC300 dan PC400
F	*Finishing*	1	semua jenis (dikerjakan oleh 2 orang)
G	*Painting*	1	semua jenis

Untuk lebih jelas mengenai urutan rute operasi *Revo Frame,* dapat dilihat pada gambar 3.1. berikut ini



Gambar 3. Rute Operasi *Revo Frame*

Karena pengerjaan pada bagian fabrikasi lebih banyak pada proses yang sangat membutuhkan pekerja dalam operasi, maka waktu operasi dihitung berdasarkan *man hour* (waktu pekerja) dalam menyelesaikan suatu operasi. Kendala alat atau mesin diabaikan karena dianggap memiliki kapasitas tak terbatas sehingga utilisasi mesin serta alat ditentukan oleh utilisasi pekerja. Satu pekerja hanya mengerjakan satu operasi pada satu alat atau mesin, kecuali pada proses *finishing* yang dikerjakan oleh dua orang sehingga asumsi waktu operasi *finishing* adalah setengah dari *man hour*. Jadi, ketika *job* 1 proses A pada mesin A

selesai, selanjutnya pekerja langsung akan mengerjakan *job* 2 proses A pada mesin A juga. Tabel 3.4. merupakan waktu operasi *Revo Frame* untuk setiap proses. Waktu operasi *Revo Frame* pada suatu rute yang bernilai nol menunjukkan bahwa jenis *Revo Frame* yang bersangkutan tidak melewati rute tersebut (sesuai gambar 3.).

Tabel 3.4. Waktu Operasi *Revo Frame*

Jenis *Revo Frame*	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
PC100F-6	50	140	15	150	0	120	0	90	155
PC200-7	60	140	15	210	0	240	0	150	155
PC300-7	120	140	15	0	210	0	300	160	180
PC300-8	120	140	15	0	210	0	300	160	180
PC300LC-8	120	140	15	0	210	0	300	160	180
PC400LC-7	120	150	15	0	240	0	340	150	200

3.2.3. Biaya Keterlambatan Setiap Pesanan (Prioritas Pengerjaan)

Semua *Revo Frame* yang telah dikerjakan akan dikirimkan ke bagian perakitan untuk merakit *Hidraulic Excavator*. *Hidraulic Excavator* ini juga berbeda-beda jenisnya sesuai dengan jenis *Revo Frame*. Penyelesaian pengerjaan *Revo Frame* yang terlambat dapat menyebabkan terjadinya keterlambatan penyelesaian perakitan *Hidraulic Excavator*. Jika produk ini terlambat dikerjakan, ada kemungkinan tidak akan terjual pada periode yang bersangkutan dan akan tercatat sebagai persediaan untuk periode berikutnya. Hal ini berarti modal yang dikeluarkan untuk membuat *Hidraulic Excavator* tidak kembali pada periode itu. Maka, prioritas untuk mengerjakan setiap pesanan *Revo Frame* dinyatakan dengan biaya penalti (bobot) yang merupakan rasio antara total biaya produksi setiap jenis *Hidraulic Excavator* dengan total biaya produksi jenis yang terkecil. Data penalti dipakai apabila *Revo Frame* terlambat diantarkan ke bagian perakitan. Penalti dikenakan untuk setiap menit keterlambatan pada setiap penyelesaian pesanan *Revo Frame*. Besar penalti untuk setiap jenis dapat dilihat pada tabel 3.5. PC100F-6 merupakan produk dengan total biaya produksi terkecil. Jadi, penalti untuk setiap produk *Revo Frame* dapat dihitung dengan membagi total biaya produksi produk *Hidraulic Excavator* untuk jenis yang bersangkutan dengan total

biaya produksi PC100F-6. Semakin besar total biaya produksi, semakin besar pula penalti yang dimilikinya.

Tabel 3.5. Biaya Penalti Keterlambatan Setiap Jenis Produk

.

Jenis Hidraulic *Excavator*	Total Biaya Produksi ($)	Penalti Keterlambatan *Revo Frame* per Menit
PC100F-6	57223.68	1.00
PC200-7	65273.83	1.14
PC300-7	107133.50	1.87
PC300-8	111418.84	1.95
PC300LC-8	116989.78	2.04
PC400LC-7	136596.20	2.39

3.2.4. Jadwal Produksi PT X Periode Januari - Februari 2008

Metode yang digunakan dalam penjadwalan produksi di PT X adalah *Earliest Due Date*. Dalam membuat jadwal produksi, jumlah pesanan untuk setiap tipe dipecah lagi menjadi satuan *job.* Sebagai contoh, jika *Revo Frame* PC100-F6 dipesan sebanyak 10 unit, dikarenakan setiap mesin hanya bisa mengerjakan satu operasi sehingga susunan mesin disebut seri (tidak ada yang paralel), maka pesanan akan diproduksi tidak sekaligus 10 unit, tetapi dipecah menjadi 10 *job*. Jadi, ada 90 *job* yang akan dijadwalkan oleh PT KI pada periode Januari – Februari 2008. *Job-job* tersebut akan melewati 9 mesin (rute proses). Selanjutnya 90 pesanan tersebut akan dibuat urutan pekerjaannya. Berikut urutan pengerjaan setiap *job* oleh PT X, yaitu dengan metode *Earliest Due Date*.

Tabel 3.6. Penjadwalan PT X Lengkap (Periode Januari – Februari 2008)

Nomor Urut	Kode Komponen	Rute Proses (menit)									Due Date	Penalti
		A	B	C	D1	D2	E1	E2	F	G		
1	PC300-7 KI84-001	120	140	15	0	210	0	300	160	180	1760	1.87
2	PC300-7 KI84-002	120	140	15	0	210	0	300	160	180	1760	1.87
3	PC300-7 KI84-003	120	140	15	0	210	0	300	160	180	1760	1.87
4	PC300-7 KI84-004	120	140	15	0	210	0	300	160	180	1760	1.87
5	PC300-7 KI84-005	120	140	15	0	210	0	300	160	180	2640	1.87
6	PC300-7 KI84-006	120	140	15	0	210	0	300	160	180	2640	1.87
7	PC300-7 KI84-007	120	140	15	0	210	0	300	160	180	3520	1.87
8	PC300-7 KI84-008	120	140	15	0	210	0	300	160	180	3520	1.87
9	PC300-7 KI84-009	120	140	15	0	210	0	300	160	180	3520	1.87
10	PC300-7 KI84-010	120	140	15	0	210	0	300	160	180	3520	1.87
11	PC300-8 KI84-001	120	140	15	0	210	0	300	160	180	3520	1.95
12	PC300-8 KI84-002	120	140	15	0	210	0	300	160	180	3520	1.95
13	PC200-7 KI84-001	60	140	15	210	0	240	0	150	155	4400	1.14
14	PC200-7 KI84-002	60	140	15	210	0	240	0	150	155	4400	1.14
15	PC200-7 KI84-003	60	140	15	210	0	240	0	150	155	4400	1.14
16	PC300-8 KI84-003	120	140	15	0	210	0	300	160	180	4400	1.95
17	PC200-7 KI84-004	60	140	15	210	0	240	0	150	155	5280	1.14
18	PC200-7 KI84-005	60	140	15	210	0	240	0	150	155	5280	1.14
19	PC200-7 KI84-006	60	140	15	210	0	240	0	150	155	5280	1.14
20	PC200-7 KI84-007	60	140	15	210	0	240	0	150	155	5280	1.14
21	PC200-7 KI84-008	60	140	15	210	0	240	0	150	155	5280	1.14
22	PC200-7 KI84-009	60	140	15	210	0	240	0	150	155	5280	1.14
23	PC200-7 KI84-010	60	140	15	210	0	240	0	150	155	6160	1.14
24	PC200-7 KI84-011	60	140	15	210	0	240	0	150	155	6160	1.14
25	PC200-7 KI84-012	60	140	15	210	0	240	0	150	155	6160	1.14
26	PC200-7 KI84-013	60	140	15	210	0	240	0	150	155	6160	1.14
27	PC200-7 KI84-014	60	140	15	210	0	240	0	150	155	7040	1.14
28	PC200-7 KI84-015	60	140	15	210	0	240	0	150	155	7040	1.14
29	PC300-8 KI84-004	120	140	15	0	210	0	300	160	180	7920	1.95
30	PC300-8 KI84-005	120	140	15	0	210	0	300	160	180	7920	1.95
31	PC300LC-8 KI84-001	120	140	15	0	210	0	300	160	180	7920	2.04
32	PC400LC-7 KI84-001	120	150	15	0	240	0	340	150	200	7920	2.39
33	PC300LC-8 KI84-002	120	140	15	0	210	0	300	160	180	8800	2.04
34	PC300LC-8 KI84-003	120	140	15	0	210	0	300	160	180	8800	2.04
35	PC300LC-8 KI84-004	120	140	15	0	210	0	300	160	180	8800	2.04
36	PC400LC-7 KI84-002	120	150	15	0	240	0	340	150	200	8800	2.39
37	PC100F-6 KI84-001	50	140	15	150	0	120	0	90	155	9680	1.00
38	PC300LC-8 KI84-005	120	140	15	0	210	0	300	160	180	9680	2.04
39	PC300LC-8 KI84-006	120	140	15	0	210	0	300	160	180	9680	2.04
40	PC300LC-8 KI84-007	120	140	15	0	210	0	300	160	180	9680	2.04
41	PC100F-6 KI84-002	50	140	15	150	0	120	0	90	155	10560	1.00
42	PC100F-6 KI84-003	50	140	15	150	0	120	0	90	155	10560	1.00
43	PC100F-6 KI84-004	50	140	15	150	0	120	0	90	155	10560	1.00
44	PC100F-6 KI84-005	50	140	15	150	0	120	0	90	155	11440	1.00
45	PC400LC-7 KI84-003	120	150	15	0	240	0	340	150	200	11440	2.39
46	PC400LC-7 KI84-004	120	150	15	0	240	0	340	150	200	11440	2.39
47	PC400LC-7 KI84-005	120	150	15	0	240	0	340	150	200	11440	2.39
48	PC400LC-7 KI84-006	120	150	15	0	240	0	340	150	200	11440	2.39
49	PC300-7 KI84-011	120	140	15	0	210	0	300	160	180	12320	1.87
50	PC300-7 KI84-012	120	140	15	0	210	0	300	160	180	12320	1.87

Tabel 3.6. (sambungan)

51	PC400LC-7 KI84-007	120	150	15	0	240	0	340	150	200	12320	2.39
52	PC400LC-7 KI84-008	120	150	15	0	240	0	340	150	200	12320	2.39
53	PC400LC-7 KI84-009	120	150	15	0	240	0	340	150	200	12320	2.39
54	PC400LC-7 KI84-010	120	150	15	0	240	0	340	150	200	12320	2.39
55	PC300-7 KI84-013	120	140	15	0	210	0	300	160	180	13200	1.87
56	PC300-7 KI84-014	120	140	15	0	210	0	300	160	180	13200	1.87
57	PC300-7 KI84-015	120	140	15	0	210	0	300	160	180	13200	1.87
58	PC300-7 KI84-016	120	140	15	0	210	0	300	160	180	13200	1.87
59	PC300-7 KI84-017	120	140	15	0	210	0	300	160	180	13200	1.87
60	PC100F-6 KI84-006	50	140	15	150	0	120	0	90	155	14080	1.00
61	PC300-7 KI84-018	120	140	15	0	210	0	300	160	180	14080	1.87
62	PC300-7 KI84-019	120	140	15	0	210	0	300	160	180	14080	1.87
63	PC100F-6 KI84-007	50	140	15	150	0	120	0	90	155	14960	1.00
64	PC100F-6 KI84-008	50	140	15	150	0	120	0	90	155	14960	1.00
65	PC100F-6 KI84-009	50	140	15	150	0	120	0	90	155	15840	1.00
66	PC100F-6 KI84-010	50	140	15	150	0	120	0	90	155	15840	1.00
67	PC300LC-8 KI84-008	120	140	15	0	210	0	300	160	180	15840	2.04
68	PC400LC-7 KI84-011	120	150	15	0	240	0	340	150	200	15840	2.39
69	PC400LC-7 KI84-012	120	150	15	0	240	0	340	150	200	15840	2.39
70	PC300LC-8 KI84-009	120	140	15	0	210	0	300	160	180	16720	2.04
71	PC300LC-8 KI84-010	120	140	15	0	210	0	300	160	180	16720	2.04
72	PC400LC-7 KI84-013	120	150	15	0	240	0	340	150	200	16720	2.39
73	PC300-8 KI84-006	120	140	15	0	210	0	300	160	180	17600	1.95
74	PC300-8 KI84-007	120	140	15	0	210	0	300	160	180	17600	1.95
75	PC300-8 KI84-008	120	140	15	0	210	0	300	160	180	17600	1.95
76	PC300-8 KI84-009	120	140	15	0	210	0	300	160	180	17600	1.95
77	PC300LC-8 KI84-011	120	140	15	0	210	0	300	160	180	17600	2.04
78	PC300LC-8 KI84-012	120	140	15	0	210	0	300	160	180	17600	2.04
79	PC300LC-8 KI84-013	120	140	15	0	210	0	300	160	180	17600	2.04
80	PC300LC-8 KI84-014	120	140	15	0	210	0	300	160	180	17600	2.04
81	PC300LC-8 KI84-015	120	140	15	0	210	0	300	160	180	17600	2.04
82	PC300LC-8 KI84-016	120	140	15	0	210	0	300	160	180	17600	2.04
83	PC300-7 KI84-020	120	140	15	0	210	0	300	160	180	18480	1.87
84	PC300-7 KI84-021	120	140	15	0	210	0	300	160	180	18480	1.87
85	PC300-7 KI84-022	120	140	15	0	210	0	300	160	180	18480	1.87
86	PC300LC-8 KI84-017	120	140	15	0	210	0	300	160	180	18480	2.04
87	PC300LC-8 KI84-018	120	140	15	0	210	0	300	160	180	18480	2.04
88	PC300LC-8 KI84-019	120	140	15	0	210	0	300	160	180	18480	2.04
89	PC300LC-8 KI84-020	120	140	15	0	210	0	300	160	180	18480	2.04
90	PC300LC-8 KI84-021	120	140	15	0	210	0	300	160	180	18480	2.04

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 dilakukan[23]. 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 ilmiah[24]

4.1.1. Langkah-Langkah Penyusunan Algoritma

Diagram alir penyusunan program algoritma *Differential Evolution* (DE) untuk menyelesaikan penjadwalan *job shop* pada PT X 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.

     Pada awal perhitungan sebelum generasi (iterasi) dimulai, dilakukan input parameter-parameter, yaitu ukuran populasi (NP), operator mutasi (F), dan

[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 June 2008)

operator pindah silang (CR). Ketiga parameter ini ditambah dengan jumlah iterasi yang akan digunakan ditentukan sesuai dengan hasil *Design of Experiments* untuk pemilihan parameter-paremeter yang akan dibahas lebih lanjut. Parameter-paremeter yang digunakan dengan mempertimbangkan hasil *DOE* tersebut yaitu: NP = 100; F = 0,6; CR = 0,5; dan jumlah iterasi = 2000.

- Menentukan populasi awal (populasi target awal)

  Ukuran populasi menyatakan jumlah individu dalam populasi. Satu individu dinyatakan dengan satu kolom. Tiap individu memiliki 90 gen/dimensi (jumlah *job*/pesanan), dinyatakan dengan jumlah baris. Populasi awal dibentuk dengan setiap dimensi untuk setiap individu dicari secara acak dengan rumus:

  batas_bawah + (batas_atas – batas_bawah) x 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 100 individu (nilai NP) dan setiap individu terdiri dari 90 dimensi, maka populasi awal merupakan matriks berukuran 90 x 100. 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-90 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 90 dengan nilai setiap dimensinya berupa indeks hasil pengurutan. Indeks-indeks tiap individu inilah yang nantinya akan menjadi urutan-urutan pengerjaan 90 pesanan sesuai data penelitian. Sebagai contoh, jika ada tiga *job* yang dikerjakan dan terdapat tiga dimensi pada suatu individu, yaitu 0,26; -0,54; dan 0,78; 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 (total biaya keterlambatan). 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 total biaya keterlambatan terkecil 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 total biaya keterlambatan 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 2000 iterasi berdasarkan percobaan yang telah dilakukan. Jadi, jika iterasi sudah mencapai 2000 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.

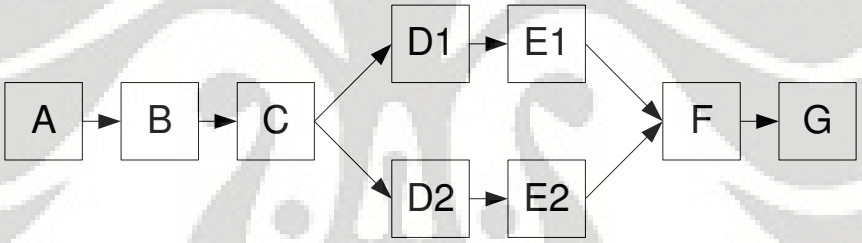
4.1.2. Verifikasi dan Validasi Program

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 total biaya keterlambatan atas nilai keterlambatan yang positif (*tardiness*). Untuk bagian-bagian selanjutnya, keterlambatan yang dimaksud adalah keterlambatan positif (*tardiness*). Artinya, keterlambatan hanya dihitung jika waktu penyelesaian lebih lama daripada waktu seharusnya diselesaikan (tidak lebih awal). Nilai keterlambatan negatif (*earliness*) tidak diperhitungkan. Oleh karena itu, pada fungsi tujuan, jika keterlambatan bernilai

negatif, maka keterlambatan yang negatif tersebut akan berubah menjadi bernilai nol sebelum dikalikan dengan penalti.

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* memiliki 7 buah operasi dengan 9 buah mesin yang akan dilewati. 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

Job	Rute dan Waktu Operasi (menit)									Due Date (menit)	Penalti
	A	B	C	D1	D2	E1	E2	F	G		
1	80	60	30	0	50	0	60	60	50	400	10
2	70	40	40	40	0	50	0	80	50	500	15
3	80	50	70	0	70	0	80	50	80	520	20
4	60	40	50	0	90	0	70	50	70	460	25
5	70	90	50	30	0	40	0	60	60	480	30

Tabel 4.2. Parameter yang Digunakan dalam Validasi

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

2.1.1. Hasil *Run* Program

Hasil *run* program menunjukkan urutan pengerjaan terbaik yaitu 4-5-2-3-1 dengan total keterlambatan yaitu 7550 menit.

2.1.2. 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
0.2423	0.5073	0.319	0.2468	-0.3792
0.1204	0.3193	-0.6333	0.3718	0.5582
-0.5119	-0.5719	0.2731	0.3547	-0.3854
0.644	0.2042	-0.6594	0.7537	0.8534
-0.4736	0.2099	0.0792	-0.9742	0.3574

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 -0,5119; -0,4736; 0,1204; 0,2423; dan 0,644. Bilangan-bilangan tersebut berturut-turut terdapat pada kolom ke 3, 5, 2, 1, dan 4. Jadi, urutan pengerjaan *job* pada individu (kolom) tersebut adalah 3-5-2-1-4. 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
3	3	4	5	3
5	4	2	1	1
2	5	5	3	5
1	2	3	2	2
4	1	1	4	4

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

   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 dikurangi dengan *due date* dari *job* yang bersangkutan, dan diperoleh nilai keterlambatan. Apabila nilai keterlambatan bernilai negatif (*earliness*), nilai tersebut akan berubah menjadi nol.
   - Selanjutnya, keterlambatan setiap *job* dikalikan dengan penalti (bobot) masing-masing sehingga didapatkan biaya keterlambatan setiap *job*. Lalu biaya keterlambatan seluruh *job* dijumlahkan sehingga didapatlah total biaya keterlambatan.

   Tabel-tabel di bawah ini menunjukkan proses perhitungan total biaya keterlambatan untuk setiap individu dalam populasi.

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

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
3	80	130	200	200	270	200	350	400	480
5	150	240	290	320	290	360	350	460	540
2	220	280	330	370	330	420	350	540	590
1	300	360	390	390	440	420	500	600	650
4	360	400	450	450	540	450	610	660	730

Tabel 4.6. Perhitungan Total Biaya Keterlambatan Individu 1 Populasi Target

Job	3	5	2	1	4
Waktu penyelesaian	480	540	590	650	730
Due date	520	480	500	400	460
Penalti	20	30	15	10	25
Keterlambatan	-40	60	90	250	270
Biaya keterlambatan	0	1800	1350	2500	6750
Total Biaya Keterlambatan	12400				

Tabel 4.7. Perhitungan Waktu Produksi setiap *Job* Individu 2 Populasi Target

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
3	80	130	200	200	270	200	350	400	480
4	140	180	250	250	360	250	430	530	610
5	210	300	350	380	360	420	430	480	540
2	280	340	390	430	390	480	430	610	660
1	360	420	450	450	500	480	560	670	720

Tabel 4.8. Perhitungan Total Biaya Keterlambatan Individu 2 Populasi Target

Job	3	4	5	2	1
Waktu penyelesaian	480	610	540	660	720
Due date	520	460	480	500	400
Penalti	20	25	30	15	10
Keterlambatan	-40	150	60	160	320
Biaya keterlambatan	0	3750	1800	2400	3200
Total Biaya Keterlambatan	11150				

Tabel 4.9. Perhitungan Waktu Produksi setiap *Job* Individu 3 Populasi Target

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
4	60	100	150	150	240	150	310	430	500
2	130	170	210	250	240	300	310	380	430
5	200	290	340	370	340	410	340	490	560
3	280	340	410	410	480	410	560	610	690
1	360	420	450	450	530	450	620	680	740

Tabel 4.10. Perhitungan Total Biaya Keterlambatan Individu 3 Populasi Target

Job	4	2	5	3	1
Waktu penyelesaian	500	430	560	690	740
Due date	460	500	480	520	400
Penalti	25	15	30	20	10
Keterlambatan	40	-70	80	170	340
Biaya keterlambatan	1000	0	2400	3400	3400
Total Biaya Keterlambatan	10200				

Tabel 4.11. Perhitungan Waktu Produksi setiap *Job* Individu 4 Populasi Target

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
5	70	160	210	240	210	280	210	340	400
1	150	220	250	250	300	280	360	420	470
3	230	280	350	350	420	350	500	610	690
2	300	340	390	430	420	480	500	560	610
4	360	400	450	450	540	480	610	660	760

Tabel 4.12. Perhitungan Total Biaya Keterlambatan Individu 4 Populasi Target

Job	5	1	3	2	4
Waktu penyelesaian	400	470	690	610	760
Due date	480	400	520	500	460
Penalti	30	10	20	15	25
Keterlambatan	-80	70	170	110	300
Biaya keterlambatan	0	700	3400	1650	7500
Total Biaya Keterlambatan	13250				

Tabel 4.13. Perhitungan Waktu Produksi setiap *Job* Individu 5 Populasi Target

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
3	80	130	200	200	270	200	350	400	480
1	160	220	250	250	320	250	410	470	530
5	230	320	370	400	370	440	410	530	590
2	300	360	410	450	410	500	410	610	660
4	360	400	460	460	550	500	620	670	740

Tabel 4.14. Perhitungan Total Biaya Keterlambatan Individu 5 Populasi Target

Job	3	1	5	2	4
Waktu penyelesaian	480	530	590	660	740
Due date	520	400	480	500	460
Penalti	20	10	30	15	25
Keterlambatan	-40	130	110	160	280
Biaya keterlambatan	0	1300	3300	2400	7000
Total Biaya Keterlambatan	14000				

4. Setelah didapat nilai total biaya keterlambatan seluruh individu, maka dipilih individu yang memiliki total biaya keterlambatan terkecil, yaitu individu 3. Individu 3 memiliki nilai total biaya keterlambatan sebesar 10200 menit dengan urutan pengerjaan *job* 4-2-5-3-1. 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 total biaya keterlambatan terkecil, yaitu individu 3. Semua kolom (individu) pada vektor target berisi bilangan-bilangan yang sama, yaitu individu 3 pada populasi target.

Tabel 4.15. Vektor Target

Individu				
1	2	3	4	5
0.3190	0.3190	0.3190	0.3190	0.3190
-0.6333	-0.6333	-0.6333	-0.6333	-0.6333
0.2731	0.2731	0.2731	0.2731	0.2731
-0.6594	-0.6594	-0.6594	-0.6594	-0.6594
0.0792	0.0792	0.0792	0.0792	0.0792

   - 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 4.16. Vektor Acak 1 (Hasil *Run* Program)

Individu				
1	2	3	4	5
0.2423	0.2423	0.5073	0.5073	-0.3792
0.1204	0.1204	0.3193	0.3193	0.5582
-0.5119	-0.5119	-0.5719	-0.5719	-0.3854
0.6440	0.6440	0.2042	0.2042	0.8534
-0.4736	-0.4736	0.2099	0.2099	0.3574

Tabel 4.17. Vektor Acak 2 (Hasil *Run* Program)

Individu				
1	2	3	4	5
0.5073	0.5073	0.2468	0.2468	0.2423
0.3193	0.3193	0.3718	0.3718	0.1204
-0.5719	-0.5719	0.3547	0.3547	-0.5119
0.2042	0.2042	0.7537	0.7537	0.6440
0.2099	0.2099	-0.9742	-0.9742	-0.4736

- Setelah didapat vektor target, vektor acak 1, dan vektor acak 2, kemudian terbentuk populasi mutan (tabel 4.18.). 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 0,1070 didapatkan dari 0,3190 + (0,2423 – 0,5073)*0,6. Angka-angka 0,3190; 0,2423; dan 0,5073 secara berurutan adalah angka-angka di baris pertama kolom pertama dari vektor target, vektor acak 1, dan vektor acak 2. Sedangkan 0,6 adalah nilai dari operator mutasi (konstanta mutasi).

Tabel 4.18. Populasi Mutan

Individu				
1	2	3	4	5
0.1070	0.1070	0.5274	0.5274	-0.1782
-0.7923	-0.7923	-0.6753	-0.6753	-0.2831
0.3210	0.3210	-0.4682	-0.4682	0.3743
-0.3076	-0.3076	-1.0989	-1.0989	-0.4919
-0.4676	-0.4676	1.0265	1.0265	0.7440

6. 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 0,5. Jika bilangan acak yang keluar bernilai kurang dari atau sama dengan 0,5; maka nilai suatu gen pada populasi *trial* berasal dari gen yang bersangkutan dari populasi mutan. Begitu pun sebaliknya, jika lebih dari 0,5; maka berasal dari populasi target. Tabel 4.19 menunjukkan populasi *trial*. Gen-gen yang bercetak tebal pada tabel tersebut adalah gen-gen yang berasal dari populasi mutan.

Tabel 4.19. Populasi *Trial*

Individu				
1	2	3	4	5
0.2423	0.5073	**0.5274**	**0.5274**	-0.3792
0.1204	0.3193	-0.6333	0.3718	0.5582
0.3210	**0.3210**	**-0.4682**	**-0.4682**	**0.3743**
0.6440	**-0.3076**	**-1.0989**	0.7537	**-0.4919**
-0.4736	0.2099	0.0792	**1.0265**	0.3574

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 4.20.
8. Setelah didapatkan urutan *job* dari setiap individu populasi *trial*, maka dari setiap individu tadi, dicari total biaya keterlambatannya masing-masing (tabel 4.21. sampai tabel 4.30.).

Tabel 4.20. Permutasi Populasi *Trial*

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

Tabel 4.21. Perhitungan Waktu Produksi setiap *Job* Individu 1 Populasi *Trial*

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
5	70	160	210	240	210	280	210	340	400
2	140	200	250	290	250	340	250	420	470
1	220	280	310	310	360	340	420	480	530
3	300	350	420	420	490	420	570	620	700
4	360	400	470	470	580	470	650	700	770

Tabel 4.22. Perhitungan Total Biaya Keterlambatan Individu 1 Populasi *Trial*

Job	5	2	1	3	4
Waktu penyelesaian	400	470	530	700	770
Due date	480	500	400	520	460
Penalti	30	15	10	20	25
Keterlambatan	-80	-30	130	180	310
Biaya keterlambatan	0	0	1300	3600	7750
Total Biaya Keterlambatan	12650				

Tabel 4.23. Perhitungan Waktu Produksi setiap *Job* Individu 2 Populasi *Trial*

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
4	60	100	150	150	240	150	310	360	430
5	130	220	270	300	270	340	310	420	490
2	200	260	310	350	310	400	310	500	550
3	280	330	400	400	470	400	550	600	680
1	360	420	450	450	520	450	610	670	730

Tabel 4.24. Perhitungan Total Biaya Keterlambatan Individu 2 Populasi *Trial*

Job	4	5	2	3	1
Waktu penyelesaian	430	490	550	680	730
Due date	460	480	500	520	400
Penalti	25	30	15	20	10
Keterlambatan	-30	10	50	160	330
Biaya keterlambatan	0	300	750	3200	3300
Total Biaya Keterlambatan	7550				

Tabel 4.25. Perhitungan Waktu Produksi setiap *Job* Individu 3 Populasi *Trial*

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
4	60	100	150	150	240	150	310	430	500
2	130	170	210	250	240	300	310	380	430
3	210	260	330	330	400	330	480	530	610
5	280	370	420	450	420	490	480	590	670
1	360	430	460	460	510	490	570	650	720

Tabel 4.26. Perhitungan Total Biaya Keterlambatan Individu 3 Populasi *Trial*

Job	4	2	3	5	1
Waktu penyelesaian	500	430	610	670	720
Due date	460	500	520	480	400
Penalti	25	15	20	30	10
Keterlambatan	40	-70	90	190	320
Biaya keterlambatan	1000	0	1800	5700	3200
Total Biaya Keterlambatan	11700				

Tabel 4.27. Perhitungan Waktu Produksi setiap *Job* Individu 4 Populasi *Trial*

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
3	80	130	200	200	270	200	350	460	540
2	150	190	240	280	270	330	350	410	460
1	230	290	320	320	370	330	430	520	590
4	290	330	380	380	470	380	540	590	660
5	360	450	500	530	500	570	540	650	720

Tabel 4.28. Perhitungan Total Biaya Keterlambatan Individu 4 Populasi *Trial*

Job	3	2	1	4	5
Waktu penyelesaian	540	460	590	660	720
Due date	520	500	400	460	480
Penalti	20	15	10	25	30
Keterlambatan	20	-40	190	200	240
Biaya keterlambatan	400	0	1900	5000	7200
Total Biaya Keterlambatan	14500				

Tabel 4.29. Perhitungan Waktu Produksi setiap *Job* Individu 5 Populasi *Trial*

Job	Waktu Proses (menit)								
	A	B	C	D1	D2	E1	E2	F	G
4	60	100	150	150	240	150	310	360	430
1	140	200	230	230	290	230	370	430	480
5	210	300	350	380	350	420	370	490	550
3	290	350	420	420	490	420	570	680	760
2	360	400	460	500	490	550	570	630	680

Tabel 4.30. Perhitungan Total Biaya Keterlambatan Individu 5 Populasi *Trial*

Job	4	1	5	3	2
Waktu penyelesaian	430	480	550	760	680
Due date	460	400	480	520	500
Penalti	25	10	30	20	15
Keterlambatan	-30	80	70	240	180
Biaya keterlambatan	0	800	2100	4800	2700
Total Biaya Keterlambatan	10400				

9. Kemudian dilakukan tahap seleksi. Setelah didapat nilai total biaya keterlambatan setiap individu pada populasi *trial*, nilai-nilai tersebut dibandingkan dengan nilai total biaya keterlambatan setiap individu populasi target. Perbandingan tersebut dapat dilihat pada tabel 4.31. Sebagai contoh, nilai total biaya keterlambatan individu 1 populasi target dibandingkan dengan nilai total biaya keterlambatan individu 1 populasi *trial*. Jika individu 1 populasi target memiliki nilai total biaya keterlambatan yang lebih kecil dari individu 1 populasi *trial*, maka nilai populasi target, total biaya keterlambatan, dan urutan (permutasi) untuk iterasi selanjutnya (iterasi 2) berturut-turut diambil dari individu 1 populasi target, total biaya keterlambatan individu 1

populasi target, dan permutasi individu 1 populasi target, begitupun sebaliknya jika nilai total biaya keterlambatan populasi target lebih besar atau sama dengan nilai total biaya keterlambatan populasi *trial*.

Tabel 4.31. Perbandingan antara Total Biaya Keterlambatan Populasi Target dan Populasi *Trial*

Total Biaya Keterlambatan Populasi Target (menit)	Total Biaya Keterlambatan Populasi *Trial* (menit)	Asal Populasi Target Iterasi Selanjutnya
12400	12650	Populasi Target
11150	**7550**	*Populasi Trial*
10200	11700	Populasi Target
13250	14500	Populasi Target
14000	**10400**	*Populasi Trial*

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 4.32. menunjukkan populasi target baru dimana populasi ini akan membentuk vektor target, dan seterusnya pada iterasi 2. Tabel 4.33. menunjukkan permutasi (urutan pengerjaan *job*) dari populasi target baru.

Tabel 4.32. Populasi Target Baru

Individu				
1	2	3	4	5
0.2423	0.5073	0.3190	0.2468	-0.3792
0.1204	0.3193	-0.6333	0.3718	0.5582
-0.5119	0.3210	0.2731	0.3547	-0.3743
0.6440	-0.3076	-0.6594	0.7537	0.8534
-0.4736	0.2099	0.0792	-0.9742	0.3574

Tabel 4.33. Permutasi Populasi Target Baru

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

11. Karena proses validasi ini hanya menggunakan satu iterasi, jadi program berakhir hanya sampai pada tahap seleksi . Jadi, dari setiap perbandingan nilai total biaya keterlambatan terkecil setiap gen antara populasi target dan populasi *trial*, langsung diambil gen yang memiliki total biaya keterlambatan terkecil. Nilai inilah yang menjadi solusi penyelesaian data *dummy* untuk validasi ini. Untuk lebih jelasnya dapat dilihat pada tabel 4.34.

Tabel 4.34. Hasil Terbaik Perhitungan Data *Dummy*

Total Biaya Keterlambatan Populasi Target (menit)	Total Biaya Keterlambatan Populasi *Trial* (menit)	Total Biaya Keterlambatan Populasi Target Baru (menit)
12400	12650	12400
11150	**7550**	7550
10200	11700	10200
13250	14500	13250
14000	**10400**	10400

Dari tabel terlihat bahwa dari setiap perbandingan total biaya keterlambatan populasi target dan populasi *trial*, nilai total biaya keterlambatan yang terkecil adalah 7550 menit, nilai tersebut terletak pada individu 2 pada nilai-nilai total biaya keterlambatan populasi target yang baru. Jadi, sesuai tabel 4.33, urutan pengerjaan *job* adalah 4 – 5 – 2 – 3 – 1 (individu 2 pada permutasi populasi target baru).

12. Maka didapatlah nilai total biaya keterlambatan terbaik, yaitu 7550 menit dengan urutan pengerjaan *job* 4 – 5 – 2 – 3 – 1.
13. Nilai total biaya keterlambatan 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 PT X yang ada pada tabel 3.6, yaitu rute dan waktu proses, *due date*, serta penalti untuk 90 *job*. Data ini akan dibuat ke dalam suatu matriks yang berukuran 90 x 11, artinya terdapat 90 *job* (baris) dan 11 kolom. Untuk kolom 1 hingga kolom 9 berisi waktu produksi (menit) setiap operasi pada setiap *job.* Kolom 10 berisi *due date* setiap *job* dan kolom 11 berisi penalti 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 total biaya keterlambatan sebagai output. DOE dilakukan untuk mendapatkan hasil yang maksimum dalam mendapatkan minimum total biaya keterlambatan. 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 desig*n. 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 x 5 = 405 percobaan.

Nilai-nilai setiap level untuk setiap faktor untuk DOE dapat dilihat pada tabel 4.35. 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 4.35. 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 14 dengan α = 5%. Setelah dilakukan percobaan sebanyak 405 kali dengan output total biaya keterlambatan (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 4.36. 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 total biaya keterlambatan.

Tabel 4.36. Kombinasi Parameter Terbaik

Faktor	Nilai
Ukuran Populasi (NP)	100
Jumlah Iterasi (JI)	2000
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) Pentium(R) CPU 1.50GHz 1.5GHz, 512 MB of RAM. *Script M File* program untuk perhitungan jadwal PT X 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 PT X

Tabel 4.37. Hasil Perhitungan Jadwal PT X

Hasil Penjadwalan PT X										
Urutan 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**
Jumlah Pesanan yang Terlambat	18									
Total Biaya Keterlambatan (menit)	33190									
Total Keterlambatan (menit)	16690									
Total Waktu Penyelesaian (menit)	20865									

4.3.2. Hasil Penjadwalan dengan Algoritma DE

Setelah parameter-parameter dari permasalahan penjadwalan ini ditentukan, maka program yang telah dibuat di-*run* sebanyak lima kali. Hasil yang terbaik diambil dari kelima hasil *run* ini, yaitu hasil *run* yang memiliki nilai total biaya keterlambatan terkecil. Tabel 4.38. sampai tabel 4.42. berikut ini menunjukkan hasil *run* sebanyak lima kali.

Tabel 4.38. Hasil *Run* 1

Hasil *Run* 1										
Urutan Pengerjaan	3	15	2	**1**	60	**4**	5	**6**	21	19
	12	22	11	**8**	**7**	13	**10**	**16**	**9**	14
	30	40	31	17	20	24	18	32	23	26
	25	33	39	29	27	36	28	35	41	34
	45	38	58	53	43	47	54	48	46	52
	51	49	50	59	65	55	57	56	62	37
	61	68	69	42	74	71	67	44	79	72
	82	70	77	78	80	81	**76**	**75**	89	**86**
	88	**90**	64	**87**	**73**	66	**84**	63	**83**	**85**
Jumlah Pesanan yang Terlambat	18									
Total Biaya Keterlambatan (menit)	28549									
Total Keterlambatan (menit)	14870									
Total Waktu Penyelesaian (menit)	20845									
Waktu Komputasi (detik)	549,48									

Tabel 4.39. Hasil *Run* 2

Hasil *Run* 2										
Urutan Pengerjaan	3	15	1	2	**4**	24	5	6	19	7
	11	**10**	22	21	**12**	**8**	**16**	**9**	14	32
	13	31	48	30	36	17	20	29	18	23
	25	26	35	33	28	34	45	40	39	52
	38	27	50	46	63	65	47	54	51	53
	49	57	55	56	37	59	58	62	67	61
	42	41	68	66	69	43	81	79	44	72
	71	74	70	82	80	64	77	78	**75**	**76**
	90	**86**	**89**	**88**	**87**	**73**	**84**	**85**	60	**83**
Jumlah Pesanan yang Terlambat	16									
Total Biaya Keterlambatan (menit)	28395									
Total Keterlambatan (menit)	14775									
Total Waktu Penyelesaian (menit)	20845									
Waktu Komputasi (detik)	548,509									

Tabel 4.40. Hasil *Run* 3

Hasil *Run* 3										
Urutan Pengerjaan	1	19	4	**2**	13	**3**	5	**6**	18	4
	11	15	12	22	**7**	**9**	**8**	**16**	**10**	35
	17	32	33	39	29	25	34	20	21	30
	31	23	26	24	36	27	28	66	45	40
	37	53	38	57	41	46	42	52	47	54
	48	51	49	43	50	56	59	63	58	60
	55	62	61	68	69	44	80	79	67	81
	70	72	65	74	71	77	78	82	64	**73**
	76	89	**90**	**88**	**86**	**87**	**83**	**85**	**84**	**75**
Jumlah Pesanan yang Terlambat	18									
Total Biaya Keterlambatan (menit)	28450									
Total Keterlambatan (menit)	14780									
Total Waktu Penyelesaian (menit)	20845									
Waktu Komputasi (detik)	580,615									

Tabel 4.41. Hasil *Run* 4

Hasil *Run* 4										
Urutan Pengerjaan	1	19	3	18	**4**	**2**	6	**5**	14	23
	17	11	12	13	**8**	**7**	**9**	**16**	**10**	38
	15	31	48	36	20	40	30	22	29	21
	24	32	49	26	25	28	35	27	34	33
	42	46	39	47	45	50	53	54	58	52
	51	59	41	37	56	55	57	62	43	60
	61	68	67	44	69	76	63	75	72	77
	66	82	70	71	80	81	79	65	**78**	**74**
	89	**88**	**87**	**86**	64	**90**	**73**	**84**	**85**	**83**
Jumlah Pesanan yang Terlambat	18									
Total Biaya Keterlambatan (menit)	28400									
Total Keterlambatan (menit)	14780									
Total Waktu Penyelesaian (menit)	20845									
Waktu Komputasi (detik)	591,601									

Tabel 4.42. Hasil *Run* 5

Hasil *Run* 5										
Urutan Pengerjaan	1	15	4	2	**3**	19	5	14	6	12
	11	**8**	13	**9**	22	**7**	**16**	**10**	17	24
	31	29	35	34	47	20	18	21	32	23
	30	25	39	26	27	36	46	33	28	40
	44	38	45	41	49	53	51	37	54	48
	50	52	42	59	43	55	57	56	58	62
	61	65	68	69	73	66	67	71	70	63
	72	77	81	82	76	80	79	**74**	**75**	87
	60	**89**	**88**	64	**86**	**90**	**78**	**85**	**83**	**84**
Jumlah Pesanan yang Terlambat	16									
Total Biaya Keterlambatan (menit)	28579									
Total Keterlambatan (menit)	14775									
Total Waktu Penyelesaian (menit)	20845									
Waktu Komputasi (detik)	583,189									

Setelah di-run sebanyak lima kali, terlihat bahwa hasil yang terbaik (total biaya keterlambatan terkecil) adalah run 2. Jadi, solusi terbaik dari masalah penjadwalan ini adalah dengan urutan jadwal seperti tertera pada hasil *run* 2. Nilai fungsi objektif, yaitu total biaya keterlambatan terkecil adalah sebesar 28395 menit dengan total keterlambatan sebesar 14775 menit dan makespan (total waktu penyelesaian seluruh *job*) sebesar 20845 menit. Jumlah *job* yang terlambat adalah sebanyak 16 *job*. Waktu *run* (waktu komputasi) untuk memperoleh hasil ini adalah selama 548,509 detik atau sekitar 9 menit. Untuk lebih jelasnya, hasil terbaik tersebut dapat terlihat seperti gambar 4.2.

Selain hasil pada tabel 4.2., hasil yang ditampilkan adalah total keterlambatan dari setiap *job* yang ditunjukkan pada gambar 4.3. Dari gambar tersebut, *job-job* yang menunjukkan keterlambatan terlihat dari nilai keterlambatan yang lebih dari nol pada posisi yang sama dengan indeks (urutan) job pada tabel 4.2., yaitu *job* 4, 10, 12, 8, 16, 9, 75, 76, 86, 89, 88, 87, 73, 84, 85, dan 83.

```
Urutan terbaik=
  Columns 1 through 9

    3   15    1    2    4   24    5    6   19

  Columns 10 through 18

    7   11   10   22   21   12    8   16    9

  Columns 19 through 27

   14   32   13   31   48   30   36   17   20

  Columns 28 through 36

   29   18   23   25   26   35   33   28   34

  Columns 37 through 45

   45   40   39   52   38   27   50   46   63

  Columns 46 through 54

   65   47   54   51   53   49   57   55   56

  Columns 55 through 63

   37   59   58   62   67   61   42   41   68

  Columns 64 through 72

   66   69   43   81   79   44   72   71   74

  Columns 73 through 81

   70   82   80   64   77   78   75   76   90

  Columns 82 through 90

   86   89   88   87   73   84   85   60   83

Total biaya keterlambatan terbaik =
     28395

Total keterlambatan =
     16690

Jumlah keterlambatan =
     16

Makespan =
     20845

Waktu Komputasi :548.509 detik
```

Gambar 4.2. Tampilan Hasil *Run* 2 (Terbaik)

Keterlambatan terbaik setiap *job* =

Columns 1 through 4

0 0 0 0

Columns 5 through 8

300 0 0 0

Columns 9 through 12

0 0 0 5

Columns 13 through 16

0 0 340 675

Columns 17 through 20

25 1240 0 0

Columns 21 through 24

0 0 0 0

Columns 25 through 28

0 0 0 0

Columns 29 through 32

0 0 0 0

Columns 33 through 36

0 0 0 0

Columns 37 through 40

0 0 0 0

Columns 41 through 44

0 0 0 0

Columns 45 through 48

0 0 0 0

Columns 49 through 52

0 0 0 0

Columns 53 through 56

0 0 0 0

Columns 57 through 60

0 0 0 0

Columns 61 through 64

0 0 0 0

Columns 65 through 68

0 0 0 0

Columns 69 through 72

0 0 0 0

Columns 73 through 76

0 0 0 0

Columns 77 through 80

0 0 245 545

Columns 81 through 84

0 265 565 865

Columns 85 through 88

1165 2345 1765 2065

Columns 89 through 90

0 2365

Gambar 4.3. Keterlambatan Setiap *Job* (*Run* 2)

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 α = 5% (0,05). Jika nilai *p-value* pada suatu faktor ≤ 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 4.43., akan dijelaskan pengaruh setiap faktor dan interaksi antarfaktor terhadap output, yaitu perubahan total biaya keterlambatan.

Tabel 4.43. ANOVA untuk Total Biaya Keterlambatan

Analysis of Variance for Total Biaya Keterlambatan, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
NP	**2**	**1.55560E+11**	**1.55560E+11**	**77780169853**	**257.73**	**0.000**
JI	**2**	**2.54009E+11**	**2.54009E+11**	**1.27004E+11**	**420.84**	**0.000**
F	**2**	**1.45972E+11**	**1.45972E+11**	**72986202344**	**241.85**	**0.000**
CR	**2**	**2.37513E+11**	**2.37513E+11**	**1.18756E+11**	**393.51**	**0.000**
NP*JI	4	1552558347	1552558347	388139587	1.29	0.275
NP*F	**4**	**6004494189**	**6004494189**	**1501123547**	**4.97**	**0.001**
NP*CR	4	954942340	954942340	238735585	0.79	0.532
JI*F	**4**	**26114712067**	**26114712067**	**6528678017**	**21.63**	**0.000**
JI*CR	**4**	**15526775303**	**15526775303**	**3881693826**	**12.86**	**0.000**
F*CR	**4**	**94846159667**	**94846159667**	**23711539917**	**78.57**	**0.000**
NP*JI*F	8	955876662	955876662	119484583	0.40	0.922
NP*JI*CR	8	2819963042	2819963042	352495380	1.17	0.318
NP*F*CR	**8**	**11378375591**	**11378375591**	**1422296949**	**4.71**	**0.000**
JI*F*CR	**8**	**14701315095**	**14701315095**	**1837664387**	**6.09**	**0.000**
NP*JI*F*CR	16	4993290414	4993290414	312080651	1.03	0.420
Error	324	97778933496	97778933496	301786832		
Total	404	1.07068E+12				

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* ≤ 0,05. Artinya keempat faktor

ini memiliki pengaruh yang signifikan terhadap perubahan output (total biaya keterlambatan). 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 jumlah iterasi, kemudian secara berurutan diikuti oleh operator pindah silang, ukuran populasi, dan operator mutasi.

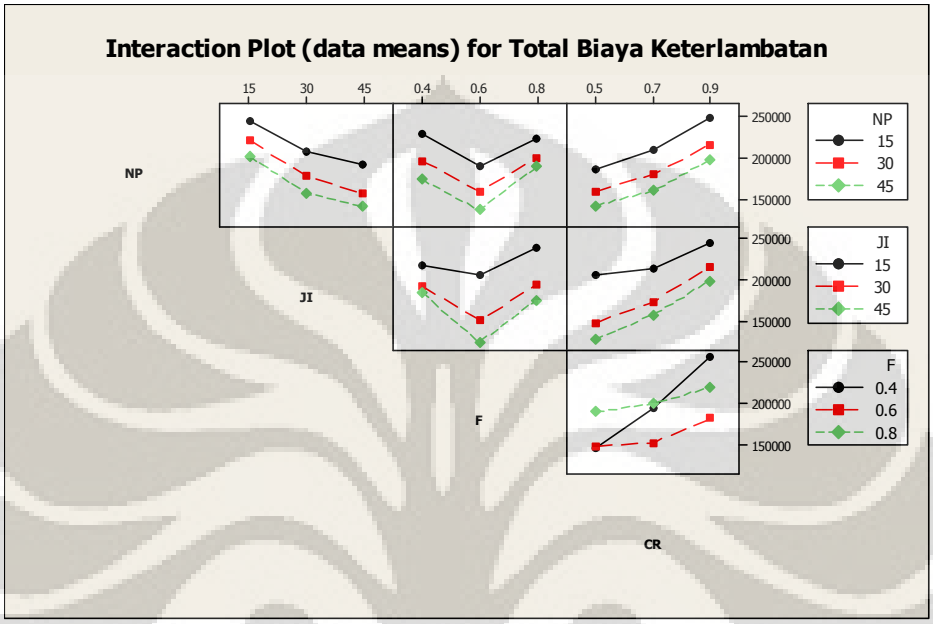
- Interaksi antara NP – JI dan NP – CR tidak berpengaruh terhadap output.
- Interaksi antara NP – F, JI – F, JI – CR, dan F – CR berpengaruh signifikan terhadap output
- Interaksi antara NP – JI – F dan NP – JI – CR tidak berpengaruh terhadap output.
- Interaksi antara NP – F - CR dan JI – F – CR berpengaruh signifikan terhadap output.
- Interaksi antara NP – JI – F – CR tidak berpengaruh 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 4.4. adalah sebagai berikut:

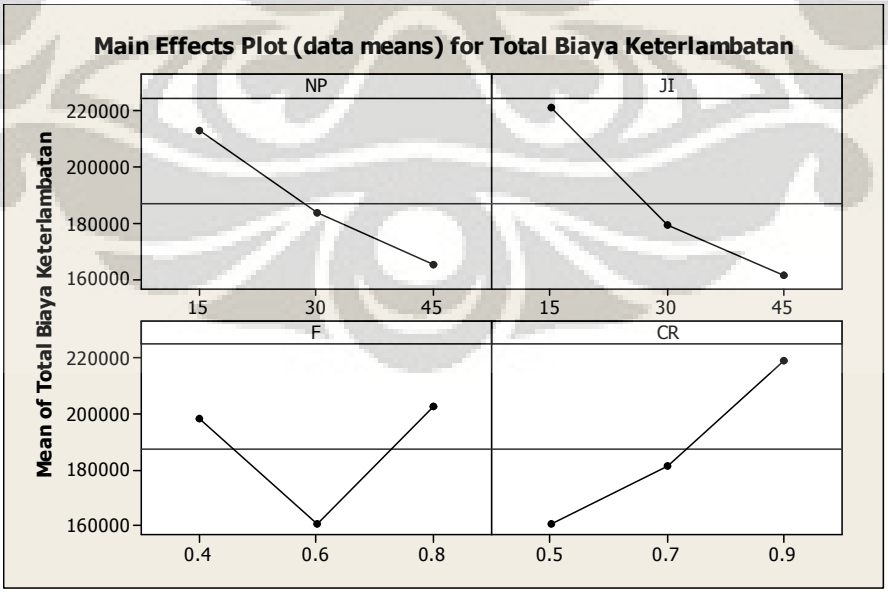
- Interaksi antara NP – JI dan NP – CR tidak berpengaruh terhadap output. Artinya jika salah satu paremeter diubah, sedangkan parameter yang lainnya tidak berubah, hasilnya tidak akan berubah secara signifikan. Sebagai contoh, untuk nilai NP paling rendah, yaitu 15 (ditunjukkan pada garis paling atas) dipadukan dengan nilai JI sebesar 15, nilai output tidak akan jauh berbeda jika nilai NP tersebut dipadukan dengan nilai JI sebesar 30 atau 45. Hal ini terlihat pada gambar, yaitu kemiringan garis yang bersangkutan hampir mendatar.

- Interaksi antara NP – F, 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 4.4. Interaksi antara Dua Faktor

Gambar 4.5. adalah *main effect plot*, yang menunjukkan efek utama dari setiap level pada setiap faktor terhadap output (total biaya keterlambatan).



Gambar 4.5. Grafik Faktor Utama untuk Total Biaya keterlambatan

Berikut adalah penjelasan gambar 4.5 :

- Ukuran populasi (NP) dan Jumlah Iterasi (JI)

Kedua faktor ini memiliki gradien (kemiringan) garis yang negatif, artinya semakin besar jumlah iterasi atau ukuran populasi, maka total biaya keterlambatan yang dihasilkan akan semakin kecil, dan sebaliknya. Maka, untuk memperkecil total biaya keterlambatan, ukuran populasi dan jumlah iterasi yang digunakan adalah yang bernilai besar.

- Operator Mutasi (F)

Dari gambar terlihat nilai output (total biaya keterlambatan) terkecil didapatkan ketika menggunakan parameter F berlevel sedang, yaitu 0,6. Oleh karena itu, untuk memperkecil total biaya keterlambatan, 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 total biaya keterlambatan yang dihasilkan juga akan semakin kecil. Maka, untuk memperkecil total biaya keterlambatan, parameter CR yang digunakan adalah yang bernilai kecil.

Untuk ukuran populasi dan jumlah iterasi, keduanya memiliki pengaruh signifikan terhadap output (total biaya keterlambatan). Jika ukuran populasi atau 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. Oleh karena itu, interaksi antara keduanya juga dilihat. Dari hasil ANOVA (tabel 4.43) terlihat bahwa interaksi antara ukuran populasi dan jumlah iterasi tidak berpengaruh signifikan terhadap output sehingga jika salah satunya sudah bernilai besar, maka sudah cukup untuk mendapatkan output yang diharapkan. Dari hasil ANOVA pula, terlihat bahwa jumlah iterasi lebih berpengaruh terhadap output dibandingkan dengan ukuran populasi sehingga nilai parameter yang akan diperbesar secara signifikan adalah jumlah iterasi.

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

1. Ukuran populasi yang relatif kecil

   Meskipun antara ukuran populasi dan jumlah iterasi, parameter yang akan lebih diperbesar adalah jumlah iterasi, 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 NP – F – CR berpengaruh signifikan terhadap output. Artinya, penentuan ukuran populasi ditentukan pula oleh dua parameter lainnya (operator mutasi dan operator pindah silang). Ukuran populasi yang terlalu kecil akan menyebabkan rendahnya variasi individu pada populasi pada proses mutasi dan pindah silang sehingga dapat terjadi konvergensi. Sedangkan ukuran populasi yang terlalu besar akan menyebabkan waktu komputasi terlalu lama. Oleh karena itu, dipilih ukuran populasi yang relatif kecil (tidak besar dan tidak terlalu kecil) yaitu 100.

2. Jumlah iterasi 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 jumlah iterasi, populasi akan semakin banyak mengalami perbaikan, artinya kemungkinan urutan yang dihasilkan dan diuji akan semakin besar sehingga solusi akan lebih mendekati optimal.

   Karena jumlah iterasi memiliki pengaruh yang sangat signifikan, maka jumlah iterasi ditetapkan dengan nilai yang sangat besar dengan mempertimbangkan output (total biaya keterlambatan) dan waktu komputasi. Untuk menentukan jumlah iterasi, dilakukan lima kali percobaan, yaitu untuk jumlah iterasi sebesar 1000, 2000, 3000, 4000, dan 5000. Dari kelima percobaan tersebut, terlihat bahwa jumlah iterasi di atas 2000 tidak memberikan peningkatan solusi yang lebih baik secara signifikan dan waktu komputasi yang dibutuhkan juga lebih lama. Dengan jumlah iterasi itu pula, waktu komputasi yang dibutuhkan tidak terlalu lama, yaitu kurang dari 10 menit. Oleh karena itu, jumlah iterasi yang digunakan adalah 2000 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-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 90, maka jumlah kemungkinan urutan pengerjaan *job* adalah 90! (90 x 89 x 88 x ... x 2 x 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

Tabel 4.44. Perbandingan Hasil Jadwal PT X dengan Usulan Jadwal

Kriteria	Jadwal PT X										Usulan Jadwal									
Urutan Pengerjaan	1	2	3	**4**	5	6	7	8	**9**	**10**	3	15	1	2	**4**	24	5	6	19	7
	11	**12**	13	14	15	**16**	17	18	19	20	11	**10**	22	21	**12**	**8**	**16**	**9**	14	32
	21	22	23	24	25	**26**	27	28	29	30	13	31	48	30	36	17	20	29	18	23
	31	32	33	34	35	36	37	38	39	40	25	26	35	33	28	34	45	40	39	52
	41	42	43	44	45	46	47	48	49	50	38	27	50	46	63	65	47	54	51	53
	51	52	53	54	55	56	57	58	59	60	49	57	55	56	37	59	58	62	67	61
	61	62	63	64	65	66	67	68	69	70	42	41	68	66	69	43	81	79	44	72
	71	72	73	74	75	76	77	78	79	**80**	71	74	70	82	80	64	77	78	**75**	**76**
	81	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	90	**86**	**89**	**88**	**87**	**73**	**84**	**85**	60	**83**
Job-job yang Terlambat	4	9	10	11	12	16	26	80	81	82	4	10	12	8	16	9	75	76	86	89
	83	84	85	86	87	88	89	90			88	87	73	84	85	63				
Jumlah *Job* yang Terlambat	18										16									
Total Biaya Keterlambatan (menit)	33190										28395									
Total Keterlambatan (menit)	16690										14775									
Total Waktu Penyelesaian (menit)	20865										20845									

Fungsi tujuan dari penyelesaian masalah pada penelitian ini adalah meminimumkan total biaya keterlambatan. Total biaya keterlambatan pada jadwal PT X adalah 33190 menit, sedangkan usulan jadwal yang dihasilkan menghasilkan total biaya keterlambatan sebesar 28385 menit. Jika dibandingkan,

total biaya keterlambatan pada usulan jadwal mengalami penurunan sebesar 14,45% dari jadwal PT X. Nilai penurunan ini tidak terlalu signifikan karena PT X telah menggunakan metode *earliest due date* untuk mengatur penjadwalan yang pada dasarnya memang ditujukan untuk meminimumkan keterlambatan *job-job*. Tetapi, metode *earliest due date* hanya memperhatikan *due date* dari setiap *job*, tidak mempertimbangkan tingkat prioritas setiap *job* (penalti). Pada perhitungan usulan jadwal, selain due date setiap *job*, besar penalti (tingkat prioritas) setiap *job* juga dipertimbangkan sehingga total biaya keterlambatan yang diperoleh akan lebih baik. Jadi, metode *earliest due date* lebih cocok digunakan untuk *job-job* yang memiliki prioritas pengerjaan yang sama.

Meskipun penyelesaian masalah memiliki fungsi tujuan hanya satu, yaitu meminimumkan total biaya keterlambatan, akan tetapi terlihat bahwa fungsi yang lain juga mengalami penurunan, yaitu :

- Jumlah *job* mengalami penurunan sebesar 11,11% (dari 18 *job* menjadi 16 *job*).
- Total keterlambatan mengalami penurunan sebesar 11,47% (dari 16690 menit menjadi 14775 menit)
- Total waktu penyelesaian (*makespan*) mengalami penurunan sebesar 0,1% (dari 20865 menit menjadi 20845 menit).

Dari hasil di atas, terlihat bahwa selain total biaya keterlambatan, total keterlambatan juga mengalami penurunan. Hal ini menunjukkan bahwa walaupun fungsi tujuan berubah, seperti meminimumkan total keterlambatan, hasil yang diperoleh dengan metode algoritma DE memiliki kemungkinan lebih baik dibandingkan metode heuristik klasik (contohnya metode *earliest due date*).

5. KESIMPULAN

Berdasarkan hasil analisis dari permasalahan penjadwalan *job shop* pada PT X menggunakan algoritma *differential evolution* dengan bantuan bahasa pemrograman MATLAB 7.0, diperoleh kesimpulan sebagai berikut:

1. Jadwal PT X memiliki total biaya keterlambatan seluruh *job* sebesar 33190 menit.
2. Usulan jadwal hasil program dengan metode algoritma *differential evolution* memiliki total biaya keterlambatan seluruh *job* sebesar 28395 menit.
3. Total biaya keterlambatan untuk usulan jadwal memiliki penurunan dibanding jadwal PT X yaitu sebesar 14,45%.
4. Selain fungsi tujuan (total biaya keterlambatan), fungsi yang lain juga mengalami penurunan dibanding jadwal PT X, yaitu:
   - Jumlah *job* yang terlambat mengalami penurunan sebesar 11,11%.
   - Total keterlambatan seluruh *job* mengalami penurunan sebesar 11,47%.
   - Makespan (total waktu penyelesaian seluruh *job*) mengalami penurunan sebesar 0,1%.
5. Waktu *run* program untuk memperoleh usulan jadwal yaitu adalah 548,509 detik.

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 thr 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 X

```
clc;
clear;
tic;

jumlah_pesanan = 90;
jumlah_rute = 7;

%---Data waktu proses, due date, dan penalti----------------
data2 = [120    140    15    0    210    0    300    160    180    1760    1.87
120    140    15    0    210    0    300    160    180    1760    1.87
120    140    15    0    210    0    300    160    180    1760    1.87
120    140    15    0    210    0    300    160    180    2640    1.87
120    140    15    0    210    0    300    160    180    2640    1.87
120    140    15    0    210    0    300    160    180    3520    1.87
120    140    15    0    210    0    300    160    180    3520    1.87
120    140    15    0    210    0    300    160    180    3520    1.87
120    140    15    0    210    0    300    160    180    3520    1.87
120    140    15    0    210    0    300    160    180    3520    1.95
120    140    15    0    210    0    300    160    180    3520    1.95
60    140    15    210    0    240    0    150    155    4400    1.14
60    140    15    210    0    240    0    150    155    4400    1.14
60    140    15    210    0    240    0    150    155    4400    1.14
120    140    15    0    210    0    300    160    180    4400    1.95
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    6160    1.14
60    140    15    210    0    240    0    150    155    6160    1.14
60    140    15    210    0    240    0    150    155    6160    1.14
60    140    15    210    0    240    0    150    155    6160    1.14
60    140    15    210    0    240    0    150    155    7040    1.14
60    140    15    210    0    240    0    150    155    7040    1.14
120    140    15    0    210    0    300    160    180    7920    1.95
120    140    15    0    210    0    300    160    180    7920    1.95
120    140    15    0    210    0    300    160    180    7920    2.04
120    150    15    0    240    0    340    150    200    7920    2.39
120    140    15    0    210    0    300    160    180    8800    2.04
120    140    15    0    210    0    300    160    180    8800    2.04
120    140    15    0    210    0    300    160    180    8800    2.04
120    150    15    0    240    0    340    150    200    8800    2.39
```

50	140	15	150	0	120	0	90	155	9680	1.00
120	140	15	0	210	0	300	160	180	9680	2.04
120	140	15	0	210	0	300	160	180	9680	2.04
120	140	15	0	210	0	300	160	180	9680	2.04
50	140	15	150	0	120	0	90	155	10560	1.00
50	140	15	150	0	120	0	90	155	10560	1.00
50	140	15	150	0	120	0	90	155	10560	1.00
50	140	15	150	0	120	0	90	155	11440	1.00
120	150	15	0	240	0	340	150	200	11440	2.39
120	150	15	0	240	0	340	150	200	11440	2.39
120	150	15	0	240	0	340	150	200	11440	2.39
120	150	15	0	240	0	340	150	200	11440	2.39
120	140	15	0	210	0	300	160	180	12320	1.87
120	140	15	0	210	0	300	160	180	12320	1.87
120	150	15	0	240	0	340	150	200	12320	2.39
120	150	15	0	240	0	340	150	200	12320	2.39
120	150	15	0	240	0	340	150	200	12320	2.39
120	150	15	0	240	0	340	150	200	12320	2.39
120	140	15	0	210	0	300	160	180	13200	1.87
120	140	15	0	210	0	300	160	180	13200	1.87
120	140	15	0	210	0	300	160	180	13200	1.87
120	140	15	0	210	0	300	160	180	13200	1.87
120	140	15	0	210	0	300	160	180	13200	1.87
50	140	15	150	0	120	0	90	155	14080	1.00
120	140	15	0	210	0	300	160	180	14080	1.87
120	140	15	0	210	0	300	160	180	14080	1.87
50	140	15	150	0	120	0	90	155	14960	1.00
50	140	15	150	0	120	0	90	155	14960	1.00
50	140	15	150	0	120	0	90	155	15840	1.00
50	140	15	150	0	120	0	90	155	15840	1.00
120	140	15	0	210	0	300	160	180	15840	2.04
120	150	15	0	240	0	340	150	200	15840	2.39
120	150	15	0	240	0	340	150	200	15840	2.39
120	140	15	0	210	0	300	160	180	16720	2.04
120	140	15	0	210	0	300	160	180	16720	2.04
120	150	15	0	240	0	340	150	200	16720	2.39
120	140	15	0	210	0	300	160	180	17600	1.95
120	140	15	0	210	0	300	160	180	17600	1.95
120	140	15	0	210	0	300	160	180	17600	1.95
120	140	15	0	210	0	300	160	180	17600	1.95
120	140	15	0	210	0	300	160	180	17600	2.04
120	140	15	0	210	0	300	160	180	17600	2.04
120	140	15	0	210	0	300	160	180	17600	2.04
120	140	15	0	210	0	300	160	180	17600	2.04
120	140	15	0	210	0	300	160	180	17600	2.04
120	140	15	0	210	0	300	160	180	17600	2.04
120	140	15	0	210	0	300	160	180	18480	1.87
120	140	15	0	210	0	300	160	180	18480	1.87

```
120   140   15   0   210   0   300   160   180   18480  1.87
120   140   15   0   210   0   300   160   180   18480  2.04
120   140   15   0   210   0   300   160   180   18480  2.04
120   140   15   0   210   0   300   160   180   18480  2.04
120   140   15   0   210   0   300   160   180   18480  2.04
120   140   15   0   210   0   300   160   180   18480  2.04];

N=size(data2);
wp=data2(1:N,1:9);
due_date=data2(1:N,10);
penalti=data2(1:N,11);

w = zeros(jumlah_pesanan, 9);
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 & b<=4
        w(a,b)=w(a,b-1)+ wp(a,b);
      end
      if a == 1 & b>=5 & b<=7
        w(a,b)=w(a,b-2)+wp(a,b);
      end
      if a > 1 & b == 1
        w(a,b)=w(a-1,b)+wp(a,b);
      end
      if a > 1 & b>=2 & b<=4
        w(a,b)=max(w(a,b-1),w(a-1,b))+ wp(a,b);
      end
      if a > 1 & b>=5 & b<=7
        w(a,b)=max(w(a,b-2),w(a-1,b))+wp(a,b);
      end
   end
end
w1=zeros(jumlah_pesanan,1);
for a = 1 : jumlah_pesanan
   if wp(a,6)== 0
      w1(a)=w(a,7);
   else
      w1(a)=w(a,6);
   end
end
index_rute8=zeros(jumlah_pesanan,1);
wp1=zeros(jumlah_pesanan,1);
for a = 1 : jumlah_pesanan
    if a == 1
      [wp1(a),index_rute8(a)]=min(w1);
```

```
        w(index_rute8(a),8)=wp1(a)+wp(index_rute8(a),8);
        w(index_rute8(a),9)=w(index_rute8(a),8)+wp(index_rute8(a),9);
    end
    if a>1
      [wp1(a),index_rute8(a)]=min(w1);
       w(index_rute8(a),8)=max(wp1(a),w(index_rute8(a-
1),8))+wp(index_rute8(a),8);
       w(index_rute8(a),9)=max(w(index_rute8(a),8),w(index_rute8(a-
1),9))+wp(index_rute8(a),9);
    end
    w1(index_rute8(a))=9999999999;
end
waktu_total=max(w(:, 9));
keterlambatan=zeros(jumlah_pesanan,1);
biaya_keterlambatan=zeros(jumlah_pesanan,1);
for a = 1:jumlah_pesanan
   keterlambatan(a)=w(a,9)-due_date(a);
   if keterlambatan(a)<0
      keterlambatan(a)=0;
   end
   biaya_keterlambatan(a)=keterlambatan(a)*penalti(a);
end
total_biaya_keterlambatan=sum(biaya_keterlambatan);
total_keterlambatan=sum(keterlambatan);
jumlah_keterlambatan=numel(find(keterlambatan));

%---Solusi Akhir-----------------------------------------------------
disp('Total Keterlambatan Setiap Job =');
disp(keterlambatan');
disp('Total Biaya Keterlambatan =')
disp(total_biaya_keterlambatan);
disp('Total Keterlambatan =')
disp(total_keterlambatan);
disp('Jumlah Keterlambatan =');
disp(jumlah_keterlambatan);
disp('Makespan =');
disp(waktu_total);
```

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

```
clc;
clear;
tic;

jumlah_pesanan = 90;
jumlah_rute = 7;
jumlah_dimensi = jumlah_pesanan;
ukuran_populasi = 100;
jumlah_iterasi = 2000;
F = 0.6; %operator mutasi
CR = 0.5; %operator pindah silang
batas_bawah = -1;
batas_atas = 1;

%---Data waktu proses, due date, dan penalti----------
%Kolom 1-9 adalah waktu proses tiap rute
%kolom 10 adalah due date tiap pesanan (job)
%kolom 11 adalah penalti keterlambatan tiap job
data2 = [120    140    15    0    210    0    300    160    180    1760
        1.87
120    140    15    0    210    0    300    160    180    1760    1.87
120    140    15    0    210    0    300    160    180    1760    1.87
120    140    15    0    210    0    300    160    180    1760    1.87
120    140    15    0    210    0    300    160    180    2640    1.87
120    140    15    0    210    0    300    160    180    2640    1.87
120    140    15    0    210    0    300    160    180    3520    1.87
120    140    15    0    210    0    300    160    180    3520    1.87
120    140    15    0    210    0    300    160    180    3520    1.87
120    140    15    0    210    0    300    160    180    3520    1.87
120    140    15    0    210    0    300    160    180    3520    1.95
120    140    15    0    210    0    300    160    180    3520    1.95
60    140    15    210    0    240    0    150    155    4400    1.14
60    140    15    210    0    240    0    150    155    4400    1.14
60    140    15    210    0    240    0    150    155    4400    1.14
120    140    15    0    210    0    300    160    180    4400    1.95
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    5280    1.14
60    140    15    210    0    240    0    150    155    6160    1.14
60    140    15    210    0    240    0    150    155    6160    1.14
60    140    15    210    0    240    0    150    155    6160    1.14
60    140    15    210    0    240    0    150    155    6160    1.14
```

60	140	15	210	0	240	0	150	155	7040	1.14
60	140	15	210	0	240	0	150	155	7040	1.14
120	140	15	0	210	0	300	160	180	7920	1.95
120	140	15	0	210	0	300	160	180	7920	1.95
120	140	15	0	210	0	300	160	180	7920	2.04
120	150	15	0	240	0	340	150	200	7920	2.39
120	140	15	0	210	0	300	160	180	8800	2.04
120	140	15	0	210	0	300	160	180	8800	2.04
120	140	15	0	210	0	300	160	180	8800	2.04
120	150	15	0	240	0	340	150	200	8800	2.39
50	140	15	150	0	120	0	90	155	9680	1.00
120	140	15	0	210	0	300	160	180	9680	2.04
120	140	15	0	210	0	300	160	180	9680	2.04
120	140	15	0	210	0	300	160	180	9680	2.04
50	140	15	150	0	120	0	90	155	10560	1.00
50	140	15	150	0	120	0	90	155	10560	1.00
50	140	15	150	0	120	0	90	155	10560	1.00
50	140	15	150	0	120	0	90	155	11440	1.00
120	150	15	0	240	0	340	150	200	11440	2.39
120	150	15	0	240	0	340	150	200	11440	2.39
120	150	15	0	240	0	340	150	200	11440	2.39
120	150	15	0	240	0	340	150	200	11440	2.39
120	140	15	0	210	0	300	160	180	12320	1.87
120	140	15	0	210	0	300	160	180	12320	1.87
120	150	15	0	240	0	340	150	200	12320	2.39
120	150	15	0	240	0	340	150	200	12320	2.39
120	150	15	0	240	0	340	150	200	12320	2.39
120	150	15	0	240	0	340	150	200	12320	2.39
120	140	15	0	210	0	300	160	180	13200	1.87
120	140	15	0	210	0	300	160	180	13200	1.87
120	140	15	0	210	0	300	160	180	13200	1.87
120	140	15	0	210	0	300	160	180	13200	1.87
120	140	15	0	210	0	300	160	180	13200	1.87
50	140	15	150	0	120	0	90	155	14080	1.00
120	140	15	0	210	0	300	160	180	14080	1.87
120	140	15	0	210	0	300	160	180	14080	1.87
50	140	15	150	0	120	0	90	155	14960	1.00
50	140	15	150	0	120	0	90	155	14960	1.00
50	140	15	150	0	120	0	90	155	15840	1.00
50	140	15	150	0	120	0	90	155	15840	1.00
120	140	15	0	210	0	300	160	180	15840	2.04
120	150	15	0	240	0	340	150	200	15840	2.39
120	150	15	0	240	0	340	150	200	15840	2.39
120	140	15	0	210	0	300	160	180	16720	2.04
120	140	15	0	210	0	300	160	180	16720	2.04
120	150	15	0	240	0	340	150	200	16720	2.39
120	140	15	0	210	0	300	160	180	17600	1.95
120	140	15	0	210	0	300	160	180	17600	1.95

```
120  140  15  0  210  0  300  160  180  17600  1.95
120  140  15  0  210  0  300  160  180  17600  1.95
120  140  15  0  210  0  300  160  180  17600  2.04
120  140  15  0  210  0  300  160  180  17600  2.04
120  140  15  0  210  0  300  160  180  17600  2.04
120  140  15  0  210  0  300  160  180  17600  2.04
120  140  15  0  210  0  300  160  180  17600  2.04
120  140  15  0  210  0  300  160  180  17600  2.04
120  140  15  0  210  0  300  160  180  18480  1.87
120  140  15  0  210  0  300  160  180  18480  1.87
120  140  15  0  210  0  300  160  180  18480  1.87
120  140  15  0  210  0  300  160  180  18480  2.04
120  140  15  0  210  0  300  160  180  18480  2.04
120  140  15  0  210  0  300  160  180  18480  2.04
120  140  15  0  210  0  300  160  180  18480  2.04
120  140  15  0  210  0  300  160  180  18480  2.04];

N=size(data2);
wp=data2(1:N,1:9);
due_date=data2(1:N,10);
penalti=data2(1:N,11);

%---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 biaya keterlambatan populasi target-------
for a = 1 : ukuran_populasi
  w = zeros(jumlah_pesanan, 9);
  keterlambatan=zeros(jumlah_pesanan,1);
  biaya_keterlambatan=zeros(jumlah_pesanan,1);
  due_date2=zeros(jumlah_pesanan,1);
  penalti2=zeros(jumlah_pesanan,1);
  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 & c<=4
        w(b,c)=w(b,c-1)+ wp(urutan_awal (b, a),c);
      end
      if b == 1 & c>=5 & c<=7
        w(b,c)=w(b,c-2)+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 & c<=4
        w(b,c)=max(w(b,c-1),w(b-1,c))+ wp(urutan_awal (b, a),c);
      end
      if b > 1 & c>=5 & c<=7
        w(b,c)=max(w(b,c-2),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),6)== 0
       w1(b)=w(b,7);
     else
       w1(b)=w(b,6);
     end
  end
  index_rute8=zeros(jumlah_pesanan,1);
  wp1=zeros(jumlah_pesanan,1);
  for b = 1 : jumlah_pesanan
    if b == 1
      [wp1(b),index_rute8(b)]=min(w1);
       w(index_rute8(b),8)=wp1(b)+wp(urutan_awal(index_rute8(b),a),8);
 w(index_rute8(b),9)=w(index_rute8(b),8)+wp(urutan_awal(index_rute8(b),a),9);
       due_date2(b)=due_date(urutan_awal(b,a));
       penalti2(b)=penalti(urutan_awal(b,a));
    end
    if b>1
      [wp1(b),index_rute8(b)]=min(w1);
       w(index_rute8(b),8)=max(wp1(b),w(index_rute8(b-
1),8))+wp(urutan_awal(index_rute8(b),a),8);
       w(index_rute8(b),9)=max(w(index_rute8(b),8),w(index_rute8(b-
1),9))+wp(urutan_awal(index_rute8(b),a),9);
       due_date2(b)=due_date(urutan_awal (b, a));
       penalti2(b)=penalti(urutan_awal (b, a));
    end
    w1(index_rute8(b))=9999999999;
  end
  for b = 1 : jumlah_pesanan
     keterlambatan(b)=w(b,9)-due_date2(b);
     if keterlambatan(b)<0
       keterlambatan(b)=0;
     end
     biaya_keterlambatan(b)=keterlambatan(b)*penalti2(b);
  end
```

```
  keterlambatan_awal(:,a)=keterlambatan;
  total_keterlambatan_awal(a)=sum(keterlambatan);
  total_biaya_keterlambatan_awal(a)=sum(biaya_keterlambatan);
  nilai_makespan_awal(a)=max(w(:, 9));
end
[total_biaya_keterlambatan_min_awal, index_vektor_target] =
min(total_biaya_keterlambatan_awal);
total_keterlambatan=total_keterlambatan_awal(:,index_vektor_target);
makespan_awal = nilai_makespan_awal(index_vektor_target);
urutan=urutan_awal(:,index_vektor_target);
keterlambatan_sekarang=keterlambatan_awal(:,index_vektor_target);
jumlah_keterlambatan_awal=numel(find(keterlambatan_sekarang));

%---Hasil Awal Sebelum Iterasi--------------------------------
disp('Keterlambatan setiap job =');
disp(keterlambatan_sekarang');
disp('Urutan =');
disp(urutan');
disp('Nilai total biaya keterlambatan Awal Minimum =');
disp(total_biaya_keterlambatan_min_awal);
disp('Nilai total keterlambatan Awal =');
disp(total_keterlambatan);
disp('Jumlah keterlambatan awal =');
disp(jumlah_keterlambatan_awal);
disp('Nilai makespan Awal =');
disp(makespan_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-------------------------------------
  [total_biaya_keterlambatan_min_awal, index_vektor_target] =
min(total_biaya_keterlambatan_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
              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 biaya keterlambatan populasi trial-------------------
keterlambatan_anak=[];
total_keterlambatan_anak = [];
total_biaya_keterlambatan_anak = [];
nilai_makespan_anak = [];
for a = 1 : ukuran_populasi
  w = zeros(jumlah_pesanan, 9);
  keterlambatan=zeros(jumlah_pesanan,1);
  biaya_keterlambatan=zeros(jumlah_pesanan,1);
  due_date2=zeros(jumlah_pesanan,1);
  penalti2=zeros(jumlah_pesanan,1);
  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 & c<=4
        w(b,c)=w(b,c-1)+ wp(urutan_trial (b, a),c);
      end
```

```
      if b == 1 & c>=5 & c<=7
        w(b,c)=w(b,c-2)+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 & c<=4
        w(b,c)=max(w(b,c-1),w(b-1,c))+ wp(urutan_trial (b, a),c);
      end
      if b > 1 & c>=5 & c<=7
        w(b,c)=max(w(b,c-2),w(b-1,c))+wp(urutan_trial (b, a),c);
      end
    end
  end
  w1=zeros(jumlah_pesanan, 1);
  for b = 1 : jumlah_pesanan
    if wp(urutan_trial (b, a),6)== 0
      w1(b)=w(b,7);
    else
      w1(b)=w(b,6);
    end
  end
  index_rute8=zeros(jumlah_pesanan,1);
  wp1=zeros(jumlah_pesanan,1);
  for b = 1 : jumlah_pesanan
   if b == 1
     [wp1(b),index_rute8(b)]=min(w1);
      w(index_rute8(b),8)=wp1(b)+wp(urutan_trial(index_rute8(b),a),8);
w(index_rute8(b),9)=w(index_rute8(b),8)+wp(urutan_trial(index_rute8(b),a),9);
      due_date2(b)=due_date(urutan_trial(b,a));
      penalti2(b)=penalti(urutan_trial(b,a));
   end
   if b>1
     [wp1(b),index_rute8(b)]=min(w1);
      w(index_rute8(b),8)=max(wp1(b),w(index_rute8(b-
1),8))+wp(urutan_trial(index_rute8(b),a),8);
      w(index_rute8(b),9)=max(w(index_rute8(b),8),w(index_rute8(b-
1),9))+wp(urutan_trial(index_rute8(b),a),9);
      due_date2(b)=due_date(urutan_trial (b, a));
      penalti2(b)=penalti(urutan_trial (b, a));
   end
   w1(index_rute8(b))=9999999999;
  end
  for b = 1 : jumlah_pesanan
    keterlambatan(b)=w(b,9)-due_date2(b);
    if keterlambatan(b)<0
      keterlambatan(b)=0;
    end
```

```
    biaya_keterlambatan(b)=keterlambatan(b)*penalti2(b);
  end
  keterlambatan_anak(:,a)=keterlambatan;
  total_keterlambatan_anak(a)=sum(keterlambatan);
  total_biaya_keterlambatan_anak(a)=sum(biaya_keterlambatan);
  nilai_makespan_anak(a)=max(w(:, 9));
end

%---Memilih antara populasi target awal atau trial--------------------
          for a = 1 : ukuran_populasi
    if total_biaya_keterlambatan_anak(a) < total_biaya_keterlambatan_awal(a)
      populasi_target(:, a) = populasi_trial(:,a);
      urutan_awal(:,a) = urutan_trial(:,a);
      keterlambatan_awal(:,a)=keterlambatan_anak(:,a);
      total_keterlambatan_awal(a) = total_keterlambatan_anak(a);
      total_biaya_keterlambatan_awal(a) = total_biaya_keterlambatan_anak(a);
      nilai_makespan_awal(a)= nilai_makespan_anak(a);
    end
  end
end
[total_biaya_keterlambatan_min,                index_vektor_target_akhir]                =
min(total_biaya_keterlambatan_awal);
total_keterlambatan=total_keterlambatan_awal(:,index_vektor_target_akhir);
makespan_akhir = nilai_makespan_awal(index_vektor_target_akhir);
urutan_akhir=urutan_awal(:,index_vektor_target_akhir);
keterlambatan_akhir=keterlambatan_awal(:,index_vektor_target_akhir);
jumlah_keterlambatan=numel(find(keterlambatan_akhir));

%---Solusi Akhir-----------------------------------------------------
disp('Keterlambatan terbaik setiap job =');
disp(keterlambatan_akhir');
disp('Urutan terbaik=');
disp(urutan_akhir');
disp('Total biaya keterlambatan terbaik =')
disp(total_biaya_keterlambatan_min);
disp('Total keterlambatan =')
disp(total_keterlambatan);
disp('Jumlah keterlambatan =');
disp(jumlah_keterlambatan);
disp('Makespan =');
disp(makespan_akhir);
disp(strcat('Waktu Komputasi :',' ', num2str(toc),' detik'));
```

Lampiran 3: Hasil *Design of Experiments*

No	Parameter				Hasil (Total Biaya Keterlambatan)				
	NP	JI	F	CR	*Run* 1	*Run* 2	*Run* 3	*Run* 4	*Run* 5
1	15	15	0.4	0.5	211240	214550	226650	234150	212360
2	15	15	0.4	0.7	211970	264870	213890	248150	205460
3	15	15	0.4	0.9	292790	314360	298040	272470	266400
4	15	15	0.6	0.5	201620	212280	234390	219040	209620
5	15	15	0.6	0.7	217400	254540	204680	227340	203230
6	15	15	0.6	0.9	256190	252590	230740	265170	237720
7	15	15	0.8	0.5	225910	252260	256410	247960	240000
8	15	15	0.8	0.7	267810	228330	268330	249170	246640
9	15	15	0.8	0.9	277714	264780	252150	275150	292140
10	15	30	0.4	0.5	208270	165360	156320	142820	161170
11	15	30	0.4	0.7	205300	212830	246010	205700	246910
12	15	30	0.4	0.9	280300	242170	300760	278570	270340
13	15	30	0.6	0.5	170920	148820	181530	148430	151410
14	15	30	0.6	0.7	173100	159580	133650	202370	187450
15	15	30	0.6	0.9	194290	224150	192550	215800	213490
16	15	30	0.8	0.5	198090	187070	194560	201870	178940
17	15	30	0.8	0.7	209580	183470	187850	218860	211340
18	15	30	0.8	0.9	246250	249340	240040	244430	251210
19	15	45	0.4	0.5	132620	168590	129460	153330	168290
20	15	45	0.4	0.7	204490	245106	203730	201360	241350
21	15	45	0.4	0.9	272890	274670	298730	262230	271280
22	15	45	0.6	0.5	144280	112280	112980	114170	125820
23	15	45	0.6	0.7	137100	155230	131940	149930	146090
24	15	45	0.6	0.9	209590	196880	212440	213530	196370
25	15	45	0.8	0.5	196270	167650	190150	182340	173770
26	15	45	0.8	0.7	209320	191160	241010	201210	181750
27	15	45	0.8	0.9	219960	205970	217330	163670	212790
28	30	15	0.4	0.5	186240	204720	176310	181160	187460
29	30	15	0.4	0.7	229570	224180	242450	174740	217700
30	30	15	0.4	0.9	258180	202730	252210	253780	208780
31	30	15	0.6	0.5	196500	209850	204180	206560	204560
32	30	15	0.6	0.7	214870	184060	189000	206530	202140
33	30	15	0.6	0.9	209750	227930	250680	205510	199710
34	30	15	0.8	0.5	197020	243540	217210	225980	224210
35	30	15	0.8	0.7	224640	206740	244350	237950	247950
36	30	15	0.8	0.9	233190	266320	250590	284310	251810
37	30	30	0.4	0.5	125580	109880	122810	130970	143250
38	30	30	0.4	0.7	152150	210900	179880	215180	189940
39	30	30	0.4	0.9	253110	269460	267160	278280	248310
40	30	30	0.6	0.5	111390	129700	133170	127710	137570
41	30	30	0.6	0.7	137570	161490	125300	131820	154390
42	30	30	0.6	0.9	176210	179850	145810	182440	180190
43	30	30	0.8	0.5	137260	184900	192400	184110	161340
44	30	30	0.8	0.7	179970	183240	186080	180020	185190
45	30	30	0.8	0.9	238230	226200	192840	218600	184600

No	Parameter				Hasil (Total Biaya Keterlambatan)				
	NP	JI	F	CR	*Run* 1	*Run* 2	*Run* 3	*Run* 4	*Run* 5
46	30	45	0.4	0.5	102320	118740	115830	99168	134380
47	30	45	0.4	0.7	150160	208320	164100	154650	206240
48	30	45	0.4	0.9	258290	243240	218240	240460	248330
49	30	45	0.6	0.5	93470	99621	124250	82498	103610
50	30	45	0.6	0.7	112810	97281	100930	100280	106910
51	30	45	0.6	0.9	109990	164600	184720	131620	160180
52	30	45	0.8	0.5	151850	178280	180680	152580	162690
53	30	45	0.8	0.7	204790	158620	120730	178200	158290
54	30	45	0.8	0.9	180870	166300	177850	174840	191780
55	45	15	0.4	0.5	146490	182800	165650	145930	164530
56	45	15	0.4	0.7	164130	159320	174040	187110	202470
57	45	15	0.4	0.9	250870	230220	251440	240060	265560
58	45	15	0.6	0.5	197320	178020	173701	182320	191710
59	45	15	0.6	0.7	168880	201980	162380	178740	167790
60	45	15	0.6	0.9	159210	165450	213540	183900	194680
61	45	15	0.8	0.5	230440	222120	200590	202290	234730
62	45	15	0.8	0.7	204610	207760	210340	222980	215440
63	45	15	0.8	0.9	231314	229710	258550	239580	260580
64	45	30	0.4	0.5	100660	75157	107070	75724	120260
65	45	30	0.4	0.7	163460	167020	159880	123880	149740
66	45	30	0.4	0.9	242530	242520	224950	228680	257120
67	45	30	0.6	0.5	112440	134570	122110	117130	128750
68	45	30	0.6	0.7	119770	115430	141390	124260	102960
69	45	30	0.6	0.9	158290	159950	118800	95333	139870
70	45	30	0.8	0.5	201750	171600	133410	162990	179340
71	45	30	0.8	0.7	169570	193780	162880	201660	163300
72	45	30	0.8	0.9	202720	203480	170250	190630	165020
73	45	45	0.4	0.5	82581	75423	80834	94041	86137
74	45	45	0.4	0.7	149440	156320	148270	148050	170630
75	45	45	0.4	0.9	248160	237860	227560	260920	226710
76	45	45	0.6	0.5	82231	98645	77661	90508	94992
77	45	45	0.6	0.7	89192	86982	124400	79142	67675
78	45	45	0.6	0.9	136900	110910	107720	130050	92756
79	45	45	0.8	0.5	142150	145620	148450	146720	127890
80	45	45	0.8	0.7	176530	175920	148930	146310	175970
81	45	45	0.8	0.9	171030	204890	172550	151350	175610