UNIVERSITAS INDONESIA

PENJADWALAN PRODUKSI DENGAN METODE ALGORITMA *DIFFERENTIAL EVOLUTION* UNTUK MEMINIMALKAN *MAKESPAN* PADA *LINE BODY PRESS* INDUSTRI OTOMOTIF

SKRIPSI

NOVANDA ASTIAN
0606043673

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INDUSTRI
DEPOK
DESEMBER 2008

UNIVERSITAS INDONESIA

PENJADWALAN PRODUKSI DENGAN METODE ALGORITMA *DIFFERENTIAL EVOLUTION* UNTUK MEMINIMALKAN *MAKESPAN* PADA *LINE BODY PRESS* INDUSTRI OTOMOTIF

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana teknik

NOVANDA ASTIAN
0606043673

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INDUSTRI
DEPOK
DESEMBER 2008

HALAMAN PERNYATAAN ORISINALITAS

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

Nama	**: Novanda Astian**
NPM	**: 0606043673**
Tanda Tangan	**:**
Tanggal	**: 23 Desember 2008**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Novanda Astian
NPM	: 0606043673
Program Studi	: Teknik Industri
Judul Skripsi	: Penjadwalan Produksi Dengan Metode Algoritma *Differential Evolution* Untuk Meminimalkan *Makespan* Pada *Line Body Press* Industri Otomotif

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. Betrianis M.Si	( )
Pembimbing	: Arian Dhini, ST, MT.	( )
Penguji	: Dr. Ir. T. Yuri M. Zagloel, MEngSc	( )
Penguji	: Ir. Amar Rachman, MEIM	( )
Penguji	: Ir. Akhmad Hidayatno, MBT	( )

Ditetapkan di : Depok
Tanggal : 23 Desember 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. Betrianis, M.Si, dan Arian Dhini, ST, MT, selaku dosen pembimbing yang telah banyak memberi bantuan, masukan dan bimbingan yang berharga bagi penulis.
- Ir. Amar Rachman, MEIM, salah satu dosen/staf pengajar di TIUI yang telah banyak memberikan bantuan serta masukan kepada penulis.
- Dosen/staf pengajar di TIUI yang telah banyak memberikan pengetahuan dan masukan kepada penulis.
- Segenap karyawan/karyawati di TIUI yang telah membantu penulis dalam pengumpulan materi – materi pendukung skripsi serta proses administrasi, baik selama perkuliahan maupun proses pengesahan skripsi ini.
- Segenap karyawan PT. X yang telah memberikan kesempatan kepada penulis untuk mengumpulkan data untuk penelitian ini dan memberikan jawaban atas pertanyaan-pertanyaan yang penulis ajukan.
- Keluarga, atas curahan kasih sayang , dukungan, dan doa yang diberikan.
- Teman-teman penulis, khususnya rekan-rekan Ekstensi TIUI 2006 yang telah memberikan dukungan, semangat, serta kebersamaan selama perkuliahan ini.

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, Desember 2008

Penulis

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

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

Nama : Novanda Astian
NPM : 0606043673
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 :

Penjadwalan Produksi Dengan Metode Algoritma *Differential Evolution* Untuk Meminimalkan *Makespan* Pada *Line Body Press* Industri Otomotif

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

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 23 Desember 2008
Yang menyatakan

(Novanda Astian)

ABSTRAK

Nama : Novanda Astian
Program Studi : Teknik Industri
Judul : Penjadwalan Produksi Dengan Metode Algoritma *Differential Evolution* Untuk Meminimalkan *Makespan* Pada *Line Body Press* Industri Otomotif

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 nilai *makespan* (waktu total penyelesaian keseluruhan *job*).

Penjadwalan yang diperoleh melalui algoritma *differential evolution* menghasilkan nilai *makespan* sebesar 3198 menit, sedangkan jadwal perusahaan menghasilkan 3209 menit. Jadi, dengan menggunakan algoritma *differential evolution* terjadi pengurangan total waktu proses seluruh *job* yaitu 11 menit. Dalam penelitian ini digunakan data waktu proses yang sama agar hasil perhitungan dapat lebih akurat terhadap fungsi tujuan yang diinginkan.

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

ABSTRACT

Name : Novanda Astian
Study of Program : Industrial Engineering
Title : Production Schedule with Differential Evolution Algorithm to Minimize *Makespan* in Line Body Press Automotive Industry

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, easy to use, speed, and robustness. The objective function in this problem is to minimize total of finish time process of all jobs.

The result of scheduling that is obtained from differential evolution algorithm produces total of finish time process is 3198 minutes, meanwhile the schedule of company produces 3209 minutes. So, there are some reducing time of total finish time process of all jobs as much as 11 minutes. In this research, we use same data in order to get more accurate calculation based on objective function.

Key words :
Scheduling, Job Shop, Differential Evolution Algorithm

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang Permasalahan

Perkembangan dunia Industri dewasa ini, khususnya industri manufaktur, dimana permintaan akan suatu barang jadi menjadi cukup tinggi sehingga diharapkan dengan waktu produksi yang ada, industri – industri tersebut dapat melakukan proses produksi secara optimal dan efisien. Dengan semakin berkembangnya teknologi industri yang ada saat ini, maka bukan merupakan suatu halangan lagi bagi para industri untuk saling bersaing dalam menghadirkan suatu produk yang baik dengan tingkat pelayanan yang tinggi. Namun demikian, perlu menjadi perhatian bahwa dengan tingginya tingkat permintaan konsumen akan suatu produk, maka perlu untuk memperhatikan proses produksi yang sudah ada. Hal ini berkaitan dengan pemenuhan permintaan konsumen terhadap suatu produk/barang jadi dengan kualitas yang baik dan waktu produksi yang relatif efisien. Dengan dilakukannya optimalisasi dan efisiensi produksi, maka dengan demikian pelayanan terhadap permintaan konsumen menjadi lebih baik sehingga mampu meningkatkan pangsa pasar dari industri itu sendiri.

Oleh karena itu perlu untuk dilakukan perbaikan – perbaikan yang berkelanjutan dan bertahap pada masing – masing industri, agar pemenuhan akan permintaan konsumen menjadi lebih baik. Salah satu hal yang perlu menjadi perhatian bagi industri adalah mengenai penjadwalan. Penjadwalan mempunyai peranan yang penting dari suatu proses produksi, karena dengan adanya penjadwalan maka perusahaan akan mengetahui kapan pesanan konsumen akan terpenuhi, bagaimana proses produksinya, dan berapa biaya produksi yang dibutuhkan untuk melakukan proses produksi tersebut. Sehingga penjadwalan yang baik dengan menggunakan metode yang baik akan membuat proses produksi dapat berjalan secara optimal dan efisien.

Suatu metode penjadwalan produksi yaitu 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. Sebagaimana diketahui bahwa *scheduling* merupakan permasalahan *combinatorial optimization*, sehingga dikategorikan sebagai permasalahan *NP-complete*[2], yaitu permasalahan yang waktu komputasinya dalam pencarian solusinya akan naik secara eksponensial seiring dengan naiknya ukuran permasalahan secara *linear*. Algoritma *heuristic* dapat menyelesaikan permasalahan *NP-complete* dengan waktu perhitungan yang lebih singkat dibandingkan dengan metode eksakta.

Dengan menggunakan metode penjadwalan tersebut, PT X yang merupakan salah satu industri otomotif di Indonesia, dimana saat ini tengah mengalami permasalahan penjadwalan khususnya pada jalur *press*, akan mempunyai solusi bagi permasalahan penjadwalan produksinya. Pada jalur *press* beberapa produk yang merupakan bagian dari *body* mobil diproses dengan menggunakan beberapa buah mesin. Masing – masing produk tersebut umumnya melalui beberapa tahapan proses. Setiap tahapan proses tersebut dikerjakan pada mesin yang berbeda. Dengan demikian maka dibutuhkan suatu model penjadwalan produksi yang baik agar kombinasi proses dari setiap produk tersebut berlangsung efektif, optimal dan efisien sehingga target produksi yang diharapkan dapat tercapai. Selain itu supply part untuk kebutuhan proses produksi selanjutnya yaitu *welding* tidak mengalami *shortage*. Dengan demikian pemenuhan akan permintaan konsumen terhadap produk yang dihasilkan semakin cepat sehingga pangsa pasar dari produk tersebut semakin bertambah.

[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.

[2] Prof. PhD. Eng. Buzatu, C; DrD. Eng. Băncilă, D. 6th International DAAAM Baltic Conference, INDUSTRIAL ENGINEERING, 24-26 April 2008, Tallinn, Estonia

1.2 Diagram Keterkaitan Permasalahan

Permasalahan yang menuntut perlu adanya suatu penjadwalan produksi yang baik pada penelitian ini terangkum dalam satu diagram keterkaitan masalah yang dapat dilihat pada gambar 1.1.

1.3 Perumusan Permasalahan

Berdasarkan latar belakang permasalahan yang terangkum pada diagram keterkaitan masalah, maka permasalahan yang dapat dirumuskan pada penelitian ini adalah mengenai belum optimalnya penjadwalan produksi yang ada di PT X, khususnya pada jalur *press*.

1.4 Tujuan Penelitian

Memperoleh suatu penjadwalan produksi *(Job Shop)* yang optimal, khususnya dalam meminimalkan makespan dengan menggunakan metode *Differential Evolution Algorithm.*

1.5 Ruang Lingkup Penelitian

Agar penelitian menjadi terarah dan pemecahan masalah dapat dilakukan, maka peneliti perlu membatasi penelitian ini pada beberapa hal berikut ini :

1. Penelitian ini menggunakan studi kasus pada jalur *Press* di PT X yang bertipe produksi *Job Shop*.
2. Data – data untuk penelitian digunakan data rencana produksi yang diambil pada November 2008.
3. Data besarnya waktu set-up dan perpindahan semua material sudah termasuk kedalam waktu proses produksi yang bersangkutan.
4. Fungsi tujuan yang akan dievaluasi dalam penelitian ini adalah fungsi *makespan* (waktu penyelesaian maksimum).
5. Penelitian ini tidak memperhitungkan faktor kendala peralatan dan kendala pekerja.
6. Kondisi mesin produksi berjalan dengan normal, tidak mengalami kondsi *repair* ataupun *breakdown* di tengah – tengah waktu proses.

7. Kondisi bahan baku merupakan komponen standar yang diasumsikan memenuhi syarat dan dalam kondisi *ready stock*.

1.6 Metodologi Penelitian

Metodologi yang digunakan dalam penelitian ini dapat dilihat pada gambar 1.2 dengan penjelasan sebagai berikut :

1. Menentukan dan merumuskan permasalahan yang dapat diangkat sebagai objek penelitian, yaitu penjadwalan *Job Shop*.
2. Menentukan tujuan yang ingin dicapai dari penelitian.
3. Mempelajari literatur – literatur mengenai prinsip dan teori yang diperlukan untuk menyelesaikan permasalahan penelitian.
4. Mengumpulkan data – data yang dibutuhkan untuk membuat penjadwalan. Pengumpulan data ini dilakukan melalui studi terhadap laporan produksi perusahaan.
5. Melakukan pengolahan data dengan menggunakan metode *Differential Evolution Algorithm*.
6. Melakukan verifikasi dan validasi terhadap model program yang telah dibuat.
7. Mencari dan menyelesaikan permasalahan hingga menghasilkan suatu solusi yang optimal.
8. Melakukan analisa terhadap hasil pengolahan data dan solusi optimal yang telah dicapai.
9. Mengambil kesimpulan.

1.7 Sistematika Penulisan

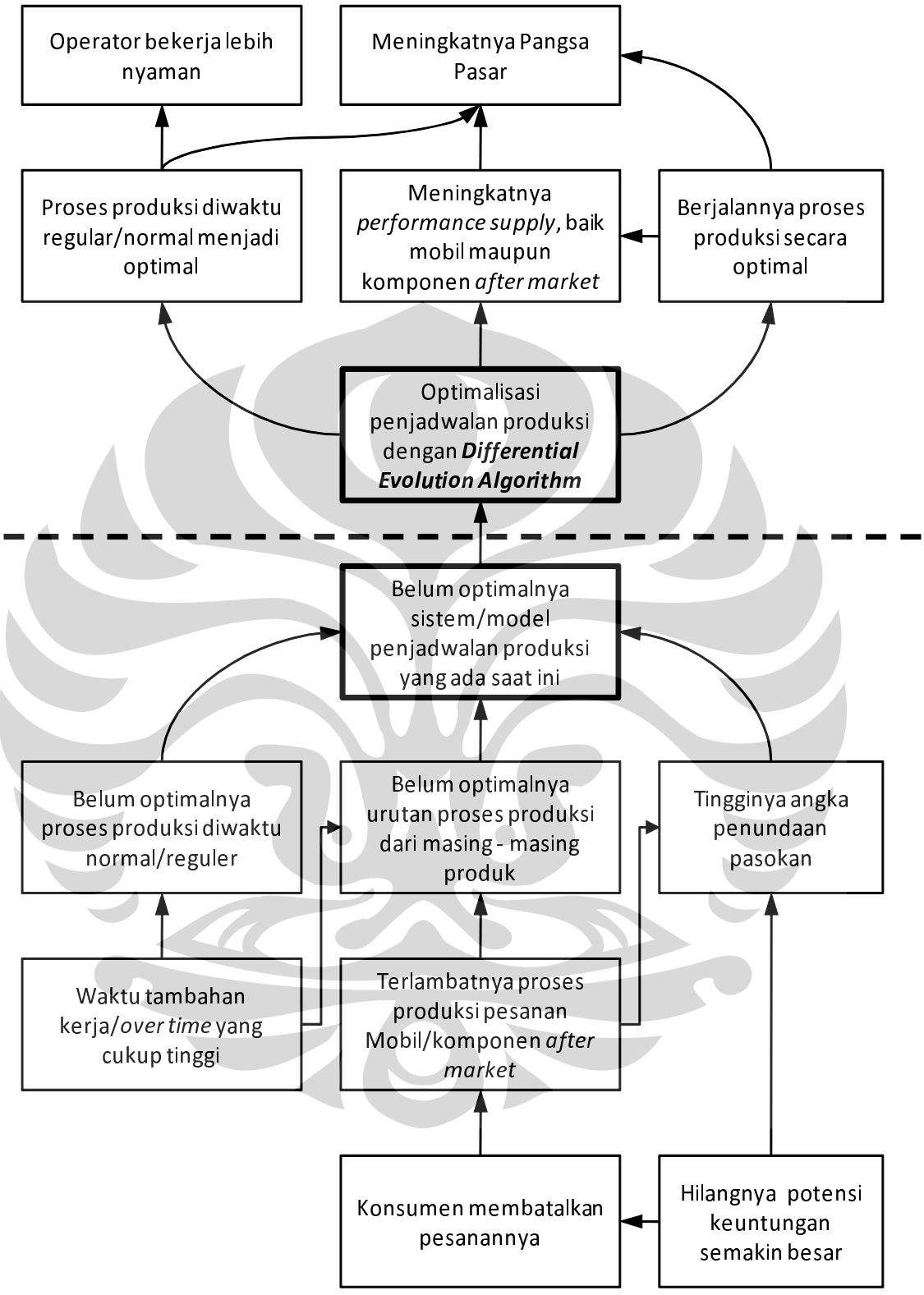
Penyusunan hasil penelitian skripsi ini mengacu pada aturan standar penyusunan skripsi yang terbagi kedalam lima bab, yaitu pendahuluan, landasan teori, pengumpulan data, pengolahan data serta analisa data, dan kesimpulan.

Pada bagian pendahuluan, bab 1, dijelaskan mengenai latar belakang permasalahan dari penelitian skripsi ini, dan diperkuat dengan diagram keterkaitan permasalahan serta perumusan masalah, tujuan yang ingin dicapai dari penelitian skripsi ini, ruang lingkup penelitian yang menjadi acuan dalam melakukan penelitian agar tujuan yang diinginkan dapat dicapai, serta dijabarkan bagaimana

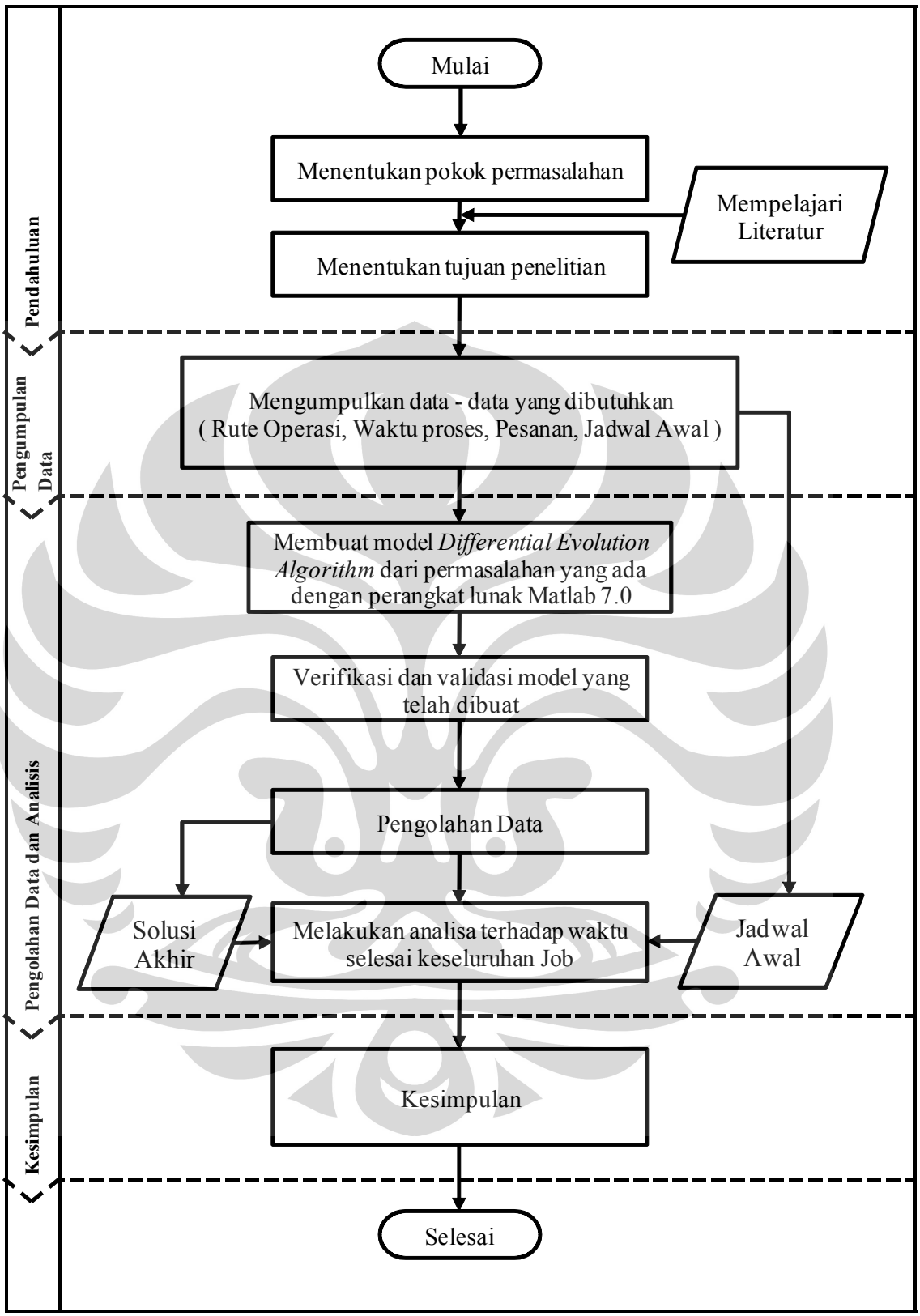
metodologi penelitian ini dilakukan, dan pemaparan sistematika penulisan agar memberikan kemudahan bagi pembaca dalam mengetahui gambaran langkah – langkah dan susunan penelitian ini.

Teori mengenai metode *Differential Evolution Algorithm* yang digunakan untuk melakukan optimalisasi penjadwalan produksi serta langkah – langkah penggunaannya akan dibahas pada bab 2, yaitu dasar teori. Kemudian mengenai cara dan metode dalam pengumpulan data, serta data yang diperlukan dalam penelitian ini akan dibahas pada bab 3, yaitu mengenai pengumpulan data. Untuk pengolahan data yang telah dikumpulkan serta analisa yang dilakukan terhadap data tersebut akan dijabarkan pada bab 4, yaitu pengolahan data dan analisis. Selain itu, pada bab 4 ini, untuk pengolahan datanya menggunakan pengembangan komputer berupa pemanfaatan software MATLAB untuk mendapatkan fungsi tujuan dari penelitian.

Hasil dari pengolahan data serta analisanya akan dituangkan pada bab 5, yaitu kesimpulan. Bagian ini berisi mengenai kesimpulan terhadap hasil dari penelitian yang telah dilakukan, sekaligus merupakan penutup dari penulisan penelitian skripsi ini.



Gambar 1.1 Diagram Keterkaitan Masalah



Gambar 1.2 Diagram Alir Metodologi Penelitian

BAB 2
DASAR TEORI

2.1 Penjadwalan Produksi

Penjadwalan merupakan salah satu bentuk dari pengambilan keputusan dimana hal tersebut memegang peranan penting, baik di industri manufaktur maupun industri jasa. Dalam kondisi saat ini dimana kompetisi sudah semakin ketatnya, maka penjadwalan produksi harus dilakukan dengan optimal dan efektif agar pemenuhan terhadap permintaan konsumen menjadi lebih baik. Hal tersebut tentunya akan memberikan dampak yang positif bagi industri tersebut.

2.1.1 Pengertian Penjadwalan Produksi

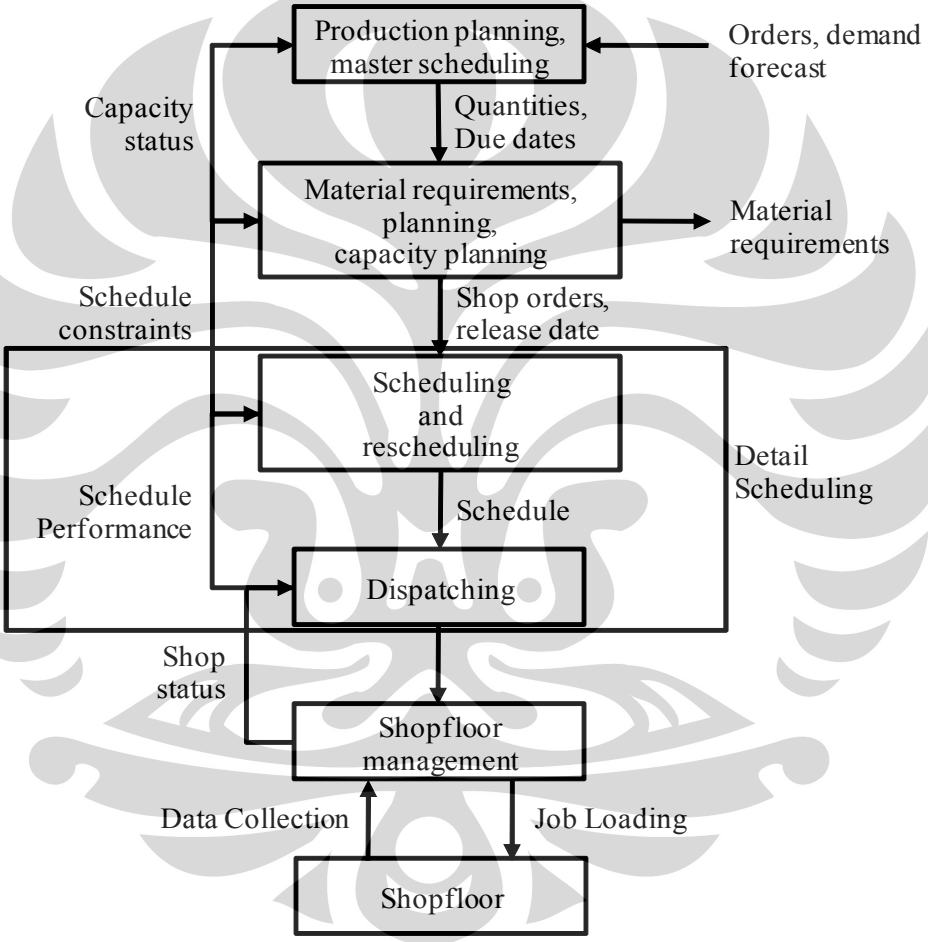
Penjadwalan atau *scheduling* mulai diperhatikan pada awal abad 20. Henry Laurence Gantt merupakan salah seorang perintis yang memperkenalkan sistem penjadwalan. Penjadwalan mulai memasuki bidang riset operasi pada tahun 1950-an. *Dynamic Programming* dan *Integer Programming* mulai digunakan untuk memecahkan masalah penjadwalan. Seiring dengan semakin berkembangnya ilmu dan teknologi komputer, penjadwalan dengan menggunakan program komputer mulai banyak digunakan. Selain itu masalah penjadwalan mulai menarik perhatian atau minat para *programmer*, ahli riset operasi, dan *industrial engineer*. Mereka terus melakukan studi mendalam tentang masalah penjadwalan yang semakin kompleks.

Perbedaan utama antara perencanaan dengan penjadwalan adalah pada kerangka waktu *(time frame)*. Perencanaan merupakan gambaran tentang kondisi yang dilakukan pada jangka waktu yang relatif panjang (bulan, semester, atau tahun). Sedangkan penjadwalan adalah merupakan suatu penugasan yang lebih terperinci dalam melakukan suatu aktivitas dalam rentang waktu yang relatif lebih pendek (jam, hari, atau minggu). Penjadwalan lebih berfokus pada alokasi sumber yang dimiliki *(limited resources)* untuk melakukan aktifitas setiap saat[3].

[3] Pinedo, Michael, Scheduling Theory,Algorithms and Systems, Prentice Hall, 1993, h. 1

2.1.2 Fungsi Penjadwalan Produksi

Sebuah penjadwalan yang terperinci tentang mesin yang digunakan, tahapan proses yang dilalui, jumlah barang yang diproduksi, dan waktu produksi akan membantu dalam proses pemenuhan pesanan dan menginformasikan kebutuhan sumber – sumber lainnya, misalnya kebutuhan material, kepada bagian yang terkait. Kemudian apabila terjadi perubahan terhadap penjadwalan yang telah dibuat, maka hal tersebut harus di informasikan juga kepada bagian yang terkait. Berikut model sistem informasi yang dibutuhkan[4] :



Gambar 2.1 Sistem Informasi Manufaktur

[4] Ibid, hal.9

2.1.3 Tipe Penjadwalan Produksi

Ada tiga jenis/tipe dari sistem penjadwalan[5], yaitu :

1. Penjadwalan Produksi Semi-*active* : yaitu suatu sistem penjadwalan dimana tidak adanya suatu proses/opearasi yang dapat dimulai lebih awal tanpa merubah urutan proses/operasi berikutnya.
2. Penjadwalan Produksi *active* : yaitu suatu sistem penjadwalan dimana tidak adanya suatu proses/operasi yang dapat dimulai lebih awal tanpa menunda beberapa proses/operasi yang lainnya.
3. Penjadwalan Produksi Non-*delay* : yaitu suatu sistem penjadwalan dimana tidak adanya suatu Mesin yang tidak beroperasi ketika beberapa proses/operasi berjalan.

2.1.4 Tujuan Penjadwalan Produksi

Adapun tujuan penjadwalan produksi antara lain[6] :

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

2.1.5 Klasifikasi Penjadwalan Produksi

Penjadwalan produksi dibagi menjadi beberapa kriteria, yaitu :

1. Berdasarkan Mesin yang digunakan dalam proses :
   a. Penjadwalan pada mesin tunggal *(single machine shop).*
   b. Penjadwalan pada mesin jamak.
2. Berdasarkan pola kedatangan *job* :
   a. Penjadwalan statis, dimana *job* datang bersamaan dan siap dikerjakan pada mesin yang tidak bekerja.
   b. Penjadwalan dinamis, dimana *job* datang tidak menentu.

[5] *José Fernando Gonçalves, Jorge José de Magalhães Mendes, Maurício G.C. Resende,* AT&T Labs Research Technical Report TD-5EAL6J, September 2002
[6] Steven Nahmias, *Production and Operation Analysis*, Mcgraw-Hill, New York, 2001, edisi ke-4, hal. 417

3. Berdasarkan lingkungan penjadwalan :

   a. *Flow Shop*

      Tiap *job* atau pesanan memiliki rute pengerjaan *(routing)* yang sama, dimana aliran dapat bersifat diskrit, kontinu, maupun semikontinu.

   b. *Job Shop*

      Setiap *job* atau pesanan memiliki rute pengerjaan yang berbeda – beda, sesuai dengan permintaan konsumen *(complex routing)*. Karena kompleksnya aliran proses, maka penjadwalan pun sangat kompleks. Aliran bersifat diskrit dan *part* tidak bersifat multiguna (*part* yang mungkin menjadi WIP-*work in process* pada *job* yang satu dan tidak bisa digunakan pada *job* yang lain).

   c. *Assembly Line*

      Hampir serupa dengan *flow shop*, akan tetapi proses hanya meliputi bagian perakitan dengan volume tinggi dan karakteristik produk yang sedikit. Tidak ditemui *buffer inventory*, kecuali pada bagian awal lini perakitan.

2.1.6 Istilah – istilah Dalam Penjadwalan Produksi

Beberapa istilah – istilah yang digunakan dalam penjadwalan produksi, antaran lain :

- Setiap *Job* i $\{i=1,2,...,n\}$ yang akan dijadwalkan pada *j* mesin $\{j=1,2,...,m\}$. 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 tersebut, 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*), *Cij*, adalah waktu penyelesaian pemrosesan *job i* pada mesin *j*.
- *Makespan* biasanya dilambangkan dengan C_{max}, yaitu waktu penyelesaian 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 prositif.
- 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*.

2.1.7 Karakteristik dan Kendala Proses Dalam Penjadwalan Produksi

Beberapa hal yang menjadi kendala dalam penjadwalan produksi, antara lain :

- Kendala *precedence*

  Kendala ini terjadi ketika suatu *job* baru mulai diproses setelah satu atau sekumpulan *job* lainnya telah selesai diproses.
- Kendala biaya dan waktu *setup* yang bergantung pada urutan *job* (*sequence-dependent*).
- *Preemption*

  Preemption berarti jika proses produksi sedang berlangsung, maka dapat dihentikan dan digantikan dengan mengerjakan *job* yang baru datang. Keadaan ini biasanya dikarenakan *job* yang berprioritas rendah dapat di sela 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.

2.1.8 Fungsi Tujuan dan Pengukuran Performa Penjadwalan Produksi

Tujuan yang biasa digunakan sebagai nilai fungsi tujuan dan untuk menilai performa penjadwalan produksi yang telah 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.

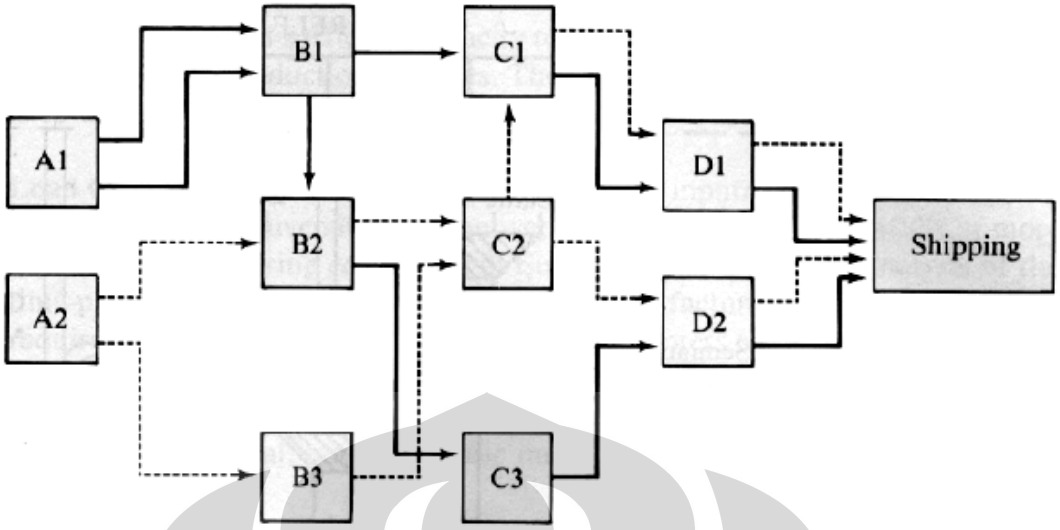
2.2 Penjadwalan *Job Shop*

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 suatu *job* oleh *job* lain).
- Permasalahan penjadwalan untuk model *job shop* merupakan salah satu permasalahan optimasi kombinatorial yang kompleks sehingga disebut *NP-hard* (*NP* merupakan singkatan dari *nondeterministic polynomial*).

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



Gambar 2.2 Rute Penjadwalan *Job Shop*

2.3 Metode Penjadwalan *Job Shop*

Metode yang digunakan dalam mengatasi permasalahan penjadwalan, yaitu diantaranya dengan mengunakan metode haeuristik yang terdiri dari :

2.3.1 Heuristik Klasik

Algoritma ini menyusun satu per satu solusi dari masalah penjadwalan. Mulai dari nol, algoritma – algoritma ini memilih mesin-mesin atau *job – job* atau operasi – operasi mana yang harus dijadwalkan terlebih dahulu. Algoritma heuristik klasik yang sering digunakan untuk menyelesaikan penjadwalan *job shop* yaitu *priority dispatch rule*. *Priority dispatch rule* adalah suatu aturan penjadwalan yang mengatur *job* mana pada suatu antrian *job* pada suatu mesin yang harus diproses terlebih dahulu berdasarkan prioritas – prioritas tertentu. Jadi, pada saat suatu mesin telah selesai memproses satu *job*, maka berdasarkan *priority dispatch rule* dipilih satu *job* yang memiliki prioritas tertinggi untuk selanjutnya diproses pada mesin tersebut.

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

1) *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.

2) *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.

3) *Minimum Slack First* (MS)

Menurut aturan ini, *job* diurutkan berdasarkan waktu *slack* yang paling kecil. Pada saat sebuah mesin selesai memproses 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*.

4) 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) *Longest Processing Time First* (LPT)

Menurut aturan ini, *job* yang memiliki waktu proses paling lama akan diproses terlebih dahulu dan kemudian dilanjutkan oleh *job – job* lainnya sampai pada *job* yang paling singkat waktu prosesnya. Aturan ini juga berguna untuk penyeimbangan beban kerja antar mesin yang disusun secara paralel.

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

1) *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.

2) *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 yang 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 – solusi tersebut dengan cara memodifikasinya berdasarkan aturan – aturan tertentu yang dikenal dengan nama *neighborhood functions*.

3) 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.

4) Algoritma *Differential Evolution*

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

2.4 Algoritma *Differential Evolution* (DE)

2.4.1 Sejarah Algoritma DE

Dalam matematika dan komputasi, algoritma merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah – perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan catatan untuk setiap masalah, ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria, dalam hal ini berbeda dengan heuristik.

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

[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

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[10]. Hingga kini, DE telah sukses diaplikasikan dalam berbagai bidang.

2.4.2 Konsep Dasar Algoritma DE

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

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

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. 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. Setelah melalui beberapa generasi dan mencapai kriteria terminasi, maka algoritma ini akan konvergen ke kromosom terbaik yang menjadi solusi penyelesaian.

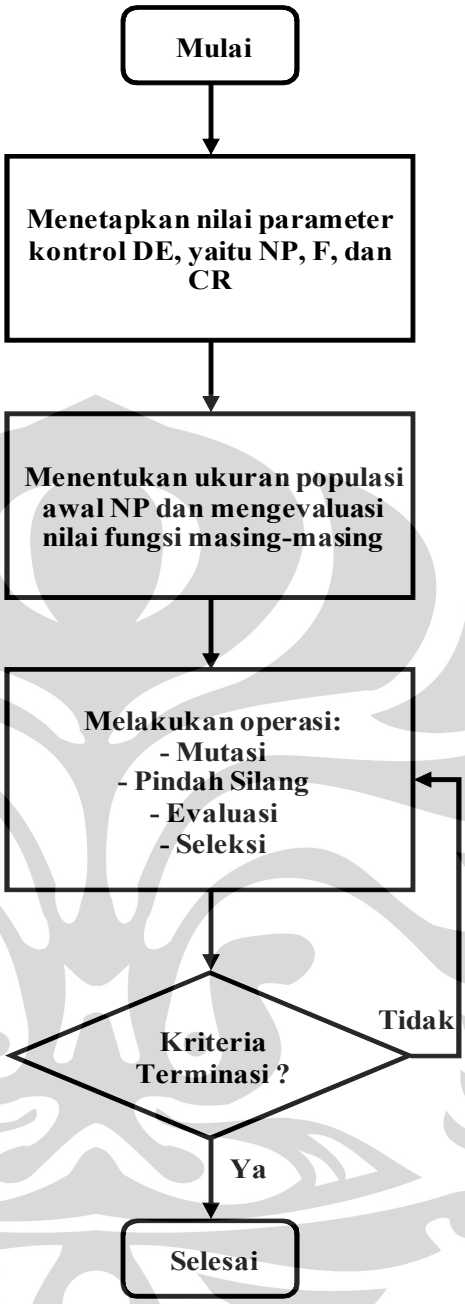
2.4.3 Tahapan Proses Pengerjaan Algoritma DE

Tahapan proses pengerjaan Algoritma DE dapat dilihat pada gambar 2.3 dan 2.4, berikut penjelasannya [11]:

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.

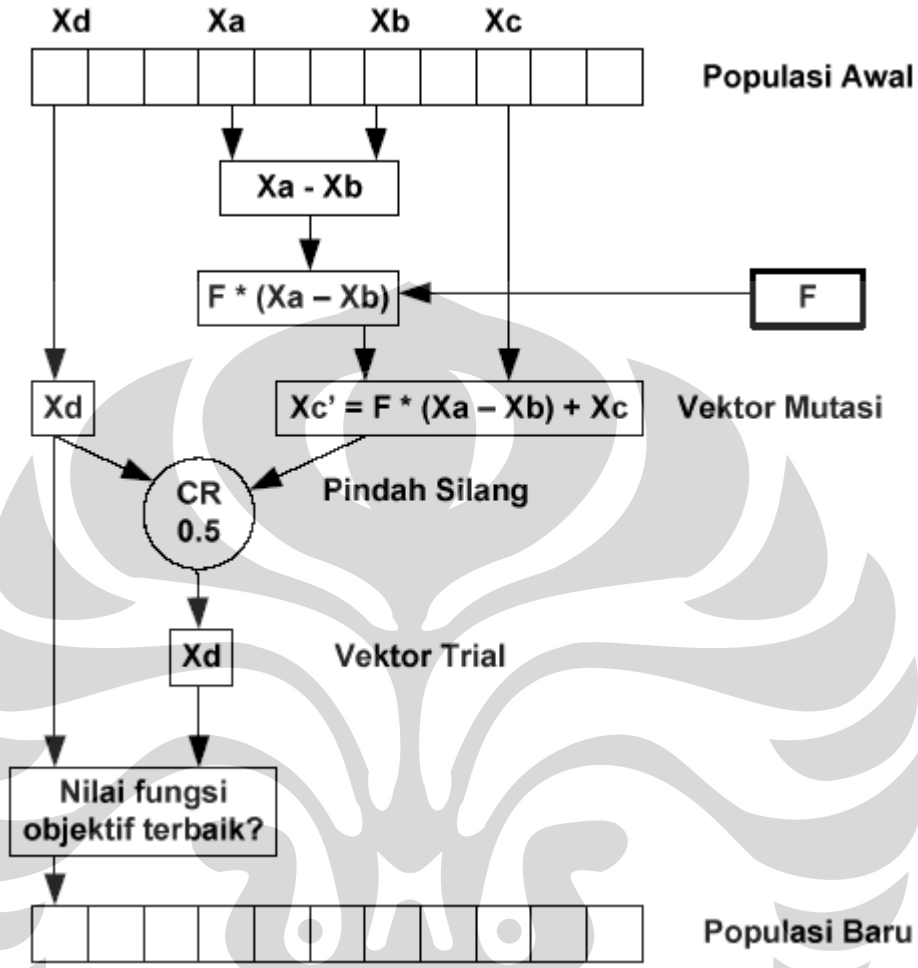
[11] Neal M., et. al., "*Applying Differential Evolution to a Whole-Farm Model to Assist Optimal Strategic Decision Making*", 2006



Gambar 2.3 Diagram Alir Proses Pengerjaan Algoritma DE

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.



Gambar 2.4 Representasi Proses *Differential Evolution*

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) 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.

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:

$$X_4' = F \times (X_2 - X_3) + X_4 \qquad (2.1)$$

Dimana:

X_4' = Vektor Mutasi

F = Parameter kontrol mutasi

X_2, X_3, X_4 = Vektor yang dipilih secara acak

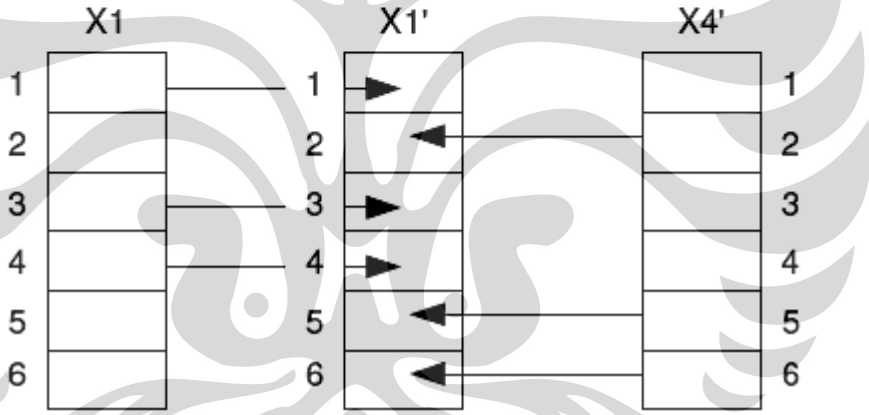
4) Pindah Silang

Pindah silang atau perkawinan silang bertujuan untuk memperoleh 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 2.5. Proses pindah silang akan dibantu dengan matriks baru yang nilainya diperoleh secara acak. Untuk kasus pada gambar 2.5, 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 2.5 Proses Pindah Silang

5) Evaluasi Vektor Trial

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

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[12].

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
```

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. DE dinotasikan dalam *DE/x/y/z*[13], 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

[12] 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.
[13] Rainer Storn and Kenneth Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces", page 345.

digunakan adalah DE/rand/1/bin[14]. Varian ini dianggap sebagai versi dasar dari DE.

2.4.4 Penerapan Algoritma DE pada Masalah Penjadwalan *Job Shop*

Algoritma DE yang akan diterapkan pada permasalahan ini menggunakan varian *DE/rand/1/bin*[15]. Adapun *pseudo code* algoritma DE yang akan digunakan adalah sebagai berikut:

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)

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[16]:

   - 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).

[14] 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
[15] 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
[16] *Ibid.*.,p.7

- 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 ket-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.
- 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 π_i^t = [π_{i1}^t, π_{i2}^t, …, $\pi_{i,n}^t$] dimana π_{ij}^t merupakan penugasan operasi ke-j individu ke-I pada iterasi ke-i.
- Konstanta mutasi : F $\in$ (0,2)
- Konstanta pindah silang : CR $\in$ (0,1)
- Fungsi objektif : Dalam suatu masalah minimasi, fungsi objektif dilambangkan dengan f_i ($\pi_i^t \leftarrow X_i^t$). Dalam penelitian ini, fungsi objektif yang digunakan adalah minimum makespan dengan persamaan matematisnya sebagai berikut :

  $$f_i\,(\pi_i^t \leftarrow X_i^t) = Max_{l\,\in Pn+1}\,\{F_l\}$$

  Dengan, P_{n+1} = Maksimum proses dari seluruh *job* yang mendahului
- Kriteria terminasi : kondisi dimana program akan berhenti berjalan, bisa dalam bentuk jumlah maksimum generasi atau waktu maksimum CPU bekerja.

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 2.6. Prosedur tersebut dijabarkan sebagai berikut:

a. Tahap Inisialisasi

❖ Menetapkan t = 0, CR, F, dan NP

Dimensi = jumlah *job*

❖ Membangun individu awal sebanyak NP, yakni $\{X_i^t, i = 1,2,\ldots,NP\}$, dimana $X_i^0 = [X_{i1}^0, X_{i2}^0, X_{i3}^0, \ldots, X_{i,nm}^0]$; $X_{ik}^0 = X_{min} + (X_{max} - X_{min})$ x r $X_{min} = -1$; $X_{max} = 1$; r = bilangan random (0,1)

❖ Melakukan operasi permutasi $\pi_i^0 = [\pi_{i1}^0, \pi_{i2}^0, \pi_{i3}^0, \ldots, \pi_{i,n}^0]$ 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 $(\pi_i^0 \leftarrow X_i^0)$ $(i = 1,2,\ldots,NP)$ untuk memilih individu target.

b. Meng-*update* generasi t = t + 1

c. Membentuk populasi mutan

Untuk setiap individu target, akan dicari individu mutan $V_i^{t+1} = [V_{i1}^{t+1}, V_{i2}^{t+1}, V_{i3}^{t+1}, \ldots, V_{i,nm}^{t+1}]$ yang diperoleh melalui operasi berikut :

$V_i^{t+1} = X_{ai}^t + F\,(X_{bi}^t - X_{ci}^t)$, $(a_i \neq b_i \neq c_i)$

d. Membentuk Populasi *Trial*

Individu *trial* $U_i^{t+1} = [u_{i1}^{t+1}, u_{i2}^{t+1}, u_{i3}^{t+1}, \ldots, u_{i,nm}^{t+1}]$ diperoleh melaui operasi berikut : $u_{ik}^{t+1} = \begin{Bmatrix} v_{ik}^{t+1}, if\ r_{ik}^{t+1} \leq CR \\ x_{ik}^{t+1}, otherwise \end{Bmatrix}$

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

e. Melakukan Operasi Permutasi *Job*

Operasi permutasi *job* dilakukan dengan menerapkan aturan *Smallest Position Value (SPV)* untuk melakukan operasi permutasi berikut :

$\varphi_i^t = [\varphi_{i1}^t, \varphi_{i2}^t, \varphi_{i3}^t, \ldots, \varphi_{i,nm}^t]$; $(i = 1,2,\ldots,NP)$

f. Mengevaluasi Populasi *Trial*

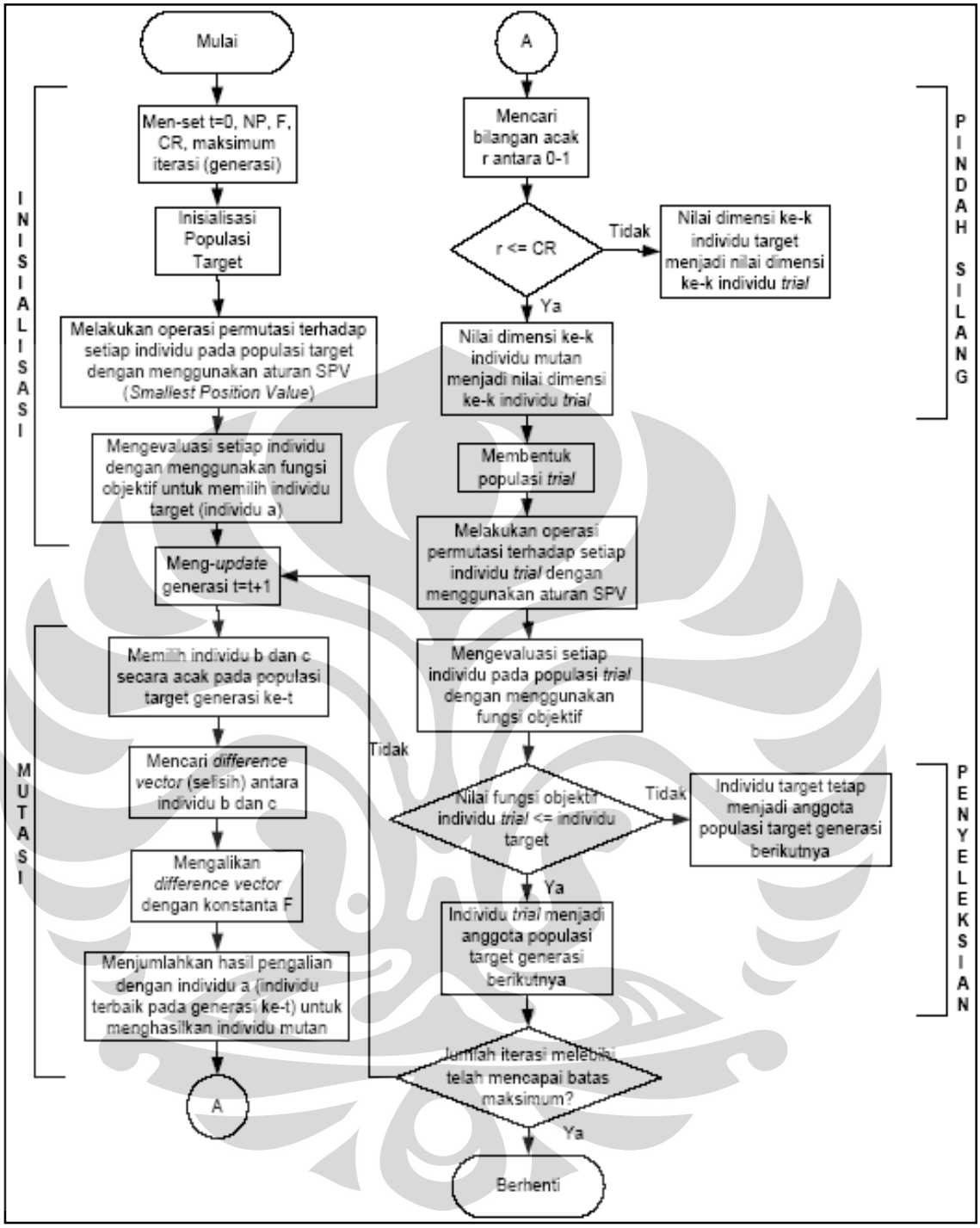
Evaluasi Populasi *trial* menggunakan fungsi objektif $f_i^{t+1}(\pi_i^{t+1} \leftarrow U_i^{t+1})$; $(i = 1,2,\ldots,NP)$

g. Melakukan Proses Seleksi

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 dari generasi berikutnya.

h. Menghentikan Operasi Pencarian

Jika jumlah iterasi sudah mencapai jumlah iterasi yang ditentukan, maka algoritma dihentikan. Namun jika hasil pencarian belum mencapai jumlah iterasi yang ditentukan, maka proses pencarian kembali ke tahap b.



Gambar 2.6 Diagram Algoritma DE untuk *Job Shop Sequencing Problem*

BAB 3
PENGUMPULAN DATA

3.1 Profil Perusahaan

PT X merupakan suatu industri manufaktur yang bergerak dibidang otomotif, dan sebagai salah satu contoh industri manufaktur yang ada di Indonesia. PT X didirikan sejak tahun 1992 dengan memproduksi berbagai jenis mobil. Sejak berdirinya perusahaan telah banyak mengalami perubahan hingga saat ini. Dengan semakin berkembangnya industri – industri manufaktur yang ada di Indonesia tentunya merupakan suatu tantangan tersendiri bagi PT X dalam mempersiapkan proses produksinya, agar produk yang dihasilkan dapat bersaing dipasaran. Dengan jumlah permintaan yang cukup tinggi saat ini, ditambah target perusahaan di tahun 2008 dengan total produksi sebanyak 250.000unit mobil, maka perlu untuk dilakukan perbaikan – perbaikan pada proses produksi yang ada saat ini, agar produktivitas dan efisiensi yang dihasilkan cukup baik.

Proses Produksi yang dilakukan oleh PT X, yaitu unit mobil, dimana proses produksi yang dilakukannya merupakan serangkaian proses. Dimulai dari proses *body press, welding, painting,* dan *assembly*. Selain itu, untuk unit *engine* sendiri diproduksi oleh PT X mulai dari proses pencetakan, *machining,* hingga proses perakitan *engine*.

3.2 Pengumpulan Data Penelitian

Data yang digunakan dalam penelitian ini merupakan data sekunder yang diperoleh dari data PT X. Data – data yang akan digunakan untuk penelitian mengenai penjadwalan *job shop* terdiri dari :

1. Data jam kerja di PT X

   PT X beroperasi mulai dari hari senin hingga hari jum'at. Setiap harinya PT X memberlakukan pembagian waktu kerja menjadi dua *shift* yaitu *shift* pagi dan malam. Untuk *shift* pagi dimulai pada pukul 07:15 hingga pukul 16:00 dengan total waktu kerja sebanyak +/- 7jam atau 455menit. Sedangkan untuk *shift* malam dimulai pada pukul 20:30 hingga pukul 04:30 dengan

total waktu kerja sebanyak +/- 7jam atau 425menit. Untuk lebih jelasnya dapat dilihat pada table 3.1 berikut.

Tabel 3.1 Jam Kerja PT X

Shift	Jadwal	Waktu Kerja	Durasi (menit)	Total Jam Kerja (menit)
P A G I	Morning Talk	07:15 - 07:25	10	**455**
	Produksi	07:25 - 09:45	140	
	Istirahat 1	09:45 - 09:50	5	
	Produksi	09:50 - 11:50	120	
	Istirahat 2	11:50 - 12:35	45	
	Produksi	12:35 - 14:05	90	
	Istirahat 3	14:05 - 14:15	10	
	Produksi	14:15 - 16:00	105	
M A L A M	Five Talk Meeting	20:30 - 20:40	10	**425**
	Produksi	20:40 - 22:00	80	
	Istirahat 1	22:00 - 22:05	5	
	Produksi	22:05 - 24:00	115	
	Istirahat 2	24:00 - 24:30	30	
	Produksi	24:30 - 02:30	120	
	Istirahat 3	02:30 - 02:40	10	
	Produksi	02:40 - 04:30	110	
TOTAL				**880**

2. Data Pesanan Part

Mengenai data pesanan dapat dilihat pada table 3.2 berikut ini.

Tabel 3.2 Data Pesanan Part

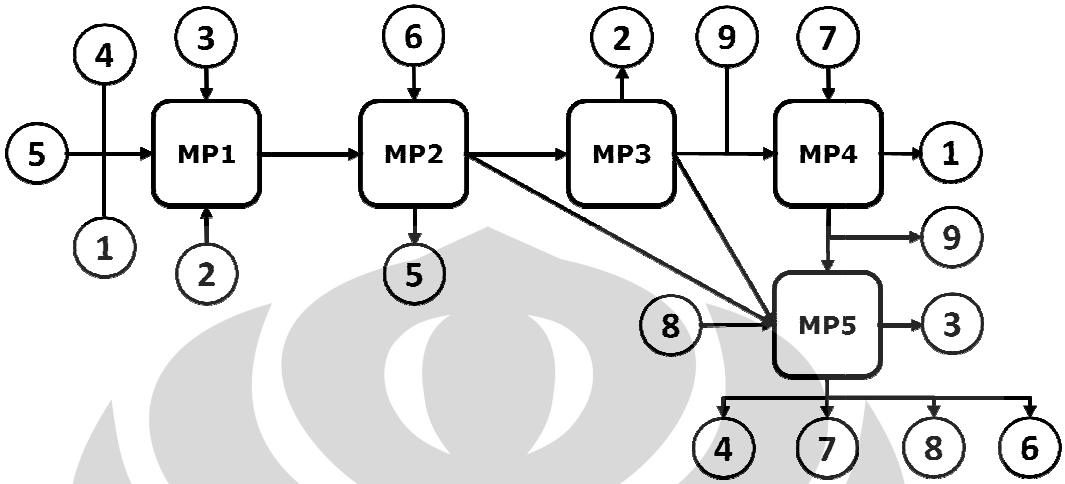
No	Part no	Part name	Pesanan Part		
			Pcs	Qty/Lot	Lot
1	55111-BZ280	PANEL, DASH	423	25	17
2	55711-BZ080	PANEL, COWL TOP, OUTER	350	50	7
3	64298-BZ010	PANEL, ROOM PARTITION CTR	150	25	6
4	57213-BZ020	EXTENSION, FR SIDE MEMBER, RH	250	25	10
5	57214-BZ020	EXTENSION, FR SIDE MEMBER, LH	250	25	10
6	57213-BZ030	EXTENSION, FR SIDE MEMBER, RH	150	25	6
7	57214-BZ030	EXTENSION, FR SIDE MEMBER, LH	150	25	6
8	57645-BZ060	REINFORCEMENT, RR SIDE MEMBER, RH	200	50	4
9	57646-BZ050	REINFORCEMENT, RR SIDE MEMBER, LH	200	50	4
10	61412-BZ060	PANEL, ROCKER, OUTER LH	225	25	9
11	53321-BZ130	PANEL, HOOD, INNER	400	50	8
12	67145-BZ050	PANEL, FR DOOR HINGE SIDE,RH	400	50	8
13	67146-BZ050	PANEL, FR DOOR HINGE SIDE,LH	400	50	8
14	53215-BZ060	SUPPORT HOOD LOCK	400	100	4
15	53735-BZ020	MEMBER, FR APRON TO COWL SIDE, OUT RR RH	350	50	7
16	53736-BZ020	MEMBER, FR APRON TO COWL SIDE, OUT RR LH	350	50	7
17	61613-BZ060	PANEL, QUARTER, INNER RH	350	50	7
18	61614-BZ060	PANEL, QUARTER, INNER LH	350	50	7
19	61615-BZ020	PANEL QUARTER INNER CTR RH	350	50	7
20	61616-BZ020	PANEL QUARTER INNER CTR LH	350	50	7

Tabel 3.2 Data Pesanan Part (Lanjutan)

No	Part no	Part name	Pesanan Part		
			Pcs	Qty/Lot	Lot
21	57113-BZ040	MEMBER FR SIDE OUTER RH	200	25	8
22	57114-BZ040	MEMBER FR SIDE OUTER LH	225	25	9
23	57411-BZ051	MEMBER, FLOOR SIDE, INNER RH	325	25	13
24	61627-BZ010	EXT, QUARTER PANEL, RR LWR RH	350	50	7
25	61628-BZ020	EXT, QUARTER PANEL, RR LWR LH	350	50	7
26	61735-BZ020	PANEL, ROOF SIDE, INNER RR RH	350	50	7
27	61736-BZ020	PANEL, ROOF SIDE, INNER RR LH	350	50	7
28	61731-BZ060	PANEL ROOF SIDE INNER RH	50	25	2
29	61732-BZ060	PANEL ROOF SIDE INNER LH	50	25	2
30	63132-BZ050	PANEL, WINDSHIELD HEADER, INNER	400	100	4
31	63134-BZ050	FRAME, BACK DOOR OPENING, INNER	50	25	2
32	53322-BZ010	R/F. HOOD PANEL	400	100	4
33	67147-BZ040	PANEL, RR DOOR HINGE SIDE, RH	350	50	7
34	67148-BZ040	PANEL, RR DOOR HINGE SIDE, LH	350	50	7
35	67349-BZ010	R/F, RR DOOR UPR FRAME, RH	350	50	7
36	67359-BZ010	R/F, RR DOOR UPR FRAME, LH	350	50	7
37	53713-BZ010	APRON FR FENDER RR RH	800	100	8
38	53714-BZ010	APRON FR FENDER RR LH	800	100	8
39	57113-BZ010	MEMBER FR SIDE OUTER RH	1000	100	10
40	57114-BZ010	MEMBER FR SIDE OUTER LH	1000	100	10
41	57181-BZ010	R/F FR SIDE MEMBER RR No.1 RH	800	100	8
42	57182-BZ010	R/F FR SIDE MEMBER RR No.1 LH	760	100	8
43	57258-BZ010	BRACKET ENGINE RR MOUNTING MEMBER LH	1000	100	10
44	57654-BZ010	MEMBER RR FLOOR CROSS NO.3	1000	100	10
45	57632-BZ010	MEMBER RR FLOOR CTR No.1	1000	100	10
46	55185-BZ040	BRACE DASH PANEL TO COWL	1000	100	10
47	55744-BZ020	REINFORCEMENT PEDAL BRACKET MTG	275	25	11
48	51545-BZ011	BRT LWR CONTROL LINK INNER RH	1000	100	10
49	51546-BZ011	BRT LWR CONTROL LINK INNER LH	1000	100	10
50	57162-BZ030	MEMBER, FR CROSS, RR	1100	100	11
51	57145-BZ011	R/F FR SIDE MEMBER RR RH	1000	100	10
52	57146-BZ011	R/F FR SIDE MEMBER RR LH	1000	100	10
53	61173-BZ010	R/F. FR BODY PILLAR, UPR INNER RH	800	100	8
54	61174-BZ010	R/F. FR BODY PILLAR, UPR INNER LH	800	100	8
55	61675-BZ010	R/F,QTR WHL HOUSE,INNER RR RH	800	100	8
56	61676-BZ010	R/F,QTR WHL HOUSE,INNER RR LH	800	100	8
57	61162-BZ051	PILLAR FR BODY INNER LH	260	20	13
58	63134-BZ010	FRAME BACK DOOR OPENING INNER	800	100	8
TOTAL			29343	3670	456

3. Data Rute dan Waktu Proses

   a. Rute Proses Produksi



Keterangan :

- MP1,MP2,MP3,MP4,MP5 : Mesin Press 1,2,3,4,5
- 1 → Produk dengan rute proses : MP1→MP2→MP3→MP4
- 2 → Produk dengan rute proses : MP1→MP2→MP3
- 3 → Produk dengan rute proses : MP1→MP2→MP3→MP5
- 4 → Produk dengan rute proses : MP1→MP2→MP5
- 5 → Produk dengan rute proses : MP1→MP2
- 6 → Produk dengan rute proses : MP2→MP3→MP5
- 7 → Produk dengan rute proses : MP4→MP5
- 8 → Produk dengan rute proses : MP5
- 9 → Produk dengan rute proses : MP4

Gambar 3.1 Rute Proses Produksi

b. Waktu Proses Produksi

Tabel 3.3 Waktu Proses Produksi

No	Part no	Part name	Press Time/Lot (Minutes)				
			MP1	MP2	MP3	MP4	MP5
1	55111-BZ280	PANEL, DASH	0,77	0,77	0,77	0,77	0
2	55711-BZ080	PANEL, COWL TOP, OUTER	2	2	2	0	0
3	64298-BZ010	PANEL, ROOM PARTITION CTR	0	0	0	1,49	1,49
4	57213-BZ020	EXTENSION, FR SIDE MEMBER, RH	0,77	0,77	0,77	0	0,77
5	57214-BZ020	EXTENSION, FR SIDE MEMBER, LH	0,74	0,74	0,74	0	0,74
6	57213-BZ030	EXTENSION, FR SIDE MEMBER, RH	0,82	0,82	0,82	0,82	0
7	57214-BZ030	EXTENSION, FR SIDE MEMBER, LH	0,79	0,79	0,79	0,79	0
8	57645-BZ060	REINFORCEMENT, RR SIDE MEMBER, RH	1,8	1,8	0	0	1,8
9	57646-BZ050	REINFORCEMENT, RR SIDE MEMBER, LH	1,87	1,87	0	0	1,87
10	61412-BZ060	PANEL, ROCKER, OUTER LH	0,98	0,98	0,98	0	0
11	53321-BZ130	PANEL, HOOD, INNER	3,15	3,15	0	0	0
12	67145-BZ050	PANEL, FR DOOR HINGE SIDE,RH	2	2	2	0	0
13	67146-BZ050	PANEL, FR DOOR HINGE SIDE,LH	1,98	1,98	1,98	0	0
14	53215-BZ060	SUPPORT HOOD LOCK	0	0	0	5,03	5,03
15	53735-BZ020	MEMBER, FR APRON TO COWL SIDE, OUT RR RH	2,03	2,03	2,03	0	0
16	53736-BZ020	MEMBER, FR APRON TO COWL SIDE, OUT RR LH	2,02	2,02	2,02	0	0
17	61613-BZ060	PANEL, QUARTER, INNER RH	2,01	2,01	2,01	0	0
18	61614-BZ060	PANEL, QUARTER, INNER LH	1,99	1,99	1,99	0	0
19	61615-BZ020	PANEL QUARTER INNER CTR RH	1,5	1,5	1,5	1,5	0
20	61616-BZ020	PANEL QUARTER INNER CTR LH	1,49	1,49	1,49	1,49	0
21	57113-BZ040	MEMBER FR SIDE OUTER RH	0	0,91	0,91	0	0,91
22	57114-BZ040	MEMBER FR SIDE OUTER LH	0	0,88	0,88	0	0,88
23	57411-BZ051	MEMBER, FLOOR SIDE, INNER RH	1,03	1,03	1,03	0	0
24	61627-BZ010	EXT, QUARTER PANEL, RR LWR RH	1,38	1,38	1,38	1,38	0
25	61628-BZ020	EXT, QUARTER PANEL, RR LWR LH	1,36	1,36	1,36	1,36	0
26	61735-BZ020	PANEL, ROOF SIDE, INNER RR RH	1,36	1,36	1,36	1,36	0
27	61736-BZ020	PANEL, ROOF SIDE, INNER RR LH	1,34	1,34	1,34	1,34	0
28	61731-BZ060	PANEL ROOF SIDE INNER RH	1,51	1,51	0	0	0
29	61732-BZ060	PANEL ROOF SIDE INNER LH	1,38	1,38	0	0	0
30	63132-BZ050	PANEL, WINDSHIELD HEADER, INNER	0	0	0	5,59	5,59
31	63134-BZ050	FRAME, BACK DOOR OPENING, INNER	0	0	0	1,45	1,45
32	53322-BZ010	R/F. HOOD PANEL	0	0	0	0	11,63
33	67147-BZ040	PANEL, RR DOOR HINGE SIDE, RH	1,51	1,51	1,51	1,51	0
34	67148-BZ040	PANEL, RR DOOR HINGE SIDE, LH	1,51	1,51	1,51	1,51	0
35	67349-BZ010	R/F, RR DOOR UPR FRAME, RH	0	0	0	0	5,44
36	67359-BZ010	R/F, RR DOOR UPR FRAME, LH	0	0	0	0	5,42
37	53713-BZ010	APRON FR FENDER RR RH	3,89	3,89	3,89	0	0
38	53714-BZ010	APRON FR FENDER RR LH	3,89	3,89	3,89	0	0
39	57113-BZ010	MEMBER FR SIDE OUTER RH	0	0	0	5,62	5,62
40	57114-BZ010	MEMBER FR SIDE OUTER LH	0	0	0	5,61	5,61
41	57181-BZ010	R/F FR SIDE MEMBER RR No.1 RH	0	0	0	5,41	5,41
42	57182-BZ010	R/F FR SIDE MEMBER RR No.1 LH	0	0	0	5,4	5,4
43	57258-BZ010	BRACKET ENGINE RR MOUNTING MEMBER LH	0	0	0	0	12,23
44	57654-BZ010	MEMBER RR FLOOR CROSS NO.3	0	0	0	0	9,93
45	57632-BZ010	MEMBER RR FLOOR CTR No.1	0	0	0	0	10,01
46	55185-BZ040	BRACE DASH PANEL TO COWL	0	0	0	0	10,28
47	55744-BZ020	REINFORCEMENT PEDAL BRACKET MTG	0	0	0	1,4	1,4
48	51545-BZ011	BRT LWR CONTROL LINK INNER RH	2,47	2,47	2,47	2,47	0
49	51546-BZ011	BRT LWR CONTROL LINK INNER LH	2,46	2,46	2,46	2,46	0
50	57162-BZ030	MEMBER, FR CROSS, RR	0	0	0	12	0
51	57145-BZ011	R/F FR SIDE MEMBER RR RH	2,85	2,85	2,85	2,85	0
52	57146-BZ011	R/F FR SIDE MEMBER RR LH	2,84	2,84	2,84	2,84	0
53	61173-BZ010	R/F. FR BODY PILLAR, UPR INNER RH	0	0	0	5,05	5,05
54	61174-BZ010	R/F. FR BODY PILLAR, UPR INNER LH	0	0	0	5,1	5,1
55	61675-BZ010	R/F,QTR WHL HOUSE,INNER RR RH	0	0	0	0	10,16
56	61676-BZ010	R/F,QTR WHL HOUSE,INNER RR LH	0	0	0	0	10,15
57	61162-BZ051	PILLAR FR BODY INNER LH	1,27	1,27	0	0	0
58	63134-BZ010	FRAME BACK DOOR OPENING INNER	0	0	0	5,92	5,92

BAB 4
PENGOLAHAN DATA DAN ANALISA

4.1 Pengolahan Data

Pengolahan data yang dilakukan pada penelitian ini menggunakan suatu metode algoritma yaitu *Differential Evolution Algorithm*. Metode tersebut membantu dalam merancang suatu penjadwalan produksi yang optimal. Setelah penjadwalan terbentuk, kemudian dilakukan proses perhitungan terhadap nilai waktu total produksi *(makespan)* yang merupakan operasi matematika. Nilai *makespan* inilah yang menjadi indikator terhadap optimalnya suatu penjadwalan produksi yang disusun. Perhitungan nilai waktu total *(makespan)* tersebut dilakukan dengan bantuan sebuah *software* yaitu Matlab. Dengan bantuan *software* tersebut diharapkan waktu komputasi dari operasi matematika yang ada, dapat dilakukan lebih cepat dan menghasilkan nilai perhitungan yang tepat.

4.1.1 Langkah – langkah Penyusunan Algoritma *Differential Evolution*

Pada gambar 2.6 telah digambarkan mengenai diagram alir dari algoritma DE dalam menyelesaikan permasalahan urutan pengerjaan *Job*. Prosedur penyusunan program algoritma tersebut dapat diuraikan sebagai berikut :

1) Inisialisasi Populasi Awal
   a. Melakukan penyetelan pada generasi ke-0

      Pada awal perhitungan yaitu sebelum generasi (iterasi) dimulai, dilakukan input parameter – parameter, yaitu ukuran populasi (NP), operator mutasi (F), operator pindah silang (CR), serta jumlah generasi (iterasi) itu sendiri. Untuk menghasilkan hasil yang efektif, perlu untuk dilakukan serangkaian percobaan agar diketahui bahwa nilai dari masing – masing parameter tersebut akan menghasilkan perhitungan yang baik terhadap fungsi tujuan yang diinginkan. Berdasarkan literatur – literatur, nilai ukuran populasi (NP) biasanya berada diantara 5*D(Dimensi) atau 10*D(Dimensi). Untuk jumlah iterasi sendiri, ditentukan berdasarkan kemampuan komputer dalam melakukan perhitungan (komputasi). Tentunya jumlah iterasi yang banyak akan menghasilkan perhitungan

(komputasi) yang baik, tetapi biasanya waktu perhitungan (komputasi) akan menjadi lebih lama. Untuk operator mutasi (F), nilainya berada diantara 0,4 dan 1. Sedangkan untuk operator pindah silang (CR), biasanya langsung ditentukan sebesar 0,9 atau 1,0. Hal ini dilakukan agar hasil perhitungan (komputasi), diperoleh lebih cepat. Setelah mempelajari literatur – literatur yang ada serta percobaan yang dilakukan terhadap data waktu proses yang ada, maka akan ditentukan nilai dari masing – masing parameter – parameter input sebagai berikut : NP = 50, JI = 50, F = 0,6, dan CR = 0,7. Selanjutnya parameter – parameter input tersebut akan digunakan untuk menghitung nilai *makespan* serta urutan prosesnya.

b. Menentukan Populasi Awal (populasi target awal)

Ukuran populasi menyatakan jumlah individu dalam populasi. Satu individu dinyatakan dengan satu kolom. Tiap individu memiliki 88 gen/dimensi (jumlah *job*/pesanan), dinyatakan dengan jumlah baris. Populasi awal dibentuk dimana setiap dimensi untuk setiap individu dicari secara acak sebagai berikut :

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 50 individu (nilai NP) dan setiap individu terdiri dari 88 dimensi, maka populasi awal nerupakan matriks berukuran 88 x 50. Pada algoritma DE ini, definisi individu dan vektor adalah sama.

c. Menentukan Vektor Permutasi (Urutan)

Setiap individu awal dari populasi awal ini diurutkan menjadi vektor permutasi sehingga dapat merepresentasikan urutan *job*. Nilai dimensi pertama hingga ke-88 setiap individu awal memiliki nilai yang berbeda – beda. Pengurutan nilai dimensi/*job* dari masing – masing individu didasarkan pada nilai dimensi/*job* yang terkecil hingga terbesar. Hasil dari pengurutan inilah yang nantinya akan menjadi urutan – urutan pengerjaan dari masing – masing *job* sesuai dengan data penelitian.

d. Mengevaluasi Setiap Individu

Setelah proses pengurutan vektor dari masing – masing individu selesai dilakukan, kemudian dilakukan evaluasi fungsi objektif dari masing – masing individu tersebut, dalam hal ini fungsi objektifnya adalah total waktu proses keseluruhan *job (makespan)*. Proses evaluasi dilakukan dengan cara menghubungkan vektor tersebut dengan waktu prosesnya, sehingga fungsi objektif dapat dihitung. Proses evaluasi ini dimaksudkan untuk mengetahui individu pada generasi (iterasi) awal yang memiliki total waktu proses keseluruhan *job* terkecil yang selanjutnya akan diturunkan menjadi generasi berikutnya.

e. Memperbaharui Generasi (iterasi)

Populasi individu pada generasi (iterasi) awal akan berevolusi membentuk populasi generasi (iterasi) baru. Masing – masing individu dari populasi setiap generasi akan melalui serangkaian proses, yang akan dimulai dengan proses mutasi, proses pindah silang, dan proses penyeleksian. Jika generasi (iterasi) awal disimbolkan dengan t=0, maka generasi (iterasi) baru disimbolkan sebagai t=t+1.

2) Proses Mutasi

Proses Mutasi merupakan bagian dari rangkaian proses Algoritma DE. Proses ini merupakan langkah pertama individu dalam berevolusi. Pada proses mutasi ini yang menjadi penekanannya adalah pada perbedaan nilai atau selisih dari dua vektor acak (vektor acak 1 dan vektor acak 2) yang memunculkan *difference vektor*. Selanjutnya *difference vektor* tersebut dikalikan dengan operasi mutasi (F) yang merupakan nilai parameter input. Setelah dilakukan proses pengalian, selanjutnya hasil/nilai dari proses tersebut ditambahkan dengan vektor target. Proses mutasi ini dianggap sangat penting, sehingga penentuan nilai parameter (F) akan menentukan hasil yang diperoleh.

3) Proses Pindah Silang

Proses pindah silang merupakan proses rekombinasi antara individu mutan dengan individu target, dimana nantinya akan menghasilkan individu *trial*. Nilai dimensi dari individu *trial* ini berasal dari sebagian individu target dan

sebagian lagi dari individu mutan, tentunya dengan mempertimbangkan nilai operator pindah silang (CR) dan bilangan acak r. Jika bilangan acak r (antara 0 sampai 1) yang diperoleh mempunyai nilai yang lebih kecil atau sama dengan nilai operator pindah silang (CR), maka yang berpeluang menjadi dimensi ke-k dari individu *trial* adalah nilai dimensi individu ke-k dari individu mutan, begitupun sebaliknya. Nilai operator pindah silang berada pada rentang 0 sampai dengan 1.

4) Proses Penyeleksian Individu

   Proses seleksi dilakukan terhadap individu target dan individu *trial* yang dihasilkan melalui rangkaian proses diatas. Proses seleksi ini bertujuan untuk menentukan individu yang layak untuk menjadi anggota pada generasi berikutnya. Proses penyeleksian individu ini didasarkan pada nilai fungsi objektif dari masing – masing individu, yaitu individu target dan individu *trial*. Individu yang memiliki nilai fungsi ojektif yang lebih baik, dalam penelitian ini yaitu nilai yang lebih kecil, akan menjadi individu anggota populasi generasi berikutnya.

5) Terminasi

   Pada penelitian ini, kriteria terminasi yang digunakan adalah jumlah iterasi (generasi). Proses pembentukan iterasi baru akan terus berlangsung hingga jumlah iterasi yang ditetapkan terpenuhi. Jumlah iterasi yang ditentukan adalah 2000 iterasi. Hal ini didasarkan pada literature yang ada bahwa jumlah iterasi yang ditetapkan semakin besar, maka nilai/hasil yang didapat akan semakin baik. Tetapi penentuan jumlah iterasi tersebut juga dipengaruhi oleh lamanya waktu komputasi. Jumlah iterasi yang besar memiliki kemungkinan untuk mencapai nilai yang optimal, tetapi biasanya akan membutuhkan waktu komputasi yang agak lama.

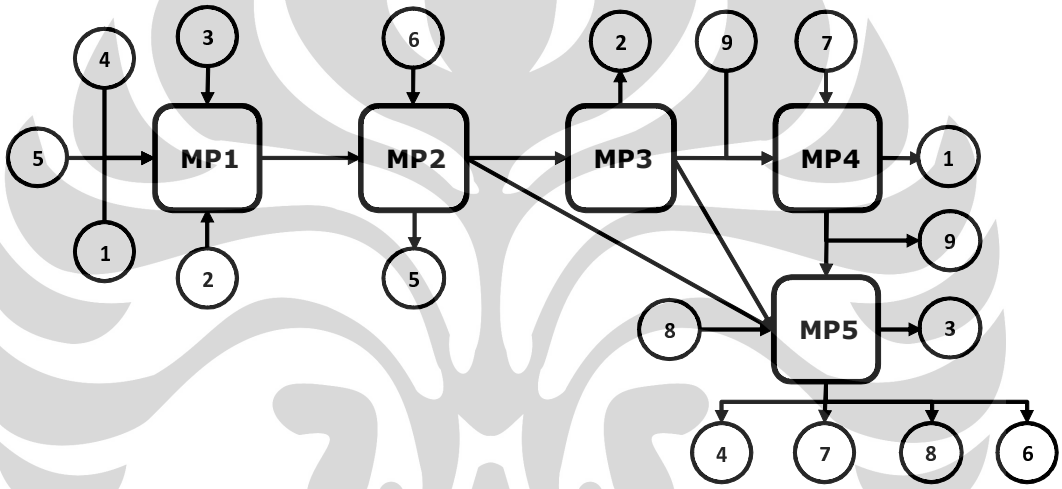
4.1.2 Verifikasi dan Validasi Program

Sebelum data penelitian diolah kedalam program yang telah disusun, verifikasi dan validasi terhadap program yang telah disusun perlu untuk dilakukan. Hal tersebut dilakukan dengan tujuan agar program yang telah disusun berjalan sesuai dengan yang diinginkan. Model program telah terverifikasi apabila

telah berjalan sesuai dengan konseptual model, yaitu adanya output berupa nilai *makespan* (waktu penyelesaian keseluruhan *job*) dan urutan pengerjaan *job*.

Selanjutnya dilakukan proses validasi program. Validasi terhadap program dilakukan dengan memasukkan data *dummy*. Hasil *run* program dengan menggunakan data *dummy* tersebut kemudian dibandingkan dengan hasil perhitungan manual. Jika hasil keduanya sama, maka program telah tervalidasi.

Data *dummy* yang digunakan adalah 9 buah *job* dimana setiap *job* memiliki rute proses yang berbeda – beda. Gambaran mengenai rute proses masing – masing *job* dapat dilihat pada gambar 4.1 berikut ini :



Gambar 4.1 Rute Proses Produksi

Untuk waktu operasi dari masing – masing *job* dapat dilhat pada tabel 4.1. Waktu operasi yang bernilai nol menunjukkan bahwa rute tersebut tidak dilalui oleh *job* yang bersangkutan. Jumlah operasi dan jumlah mesin yang digunakan sesuai dengan data penelitian.

Untuk parameter algoritma DE yang digunakan pada proses validasi program juga ditentukan, dan dapat dilihat pada tabel 4.2.

Tabel 4.1 Data *dummy* untuk validasi program

Job	Rute dan waktu operasi (menit)				
	MP1	MP2	MP3	MP4	MP5
1	2	2	2	2	0
2	5	5	5	0	0
3	1	1	1	0	1
4	4	4	0	0	4
5	6	6	0	0	0
6	0	3	3	0	3
7	0	0	0	7	7
8	0	0	0	0	8
9	0	0	0	9	0

Tabel 4.2 Parameter untuk validasi program

Parameter	Nilai
Ukuran Populasi	5
Jumlah iterasi	1
Operator Mutasi	0,6
Operator Pindah Silang	0,7

1. Hasil *Run* Program

   Hasil *run* program menunjukkan bahwa urutan proses terbaik yaitu 8-7-2-4-6-3-9-1-5 dengan nilai *makespan* yaitu 26 menit.

2. Hasil Perhitungan Manual

   Langkah – langkah dalam perhitungan manual dilakukan sebagai berikut :

   a. 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 dari data *dummy* ini berukuran 5 x 9 (ukuran populasi x jumlah *job*). Matriks populasi target ini adalah matriks yang memiliki populasi sebanyak 5 individu atau vektor. Jadi satu kolom akan merepresentasikan satu individu. Setiap kolom memiliki sembilan baris, artinya setiap individu memiliki sembilan buah gen (*job*).

Tabel 4.3 Populasi Target (hasil *run* program)

Individu				
1	**2**	**3**	**4**	**5**
0,6680	0,8589	-0,1795	1,0912	0,6924
-0,5377	-0,2039	0,7873	-0,9496	0,0503
0,2137	-0,6672	-0,8842	-0,4981	-0,5947
-0,0280	0,8436	-0,2943	-0,9695	0,3443
0,7826	0,8725	0,6263	0,4936	0,6762
0,0019	0,3245	-0,9803	0,8468	-0,9607
-1,0386	-0,4072	-0,7222	0,5443	0,3626
-1,2789	-1,8422	-0,5945	-1,5264	-0,2410
0,3789	-0,2190	-0,6026	0,0449	0,6636

b. Melakukan permutasi pada setiap individu dari populasi target

Proses pengurutan (permutasi) dari populasi target dilakukan dengan mengurutkan bilangan acak pada masing – masing kolom dari tabel 4.3 diatas. Urutan dari masing – masing kolom tersebut dimulai dari nilai yang terkecil hingga yang terbesar. Sebagai contoh, untuk kolom 1 (individu 1), urutannya adalah : -1,2789; -1,0386; -0,5377; -0,0280; 0,0019; 0,2137; 0,3789; 0,6680; 0,7826. Nilai – nilai tersebut berada pada urutan ke 8, 7, 2, 4, 6, 3, 9, 1, dan 5. Jadi urutan pengerjaan *job* untuk kolom 1 (individu 1) adalah 8-7-2-4-6-3-9-1-5. Tabel 4.4 menyatakan permutasi (urutan pengerjaan *job*) populasi target untuk masing – masing individu.

Tabel 4.4 Permutasi Populasi Target

Individu				
1	**2**	**3**	**4**	**5**
8	8	6	8	6
7	3	3	4	3
2	7	7	2	8
4	9	9	3	2
6	2	8	9	4
3	6	4	5	7
9	4	1	7	9
1	1	5	6	5
5	5	2	1	1

c. Mencari nilai fungsi objektif (fungsi tujuan) untuk setiap individu

Setelah urutan pengerjaan *job* dari masing – masing individu didapatkan, maka selanjutnya dicari nilai fungsi objektif (fungsi tujuan) untuk setiap individu. Adapun prosesnya adalah sebagai berikut :

- Mengeset waktu mula – mula adalah sama dengan nol.
- Untuk setiap individu, sesuai dengan urutan *job* masing – masing, dicari waktu penyelesaian untuk setiap *job (makespan)* dengan terus menambahkan waktu proses disetiap proses yang bersangkutan dengan waktu penyelesaian *predecessor* (*job* sebelumnya).

Tabel – tabel berikut ini menunjukkan proses perhitungan nilai *makespan* untuk setiap individu dalam suatu populasi.

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

Job	Rute dan waktu operasi (menit)				
	MP1	MP2	MP3	MP4	MP5
8	0	0	0	0	8
7	0	0	0	7	15
2	5	10	15	7	15
4	9	14	15	7	19
6	9	17	20	7	23
3	10	18	21	7	24
9	10	18	21	16	24
1	12	20	23	25	24
5	18	26	23	25	24

Tabel 4.6 Perhitungan Nilai *Makespan* Individu 1 Populasi Target

Job	**8**	**7**	**2**	**4**	**6**	**3**	**9**	**1**	**5**
Nilai *makespan*	8	15	15	14	23	24	16	25	26
Nilai *makespan* maksimal	26								

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

Job	Rute dan waktu operasi (menit)				
	MP1	MP2	MP3	MP4	MP5
8	0	0	0	0	8
3	1	2	3	0	9
7	1	2	3	7	16
9	1	2	3	16	16
2	6	11	16	16	16
6	6	14	19	16	22
4	10	18	19	16	26
1	12	20	22	24	26
5	18	26	22	24	26

Tabel 4.8 Perhitungan Nilai *Makespan* Individu 2 Populasi Target

Job	**8**	**3**	**7**	**9**	**2**	**6**	**4**	**1**	**5**
Nilai *makespan*	8	9	16	16	16	22	26	24	26
Nilai *makespan* maksimal	26								

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

Job	Rute dan waktu operasi (menit)				
	MP1	MP2	MP3	MP4	MP5
6	0	3	6	0	9
3	1	4	7	0	10
7	1	4	7	7	17
9	1	4	7	16	17
8	1	4	7	16	25
4	5	8	7	16	29
1	7	10	12	18	29
5	13	19	12	18	29
2	18	24	29	18	29

Tabel 4.10 Perhitungan Nilai *Makespan* Individu 3 Populasi Target

Job	**6**	**3**	**7**	**9**	**8**	**4**	**1**	**5**	**2**
Nilai *makespan*	9	10	17	16	25	29	18	19	29
Nilai *makespan* maksimal	29								

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

Job	Rute dan waktu operasi (menit)				
	MP1	MP2	MP3	MP4	MP5
8	0	0	0	0	8
4	4	8	0	0	12
2	9	14	19	0	12
3	10	15	20	0	21
9	10	15	10	9	21
5	16	22	10	9	21
7	16	22	10	16	28
6	16	25	28	16	31
1	18	27	30	32	31

Tabel 4.12 Perhitungan Nilai *Makespan* Individu 4 Populasi Target

Job	**8**	**4**	**2**	**3**	**9**	**5**	**7**	**6**	**1**
Nilai *makespan*	8	12	19	21	9	22	28	31	32
Nilai *makespan* maksimal	32								

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

Job	Rute dan waktu operasi (menit)				
	MP1	MP2	MP3	MP4	MP5
6	0	3	6	0	9
3	1	4	7	0	10
8	1	4	7	0	18
2	6	11	16	0	18
4	10	15	16	0	22
7	10	15	16	7	29
9	10	15	16	16	29
5	16	22	16	16	29
1	18	24	26	28	29

Tabel 4.14 Perhitungan Nilai *Makespan* Individu 5 Populasi Target

Job	**6**	**3**	**8**	**2**	**4**	**7**	**9**	**5**	**1**
Nilai *makespan*	9	10	18	16	22	29	16	22	28
Nilai *makespan* maksimal	29								

d. Pemilihan individu untuk menjadi vektor target

Setelah nilai *makespan* dari masing – masing individu diperoleh, selanjutnya dilakukan pemilihan individu untuk dijadikan sebagai vektor target untuk proses berikutnya. Pemilihan individu untuk menjadi vektor target didasarkan pada nilai *makespan* terkecil dari seluruh individu. Berdasarkan tabel – tabel diatas, nilai *makespan* terkecil dihasilkan oleh individu ke satu dan dua. Berdasarkan urutan proses, maka ditentukan individu ke satu sebagai vektor target dengan urutan pengerjaan *job* 8-7-2-4-6-3-9-1-5 dengan nilai *makespan* sebesar 26 menit.

e. Populasi mutan dengan input vektor target dan dua vektor acak

Proses pembentukan populasi mutan dijelaskan sebagai berikut :

- Seperti pada pembahasan sebelumnya bahwa vektor target didapat dari individu populasi target yang memiliki nilai *makespan* terkecil, yaitu individu ke satu dan dua. Dalam penelitian ini, individu ke satu ditetapkan sebagai vektor target. Semua kolom (individu) pada vektor target berisi bilangan – bilangan yang sama, yaitu individu ke satu pada populasi target.

Tabel 4.15 Vektor Target

Individu				
1	**2**	**3**	**4**	**5**
0,6680	0,6680	0,6680	0,6680	0,6680
-0,5377	-0,5377	-0,5377	-0,5377	-0,5377
0,2137	0,2137	0,2137	0,2137	0,2137
-0,0280	-0,0280	-0,0280	-0,0280	-0,0280
0,7826	0,7826	0,7826	0,7826	0,7826
0,0019	0,0019	0,0019	0,0019	0,0019
-1,0386	-1,0386	-1,0386	-1,0386	-1,0386
-1,2789	-1,2789	-1,2789	-1,2789	-1,2789
0,3789	0,3789	0,3789	0,3789	0,3789

- 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 dimana nilainya berasal dari populasi target pada kolom 3, 1, dan 2. Ketiga kolom tersebut berbeda, tidak sama.

Tabel 4.16 Vektor Acak 1 (hasil *run* program)

Individu				
1	**2**	**3**	**4**	**5**
-0,1795	-0,1795	0,2076	0,2076	0,2076
0,7873	0,7873	-0,4556	-0,4556	-0,4556
-0,8842	-0,8842	-0,6024	-0,6024	-0,6024
-0,2943	-0,2943	-0,9695	-0,9695	-0,9695
0,6263	0,6263	0,4936	0,4936	0,4936
-0,9803	-0,9803	-0,1098	-0,1098	-0,1098
-0,7222	-0,7222	0,8636	0,8636	0,8636
-0,5945	-0,5945	-0,0680	-0,0680	-0,0680
-0,6026	-0,6026	-0,1627	-0,1627	-0,1627

Tabel 4.17 Vektor Acak 2 (hasil *run* program)

Individu				
1	**2**	**3**	**4**	**5**
0,2076	-0,1106	-0,1106	-0,1106	0,6924
-0,4556	0,2309	0,2309	0,2309	0,0503
-0,6024	0,5839	0,5839	0,5839	-0,5947
-0,9695	0,8436	0,8436	0,8436	0,3443
0,4936	0,4764	0,4764	0,4764	0,6762
-0,1098	-0,6475	-0,6475	-0,6475	-0,9607
0,8636	-0,1886	-0,1886	-0,1886	0,3626
-0,0680	0,8709	0,8709	0,8709	-0,2410
-0,1627	0,8338	0,8338	0,8338	0,6636

- Setelah vektor target, vektor acak 1, dan vektor acak 2 diperoleh, selanjutnya dibentuklah populasi mutan (tabel 4.18). Populasi mutan diperoleh sebagai berikut :

  Populasi mutan = vektor target + [(vektor acak 1 – vektor acak 2) x konstanta mutasi (F)]

Persamaan diatas berlaku untuk setiap gen dari masing – masing individu.

Tabel 4.18 Populasi Mutan

Individu				
1	**2**	**3**	**4**	**5**
0,4358	0,6267	0,8589	0,8589	0,3771
0,2080	-0,2039	-0,9496	-0,9496	-0,8413
0,0446	-0,6672	-0,4981	-0,4981	0,2091
0,3771	-0,7108	-1,1159	-1,1159	-0,8163
0,8623	0,8725	0,7929	0,7929	0,6730
-0,5204	-0,1978	0,3245	0,3245	0,5125
-1,9901	-1,3588	-0,4073	-0,4073	-0,7380
-1,5948	-2,1581	-1,8423	-1,8423	-1,1751
0,1150	-0,4829	-0,2190	-0,2190	-0,1169

f. Pembentukan populasi *trial*

Setelah populasi mutan dibentuk, proses selanjutnya adalah pembentukan populasi *trial*. Setiap gen individu dari populasi *trial* berasal dari gen populasi target atau gen individu populasi mutan. Untuk memilih gen dari populasi target atau populasi mutan, digunakanlah bilangan acak antara 0 dan 1 serta operator pindah silang (CR) sebesar 0,7. Jika bilangan acak yang terpilih bernilai kurang dari 0,7 maka nilai gen untuk populasi *trial* berasal dari gen populasi mutan. Namun jika nilai bilangan acak lebih dari 0,7 maka nilai gen untuk populasi *trial* berasal dari gen populasi target. Pada tabel 4.19 dapat dilihat populasi *trial* dengan gen yang berasal dari populasi mutan diberi blok.

g. Melakukan pengurutan (permutasi) pada setiap individu dari populasi *trial*

Proses permutasi untuk setiap individu dari populasi *trial* dilakukan seperti halnya dalam proses permutasi untuk setiap individu dari populasi target sebelumnya. Hasil permutasi dapat dilihat pada tabel 4.20.

h. Menghitung nilai *makespan* dari setiap individu dari populasi *trial*

Setelah proses permutasi dari populasi *trial* dilakukan, selanjutnya dilakukan proses penghitungan nilai *makespan* untuk setiap individu.

Tabel 4.19 Populasi *Trial*

Individu				
1	**2**	**3**	**4**	**5**
0,6680	0,6680	0,6680	0,6680	0,6680
-0,5377	-0,2039	-0,5377	-0,9496	-0,8413
0,2137	-0,6672	-0,4981	-0,4981	0,2091
-0,0280	-0,0280	-1,1159	-0,0280	-0,8163
0,7826	0,8725	0,7929	0,7826	0,7826
0,0019	0,0019	0,0019	0,0019	0,0019
-1,0386	-1,0386	-1,0386	-1,0386	-1,0386
-1,2789	-1,2789	-1,2789	-1,2789	-1,2789
0,3789	0,3789	0,3789	0,3789	0,3789

Tabel 4.20 Permutasi Populasi *Trial*

Individu				
1	**2**	**3**	**4**	**5**
8	8	8	8	8
7	7	4	7	7
2	3	7	2	2
4	2	2	3	4
6	4	3	4	6
3	6	6	6	3
9	9	9	9	9
1	1	1	1	1
5	5	5	5	5

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

Job	**Rute dan waktu operasi (menit)**				
	MP1	**MP2**	**MP3**	**MP4**	**MP5**
8	0	0	0	0	8
7	0	0	0	7	15
2	5	10	15	7	15
4	9	14	15	7	19
6	9	17	20	7	23
3	10	18	21	7	24
9	10	18	21	16	24
1	12	20	23	25	24
5	18	26	23	25	24

Tabel 4.22 Perhitungan Nilai *Makespan* Individu 1 Populasi *Trial*

Job	**8**	**7**	**2**	**4**	**6**	**3**	**9**	**1**	**5**
Nilai *makespan*	8	15	15	14	23	24	16	25	26
Nilai *makespan* maksimal	26								

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

Job	**Rute dan waktu operasi (menit)**				
	MP1	**MP2**	**MP3**	**MP4**	**MP5**
8	0	0	0	0	8
7	0	0	0	7	15
3	1	2	3	7	16
2	6	11	16	7	16
4	10	15	16	7	20
6	10	18	21	7	24
9	10	18	21	16	24
1	12	20	23	25	24
5	18	26	23	25	24

Tabel 4.24 Perhitungan Nilai *Makespan* Individu 2 Populasi *Trial*

Job	**8**	**7**	**3**	**2**	**4**	**6**	**9**	**1**	**5**
Nilai *makespan*	8	15	16	16	20	24	16	25	26
Nilai *makespan* maksimal	26								

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

Job	**Rute dan waktu operasi (menit)**				
	MP1	**MP2**	**MP3**	**MP4**	**MP5**
8	0	0	0	0	8
4	4	8	0	0	12
7	4	8	0	7	19
2	9	14	19	7	19
3	10	15	20	7	21
6	10	18	23	7	26
9	10	18	23	16	26
1	12	20	25	28	26
5	18	26	25	28	26

Tabel 4.26 Perhitungan Nilai *Makespan* Individu 3 Populasi *Trial*

Job	**8**	**4**	**7**	**2**	**3**	**6**	**9**	**1**	**5**
Nilai *makespan*	8	12	19	19	21	26	16	28	26
Nilai *makespan* maksimal	28								

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

Job	**Rute dan waktu operasi (menit)**				
	MP1	**MP2**	**MP3**	**MP4**	**MP5**
8	0	0	0	0	8
7	0	0	0	7	15
2	5	10	15	7	15
3	6	11	16	7	17
4	10	15	16	7	21
6	10	18	21	7	24
9	10	18	21	16	24
1	12	20	23	25	24
5	18	26	23	25	24

Tabel 4.28 Perhitungan Nilai *Makespan* Individu 4 Populasi *Trial*

Job	**8**	**7**	**2**	**3**	**4**	**6**	**9**	**1**	**5**
Nilai *makespan*	8	15	15	17	21	24	16	25	26
Nilai *makespan* maksimal	26								

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

Job	**Rute dan waktu operasi (menit)**				
	MP1	**MP2**	**MP3**	**MP4**	**MP5**
8	0	0	0	0	8
7	0	0	0	7	15
2	5	10	15	7	15
4	9	14	15	7	19
6	9	17	20	7	23
3	10	18	21	7	24
9	10	18	21	16	24
1	12	20	23	25	24
5	18	26	23	25	24

Tabel 4.30 Perhitungan Nilai *Makespan* Individu 5 Populasi *Trial*

Job	**8**	**7**	**2**	**4**	**6**	**3**	**9**	**1**	**5**
Nilai *makespan*	8	15	15	14	23	24	16	25	26
Nilai *makespan* maksimal	26								

i. Proses penyeleksian individu

Setelah nilai *makespan* dari setiap individu pada populasi *trial* diperoleh, selanjutnya dilakukan proses seleksi dengan membandingkan nilai *makespan* dari populasi target dengan nilai *makespan* dari populasi *trial*. Perbandingan nilai *makespan* tersebut dapat dilihat pada tabel 4.31 berikut.

Tabel 4.31 Perbandingan Nilai *Makespan* Populasi Target dan Populasi *Trial*

Nilai *Makespan* Populasi Target (menit)	**Nilai *Makespan* Populasi *Trial* (menit)**	**Asal Populasi Target Iterasi selanjutnya**
26	26	Populasi Target
26	26	Populasi Target
29	28	Populasi *Trial*
32	26	Populasi *Trial*
29	26	Populasi *Trial*

Berdasarkan data tabel 4.31 diatas terlihat bahwa nilai *makespan* minimal terdapat pada individu ke satu dan dua dari populasi target serta individu ke satu, dua, empat, dan lima dari populasi *trial*. Masing – masing individu tersebut memiliku nilai *makespan* yang sama yaitu sebesar 26 menit. Akan tetapi, individu yang akan dipilih sebagai individu baru untuk proses iterasi selanjutnya hanya satu individu saja.

Oleh karena itu, berdasarkan urutan pengerjaan dari masing – masing individu, maka individu yang dipilih sebagai individu baru untuk proses iterasi selanjutnya adalah individu ke satu dari populasi target. Individu ke satu tersebut memiliki proses pengerjaan *job* sebagai berikut : 8-7-2-4-6-3-9-1-5 dengan nilai *makespan* 26 menit. Berdasarkan hasil perhitungan manual ini, bahwa nilai *makespan* perhitungan manual sama dengan hasil output dari program, maka program telah tervalidasi.

4.1.3 Perhitungan Nilai Total Waktu Proses Keseluruhan *Job (makespan)*

1) Nilai Makespan untuk Penjadwalan Produksi yang sudah ada di PT X

Untuk melakukan perbandingan antara penjadwalan yang ada di PT X dengan penjadwalan Algoritma DE, maka dilakukan perhitungan nilai total waktu proses keseluruhan *job (makespan)* untuk penjadwalan yang ada di PT X. Penjadwalan yang ada di PT X beserta waktu proses dan urutan prosesnya dapat dilihat pada tabel 4.32.

Tabel 4.32 Penjadwalan PT X

No	Grup	Process	Sequence of Process					Press Time (Minutes)				
			MP1	MP2	MP3	MP4	MP5	MP1	MP2	MP3	MP4	MP5
1	A	1	57145-BZ011	57145-BZ011	57145-BZ011	57145-BZ011	61173-BZ010	28.5	28.5	28.5	28.5	40.4
2		2	61627-BZ010	61627-BZ010	61627-BZ010	61627-BZ010	61173-BZ010	9.66	9.66	9.66	9.66	40.4
3		3	57411-BZ051	57411-BZ051	57411-BZ051	57181-BZ010	61173-BZ010	13.39	13.39	13.39	5.41	40.4
4		4	63134-BZ010	63134-BZ010	61675-BZ010	57181-BZ010	57162-BZ030	5.92	5.92	81.28	5.41	132
5		5	55711-BZ080	55711-BZ080	55711-BZ080	53322-BZ010	57181-BZ010	14	14	14	46.52	5.41
6		6	61731-BZ060	61731-BZ060				3.02	3.02	0	0	0
7	B	1	51545-BZ011	51545-BZ011	51545-BZ011	51545-BZ011	57632-BZ010	24.7	24.7	24.7	24.7	100.1
8		2	61735-BZ020	61735-BZ020	61735-BZ020	61735-BZ020	57258-BZ010	9.52	9.52	9.52	9.52	122.3
9		3	53713-BZ010	53713-BZ010	53713-BZ010	57258-BZ010	57258-BZ010	31.12	31.12	31.12	122.3	122.3
10		4	57113-BZ010	57113-BZ010	57113-BZ010	63132-BZ050	63132-BZ050	56.2	56.2	56.2	22.36	22.36
11		5				57654-BZ010	55185-BZ040	0	0	0	99.3	102.8
12		6		57113-BZ040	57113-BZ040		57113-BZ040	0	7.92	7.92	0	7.92
13		7	57213-BZ030	57213-BZ030	57213-BZ030	57213-BZ030		4.92	4.92	4.92	4.92	0
14	C	1	57145-BZ011	57145-BZ011	57145-BZ011	57145-BZ011	61173-BZ010	28.5	28.5	28.5	28.5	40.4
15		2	67147-BZ040	67147-BZ040	67147-BZ040	67147-BZ040	61173-BZ010	10.57	10.57	10.57	10.57	40.4
16		3	61615-BZ020	61615-BZ020	61615-BZ020	61615-BZ020	61173-BZ010	10.5	10.5	10.5	10.5	40.4
17		4	53735-BZ020	53735-BZ020	53735-BZ020	57181-BZ010	57162-BZ030	14.21	14.21	14.21	5.41	132
18		5	63134-BZ010	63134-BZ010	61675-BZ010	57181-BZ010	57181-BZ010	5.92	5.92	81.28	5.41	5.41
19		6	67145-BZ050	67145-BZ050	67145-BZ050	55744-BZ020	55744-BZ020	16	16	16	15.4	15.4
20	D	1	51545-BZ011	51545-BZ011	51545-BZ011	51545-BZ011	57258-BZ010	24.7	24.7	24.7	24.7	122.3
21		2	61613-BZ060	61613-BZ060	61613-BZ060	57258-BZ010	57258-BZ010	14.07	14.07	14.07	122.3	122.3
22		3	53713-BZ010	53713-BZ010	53713-BZ010	57632-BZ010	53215-BZ060	31.12	31.12	31.12	100.1	20.12
23		4	57113-BZ010	57113-BZ010	57113-BZ010	67349-BZ010	53215-BZ060	56.2	56.2	56.2	38.08	20.12
24		5	61162-BZ051	61162-BZ051		57654-BZ010	55185-BZ040	16.51	16.51	0	99.3	102.8
25		6	57645-BZ060	57645-BZ060			57645-BZ060	7.48	7.48	0	0	7.48
26	A2	1	57145-BZ011	57145-BZ011	57145-BZ011	57145-BZ011	61173-BZ010	28.5	28.5	28.5	28.5	40.4
27		2	61627-BZ010	61627-BZ010	61627-BZ010	61627-BZ010	61173-BZ010	9.66	9.66	9.66	9.66	40.4
28		3	57411-BZ051	57411-BZ051	57411-BZ051	57181-BZ010	61173-BZ010	13.39	13.39	13.39	5.41	40.4
29		4	63134-BZ010	63134-BZ010	61675-BZ010	57181-BZ010	57162-BZ030	5.92	5.92	81.28	5.41	132
30		5	53321-BZ130	53321-BZ130		53322-BZ010	57181-BZ010	25.2	25.2	0	46.52	5.41
31		6				64298-BZ010	64298-BZ010	0	0	0	8.94	8.94
32	B2	1	51545-BZ011	51545-BZ011	51545-BZ011	51545-BZ011	57632-BZ010	24.7	24.7	24.7	24.7	100.1
33		2	61735-BZ020	61735-BZ020	61735-BZ020	61735-BZ020	57258-BZ010	9.52	9.52	9.52	9.52	122.3
34		3	53713-BZ010	53713-BZ010	53713-BZ010	57258-BZ010	57258-BZ010	31.12	31.12	31.12	122.3	122.3
35		4	57113-BZ010	57113-BZ010	57113-BZ010	63132-BZ050	63132-BZ050	56.2	56.2	56.2	22.36	22.36
36		5	61412-BZ060	61412-BZ060	61412-BZ060	57654-BZ010	55185-BZ040	8.82	8.82	8.82	99.3	102.8
37		6				63134-BZ050	63134-BZ050	0	0	0	2.9	2.9
38	C2	1	57145-BZ011	57145-BZ011	57145-BZ011	57145-BZ011	61173-BZ010	28.5	28.5	28.5	28.5	40.4
39		2	67147-BZ040	67147-BZ040	67147-BZ040	67147-BZ040	61173-BZ010	10.57	10.57	10.57	10.57	40.4
40		3	61615-BZ020	61615-BZ020	61615-BZ020	61615-BZ020	61173-BZ010	10.5	10.5	10.5	10.5	40.4
41		4	53735-BZ020	53735-BZ020	53735-BZ020	57181-BZ010	57162-BZ030	14.21	14.21	14.21	5.41	132
42		5	63134-BZ010	63134-BZ010	61675-BZ010	57181-BZ010	57181-BZ010	5.92	5.92	81.28	5.41	5.41
43		6	55111-BZ280	55111-BZ280	55111-BZ280	55111-BZ280		13.09	13.09	13.09	13.09	0
44	D2	1	51545-BZ011	51545-BZ011	51545-BZ011	51545-BZ011	57258-BZ010	24.7	24.7	24.7	24.7	122.3
45		2	61613-BZ060	61613-BZ060	61613-BZ060	57258-BZ010	57258-BZ010	14.07	14.07	14.07	122.3	122.3
46		3	53713-BZ010	53713-BZ010	53713-BZ010	57632-BZ010	53215-BZ060	31.12	31.12	31.12	100.1	20.12
47		4	57113-BZ010	57113-BZ010	57113-BZ010	67349-BZ010	53215-BZ060	56.2	56.2	56.2	38.08	20.12
48		5				57654-BZ010	55185-BZ040	0	0	0	99.3	102.8
49		6	57213-BZ020	57213-BZ020	57213-BZ020		57213-BZ020	7.7	7.7	7.7	0	7.7

Berdasarkan penjadwalan PT X diatas, maka dilakukanlah perhitungan nilai *makespan* dari penjadwalan tersebut. Setelah dilakukan perhitungan dengan menggunakan software Matlab (data *script M-file* dari program perhitungan nilai *makespan* penjadwalan PT X, dapat dilihat pada lampiran 1), didapat nilai *makspan*-nya adalah 3209 menit.

2) Nilai *Makespan* untuk Penjadwalan dengan Algoritma DE

Perhitungan nilai *makespan* untuk penjadwalan dengan algoritma DE dilakukan dengan menggunakan suatu software yaitu matlab. Hal ini dilakukan agar waktu perhitungan yang dilakukan menjadi lebih cepat sekaligus memberikan hasil yang lebih akurat. Dengan menggunakan sotware matlab tersebut, output yang dapat dihasilkan oleh software tersebut antara lain ialah nilai *makespan*, urutan *job*, dan waktu perhitungannya. Untuk bahasa pemrograman penjadwalan dengan algoritma DE dapat dilihat pada lampiran 2 script M-file. Berikut hasil dari run program tersebut :

Urutan terbaik=

Columns 1 through 10

80 11 60 69 77 39 9 38 13 26

Columns 11 through 20

15 84 36 43 17 61 58 57 16 66

Columns 21 through 30

54 6 32 44 42 10 24 7 30 5

Columns 31 through 40

31 55 78 88 86 64 71 63 85 1

Columns 41 through 50

72 4 62 40 82 19 59 52 27 22

Columns 51 through 60

81 83 73 28 47 79 41 56 20 67

Columns 61 through 70

37 34 35 74 18 25 53 29 65 70

Columns 71 through 80

46 21 45 2 48 8 51 33 14 87

Columns 81 through 88

50 68 76 23 49 3 75 12

Makespan =
3.1980e+003

Waktu Komputasi :84.7765 detik

Setelah dilakukan *run program* dengan hasil seperti diatas, maka urutan proses untuk penjadwalan dengan algoritma *differential evolution* dapat dilihat pada tabel 4.33.

Tabel 4.33 Urutan *Job* untuk Penjadwalan dengan Algoritma DE

Urutan Proses	Produk	Part no. Produk	Press Time (Minutes)				
			MP1	MP2	MP3	MP4	MP5
1	80	55185-BZ040-4	0	0	0	0	102,8
2	11	67147-BZ040-2	10,57	10,57	10,57	10,57	0
3	60	53322-BZ010-1	0	0	0	0	46,52
4	69	57654-BZ010-1	0	0	0	0	99,3
5	77	55185-BZ040-1	0	0	0	0	102,8
6	39	61731-BZ060-1	3,02	3,02	0	0	0
7	9	61735-BZ020-2	9,52	9,52	9,52	9,52	0
8	38	53215-BZ060-2	0	0	0	20,12	20,12
9	13	51545-BZ011-2	24,7	24,7	24,7	24,7	0
10	26	61613-BZ060-1	14,07	14,07	14,07	0	0
11	15	51545-BZ011-4	24,7	24,7	24,7	24,7	0
12	84	57162-BZ030-4	0	0	0	132	0
13	36	53321-BZ130-1	25,2	25,2	0	0	0
14	43	57113-BZ010-1	0	0	0	56,2	56,2
15	17	57145-BZ011-2	28,5	28,5	28,5	28,5	0
16	61	53322-BZ010-2	0	0	0	0	46,52
17	58	63134-BZ010-3	0	0	0	47,36	47,36
18	57	63134-BZ010-2	0	0	0	47,36	47,36
19	16	57145-BZ011-1	28,5	28,5	28,5	28,5	0
20	66	57258-BZ010-3	0	0	0	0	122,3
21	54	61173-BZ010-3	0	0	0	40,4	40,4
22	6	61627-BZ010-1	9,66	9,66	9,66	9,66	0
23	32	53713-BZ010-2	31,12	31,12	31,12	0	0
24	44	57113-BZ010-2	0	0	0	56,2	56,2
25	42	63134-BZ050-1	0	0	0	2,9	2,9
26	10	67147-BZ040-1	10,57	10,57	10,57	10,57	0
27	24	53735-BZ020-1	14,21	14,21	14,21	0	0
28	7	61627-BZ010-2	9,66	9,66	9,66	9,66	0
29	30	57411-BZ051-2	13,39	13,39	13,39	0	0
30	5	61615-BZ020-2	10,5	10,5	10,5	10,5	0
31	31	53713-BZ010-1	31,12	31,12	31,12	0	0
32	55	61173-BZ010-4	0	0	0	40,4	40,4
33	78	55185-BZ040-2	0	0	0	0	102,8
34	88	61675-BZ010-4	0	0	0	0	81,28
35	86	61675-BZ010-2	0	0	0	0	81,28

Tabel 4.33 Urutan *Job* untuk Penjadwalan dengan Algoritma DE (lanjutan)

Urutan Proses	Produk	Part no. Produk	Press Time (Minutes)				
			MP1	MP2	MP3	MP4	MP5
36	64	57258-BZ010-1	0	0	0	0	122,3
37	71	57654-BZ010-3	0	0	0	0	99,3
38	63	67349-BZ010-2	0	0	0	0	38,08
39	85	61675-BZ010-1	0	0	0	0	81,28
40	1	55111-BZ280-1	13,09	13,09	13,09	13,09	0
41	72	57654-BZ010-4	0	0	0	0	99,3
42	4	61615-BZ020-1	10,5	10,5	10,5	10,5	0
43	62	67349-BZ010-1	0	0	0	0	38,08
44	40	63132-BZ050-1	0	0	0	22,36	22,36
45	82	57162-BZ030-2	0	0	0	132	0
46	19	57145-BZ011-4	28,5	28,5	28,5	28,5	0
47	59	63134-BZ010-4	0	0	0	47,36	47,36
48	52	61173-BZ010-1	0	0	0	40,4	40,4
49	27	61613-BZ060-2	14,07	14,07	14,07	0	0
50	22	61412-BZ060-1	8,82	8,82	8,82	0	0
51	81	57162-BZ030-1	0	0	0	132	0
52	83	57162-BZ030-3	0	0	0	132	0
53	73	57632-BZ010-1	0	0	0	0	100,1
54	28	57113-BZ040-1	0	7,28	7,28	0	7,28
55	47	57181-BZ010-1	0	0	0	43,28	43,28
56	79	55185-BZ040-3	0	0	0	0	102,8
57	41	63132-BZ050-2	0	0	0	22,36	22,36
58	56	63134-BZ010-1	0	0	0	47,36	47,36
59	20	55711-BZ080-1	14	14	14	0	0
60	67	57258-BZ010-4	0	0	0	0	122,3
61	37	53215-BZ060-1	0	0	0	20,12	20,12
62	34	53713-BZ010-4	31,12	31,12	31,12	0	0
63	35	64298-BZ010-1	0	0	0	8,94	8,94
64	74	57632-BZ010-2	0	0	0	0	100,1
65	18	57145-BZ011-3	28,5	28,5	28,5	28,5	0
66	25	53735-BZ020-2	14,21	14,21	14,21	0	0
67	53	61173-BZ010-2	0	0	0	40,4	40,4
68	29	57411-BZ051-1	13,39	13,39	13,39	0	0
69	65	57258-BZ010-2	0	0	0	0	122,3
70	70	57654-BZ010-2	0	0	0	0	99,3

Tabel 4.33 Urutan *Job* untuk Penjadwalan dengan Algoritma DE (lanjutan)

Urutan Proses	Produk	Part no. Produk	Press Time (Minutes)				
			MP1	MP2	MP3	MP4	MP5
71	46	57113-BZ010-4	0	0	0	56,2	56,2
72	21	57645-BZ060-1	7,2	7,2	0	0	7,2
73	45	57113-BZ010-3	0	0	0	56,2	56,2
74	2	57213-BZ020-1	7,7	7,7	7,7	0	7,7
75	48	57181-BZ010-2	0	0	0	43,28	43,28
76	8	61735-BZ020-1	9,52	9,52	9,52	9,52	0
77	51	55744-BZ020-1	0	0	0	15,4	15,4
78	33	53713-BZ010-3	31,12	31,12	31,12	0	0
79	14	51545-BZ011-3	24,7	24,7	24,7	24,7	0
80	87	61675-BZ010-3	0	0	0	0	81,28
81	50	57181-BZ010-4	0	0	0	43,28	43,28
82	68	57258-BZ010-5	0	0	0	0	122,3
83	76	57632-BZ010-4	0	0	0	0	100,1
84	23	67145-BZ050-1	16	16	16	0	0
85	49	57181-BZ010-3	0	0	0	43,28	43,28
86	3	57213-BZ030-1	4,92	4,92	4,92	4,92	0
87	75	57632-BZ010-3	0	0	0	0	100,1
88	12	51545-BZ011-1	24,7	24,7	24,7	24,7	0

4.2 Analisa

Setelah proses pengolahan selesai dilakukan, maka perlu untuk dilakukan suatu analisa terhadap hasil yang didapatkan. Pada penelitian ini, proses analisa akan dilakukan pada :

1) Analisa terhadap waktu komputasi (waktu *run*)

   Waktu komputasi perhitungan akan sangat bergantung pada jumlah iterasi yang ditentukan. Semakin banyak iterasi yang dipilih, maka waktu komputasi akan semakin lama. Akan tetapi, dengan semakin banyaknya jumlah iterasi, maka kemungkinan nilai yang akan dicapai pada waktu komputasi tertentu akan mendekati nilai optimalnya. Hal ini dikarenakan proses pencarian solusi dari model algoritma akan terus berlangsung hingga jumlah iterasi yang ditetapkan telah tercapai. Proses pencarian yang berulang tersebut dilakukan agar solusi yang didapatkan dapat mendekati nilai optimalnya. Sehingga dengan proses pencarian solusi yang dilakukan secara terus menerus tersebut diharapkan tidak terjadi optimal lokal, yaitu

suatu nilai dari solusi yang didapatkan hanya pada iterasi tertentu saja, sehingga hasil yang didapat menjadi kurang baik.

2) Analisa terhadap nilai *makespan*

Dari hasil pengolahan data di atas terlihat bahwa nilai *makespan* yang dihasilkan berbeda antara penjadwalan PT X dengan penjadwalan dengan algoritma DE. Hal ini dikarenakan pada penjadwalan dengan algoritma DE dilakukan dengan beberapa tahapan proses yaitu inisialisasi populasi, proses mutasi, proses pindah silang, dan proses seleksi. Pada proses inisialisasi populasi dilakukan pembentukan individu – individu dimana masing – masing individu – individu tersebut memiliki gen/kromosom yang berbeda – beda. Dalam penelitian ini, gen/kromosom tersebut dapat disebut sebagai *job*. Pada tahap ini, masing – masing individu – individu tersebut akan menghasilkan nilai *makespan* yang berbeda – beda sehingga perlu untuk dilakukan pemilihan salah satu individu terbaik yang akan dijadikan sebagai individu target. Individu yang menjadi target tersebut dipilih atau diambil dari suatu populasi yang disebut populasi target. Individu target inilah yang nantinya akan menjadi acuan atau perbandingan pada saat proses pemilihan atau seleksi terhadap individu baru hasil dari proses mutasi dan pindah silang. Kemudian setelah individu target ditentukan, maka siap untuk dilakukan proses berikutnya yaitu proses mutasi.

Proses Mutasi merupakan salah satu dari bagian proses algoritma. Pada proses ini terjadi pertukaran gen/kromosom pada masing – masing individu. Proses mutasi ini yang akan menghasilkan perbedaan urutan gen/kromosom, dalam hal ini urutan *job*. Dengan dilakukannya proses mutasi tersebut, maka individu – individu baru yang dihasilkan akan memiliki urutan pengerjaan *job* yang berbeda – beda, dan hal inilah yang juga akan berpengaruh terhadap nilai *makespan*. Dari hasil setiap iterasi akan dihasilkan urutan – urutan pengerjaan *job* yang berbeda – beda. Namun individu yang terpilih nantinya sebagai generasi baru sudah membawa urutan yang terbaik. Selain itu proses mutasi juga dipengaruhi oleh suatu operator, yaitu vektor mutasi. Vektor mutasi inilah yang nantinya akan berpengaruh pada hasil mutasi dan proses berikutnya. Penggunaan vektor mutasi ini adalah dengan

mengalikannya dengan selisih antara nilai batas atas dan batas bawah dari suatu individu. Kemudian hal lain yang juga berpengaruh pada nilai *makespan* adalah proses pindah silang. Proses pindah silang ini dilakukan agar gen/kromosom yang dimiliki oleh masing – masing individu dalam suatu populasi menjadi lebih beragam. Proses pindah silang ini berpengaruh pada nilai *makespan* dan urutan pengerjaan *job* adalah karena terjadinya proses pertukaran gen antar individu. Individu yang mengalami proses pertukaran gen berasal dari individu hasil proses mutasi dengan individu target. Pertukaran gen yang terjadi juga dipengaruhi oleh suatu faktor yaitu yang disebut faktor pindah silang. Proses pertukaran gen antar individu tersebut tidak terjadi secara berurutan, yaitu sejumlah gen yang berasal dari suatu individu. Individu hasil proses pindah silang ini disebut individu trial. Susunan gen dari individu trial ini bearsal dari dua individu, yaitu individu hasil mutasi dan individu yang menjadi target. Urutan gen yang ada pada individu trial tersebut, merupakan gabungan dari kedua individu tersebut dan urutannya tidak selalu berurutan, misalkan urutan pertama berasal dari individu mutan, kemudian urutan kedua dari individu target, urutan ketiga dan keempat dari individu mutan, dan seterusnya. Penempatan urutan gen tersebut dipengaruhi oleh faktor pindah silang yaitu CR. Keanekaragaman susunan gen tersebut yang akan berpengaruh juga pada nilai *makespan* dan urutan pengerjaan *job*. Proses selanjutnya adalah proses seleksi. Pada proses ini akan ditentukan apakah individu baru yang dihasilkan akan menjadi individu generasi baru atau tidak. Setelah proses seleksi selesai dilakukan, maka nilai *makespan* dari masing – masing individu yang terlihat. Masing – masing proses mempunyai peranan yang penting. Proses seleksi yang merupakan proses terakhir dalam rangka menghasilkan individu baru, merupakan proses yang menentukan. Pada proses inilah ditentukan apakah individu baru hasil mutasi dan pindah silang dapat diteruskan atau tidak sebagai generasi baru yang lebih baik dari generasi sebelumnya. Sehingga apabila individu baru tersebut memiliki nilai yang lebih baik (fungsi ojektifnya) dibandingkan dengan individu sebelumnya, maka individu baru tersebut akan diteruskan sebagai generasi selanjutnya, dan gen penyusunnya

merupakan urutan *job* yang lebih baik dari urutan – urutan yang ada sebelumnya.

3) Analisa terhadap urutan pengerjaan *job*

Mengenai urutan pengerjaan *job* yang dihasilkan dari proses algoritma, hal ini akan tergambarkan pada proses berjalannya masing – masing mesin. Hal ini dapat dilihat pada tabel 4.33 diatas mengenai urutan pengerjaan *job*. Pada tabel 4.33 tersebut dapat terlihat bahwa masing – masing mesin akan bekerja sesuai dengan rute proses dari masing – masing produk. Untuk proses ke satu, mesin MP5 mengerjakan produk/*job* dengan nomor 80. Pada tabel tersebut terlihat bahwa produk/*job* dengan nomor 80 tersebut membutuhkan waktu proses sebesar 102,8 menit. Hal ini menggambarkan bahwa proses produksi dimulai dengan produk/*job* yang membutuhkan waktu pengerjaan cukup lama dan hanya memiliki satu rute proses saja. Bila diperhatikan pada tabel 4.33 tersebut, maka akan terlihat bahwa urutan pengerjaan *job* yang dilakukan tidak dilakukan atau dimulai untuk produk – produk yang hanya memiliki satu kali proses saja, ataupun sebaliknya proses pengerjaan *job* tidak dimulai untuk mengerjakan produk – produk dengan jumlah proses yang harus melalui empat mesin. Akan tetapi proses pengerjaan *job* dilakukan dengan memaksimalkan kondisi mesin yang ada. Data dari masing – masing produk menggambarkan bahwa rute proses yang paling panjang adalah produk dengan rute empat kali proses, artinya suatu produk harus melalui empat tahapan proses dengan menggunakan satu mesin pada masing – masing tahapannya. Begitu juga dengan produk yang memiliki rute proses yang lainnya, yaitu tiga proses, dua proses, dan satu proses. Dari masing – masing rute proses tersebut oleh algoritma disusun sedemikian rupa agar urutan proses dari masing – masing produk dapat dilakukan dengan maksimal oleh mesin – mesin yang ada. Urutan proses yang dihasilkan dari metode algoritma terlihat bahwa pada saat satu mesin mengerjakan produk, maka mesin – mesin yang lainnya juga akan mengerjakan produk, meskipun produk – produk atau *job* yang dikerjakan berbeda. Proses pengerjaan dari masing – masing produk tetap sesuai dengan rute atau tahapan proses dari masing – masing produk dan proses

produksinya dilakukan secara berkelanjutan sesuai dengan urutan prosesnya. Artinya apabila produk tersebut membutuhkan empat tahapan proses, maka mulai dari proses pertama hingga keempat akan dikerjakan secara berkelanjutan hingga produk tersebut selesai. Namun apabila produk tersebut telah selesai dikerjakan pada tahap proses yang pertama, maka mesin dapat memproses produk berikutnya. Seperti pembahsan sebelumnya bahwa urutan pengerjaan *job* yang dihasilkan oleh algoritma *differential evolution* ini menunjukkan bahwa pengerjaan *job* didasarkan pada tahapan proses dari masing – masing produk, sehingga urutan pengerjaan *job* dilakukan dengan memperhatikan penggunaan mesin secara optimal. Hal ini sesuai dengan tujuan dari algoritma *differential evolution* yaitu optimalisasi dalam pengerjaan *job* yang dilakukan oleh beberapa mesin. Dengan optimalisasi tersebut diharapkan nilai *makespan* yang dihasilkan menjadi lebih kecil. Urutan pengerjaan *job* yang optimal tersebut terjadi karena adanya proses mutasi dan pindah silang dari algoritma. Kedua tahapan proses dari algoritma *differential evolution* tersebut bertujuan untuk menghasilkan keanekaragaman gen/kromosom sehingga individu – individu yang dihasilkan akan memiliki gen/kromosom yang lebih baik dibandingkan dengan individu sebelumnya. Gen atau urutan pengerjaan *job* tersebut sangat berpengaruh terhadap nilai *makespan*. Bila dibandingkan dengan penjadwalan yang dibuat oleh PT X, dimana nilai *makespan* yang dihasilkan adalah 3209 menit sedangkan nilai *makespan* yang dihasilkan oleh algoritma *differential evolution* adalah 3189 menit, maka dapat disimpulkan bahwa urutan pengerjaan *job* akan mempengaruhi nilai *makespan* dari urutan pengerjaan *job* tersebut.

Penjadwalan yang dibuat oleh PT X hanya didasarkan pada optimalisasi dari pemanfaatan masing – masing mesin. Maksudnya adalah masing – masing mesin akan mempunyai volume pengerjaan *job* yang hampir merata untuk seluruh mesin. Sedangkan penjadwalan dengan algoritma *differential evolution* dilakukan dengan didasarkan pada urutan proses dari masing – masing produk, optimalisasi masing – masing mesin, dan nilai *makespan* yang dihasilkan. Selain itu, dengan algoritma *differential evolution* alternatif

dari penjadwalan bisa diperoleh, yaitu melalui proses iterasi, dimana pengerjaannya dilakukan dengan bantuan suatu software, yaitu matlab, maka hasil dari masing – masing alternatif penjadwalan dapat dengan cepat diperoleh. Sedangkan PT X yang masih melakukan pembuatan penjadwalannya secara manual, akan mengalami kesulitan bila harus membuat beberapa atau banyak alternatif penjadwalan. Sehingga pembuatan suatu penjadwalan produksi dengan menggunakan suatu metode tertentu perlu dilakukan, dan juga penggunaan suatu software yang dapat membantu dalam pengerjaannya akan lebih baik diterapkan di PT X agar optimalisasi produksi khususnya dalam hal penjadwalan produksi sebagaimana yang diharapkan, dapat dicapai.

BAB 5
KESIMPULAN

Berdasarkan hasil analisa terhadap pengolahan data yang dilakukan diperoleh bahwa hasil penjadwalan urutan pengerjaan *job* dengan menggunakan Algoritma *Differential Evolution* menghasilkan penjadwalan yang lebih optimal bila dibandingkan dengan penjadwalan yang dibuat oleh PT X. Hal ini dapat dilihat pada nilai *makespan* (waktu penyelesaian keseluruhan *job*) yang lebih kecil bila dibandingkan dengan penjadwalan urutan pengerjaan *job* yang dilakukan oleh PT X, yaitu sebesar 3198 menit. Hasil penjadwalan urutan pengerjaan *job* ini merupakan sebuah solusi yang diberikan oleh algoritma *Differential Evolution* dalam permasalahan penjadwalan produksi *(Job Shop Scheduling Problem)*.

Penjadwalan produksi yang dilakukan oleh PT X, memberikan penjadwalan urutan pengerjaan *job* yang membutuhkan waktu penyelesaian keseluruhan *job* (*makespan*) sedikit lebih besar, yaitu 3209 menit. Hal ini memberikan gambaran bahwa penjadwalan urutan pengerjaan *job* yang dihasilkan oleh algoritma *Differential Evolution* lebih baik bila dibandingkan dengan penjadwalan urutan pengerjaan *job* yang dilakukan oleh PT X.

DAFTAR REFERENSI

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

Buzatu C, Prof. PhD. Eng., DrD. Eng. Băncilă, D. 6th International DAAAM Baltic Conference, INDUSTRIAL ENGINEERING, 24-26 April 2008, Tallinn, Estonia

Hui-Yuan Fan, Lampinen, Jouni., and Levy, Yeshayahou, "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

Gonçalves, José Fernando., Magalhães Mendes, Jorge José de, Maurício G.C. Resende, AT&T Labs Research Technical Report TD-5EAL6J, September 2002

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

Michael, Pinedo,Scheduling Theory,Algorithms and Systems, Prentice Hall,1993, h. 1

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

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

Nahmias, Steven., *Production and Operation Analysis*, (4th ed.), New York: Mcgraw-Hill, 2001, hal. 401

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

Storn, Rainer., & Price, Kenneth, "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

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

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

Lampiran 1 : *Script M-file* Penjadwalan PT X

```
clc;
clear;
tic;
jumlah_pesanan = 49;
jumlah_rute = 5;
data2 = [28.5   28.5    28.5    28.5    40.4
9.66    9.66    9.66    9.66    40.4
13.39   13.39   13.39   5.41    40.4
5.92    5.92    81.28   5.41    132
14  14  14  46.52   5.41
3.02    3.02    0   0   0
24.7    24.7    24.7    24.7    100.1
9.52    9.52    9.52    9.52    122.3
31.12   31.12   31.12   122.3   122.3
56.2    56.2    56.2    22.36   22.36
0   0   0   99.3    102.8
0   7.92    7.92    0   7.92
4.92    4.92    4.92    4.92    0
28.5    28.5    28.5    28.5    40.4
10.57   10.57   10.57   10.57   40.4
10.5    10.5    10.5    10.5    40.4
14.21   14.21   14.21   5.41    132
5.92    5.92    81.28   5.41    5.41
16  16  16  15.4    15.4
24.7    24.7    24.7    24.7    122.3
14.07   14.07   14.07   122.3   122.3
31.12   31.12   31.12   100.1   20.12
56.2    56.2    56.2    38.08   20.12
16.51   16.51   0   99.3    102.8
7.48    7.48    0   0   7.48
28.5    28.5    28.5    28.5    40.4
9.66    9.66    9.66    9.66    40.4
13.39   13.39   13.39   5.41    40.4
5.92    5.92    81.28   5.41    132
25.2    25.2    0   46.52   5.41
0   0   0   8.94    8.94
24.7    24.7    24.7    24.7    100.1
9.52    9.52    9.52    9.52    122.3
31.12   31.12   31.12   122.3   122.3
56.2    56.2    56.2    22.36   22.36
8.82    8.82    8.82    99.3    102.8
0   0   0   2.9 2.9
28.5    28.5    28.5    28.5    40.4
10.57   10.57   10.57   10.57   40.4
10.5    10.5    10.5    10.5    40.4
14.21   14.21   14.21   5.41    132
5.92    5.92    81.28   5.41    5.41
13.09   13.09   13.09   13.09   0
24.7    24.7    24.7    24.7    122.3
14.07   14.07   14.07   122.3   122.3
31.12   31.12   31.12   100.1   20.12
56.2    56.2    56.2    38.08   20.12
0   0   0   99.3    102.8
7.7 7.7 7.7 0   7.7];
```

Lampiran 1 : *Script M-file* Penjadwalan PT X (lanjutan)

```
N=size(data2);
wp=data2(1:N,1:5);
w = zeros(jumlah_pesanan, 5);
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 == 5
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 == 1
w(a,b)=w(a-1,b)+wp(a,b);
end
if a > 1 & b >= 2 & b <= 3
w(a,b)=max(w(a,b-1),w(a-1,b))+ wp(a,b);
end
if a >= 2 & a <= 3 & b == 4
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a >= 4 & a <= 6 & b == 4
w(a,b) = w(a-1,b) + wp(a,b);
end
if a >= 7 & a<= 8 & b == 4
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a == 45 & b == 4
w(a,b) = w(a-1,b+1) + wp(a,b+1) + wp(a,b);
end
if a >= 10 & a <= 20 & b == 4
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a >= 22 & a <= 44 & b == 4
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a >= 46 & a <= 49 & b == 4
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a >=2 & a <= 4 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a == 5 & b == 5
w(a,b) = max(w(a-1,b),w(a-1,b-1)) + wp(a,b);
end
if a == 30 & b == 5
w(a,b) = max(w(a-1,b),w(a-1,b-1)) + wp(a,b);
end
if a >=6 & a <= 9 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a == 10 & b == 5
```

Lampiran 1 : *Script M-file* Penjadwalan PT X (lanjutan)

```
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a == 18 & b == 5
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a == 19 & b == 5
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a == 31 & b == 5
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a == 35 & b == 5
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a == 37 & b == 5
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a == 42 & b == 5
w(a,b) = max(w(a,b-1),w(a-1,b)) + wp(a,b);
end
if a == 12 & b == 5
w(a,b) = max(w(a-1,b),w(a,b-2)) + wp(a,b);
end
if a == 49 & b == 5
w(a,b) = max(w(a-1,b),w(a,b-2)) + wp(a,b);
end
if a == 11 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a == 36 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a >= 13 & a <= 17 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a >= 20 & a <= 29 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a >= 32 & a <= 34 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a >= 38 & a <= 41 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a >= 43 & a <= 48 & b == 5
w(a,b) = w(a-1,b) + wp(a,b);
end
if a == 9 & b == 4
w(a,b) = wp(a,b) + wp(1,5) + wp(2,5) + wp(3,5) + wp(4,5) + wp(5,5)
+ wp(6,5) + wp (7,5) + wp(8,5) + wp(9,5);
end
if a == 21 & b == 4
w(a,b) = w(a-1,b+1) + wp(a,b+1) + wp(a,b);
end
```

Lampiran 1 : *Script M-file* Penjadwalan PT X (lanjutan)

```
end
end
waktu_total=max(w(49,:));
%---Solusi Akhir-----------------------------------------------------
disp('Makespan =');
disp(waktu_total);
```

Lampiran 2 : *Script M-file* Penjadwalan Algoritma DE

```
clc;
clear;
tic;
jumlah_pesanan = 88;
jumlah_rute = 5;
jumlah_dimensi = jumlah_pesanan;
ukuran_populasi = 50;
jumlah_iterasi = 50;
F = 0.6; %operator mutasi
CR = 0.7; %operator pindah silang
batas_bawah = -1;
batas_atas = 1;
%---Data waktu proses-----------------
%Kolom 1-5 adalah waktu proses tiap rute
data2 = [13.09  13.09   13.09   13.09   0
7.7 7.7 7.7 0   7.7
4.92    4.92    4.92    4.92    0
10.5    10.5    10.5    10.5    0
10.5    10.5    10.5    10.5    0
9.66    9.66    9.66    9.66    0
9.66    9.66    9.66    9.66    0
9.52    9.52    9.52    9.52    0
9.52    9.52    9.52    9.52    0
10.57   10.57   10.57   10.57   0
10.57   10.57   10.57   10.57   0
24.7    24.7    24.7    24.7    0
24.7    24.7    24.7    24.7    0
24.7    24.7    24.7    24.7    0
24.7    24.7    24.7    24.7    0
28.5    28.5    28.5    28.5    0
28.5    28.5    28.5    28.5    0
28.5    28.5    28.5    28.5    0
28.5    28.5    28.5    28.5    0
14  14  14  0   0
7.2 7.2 0   0   7.2
8.82    8.82    8.82    0   0
16  16  16  0   0
14.21   14.21   14.21   0   0
14.21   14.21   14.21   0   0
14.07   14.07   14.07   0   0
14.07   14.07   14.07   0   0
0   7.28    7.28    0   7.28
13.39   13.39   13.39   0   0
13.39   13.39   13.39   0   0
31.12   31.12   31.12   0   0
31.12   31.12   31.12   0   0
31.12   31.12   31.12   0   0
31.12   31.12   31.12   0   0
0   0   0   8.94    8.94
25.2    25.2    0   0   0
0   0   0   20.12   20.12
0   0   0   20.12   20.12
3.02    3.02    0   0   0
0   0   0   22.36   22.36
```

Lampiran 2 : *Script M-file* Penjadwalan Algoritma DE (lanjutan)

```
0   0   0   22.36   22.36
0   0   0   2.9 2.9
0   0   0   56.2    56.2
0   0   0   56.2    56.2
0   0   0   56.2    56.2
0   0   0   56.2    56.2
0   0   0   43.28   43.28
0   0   0   43.28   43.28
0   0   0   43.28   43.28
0   0   0   43.28   43.28
0   0   0   15.4    15.4
0   0   0   40.4    40.4
0   0   0   40.4    40.4
0   0   0   40.4    40.4
0   0   0   40.4    40.4
0   0   0   47.36   47.36
0   0   0   47.36   47.36
0   0   0   47.36   47.36
0   0   0   47.36   47.36
0   0   0   0   46.52
0   0   0   0   46.52
0   0   0   0   38.08
0   0   0   0   38.08
0   0   0   0   122.3
0   0   0   0   122.3
0   0   0   0   122.3
0   0   0   0   122.3
0   0   0   0   122.3
0   0   0   0   99.3
0   0   0   0   99.3
0   0   0   0   99.3
0   0   0   0   99.3
0   0   0   0   100.1
0   0   0   0   100.1
0   0   0   0   100.1
0   0   0   0   100.1
0   0   0   0   102.8
0   0   0   0   102.8
0   0   0   0   102.8
0   0   0   0   102.8
0   0   0   132 0
0   0   0   132 0
0   0   0   132 0
0   0   0   132 0
0   0   0   0   81.28
0   0   0   0   81.28
0   0   0   0   81.28
0   0   0   0   81.28];
N=size(data2);
wp=data2(1:N,1:5);
%---Membentuk populasi target----------------------------
populasi_target = batas_bawah + (batas_atas - batas_bawah) *
rand(jumlah_pesanan, ukuran_populasi);
```

Lampiran 2 : *Script M-file* Penjadwalan Algoritma DE (lanjutan)

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

Lampiran 2 : *Script M-file* Penjadwalan Algoritma DE (lanjutan)

```
if b > 1 & c>=2 & c<=4 & wp(urutan_awal (b, a),c)==0
w(b,c)=w(b-1,c);
end
if b > 1 & c>=2 & c<=4 & wp(urutan_awal (b, a),c)>0 &
wp(urutan_awal(b,a),c-1)==0
w(b,c)=w(b-1,c);
end
if b > 1 & c>=2 & c<=4 & wp(urutan_awal (b, a),c)>0 &
wp(urutan_awal(b,a),c-1)>0
w(b,c)=max(w(b-1,c),w(b,c-1))+wp(urutan_awal (b, a),c);
end
if b > 1 & c == 5 & wp(urutan_awal (b, a),c)==0
w(b,c)=w(b-1,c);
end
if b > 1 & c == 5 & wp(urutan_awal (b, a),c)>0 &
wp(urutan_awal(b,a),c-1)==0 & wp(urutan_awal(b,a),c-2)==0 &
wp(urutan_awal(b,a),c-3)==0 & wp(urutan_awal(b,a),c-4)==0
w(b,c)=w(b-1,c)+wp(urutan_awal (b, a),c);
end
if b > 1 & c == 5 & wp(urutan_awal (b, a),c)>0 &
wp(urutan_awal(b,a),c-1)>0
w(b,c)=max(w(b-1,c),w(b,c-1))+wp(urutan_awal (b, a),c);
end
if b > 1 & c == 5 & wp(urutan_awal (b, a),c)>0 &
wp(urutan_awal(b,a),c-1)==0 & wp(urutan_awal(b,a),c-2)>0
w(b,c)=max(w(b-1,c),w(b,c-2))+wp(urutan_awal (b, a),c);
end
if b > 1 & c == 5 & wp(urutan_awal (b, a),c)>0 &
wp(urutan_awal(b,a),c-1)==0 & wp(urutan_awal(b,a),c-2)==0 &
wp(urutan_awal(b,a),c-3)>0
w(b,c)=max(w(b-1,c),w(b,c-3))+wp(urutan_awal (b, a),c);
end
end
end
nilai_makespan_awal(a)=max(w(:,5));
end
[nilai_makespan_min_awal, index_vektor_target] =
min(nilai_makespan_awal);
makespan_awal = nilai_makespan_awal(index_vektor_target);
urutan=urutan_awal(:,index_vektor_target);
%---Hasil Awal Sebelum Iterasi---------------------------------
disp('Urutan =');
disp(urutan');
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
```

Lampiran 2 : *Script M-file* Penjadwalan Algoritma DE (lanjutan)

```
%---Mencari vektor target----------------------------------------
[nilai_makespan_min_awal, index_vektor_target] =
min(nilai_makespan_awal);
vektor_target = populasi_target(:,index_vektor_target);
for a = 2 : ukuran_populasi
vektor_target(:, a) = vektor_target(: , a-1);
end
%---Mencari Populasi Mutan----------------------------------------
%---Mencari 2 vektor acak---
for a = 1 : ukuran_populasi
index_vektor_acak1 = 1;
index_vektor_acak2 = 1;
while (index_vektor_acak1 == index_vektor_acak2) |
(index_vektor_acak1 == index_vektor_target) | (index_vektor_acak2
== index_vektor_target)
index_vektor_acak1 = randint(1,1,ukuran_populasi) + 1;
index_vektor_acak2 = randint(1,1,ukuran_populasi) + 1;
end
vektor_acak1(:,a) = populasi_target(:,index_vektor_acak1);
vektor_acak2(:,a) = populasi_target(:,index_vektor_acak2);
end
%---Membentuk populasi mutan---
populasi_mutan = (vektor_acak1 - vektor_acak2) * F +
vektor_target;
%---Membentuk populasi trial--------------------------------------
for a = 1 : jumlah_pesanan
for b = 1 : ukuran_populasi
if (rand() <= CR)
populasi_trial(a,b) = populasi_mutan(a,b);
else
populasi_trial(a,b) = populasi_target(a,b);
end
end
end
%---Mencari urutan pengerjaan populasi trial----------------------
for a = 1 : ukuran_populasi;
[hasil, urutan_trial(:,a)] = sort(populasi_trial(:,a));
end
%---Menghitung total makespan populasi trial-------------------
nilai_makespan_anak = [];
for a = 1 : ukuran_populasi
w = zeros(jumlah_pesanan, 5);
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 & wp(urutan_trial(b, a),c)==0
w(b,c)=wp(urutan_trial(b, a),c);
end
if b == 1 & c>=2 & c<=4 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0
w(b,c)=wp(urutan_trial(b, a),c);
end
```

Lampiran 2 : *Script M-file* Penjadwalan Algoritma DE (lanjutan)

```
if b == 1 & c>=2 & c<=4 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)>0
w(b,c)=w(b,c-1)+wp(urutan_trial(b, a),c);
end
if b == 1 & c == 5 & wp(urutan_trial(b, a),c)==0
w(b,c)=wp(urutan_trial(b, a),c);
end
if b == 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0 & wp(urutan_trial(b,a),c-2)==0 &
wp(urutan_trial(b,a),c-3)==0 & wp(urutan_trial(b,a),c-4)==0
w(b,c)=wp(urutan_trial (b, a),c);
end
if b == 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)>0
w(b,c)=w(b,c-1)+wp(urutan_trial(b, a),c);
end
if b == 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0 & wp(urutan_trial(b,a),c-2)>0
w(b,c)=w(b,c-2)+wp(urutan_awal (b, a),c);
end
if b == 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0 & wp(urutan_trial(b,a),c-2)==0 &
wp(urutan_trial(b,a),c-3)>0
w(b,c)=w(b,c-3)+wp(urutan_trial(b, a),c);
end
if b == 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0 & wp(urutan_trial(b,a),c-2)==0 &
wp(urutan_trial(b,a),c-3)==0 & wp(urutan_trial(b,a),c-4)>0
w(b,c)=w(b,c-4)+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 & wp(urutan_trial(b, a),c)==0
w(b,c)=w(b-1,c);
end
if b > 1 & c>=2 & c<=4 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0
w(b,c)=w(b-1,c)+wp(urutan_trial(b, a),c);
end
if b > 1 & c>=2 & c<=4 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)>0
w(b,c)=max(w(b-1,c),w(b,c-1))+wp(urutan_trial(b, a),c);
end
if b > 1 & c == 5 & wp(urutan_trial(b, a),c)==0
w(b,c)=w(b-1,c);
end
if b > 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0 & wp(urutan_trial(b,a),c-2)==0 &
wp(urutan_trial(b,a),c-3)==0 & wp(urutan_trial(b,a),c-4)==0
w(b,c)=w(b-1,c)+wp(urutan_trial(b, a),c);
end
```

Lampiran 2 : *Script M-file* Penjadwalan Algoritma DE (lanjutan)

```
if b > 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)>0
w(b,c)=max(w(b-1,c),w(b,c-1))+wp(urutan_trial(b, a),c);
end
if b > 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0 & wp(urutan_trial(b,a),c-2)>0
w(b,c)=max(w(b-1,c),w(b,c-2))+wp(urutan_trial(b, a),c);
end
if b > 1 & c == 5 & wp(urutan_trial(b, a),c)>0 &
wp(urutan_trial(b,a),c-1)==0 & wp(urutan_trial(b,a),c-2)==0 &
wp(urutan_trial(b,a),c-3)>0
w(b,c)=max(w(b-1,c),w(b,c-3))+wp(urutan_trial(b, a),c);
end
end
end
nilai_makespan_anak(a)=max(w(:, 5));
end
%---Memilih antara populasi target awal atau trial----------------
---
for a = 1 : ukuran_populasi
if min(nilai_makespan_anak(a)) < min(nilai_makespan_awal(a))
populasi_target(:, a) = populasi_trial(:,a);
urutan_awal(:,a) = urutan_trial(:,a);
nilai_makespan_akhir(a)= nilai_makespan_anak(a);
else
nilai_makespan_akhir(a)= nilai_makespan_awal(a);
end
end
end
[nilai_makespan_min_akhir,index_vektor_target_akhir] =
min(nilai_makespan_akhir);
makespan_akhir = nilai_makespan_akhir(index_vektor_target_akhir);
urutan_akhir=urutan_awal(:,index_vektor_target_akhir);
%---Solusi Akhir--------------------------------------------------
----
disp('Urutan terbaik=');
disp(urutan_akhir');
disp('Makespan =');
disp(makespan_akhir);
disp(strcat('Waktu Komputasi :',' ', num2str(toc),' detik'));
```