APPLYING DIFFERENTIAL EVOLUTION ALGORITHM FOR MINIMIZING MAKESPAN IN JOB SHOP SCHEDULING PROBLEM

By
Yandhika
ID No. 004201300045

A Thesis presented to the Faculty of Engineering President University in partial fulfillment of the requirements of Bachelor Degree in Engineering Major in Industrial Engineering

2017

THESIS ADVISOR
RECOMMENDATION LETTER

This thesis entitled **“Applying Differential Evolution Algorithm for Minimizing Makespan in Job Shop Scheduling Problem”** prepared and submitted by **Yandhika Heryanto** in partial fulfillment of the requirements for the degree of Bachelor Degree in the Faculty of Engineering has been reviewed and found to have satisfied the requirements for a thesis fit to be examined. I therefore recommend this thesis for Oral Defense.

Cikarang, Indonesia, January 31th, 2017

<u>Anastasia Lidya Maukar, ST.,MSc., M.MT.</u>

DECLARATION OF ORIGINALITY

I declare that this thesis, entitled **"Applying Differential Evolution Algorithm for Minimizing Makespan in Job Shop Scheduling Problem"** is, to the best of my knowledge and belief, an original piece of work that has not been submitted, either in whole or in part, to another university to obtain a degree.

Cikarang, Indonesia, January 31th, 2017

Yandhika Heryanto

APPLYING DIFFERENTIAL EVOLUTION ALGORITHM FOR MINIMIZING MAKESPAN IN JOB SHOP SCHEDULING PROBLEM

By

Yandhika Heryanto

ID No. 004201300045

Approved by

Anastasia L. Maukar,S.T.,M.Sc.,M.MT.	**Arthur P. Silitonga, S.T.,M.Sc.**
Thesis Advisor 1	**Thesis Advisor 2**

Andira Taslim, S.T., M.T.

Program Head of Industrial Engineering

ABSTRACT

Job shop scheduling problem can become complex when there are a set of different machines that perform several operations in job shop production system. Improper or wrong scheduling can affect the schedule because the job cannot be completed on time. Differential Evolution Algorithm can arrange the scheduling to solve job shop scheduling problem because this algorithm can produce the best solution easily with minimum time. Design of Experiment method will help to find the best parameters. The best combination parameter for Differential Evolution model to solve job shop scheduling problem are mutation factor 0.7, cross over rate 0.5, number of population 10 individuals and run with 15 iterations. Differential Evolution Algorithm is done through several steps, including initialization, mutation, crossover and selection process. By using combination of parameters that is calculated from Design of Experiment method and Differential Evolution Algorithm, it requires 34,200 seconds or 570 minutes to finish 6 jobs with 5 machines. Compared with Genetic Algorithm from previous research, Differential Evolution can reduce makespan around 5% from Genetic Algorithm schedule and reduce around 10% from current schedule.

Keywords: *Job Shop Scheduling, Differential Evolution Algorithm, Design of Experiment, Best Parameter, Genetic Algorithm, Meta-Heuristic Method*

ACKNOWLEDGEMENT

This thesis is hardly to be done without big support. Therefore, I would like to express my gratitude to:

1. Jesus Christ, the source of everything. Thank you for unconditional love that You have given to me.
2. My Thesis Advisor, Anastasia L. Maukar and Arthur Silitonga. Thank you for all advices and support given to me during my thesis period and finally done this Thesis.
3. My Father and Mother (OK Heryanto and Melia Jayalaksana), my brother (Christian Heryanto) and my big family. Thank you for believing and loving me. I love you all.
4. My special one, Rachel Melliana Prasetyorini who always gives me support and become my inspiration.
5. Head of Industrial Engineering Study Program, Ir. Andira T. who is always giving me a support, a motivation and also chance to finish my thesis.
6. My Industrial Engineering lecturers during my study from semester I until semester X. (Mam Andira, Prof. Budi, Mam Anas, Mr. Burhan, Mr. Kinley, Mr. Bagus, Mr. Herwan, Mam Lidya and Mam Ineu. I learned many things from all of you. Thank you for your lessons and knowledge.
7. My best classmate, Industrial Engineering 2013, WinJe Group, Erwin, Yossa, James, Feli, Jeje, Novi, Dede, Desire, Jeff, Natali and Nia who are always helping me in finishing my study in President University.
8. My organization family, PUCatSo, thank you for moment that I got in President University.
9. My others family, De Rombongan, Gouw, Billy, Kevin, Ian, Rendy, Ben, Fahri, Indra, Ahe, Bobby, Cyrus, Eric, Varianto, William, Varianto, Angga, Rana, and Steven who are coloring my University life.
10. My others family, House Van G, Hendry, Reinhart, Andres, Enrico, Mote, Calvin, Heri, Adrian, Ryan, Fabi who gave me inspirations.
11. Others party that cannot be mentioned one by one. Thank you for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF TERMINOLOGIES

Algorithm	:	A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
Crossover	:	Process that is purposed in adding the variances of the gen in population that will enter the next generation by crossing gen that is owned by population of target vector and mutant vector.
Gantt Chart	:	A type of bar chart that illustrates the project schedule
Generation	:	An iteration step of algorithm, a complete cycle of creating and evaluating individual
Heuristic	:	Any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals.
Initialization	:	Collection of initial solution that can be obtained from heuristic method or random method.
Iteration	:	Repetition of a mathematical or computational procedure
Job Shop Scheduling	:	An optimization problem in computer science and operations research in which ideal jobs are assigned to resources at particular times
Makespan	:	The total length of the schedule

Matlab	:	Matrix Laboratory, Multi-paradigm numerical computing environment and fourth-generation programming language.
Meta Heuristic	:	A higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem
Mutant Population	:	Population that is produced after mutation process
Mutation	:	An operation in generating mutation vector that is obtained from multiplying the differences of two vectors in present generation that is chosen random with mutation control parameter then added by the third vector that is randomly chosen.
Parameter	:	A numerical or other measurable factor forming one of a set that defines a system or sets the conditions of its operation.
Population	:	A particular section, group of solution
Random Vector	:	A multidimensional generalization of the concept of random variable.
Selection	:	Process of comparing the objective value of target vector and trial vector
Smallest Position Value	:	Method to state the job with the smallest position value is scheduled first

Trial Population	:	Population that is produced after crossover process
Vector	:	Media or Individual
Vector Target	:	Vector or media that have potential become the optimum solution

CHAPTER I

INTRODUCTION

1.1.Problem Background

Production scheduling in industry has important role in improving the productivity. Scheduling is the activity to organize and decide the resources that will be used to fulfill the desired output at the right time with resources and activity constraints (Morton and Pentico, 2013). Allocation of machine resources in production process usually causes the bottleneck that cannot be solved optimally.

In maintaining inventory level and increasing utilization of the resource, production scheduling can become one of the solutions. The problem can become complex when there are a set of different machines that perform operations in job shop production system. A job shop production system has a specific process sequence for each job. This means the production flow of each job is different with another. Thus, job shop scheduling problem is one of the most difficult combinatorial optimization problem, which is used in complex equipment manufacturing system to validate the performance of heuristic algorithms (Surekha and Sumathi, 2010). There are many possibilities in job shop scheduling.

By using algorithm, the solution of job shop scheduling can be obtained the optimum solution. Not only job shop scheduling problem, but there are also other problems that can be solved by using algorithm such as facility design layout, routing and transportation problem. A new heuristic algorithm, known as Differential Evolution Algorithm, is appeared and developed to overcome weaknesses of previous algorithm. Among the Elementary Algorithms, Differential Evolution Algorithm is the newest evolutionary optimization technique. Rainer Storn and Kenneth Price (1997) said Differential Evolution Algorithm was designed to meet users demand that a practical minimization technique should fulfill some requirements, such as: ability to handle non-differentiable, nonlinear and multimodal cost functions, parallelizability to cope with computation intensive cost functions, ease of use, for example, few control variables to steer the minimization and these variables should also be robust and easy to choose. The last requirement is good convergence properties, for example, consistent convergence to the global minimum in consecutive independent trials.

Excess Differential Evolution algorithm compared to previous evolutionary algorithm method is the evolution experienced by each individual in a population where differentiation and crossovers occur sequentially in each individual randomly selected from the population at any time. The results of these variations, known as a child or individual trial that will replace parents in the population if the resulting fitness is better or equal to that produced by parents. In the research by Jakob Vesterstrøm and Rene Thomsen (2004) about comparative study of Differential Evolution, Particle Swarm Optimization and Evolutionary Algorithms, it stated that DE generally outperforms the other algorithms. It is simple, robust, converges fast, and finds the optimum in almost every run. In addition, it has few parameters to set, and the same settings can be used for many different problems. The DE has shown its worth on real-world problems.

In previous study of Differential Evolution algorithm for job shop scheduling problem (Bhaskara et al, 2015), it stated that Differential Evolution (DE) algorithm is the effective tool to solve the flexible job shop scheduling problem because Differential Evolution employs simple mutation and cross-over to generate new candidate solutions and applies one to one competition scheme to parsimoniously determine whether the new candidate or its present will survive in the next generation.

For obtaining the optimum solution, it needs some combination method. Function of Design of Experiment method is used to determine the optimum parameters that will be used for Differential Evolution Algorithm. This research will discuss application of Differential Evolution Algorithm for job shop scheduling in order to minimize makespan. The parameters will become the factor of the experiment. There are four factors which are permutation factor (F), crossovers factor (CR), the size of the population (NP) and total iteration. Each factor has three levels which are low, medium and high. Because the factors that are used more than two factors, the DOE type used is a full factorial design.

1.2.Problem Statement

From the problem background, there are several questions that need to be answered, which are:

- How is Differential Evolution algorithm applied to generate job order scheduling for minimizing total makespan?
- What is the best combination of parameters in Differential Evolution algorithm?

1.3.Research Objectives

There are several objectives need to be achieved in this research, which are:

- To apply Differential Evolution algorithm for job shop scheduling in minimizing makespan
- To determine the best combination of parameters of Differential Evolution job shop scheduling for minimizing makespan

1.4.Scopes

Due to limited time and resources in doing this research, there will be some scope in the research, which are:

- The research is focused on comparing the makespan of the optimum scheduling using Differential Evolution (DE) Algorithm with optimum scheduling using Genetic Algorithm (GA).
- Only selected job and machine that is used as experiment.
- Research used hypothetical data and secondary data.

1.5.Assumptions

Some assumptions must be made in order to cover the project, which are:

- Scheduling is categorized static and non-preemptive (it means all orders are received and scheduling is conducted in the beginning of the period).
- Setup time and transfer time from one machine to another are neglected.
- In one time, one machine can only process one job.
- The material is always available.
- There is no circular order in job shop model.
- No machine breakdown during the production process.

1.6.Research Outline

Chapter I	**Introduction** This chapter consists of the background of the research, problem identification, objective, scope and assumption of the study.
Chapter II	**Literature Study** This chapter delivers the previous study about job shop scheduling, differential evolution (DE) algorithm and other tools which support this research.
Chapter III	**Research Methodology** This chapter contains a detailed process flow and detail explanation of each phase procedure step by step used to conduct this research start from initial observation until the conclusion.
Chapter IV	**Data Analysis** The result of data analysis is a new production scheduling from differential evolution (DE) algorithm method. The numerical computation software will be used to compute the problem into programming model in order to minimize makespan and find the optimization.
Chapter V	**Conclusion and Recommendation** This chapter will give the conclusion result of this final project, and also recommendation for future research.

CHAPTER II

LITERATURE STUDY

2.1. Production Scheduling

Scheduling is the part of the role of production planning department. It has meaning as the timing of all operations completed on time. There are some definitions of production scheduling. Scheduling is known as the real task of stating and completion dates to operations to show when the order should be completed on time. Production scheduling has other definition which is keeping decision making process in which people make plans, share information and react to unexpected events (Hermann, 2006).

According to Pinedo and Chao (1999), scheduling is the decision-making process that holdsan important role in manufacturing and production systems.The purpose of scheduling is to improve effectivenessand efficient of resources usage, reducethe accumulation of work in process inproduction line, reducing delays and can help in making decisions aboutcapacity planning plant.

2.1.1. Classification of Production Scheduling

Production scheduling based on Pinedo and Chao (1999) is classified into some criteria, which are:

- Based on machine that is used in the process:
    - Single machine shop
    - Multiple machine (m machine)
- Based on the job arrival
    - Static scheduling
      Jobs arrived together and ready to be done by available machine
    - Dynamic scheduling
      Jobs arrived on different time. Approaching that is usually used in this scheduling is different dispatching rule for each work station.

- Based on the area scheduling
    - Flow shop

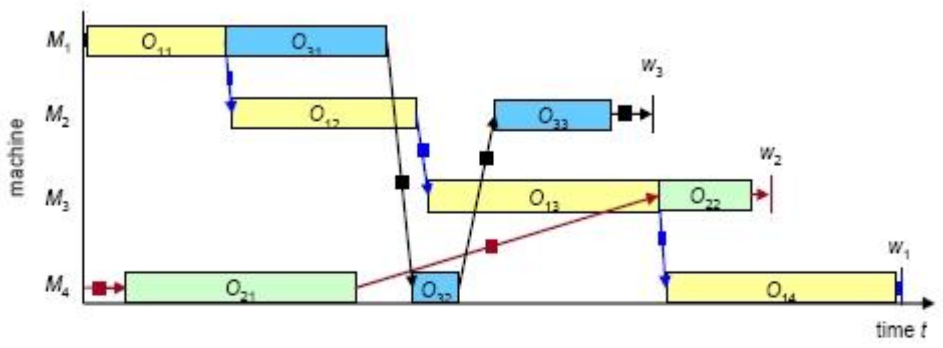
      Each job has same routing. The characteristic of the flow is discrete, continue and semi-continue.
    - Job shop

      Each job has different routing based on consumer demand (complex routing). Because the flow is complex, it makes the scheduling also complex. The characteristic of the flow is discrete and part is not multifunction (part that possible become WIP in one job that cannot be used for another job).
    - Assembly Line

      It is similar with the flow shop but the process only consists of assembly area with the high volume.

2.1.2. Gantt Chart

In 19th century, a man called Henry Gantt tried to increase the productivity by using proper scheduling. The tool that usually Gantt used was a representation of a schedule and nowadays known as Gantt chart. The objective of the chart is to show graphically the condition of each resource at each time. The components of the Gantt chart are time (represented on x axis) and bar of each machine (represented on y axis). In the y-axis, it can be constructed placing jobs, beside of machines. Gantt chart is represented the start and completion of jobs or machines (Sipper and Bulfin, 1997).



Source: Gen and Lin, 2012

Figure 2.1 Job Shop Gantt Chart

As Wilson (2003) statement, Gantt chart can give the significant impact for the successful project management. Gantt chart tool can help the managers in making decision in planning the production scheduling. This tool becomes the standard tool for industrial engineers in making a decision. As the manager or supervisor, Gantt chart can be a principal of the production whether it is on time, behind the schedule or ahead of schedule.

2.1.3. Objectives and Performance Measurement of Scheduling

In measuring the performance of production scheduling, there are some objectives that can evaluate the performance, which are:

- Minimizing flow time and makespan
- Maximizing the utilization (minimizing idle time of machine and operator in production)
- Minimizing inventory and work in process
- Minimizing lateness (earliness and tardiness)
- Minimizing the number of tardy task
- Minimizing total penalty cost because of lateness

2.2. Job Shop Scheduling

The job shop scheduling is one of the most famous production systems and generally used in factory, which is difficult to solve with NP (non deterministic-polynomial) hard nature. Each job in job shop may have a unique routing and the operation routing of each job is different. Characteristics of job shop scheduling are:

- There are m machine and n job
- Each job consists of one routing that is different one and another
- Each operation of a job is proceed by only one machine
- There is no preemption (delay one job by another job)
- Job shop scheduling is one complex problem and called as NP-hard

Job shop is classified as NP – hard because job shops are difficult to schedule. There are $(n!)^m$ possible schedules for an n-job, m-machine job shop (Sipper and Bulfin, 1997). Job shop scheduling problem (JSSP) is divided into four job shop scheduling (Artigues *et al,* 2005):

1. Feasible Schedule

   By using feasible scheduling, the number of solution will grow exponentially. This scheduling type has a lot of solutions. As a result, it will be impossible to solve big case problem optimally.
2. Semi – Active Schedule

   Semi – Active Scheduling is placing job by prioritize level or a feasible schedule without local left shift that leads to another feasible schedule. Each operation has at most one tight constraint (job or resource precedence). The concept of semi-active schedule is choosing available operation as soon as possible on the machine.
3. Active Schedule

   Active scheduling is the improvement of semi active schedule. The different is in active scheduling there is local left shifting and there is no global left shift leads to feasible schedule. Using active scheduling can reduce idle time of the machine.
4. Non-delay Schedule

   By using non-delay schedule, there is no machine idle or performing a setup at any time. Non-delay schedule can be an active schedule without idle time.

2.3. Method in Solving Production Scheduling

Production scheduling problem can be solved by heuristic method that can be classified into:

- Classic - Heuristic

  Algorithm of classic heuristic will arrange one by one solution from scheduling problem. From zero value, algorithm will choose the job or machine that should be completed first. The popular algorithm of classic heuristic is priority dispatch rule. This algorithm will arrange job based on stated priority. There are some basic priority dispatch rules which are (Sipper and Bulfin, 1997):

- First Come First Serve (FCFS)
  This rule states that the job that comes first will be completed fist.
- Earliest Due Date First (EDD)
  EDD rule focus on the due date time of the job. Job that has near due date, it will be completed first rather than job that has high due date. This rule usually used for minimizing the maximum lateness.
- Minimum Slack First (MS)
  Based on this rule, job will be arranged depends on slack time. The smallest slack time will be completed first. Purpose of this rule is used same as EDD which as minimizing lateness and tardiness.
- Shortest Processing Time First (SPT)
  For SPT rule, it based on processing time. Job that has small processing time should be completed first.

- Meta-Heuristic (Modern)
  This algorithm solves the scheduling problem by conducting improvement from the initial solution. Initial solution can be obtained randomly and can be obtained by using specific heuristic method. Meta-heuristic algorithms that can be used to solve job shop scheduling problem are (Malikarjuna et al., 2014):
  - Simulated Annealing (SA)
    This algorithm usually used for solving problem that is the changing of the condition from one condition to another needs wide space.
  - Tabu Search (TS)
    TS algorithm is an optimization method that is used short-term memory to keep the searching process is not stuck on local optima.
  - Genetic Algorithm (GA)
    GA algorithm has same concept like natural process which is natural selection by Darwin. Each individual is representing the

unique solution. Genetic algorithm works to find the best individual structure in the population.

- Differential Evolution (DE) Algorithm

  DE algorithm is the latest evolution of evolutionary algorithm. This algorithm is the development of Genetic Algorithm.

2.4. Differential Evolution Algorithm

The idea of Differential Evolution (DE) Algorithm is development from the previous algorithm which is Genetic Algorithm (GA). DE algorithm is an efficient global optimization algorithm that is principled to evolution concept (Price and Storn, 1997). By evolving the population of predicted solutions using some alterations and choosing operators, DE algorithm provides optimization problems. The algorithm is inspired from biology operations with mutation, simple crossover and selection operation in minimizing the objective function. (Ardia *et al.*, 2011).

Differential evolution algorithm and genetic algorithm have similar procedures. The significant difference is shown in generating the solutions. DE algorithm is dependent on mutation operation but genetic algorithm is dependent on crossover operation. The main operation of DE is principled on the variance of sampled pairs of solutions which are random in the population. In searching and selecting operation mechanism, DE algorithm adapts mutation operation to find the candidate area in the search area. The crossover will mix the information about good combinations and find for better solution (Karaboga and Okdem, 2004). Figure 2.2. shows the following steps in Differential Evolution Algorithm:

```
Initialization
Evaluation
Repeat
Mutation
  Recombination
Evaluation
Selection
Until (Termination criteria are met)
```

Source: Karaboga and Okdem, 2004, p. 54

Figure 2.2 Differential Evolution Algorithm

2.4.1. Initialization

In this phase, the initial population (generation – 0) and control parameter should be determined. Initial population is all individual in the population before DE iteration started. Individual in the population is a collection of initial solution that can be obtained from heuristic method or random method. Three DE parameters are population size (NP), mutation coefficient (F) and crossover coefficient (Cr). NP is all individual in one generation and the value will not be changed in searching process. However, if searching faces the problem, the value of NP can be increased. Generally, the value of Np = 10 x d, where d is dimension size (row matrix).

2.4.2. Mutation

Next phase is mutation operation. Mutation is an operation in generating mutation vector ($V_{i,g}$) that is obtained from multiplying the differences of two vectors in present generation that is chosen random with mutation control parameter (F) then added by the third vector that is randomly chosen. This process is formulated as Equation (2-1).

$$V_{i,g} = x_{r0,g} + F.(x_{r1,g} - x_{r2,g}) \tag{2-1}$$

Where:

$V_{i,g}$	= mutation vector i in generation g
$x_{r0,g}$, $x_{r1,g}$, $x_{r2,g}$	= vector that is randomly chosen in generation g
F	= mutation coefficient

2.4.3. Crossover

Crossover is a process that is purposed in adding the variances of the gen in population that will enter the next generation by crossing gen that is owned by population of target vector and mutant vector. Crossover operation is element that is determining gen that is gained from the vector target and mutant for inherited to trial vector. The determination is done by comparing the value of Cr with the random value.

If the value of Cr is bigger than random value, therefore gen from mutation vector will pass to go to the trial vector. But if the value of Cr is lower or equal than random value, gen from target vector will pass to go to the trial vector. After obtained the population from trial vector, then objective value of trial vector and target vector will be evaluated that the value will be used for the next step. The general formulas of trial vector are shown as Equation (2-2) and (2-3).

$$U_{i,G+1} = (U_{1i,G+1} + U_{2i,G+1} + \ldots + U_{Di,G+1}) \quad (2\text{-}2)$$

$$u_{ji,G+1} = \begin{cases} V_{ji;G+1} \text{ if } (\mathrm{randb}(j) \leq \mathrm{CR}) \text{ or } j = \mathrm{rnbr}(i) \\ X_{ji;G} \text{ if } (\mathrm{randb}(j) > CR)\ and\ j \neq \mathrm{rnbr}(i) \end{cases}$$

$$j = 1, 2, \ldots, D \quad (2\text{-}3)$$

From equation 2-3, randb(j) is the jth evaluation of a uniform random number generator with outcome between 0 or 1.

2.4.4. Selection

In this stage, target vector and trial vector will be selected become the population of the next generation. Selection will be done by comparing the value of the objective evaluation in target vector and trial vector. Vector that will be passed to the next generation is the vector that has higher evaluation score. Trial vector can replace the target vector in the next generation if the objective evaluations score of trial vector are higher than target vector.

2.4.5. Termination State

Termination is a condition when it meets the criteria. However, if criteria of termination still not fulfilled, it will generate the next generation repeat the previous step. Generally, termination criteria are:

- Maximum total iteration
- Maximum computation time
- Stuck in convergent (the value of objective value does not change again)

2.5. Application of DE Algorithm for Job Shop Scheduling

This sub chapter will explain the basic element of DE algorithm that will be used in implementing DE to JSSP. Based on Tasgetiren et al. (2004), basic elements of DE algorithm are:

- Target vector or target individual (X_i^t)
- Mutant individual (V_i^t)
- Trial vector or trial individual (U_i^t)
- Population target (X^t)
- Mutant population (V^t)
- Trial population (U^t)
- Permutation operation (π_i^t)
- Mutation coefficient (F $\in$ (0,1))
- Crossover coefficient (CR $\in$ (0,1))
- Objective function ($f_{i,t+1}(\pi_{i,t+1} \leftarrow U_{i,t+1})$)
- Termination criteria

In formulating DE algorithm in solving JSSP, there is a general procedure. Below is the procedure in conducting DE algorithm that already adapts for solving JSSP (Tasgetiren et al., 2004):

1. Initialization Stage

   In this stage (t=0), CR, F and NP should be determined with the dimension is total job. After that, the initial individual should be designed as many as NP. For the next step, permutation should be conducted to initial individual by implementing Smallest Position Value (SPV) and evaluate each individual in population by using objective function to choose vector target.
2. Updating Generation (t=t+1)
3. Creating mutant population

   For each vector target, it will be searched mutant individual that can be obtained by using Equation (2-1). Equation (2-4) is the formula that is adjusted for JSSP.

$$V_{i,\, t+1} = X_{ai,\, t} + F.(X_{bi,\, t} - X_{ci,\, t}) \qquad (2\text{-}4)$$

4. Creating trial population

   In creating trial individual, it can be obtained from the Equation (2-3). Equation (2-5) is the formula that is adjusted for JSSP.

$$u_{ji,t+1} = \begin{cases} V_{ji;t+1}, \text{if } r_{ji,t+1} \leq CR \\ X_{ji;t} \text{ otherwise} \end{cases} \quad (2\text{-}5)$$

   CR is the crossover coefficient in the range (0,1) and $r_{ji,t+1}$ is the uniform random number between 0 until 1.

5. Conducting permutation of job

   Permutation of the job is conducted by implementing SPV rule.

6. Evaluating trial population

   Trial population will be evaluated by objective function. Equation (2-6) is the formula of the objective function.

$$f_{i,t+1}(\pi_{i,t+1} \leftarrow U_{i,t+1}) = \min \sum_{j=1}^{n}(C) \quad (2\text{-}6)$$

   Where:

   C = Makespan

7. Selecting the individual

   Value of objective function of trial individual will be compared to the vector target of previous generation. It is conducted for deciding that trial individual suitable becomes a member of the next generation population.

8. Generating operation stop

   If total iteration is already reaching total iteration that already stated, therefore DE algorithm will be stop. However if it is not reached, the procedure will back to step 2.

2.6. Design of Experiment Method

Experimentation is a crucial part of the scientific method. However, most problems need observation of the system at work and experimentation to elucidate information about why and how it works (Montgomery, 2009). Design of Experiment can be an application in developing new processes. DOE is related with the manipulation of factor that is controlled in order to find the combination of factor that can produce maximum output. In designing experiment, there are some types of experiments which are:

- Trial and Error Experiments
  This experiment just manipulate one factor without consider other factor. It needs much time, high cost, low accuracy and not efficient.
- One Factor at a Time Experiments
  This experiment is the development of trial and error experiments. There are combination factor that is observed but only one factor that is changed. It has many weaknesses such as not efficient, may get wrong conclusion and take a long time.
- Full Factorial Design
  For this experiment, there are some combination factors that are tested. It can give more accurate conclusion. In other side, the number of testing will be increased when the factor that is observed increased.
- Fractional Factorial Design
  Total of testing by using full factorial design can take a long time and get big cost. Therefore there is fractional factorial design experiment that test part of the combination factor. This experiment is generally used in screening the combination of experiments.

2.7. Three-Level (3^k) Factorial Design

This research used a three-level factorial design, better known by 3^k factorial design, which is an arrangement with the k factor that each factor is consisting of three levels which are low, medium and high. There are some differences in the use of notation for the presentation of the level of these factors, the likely level of factors presented by the digit -1 (low), 0 (medium), and 1 (high). General full factorial design is one method of experimental design that can be used, for factors with different levels, for example, there is a level for factor A, b level for factor B, c level for factor C, and so on (Montgomery, 2009). For example it can be seen in Table 2.1 that describes interactions between factors.

Table 2.1 Combination of 3^k Experiment Design

Factor K		-1			0			1			…
…		…			…			…			…
Factor B		-1	0	1	-1	0	1	-1	0	1	…
Factor A	-1	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	…
	0	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	…
	1	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	n1 n2 n..	…

From the design, there will be analysis of variance (ANOVA) to test the effect of these K factors at the level of significance (α) that have been determined. In ANOVA calculation, there will be sum square (consist of sum square total, main effects, two factor interactions, three factor interactions, until K factor interactions), degrees of freedom and mean square. Equation (2-7) to Equation (2-14) is the sum square formula for three factor ANOVA model:

Sum square for main effects:

Main Effect A $$SS_A = \frac{1}{bcn}\sum_{i=1}^{a}(y_{i...})^2 - \frac{(y_{....})^2}{abcn} \quad (2\text{-}7)$$

Main Effect B $$SS_B = \frac{1}{acn}\sum_{j=1}^{b}(y_{.j..})^2 - \frac{(y_{....})^2}{abcn} \quad (2\text{-}8)$$

Main Effect C $$SS_C = \frac{1}{abn}\sum_{k=1}^{c}(y_{..k.})^2 - \frac{(y_{....})^2}{abcn} \quad 2\text{-}9)$$

Sum square for total:

$$SS_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{l=1}^{n}(y_{ijkl})^2 - \frac{(y_{....})^2}{abcn} \quad (2\text{-}10)$$

Sum square for two factors interaction:

Interaction AB $$SS_{AB} = \frac{1}{cn}\sum_{i=1}^{a}\sum_{j=1}^{b}(y_{ij..})^2 - \frac{(y_{....})^2}{abcn} - SS_A - SS_B \quad (2\text{-}11)$$

Interaction AC $$SS_{AC} = \frac{1}{bn}\sum_{i=1}^{a}\sum_{k=1}^{c}(y_{i.k.})^2 - \frac{(y_{....})^2}{abcn} - SS_A - SS_C \quad (2\text{-}12)$$

Interaction BC $$SS_{BC} = \frac{1}{an}\sum_{j=1}^{b}\sum_{k=1}^{c}(y_{.jk.})^2 - \frac{(y_{....})^2}{abcn} - SS_B - SS_C \quad (2\text{-}13)$$

Sum Square for three factors interaction:

$$SS_{ABC} = \frac{1}{n}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}(y_{i.k.})^2 - \frac{(y_{....})^2}{abcn} - SS_A - SS_B - SS_C - SS_{AB} - SS_{AC} - SS_{BC} \quad (2\text{-}14)$$

Table 2.2 is the ANOVA table for three factor model:

Table 2.2 ANOVA for 3^3 Experiment Design

Source of Variation	Sum of Square	Degree of Freedom	Mean Square	F_0
A	SS_A	a – 1	$MS_A = \frac{SS_A}{dof}$	$F_0 = \frac{MS_A}{MS_E}$
B	SS_B	b – 1	$MS_B = \frac{SS_B}{dof}$	$F_0 = \frac{MS_B}{MS_E}$
C	SS_C	c – 1	$MS_C = \frac{SS_C}{dof}$	$F_0 = \frac{MS_C}{MS_E}$
AB	SS_{AB}	(a-1)(b-1)	$MS_{AB} = \frac{SS_{AB}}{dof}$	$F_0 = \frac{MS_{AB}}{MS_E}$
AC	SS_{AC}	(a-1)(c-1)	$MS_{AC} = \frac{SS_{AC}}{dof}$	$F_0 = \frac{MS_{AC}}{MS_E}$
BC	SS_{BC}	(b-1)(c-1)	$MS_{BC} = \frac{SS_{BC}}{dof}$	$F_0 = \frac{MS_{BC}}{MS_E}$
ABC	SS_{ABC}	(a-1)(b-1)(c-1)	$MS_{ABC} = \frac{SS_{Abc}}{dof}$	$F_0 = \frac{MS_{ABC}}{MS_E}$
Error	SS_{Error}	abc (n-1)	$MS_E = \frac{SS_{Error}}{dof}$	-
Total	SS_T	abcn - 1	-	-

From level of significance, critical area of testing value (Fα) can be determined. After defining the critical area of testing value, do a comparison of actual value (F_0) to the value of testing (Fα). With the comparisons value before, conclusions of the research can be taken. The concept of small replication can be delivered to 3k factorial design. Because completeness replication of 3k factorial design requires a greater number than the amount fixed for intermediate values of k, small replication of this design is important.

2.8. State of the Art of Differential Evolution Algorithm

There are many models and researches about job shop scheduling that is solved by many kinds of methods. Job shop scheduling problem is one of the complex problem and classified as NP (non deterministic polynomial) hard. The development of optimization methods are already increased in complexity problem. A lot of methods have been developed to overcome the limitations of exact enumeration techniques. These development techniques include genetic algorithms (GA), tabu search (TS), differential evolution algorithm (DE), neural networks (NN), simulated annealing (SA) and particle swamp optimization (PSO) (Malikarjuna, 2014).

Table 2.3 Position of Research

Production Scheduling Class	**Job Shop Scheduling Type**	**Meta Heuristic Method**
Pinedo and Chao, 1999	*Sipper and Bulfin, 1997*	*Malikarjuna et al., 2014*
Single Machine Shop	Feasible Schedule	Simulated Annealing
Multiple Machine	**Semi-Active Schedule**	Tabu Search
Static Scheduling	Active Schedule	Genetic Algorithm
Dynamic Scheduling	Non Delay Schedule	**Differential Evolution Algorithm**
Flow Shop		
Job Shop		
Assembly Line		

There are three main control parameters in Differential Evolution Algorithm which are NP (Population Size), F (Mutation coefficient) and Cr (Crossover coefficient). In this section, the state of the art is focusing on tuning all parameters and effect of these parameters on the performance of DE algorithm. There are many researches about DE algorithm (Das and Suganthan, 2011). Based on Storm and Price (1997), the reasonable value for NP could be chosen between 5D – 10D (D represents the dimensionality of the job shop). For F, the good initial choice is 0.5 but for effective range F is usually between 0.4 and 1. For parameter Cr, a lot of research uses two classifications which are small number and high number of parameters is changed in each generation (Das and Suganthan, 2011).

Table 2.4 Comparison of Setting Parameter in DE Algorithm

Researcher	**Method in Setting Parameter of DE Algorithm (NP, F and Cr)**
Gamperle*et al.*	Evaluated different parameter settings for DE algorithm on Sphere, Rosenbrock's, and Rastrigin's functions.
Liu and Lampinen	Fuzzy adaptive differential evolution using fuzzy logic controllers whose inputs incorporate and relative function values and individuals of successive generations to adapt the parameters for the mutation and crossover operation.
Brest *et al.*	Proposed a self-adaptation scheme for the DE control parameters.
Zaharie	Proposed a parameter adaptation strategy for DE based on the idea of controlling the population diversity and implemented a multi-population approach.

Source: Das and Suganthan, 2011, pp. 11 – 12.

Beside from parameters settings, there are some researches about job shop scheduling in minimizing makespan. Research from Rendy (2016) with title

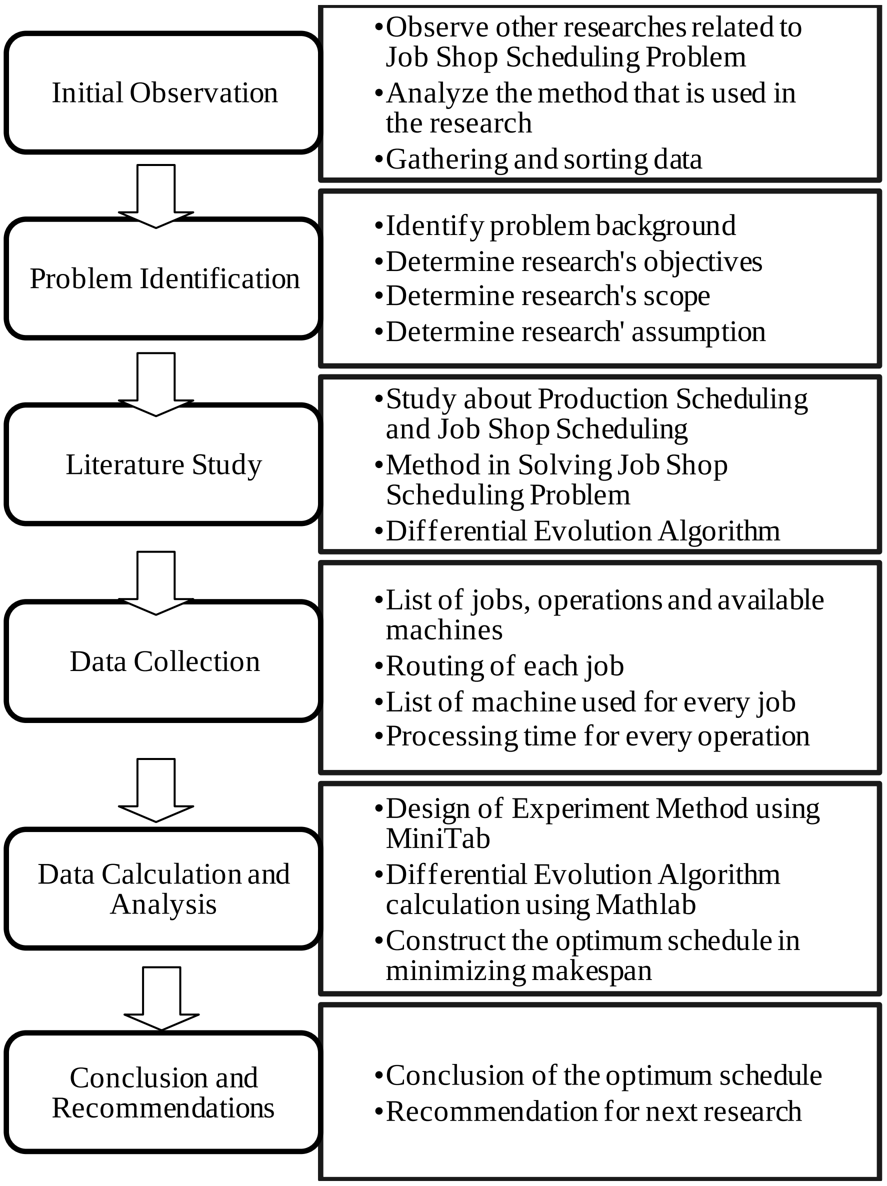
“Determining Job Shop Scheduling using Genetic Algorithms (GA) to Minimize Makespan in Automotive Bodies Manufacturing of PT. XYZ” is using Genetic Algorithms for minimizing makespan. The result of research by using GA will be compared with the result of this research by using DE algorithm with setting parameters with DOE method.

CHAPTER III

RESEARCH METHODOLOGY

3.1. Research Methodology

Choosing the appropriate methodology in research will affect the system in observing and analyzing the object. Figure 3.1 shows the research framework.



Initial Observation
•Observe other researches related to Job Shop Scheduling Problem
•Analyze the method that is used in the research
•Gathering and sorting data
Problem Identification
•Identify problem background
•Determine research's objectives
•Determine research's scope
•Determine research' assumption
Literature Study
•Study about Production Scheduling and Job Shop Scheduling
•Method in Solving Job Shop Scheduling Problem
•Differential Evolution Algorithm
Data Collection
•List of jobs, operations and available machines
•Routing of each job
•List of machine used for every job
•Processing time for every operation
Data Calculation and Analysis
•Design of Experiment Method using MiniTab
•Differential Evolution Algorithm calculation using Mathlab
•Construct the optimum schedule in minimizing makespan
Conclusion and Recommendations
•Conclusion of the optimum schedule
•Recommendation for next research

Figure 3.1 Research Methodology

Sekaran and Bougie (2013) said that research is the process of stating the solutions to a problem after conducting study, observation and analysis of the situational condition. In this chapter, it contains the procedure to conduct the research. The explanation is consists of initial observation, problem identification, literature study, data collection, data analysis and conclusion.

3.1.1. Initial Observation

In formulating the problem background, initial observation is needed. From initial observation, it can be obtained the calculation of job shop scheduling from some methods. It can be used for the comparison for this research.

3.1.2. Problem Identification

Depart from the similar problem; this research will compare the previous method with the combination method that already stated. It will see the effect of combination that has been proposed in minimizing makespan. The objective of this research is finding the influence of combination method between DOE with Differential Evolution Algorithm.

3.1.3. Literature Study

This research uses some studies from some sources that can strength the result of the research. The literature study is coming from books, journals, website and other research that has similar topic. The study is used to convince the basic of the research.

3.1.4. Data Collection

There are some input data for Differential Evolution (DE) algorithm. These data are type of job, routing of each job, processing time on each machine, available of machine and operation.

3.1.5. Calculation and Data Analysis

This research will employ hypothetic data for the input of Differential Evolution (DE) algorithm. These data are type of job, routing of each job, processing time of each machine, available of machine and data of customer demand each month. The secondary data obtained from thesis with title "Determining Job ShopScheduling using Genetic Algorithms (GA) to Minimize Makespan in

Automotive Bodies Manufacturing of PT. XYZ" (Rendy, 2016). These data will be used to compare the calculation result between Genetic Algorithms and Differential Evolution algorithm.

By using Design of Experiment (DOE) method, it will produce the best combination of the parameters that can give optimum makespan. After the first population is already defined, the operation of permutation should be conducted by using Smallest Position Value (SPV) and evaluated by objective function. The next phase is mutation to produce mutant individual. The mutant individual will be evaluated before go to the next phase. After mutation, it should be conducted the crossover to get trial individual that become trial population. In this phase, there is also the evaluation of trial individual that is suitable for the next generation population. The last phase is selection the best individual. This individual is the new solution of production scheduling. For this research the calculation uses MATLAB in calculating DE algorithm and Statistical software for finding optimum parameters.

3.1.6. Conclusion and Recommendations

Conclusions of this research can be obtained from the analysis. The conclusion is the picture of the solution of the problem and summarize of the analysis. In the conclusion, the research objective will meet the solution. In this chapter, there are also the recommendations and suggestions for the next research related to this topic.

3.2. Detail Research Framework

The detail framework of research methodology is shown in Figure 3.2 below. This figure shows the preparation step before conducting Differential Evolution Algorithm. The step starts from generating job, routing of each job, determine processing time each job, determine some factors for permutation and crossover then define total iteration with size of population and start to create population for Differential Evolution Algorithm. For permutation, crossover, total iteration and size of population, it will be defined using DOE method to get the optimum value.

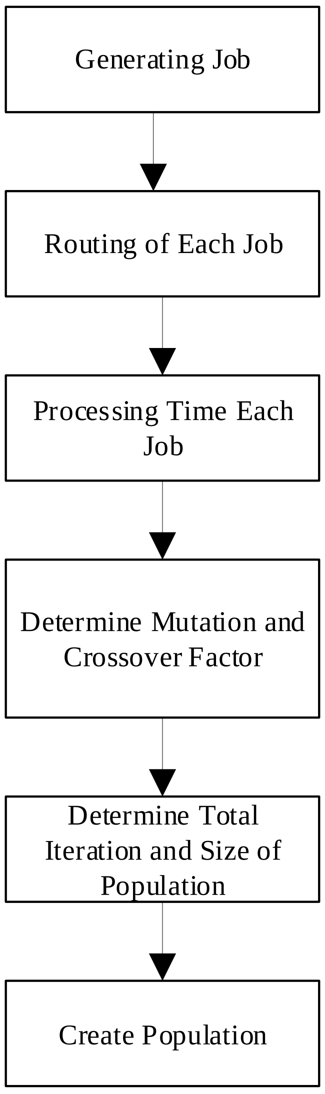

Generating Job
Routing of Each Job
Processing Time Each Job
Determine Mutation and Crossover Factor
Determine Total Iteration and Size of Population
Create Population

Figure 3.2 Detail Research Framework

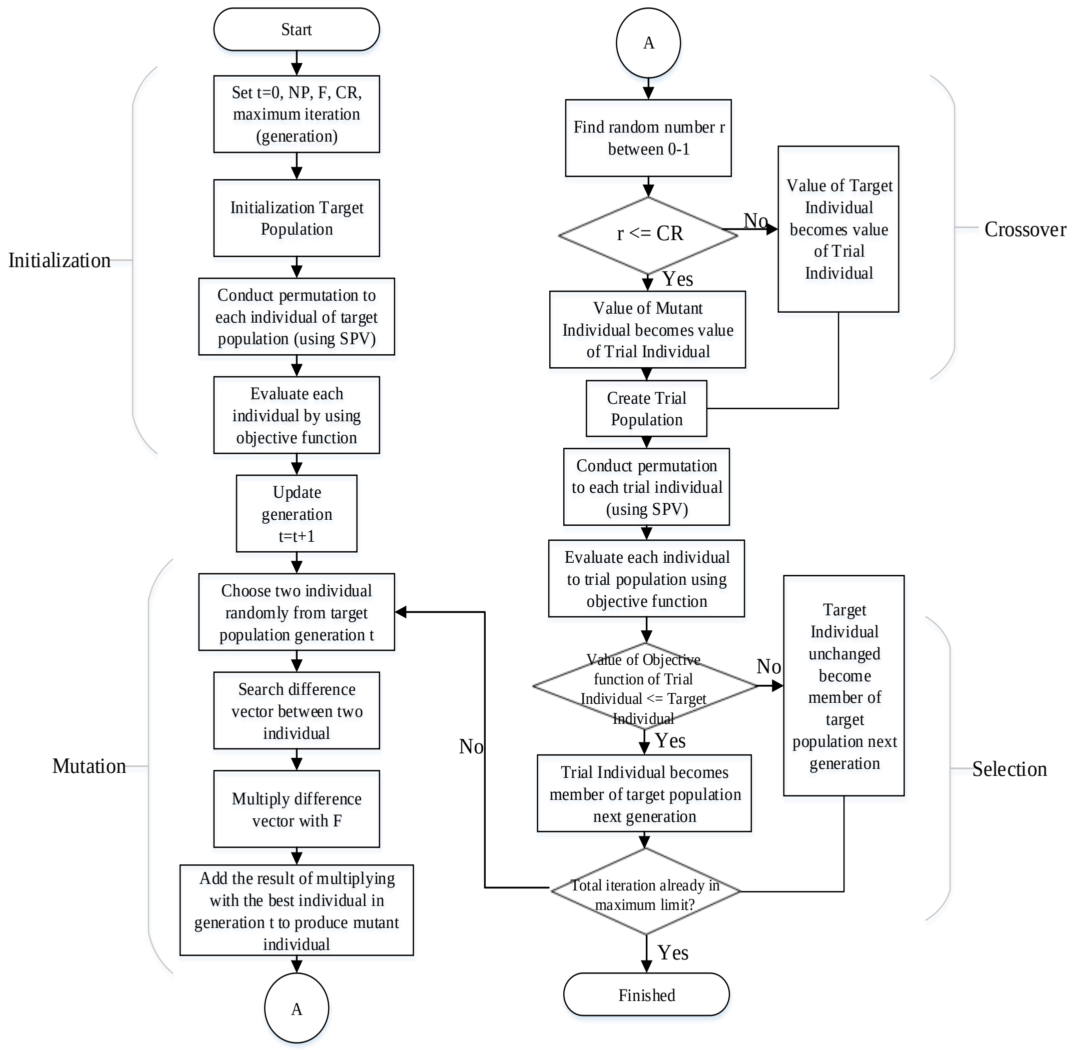
There are four factors that will be calculated using DOE method which are permutation factor, crossover factor, total iteration and size of population. There will be three levels which are low, medium and high. Tonge et al. (2012) said the interval with range 0.4 to 1 could be considered. The good first choice for crossover factor is 0.1 whereas in general crossover that is used relatively big, which is 0.9 to 1 (Storm and Price, 1997). From the same literature, it also said that effectiveness of population size is 5 to 10 dimensions or individual per population. If the total number of iteration is high, the solution can become more optimal. Table 3.1 shows the values of each levels and factor.

Table 3.1 Level of Each Parameter in DOE

Level	F	CR	Population Size	Iteration
Low (-1)	0.4	0.1	5	5
Medium (0)	0.7	0.5	8	10
High (1)	1.0	1.0	10	15

3.3. Flowchart of Differential Evolution Algorithm Scheduling

Figure 3.3 shows the flowchart in conducting the Differential Evolution Algorithm. This flowchart is divided in to four phases which are initialization, mutation, crossover and selection. The initialization is got from calculation of DOE method. The objective function is minimizing the makespan.



Start
Set t=0, NP, F, CR, maximum iteration (generation)
Initialization Target Population
Conduct permutation to each individual of target population (using SPV)
Evaluate each individual by using objective function
Initialization
Update generation t=t+1
Choose two individual randomly from target population generation t
Search difference vector between two individual
Multiply difference vector with F
Add the result of multiplying with the best individual in generation t to produce mutant individual
Mutation
A
A
Find random number r between 0-1
r <= CR
No
Yes
Value of Target Individual becomes value of Trial Individual
Crossover
Value of Mutant Individual becomes value of Trial Individual
Create Trial Population
Conduct permutation to each trial individual (using SPV)
Evaluate each individual to trial population using objective function
Value of Objective function of Trial Individual <= Target Individual
No
Yes
Target Individual unchanged become member of target population next generation
Selection
Trial Individual becomes member of target population next generation
Total iteration already in maximum limit?
No
Yes
Finished

Figure 3.3 Flow Chart of Differential Evolution Algorithm

After initialization, the generation should be updated. In creating a new generation there is mutation and crossover phase that produce the individual. The selection of the individual is based on objective functions. All the steps will be repeated until termination criterion is reached. Figure 3.3 shows the flow chart of Differential Evolution Algorithm procedures until the model created.

CHAPTER IV

DATA ANALYSIS

Data processing is done after the process of collecting the necessary data in the study is completed in order to produce output that is consistent with the purpose of research. The analysis process will be divided into five stages which are modeling algorithm, create algorithm in the program, verification and validation of the model against the program, input data and input parameters used.

4.1. Differential Evolution Algorithm Modeling and Program

In order to search for the best scheduling solution of this study, data was collected were processed using the programming language MATLAB (Matrix Laboratory). MATLAB is widely used in the implementation of numerical algorithms for various applications. MATLAB works with the concept of matrix and has a library of mathematical functions and engineering are complete and used for technical calculations such as industrial, electro, civil, geodetic and even economics.

Figure 3.3 shows flow chart of Differential Evolution (DE) algorithm to find job shop scheduling. For the program code, it can be seen in appendix III. Procedure of the program can be described as follow:

- Initialization Process
    - Setting Generation-0

      In applying DE algorithm, it takes control parameters that will be used at various stages of mutation factor (F) and crossover factor (CR). In addition, there are two parameters used which are population size parameters (NP) and the number of iterations that will be used for the entire process generation. Determination of these parameters are using statistical software application assistance program, the Design of Experiment (DOE). Further explanation for finding parameter using DOE can be seen in result analysis. Result of calculation DOE for the parameter are F = 0.7; CR = 0.5; NP = 10; and total iteration = 15.

- Determine Initial Population

  This initial population is defined if there is no initial job scheduling. Total population illustrates the number of individuals in a population. In the matrix of the target population, individuals are arrays of columns in which the individual is represented by a single column. Genes are arrays of row target population matrix.

$$\left\{\begin{array}{cccc} \text{Gen 1-1} & \text{Gen 2-1} & \dots & \text{Gen N-1} \\ \text{Gen 1-2} & \text{Gen 2-2} & \dots & \text{Gen N-2} \\ \dots & \dots & \dots & \dots \\ \text{Gen 1-n} & \text{Gen 2-n} & \dots & \text{Gen N-n} \end{array}\right\}$$

Individual 1	**Individual 2**	**…**	**Individual N**

Figure 4.1 Illustration Population Matrix

Initial population is made from some individual and genes with formulation:

Initial Population Gen = LL + (UL – LL) x RN

Where:

LL	: Lower Limit
UL	: Upper Limit
RN	: Random Number

Input from formula above is -1 for lower limit, +1 for upper limit and random number between 0 and 1.

- Determine Order of Gen

  Each initial individual in initial population is composed by gen that has different value from random number. All of genes will be sorted from the smallest until the biggest values. For example:

Table 4.1 Illustration Order of Gen

Individual 1		**Individual 1**
Gen 1-1 = 0.53	**Sorted to**	Gen 1-2 = -0.32
Gen 1-2 = -0.32		Gen 1-3 = 0.14
Gen 1-3 = 0.14		Gen 1-1 = 0.53

From example table 4.1, the order of the gen become gen 2, gen 3 and gen 1. Gen for job shop scheduling problem is a job then the order of the job for individual 1 is job 2, job 3 and job 1.

- Evaluate Each Individual

  Each individual will be evaluated from objective function that is produced each individual. Objective function in this research is makespan. Each individual that has job order will be combined with machine and sequence matrix as input for calculating makespan. Objective of evaluation is finding the individual of first generation that has the smallest makespan. Individual with the smallest makespan in that generation will be survive to next generation.

- Create new generation or iteration

  Population in the initial iteration will evolve become new population of new iteration. The population evolves by some processes which are mutation, crossover and selection process. Initial generation or iteration is symbolized with t=0 and for new iteration is symbolized t=t+1.

- Mutation Process

  After the initialization process, DE will be mutating and recombining the initial population to produce new populations. Mutations in the context of Genetic have definition to change become random element. Therefore, the DE algorithm, the mutation process involves two random vectors in which the difference between the two will produce a difference vector. Difference vector will be multiplied by a factor permutation (F) and added to the target vector to generate a vector mutant population based on equation (2-1).

- Crossover Process

  In completing the strategy of searching differential mutation, DE uses the process of crossing with purpose to increase the diversity of the population parameters. Crossover trial vector construct of the parameter values have been copied from two different vectors (Price and Storn, 2005, pp. 39). Vector or individual trial is the result of a crossover between individual targets and individual mutants. Therefore, most of the genes in an individual trial come from individuals in the target population and others come from individuals in the mutant population. Intake of genes carried by the comparison between the random numbers is generated for each of the genes concerned with the operator crossovers. Crossover process uses formulation in equation (2-2) and (2-3).

- Selection Process

  The selection process aims to select the individual who is eligible to enter the next iteration. Selection of individuals is done by comparing the objective function is generated each individual in the target population with each individual in the population trial. For example, an individual one in the target population generates makespan is smaller than the individual one on trial population, then used in the next iteration process is an individual one of the target population. With the selection process, the population of the next generation will be better.

- Termination Process

  After the new population is established, the process of mutation, crossover and selection will occur repeatedly. This process will continue until the termination criteria are already determined. In this study, the criteria that will be used is the number of iterations. The computer program will automatically stop the iteration process if the total iteration is already appropriate with total iteration that is determined before. Determination of the number of iterations is very large have the possibility to achieve optimal results, but the computation time required will be very long.

4.2. Concept of DE Algorithm in Job Shop Scheduling Problem

Every step from section 4.1 will be applied to Job Shop Scheduling Problem. From table 4.2 and 4.3, it can be seen the sequence, machine and time of the jobs that will be used for problem.

Table 4.2 Sequence of Job

Job	Sequence				
	Sequence 1	**Sequence 2**	**Sequence 3**	**Sequence 4**	**Sequence 5**
1	Machine 1	Machine 2	Machine 5	Machine 3	Machine 4
2	Machine 5	Machine 1	Machine 3	Machine 2	Machine 4
3	Machine 5	Machine 1	Machine 3	Machine 2	Machine 4
4	Machine 1	Machine 3	Machine 2	Machine 4	Machine 5
5	Machine 1	Machine 2	Machine 5	Machine 3	Machine 4
6	Machine 3	Machine 1	Machine 2	Machine 4	Machine 5

Source: Rendy, 2016

It will use 6 jobs with 5 different machines. Table 4.2 shows the order of machine that is used for every job. For example, job 1 will be finished after doing process machine 1 – 2 – 5 – 3 – 4.

Table 4.3 Run Time Machine per Job

Job	Machine				
	Machine 1	**Machine 2**	**Machine 3**	**Machine 4**	**Machine 5**
1	1600 sec	1600 sec	1600 sec	1600 sec	1600 sec
2	2800 sec	2800 sec	2800 sec	4000 sec	6000 sec
3	2800 sec	2800 sec	2800 sec	4000 sec	6000 sec
4	2000 sec	1200 sec	1600 sec	1600 sec	1600 sec
5	1200 sec	1000 sec	1600 sec	1600 sec	2000 sec
6	1000 sec	800 sec	1400 sec	800 sec	1600 sec

Source: Rendy, 2016

From table 4.3, it can be seen the run time of each machine for each job. The time is described in seconds.

Table 4.4 Values of Parameter for Verification and Validation

DE Parameter	**Values**
Permutation Factor (F)	0.6
Crossover (CR)	0.5
Population Size (NP)	5
Total Iteration	1

DE algorithm parameter values that are used for this problem are 0.6 for permutation factor (F), 0.5 for operator crossovers (CR) and 5 for the size of the population. In simplifying the calculation, total iteration that is used for this explanation is one. Table 4.4 shows the summary for parameter that is used for this problem.

After all data are already prepared, the calculation can be done. Stages of the manual calculation of the DE algorithm for Job Shop Scheduling Problem are as follows:

- Permutation on each Individual in the Target Population

  The target population for the verification and validation process is a 6 x 5 matrix where the columns in the matrix represent the population of individuals in the population while the line stating job. This matrix contains a random number with a range of values between -1 and 1. The permutations aim to find a job execution order for each individual. Permutation is done by sorting the random numbers in a column from the smallest value to the largest value.

Table 4.5 Target Population Matrix

Job/Individual	**Target Population**				
	Individual 1	**Individual 2**	**Individual 3**	**Individual 4**	**Individual 5**
1	0.962	0.788	0.734	0.776	0.263
2	0.924	-0.919	-0.817	0.845	-0.076
3	-0.833	0.921	-0.428	0.462	-0.997
4	0.313	-0.639	-0.058	0.173	0.857
5	0.400	0.763	-0.298	0.913	-0.570
6	-0.937	-0.175	-0.409	0.308	-0.972

First job in individual 1 that should be done is job 6 because the smallest random number in individual 1 is in job 6. After job 6, the second of the smallest random number is job 3 and it means the second order of the job is job 3. It will continue until the biggest random number.

Table 4.6 Job Order after Permutation

	Job Order After Permutation				
	Individual 1	Individual 2	Individual 3	Individual 4	Individual 5
Job	6	2	2	4	3
	3	4	3	6	6
	4	6	6	3	5
	5	5	5	1	2
	2	1	4	2	1
	1	3	1	5	4

- Calculate Objective Function each Individual in Target Population

  Objective function in this study is makespan of each individual. In calculating the makespan, it will use semi-active schedule algorithm. In calculating makespan, the first step is setting initial time equal with zero. For each individual, in order of each job, it will be searched for the completion time for each job. Job completion time is calculated by adding the time the job is concerned with the time of the previous process (predecessor). Time predecessor that is used is the greatest value that is selected from processing time of the job that is concerned at the previous work station or previous job processing time at the same work station. Table below shows the summary of makespan for each individual inside of population. Individual with the smallest makespan in first population is individual 3 with total makespan 41,200 seconds with job order 2-3-6-5-4-1. This individual will become gen for population in vector target.

Table 4.7 Makespan of Each Job in Target Population

	Job Order				
	Individual 1	Individual 2	Individual 3	Individual 4	Individual 5
Job	6	2	2	4	3
	3	4	3	6	6
	4	6	6	3	5
	5	5	5	1	2
	2	1	4	2	1
	1	3	1	5	4
Makespan (Seconds)	**50,000**	**45,200**	**41,200**	**47,800**	**48,000**

- Create Population of Target Vector and Mutant

  After getting the best individuals in the population target, then that individual genes will be used for vector targets population in order to mutation process. Mutations happen to create a mutant population with additional of two random vectors. The difference from the value of each gene in two random vectors will be multiplied by a permutation factor (F) and then added to the value of the relevant gene in the target vector. For this problem, the value of permutation factor is 0.6. The formula for mutation is from equation (2-4).

Table 4.8 Vector Target Population

Vector Target Population				
Individual 1	**Individual 2**	**Individual 3**	**Individual 4**	**Individual 5**
0.734	0.734	0.734	0.734	0.734
-0.817	-0.817	-0.817	-0.817	-0.817
-0.428	-0.428	-0.428	-0.428	-0.428
-0.058	-0.058	-0.058	-0.058	-0.058
-0.298	-0.298	-0.298	-0.298	-0.298
-0.409	-0.409	-0.409	-0.409	-0.409

Table 4.8 shows vector target population. Individual from vector target population is coming from individual 3 target population. Below is the table of random vector population 1. The individual of random vector is coming from individual 1, 2, 4 and 5 that randomly ordered.

Table 4.9 Random Vector Population 1

Random Vector Population 1				
Individual 1	**Individual 2**	**Individual 3**	**Individual 4**	**Individual 5**
0.788	0.263	0.788	0.962	0.776
-0.919	-0.076	-0.919	0.924	0.845
0.921	-0.997	0.921	-0.833	0.462
-0.639	0.857	-0.639	0.313	0.173
0.763	-0.570	0.763	0.400	0.913
-0.175	-0.972	-0.175	-0.937	0.308

Table 4.10 shows the random vector population 2. As previous explanation, vector target, random vector 1 and random vector 2 will become input in mutation process and resulting population mutant.

Table 4.10 Random Vector Population 2

Random Vector Population 2				
Individual 1	**Individual 2**	**Individual 3**	**Individual 4**	**Individual 5**
0.776	0.776	0.263	0.788	0.962
0.845	0.845	-0.076	-0.919	0.924
0.462	0.462	-0.997	0.921	-0.833
0.173	0.173	0.857	-0.639	0.313
0.913	0.913	-0.570	0.763	0.400
0.308	0.308	-0.972	-0.175	-0.937

Individuals on random vectors derived from the target population of individuals chosen randomly by the program. However, individuals at random vector may not occupy the same position there are other random vector and the vector of the target. For example, the first vector of the target individual, random vectors 1 and random vectors 2come from individual 3, 2 and 4 in the target population. The third column is different there should not be the same. Table 4.11 is the table of population mutant that is created from mutation process between vector target population, random vector population 1 and random vector population 2.

Table 4.11 Mutant Population

Mutant Population				
Individual 1	**Individual 2**	**Individual 3**	**Individual 4**	**Individual 5**
0.741	0.426	1.049	0.838	0.622
-1.875	-1.370	-1.323	0.289	-0.864
-0.153	-1.303	0.723	-1.480	0.349
-0.545	0.352	-0.956	0.513	-0.142
-0.388	-1.188	0.502	-0.516	0.010
-0.699	-1.177	0.069	-0.866	0.338

From these tables, the value of the gen (the first line) in the first individual mutant population was obtained from the calculation as follows:

$V_{1,1}$ (Gen 1 of Individual 1) = $X_{1,1} + F.(X_{1,1} – X_{1,1})$

= 0.734 + ((0.788-0.776)*0.6)

= 0.741

- Conduct Crossover to Create Trial Population

  The next step is crossover. The crossover process aims to create a trial population in which genes in the trial population derived from the target population and the population of mutants that were selected by using the crossover operator (CR). The process starts with create a random number for each gene and then compare the value of the random number with the crossover rate (CR). If the random number value is less than or equal to crossover rate (CR), the trial population gene taken from a gene mutant population. Conversely, if the value of the random number is greater than the value of CR, the gene that is used in the trial population is the target population gene. In this problem, the value of crossover rate (CR) is 0.5. Table 4.12 contains the gene of trial population. Number with bold will represent the gene from target population.

Table 4.12 Trial Population

Trial Population				
Individual 1	**Individual 2**	**Individual 3**	**Individual 4**	**Individual 5**
0.962	0.426	1.049	0.838	0.622
0.924	-1.370	**-0.817**	0.289	**-0.076**
-0.153	**0.921**	0.723	-1.480	**-0.997**
-0.545	0.352	-0.956	**0.173**	-0.142
0.400	-1.188	0.502	**0.913**	0.010
-0.699	-1.177	**-0.409**	**0.308**	**-0.972**

- Calculate Objective Function each Individual in Trial Population

  After the trial population is created, the objective function of each individual should be calculated. All steps that are conducted for calculating makespan are similar with the step in calculating makespan of target population. It is started from job order and continued with calculation of makespan of each individual. Below table shows makespan from trial population.

Table 4.13 Makespan of Each Job in Trial Population

	Job Order				
	Individual 1	**Individual 2**	**Individual 3**	**Individual 4**	**Individual 5**
Job	6	2	4	3	3
	4	5	2	4	6
	3	6	6	2	4
	5	4	5	6	2
	2	1	3	1	5
	1	3	1	5	1
Makespan (Seconds)	**48,600**	**49,400**	**51,600**	**49,200**	**44,600**

- Compare Target Population and Trial Population (Selection Process)

  The next step is comparing makespan that is calculated from each individual in target population and trial population. Gen from individual that gives optimum makespan will become gen in target population for next iteration. Table 4.14 shows the selection process for the next target population. For individual 1, trial population has smaller makespan than target population then the gen for individual 1 in next target population is come from trial population.

Table 4.14 Selection Process

Individual	**Makespan of Target Population (Seconds)**	**Makespan of Trial Population (Seconds)**	**Origin of Next Target Population**
Individual 1	50,000	**48,600**	Trial Population
Individual 2	**45,200**	49,400	Target Population
Individual 3	**41,200**	51,600	Target Population
Individual 4	**47,800**	49,200	Target Population
Individual 5	48,000	**44,600**	Trial Population

After all individuals are compared, the next target population can be derived. From the comparison above, gen for individual 1 and 5 in next target population is got from trial population and gen for individual 2, 3 and 5 in target population is got from target population. Table 4.15 shows new target population.

Because the iteration process for this problem is only done once, then the calculation will be stop to obtain the new target population. Based on a comparison of makespan, it can be known that the best solution generated in this problem is the schedule with job order of 2-3-6-5-4-1 and makespan of 41,200 seconds.

Table 4.15 New Target Population

New Target Population				
Individual 1	**Individual 2**	**Individual 3**	**Individual 4**	**Individual 5**
0.962	0.788	0.734	0.776	0.622
0.924	-0.919	-0.817	0.845	-0.076
-0.153	0.921	-0.428	0.462	-0.997
-0.545	-0.639	-0.058	0.173	-0.142
0.400	0.763	-0.298	0.913	0.010
-0.699	-0.175	-0.409	0.308	-0.972

4.3. Verification and Validation of Program

Before the program or software used to process the entire data job, there should be verification and validation of the program. The purpose of the verification program is to ensure that the program runs accordingly to a predetermined concept. In this study, a concept model of a program created is the change in makespan. This value is the total of the time to complete the entire job. Furthermore, the validation program conducted with the aim of comparing the output produced by the program with manual calculations. Validation program conducted using data from previous research (Rendy, 2016). If the output from program and manual calculation is same, then the program has been validated. Program for research data processing is executed by the computer specification, Intel Celeron 877 CPU @ 1:40 GHz, 1:40 GHz, 4:00 GB RAM. Script M File program for makespan calculation can be found in the Appendix III.

4.3.1. Verification of Program

In accordance with the purpose of research, the concept of the program is finding nearly optimal scheduling solution that can reduce makespan. It is needed to calculate makespan of before and after applying DE algorithm to search the best solution of scheduling on the program that has been created.

Beside verification for Differential Evolution itself, it needs verification for the makespan calculation. The first run of program will be used for verification. Figure 4.2 shows the Gantt chart (output from program) of the job order 2-3-6-5-4-1 and makespan of 41,200 seconds. From the Gantt chart, there is no overlapping of machine and job order.

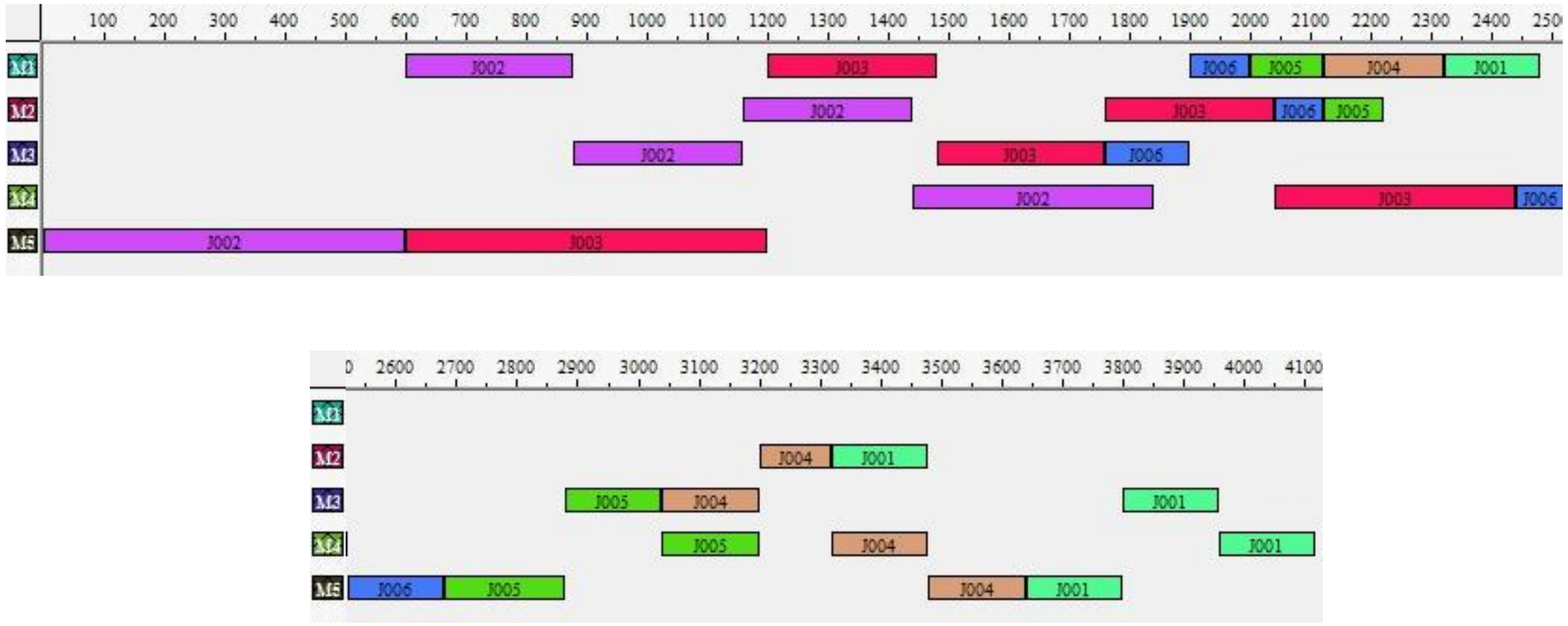


Figure 4.2 Gantt Chart of First Program Running

Table 4.16 shows the result of program verification which comparison of makespan between before and after program running 10 times iterations.

Table 4.16 Result of Program Verification

Performance	Before (1 iteration)	After (10 iterations)
Makespan	41,200 sec	39,800 sec

From Table 4.16 above, makespan decreases from 41,200 seconds to 39,800 seconds. The makespan can be reduced which is appropriate with the concept of the program that is finding optimum solution (small makespan). It means the program is already verified.

4.3.2. Validation of Program

The results run program for first iteration with target population shows the order of the job is 2-3-6-5-4-1 with total makespanof 41,200 seconds. This calculation is totally same with the manual calculation and the concept of DE algorithm in section 4.2. There is no different in calculation. It means the program is already validated.

4.4. Input of Program

In section 4.2, it can be seen there are some input data for calculating makespan using Differential Evolution Algorithm. These data input is categorized in to two types which are Differential Evolution Parameter Input and Scheduling Data Input. Scheduling data consists of job, machine, processing time and sequence data.

4.4.1. Parameter of DE Algorithm Input

In the selection of parameters, it will be used design of experiments. The objective of the DOE is to get a combination of parameters that produce the best output. DE algorithm uses four parameters, namely the permutation factor (F), factor crossovers (CR), the size of the population (NP) and the total iteration. Because the factors used more than two factors, the DOE type used is a full factorial design. Each factor has three levels, which are low, medium and high. Thus, the number of combinations of parameters obtained is as many as 3^4 = 81 combinations of parameters. For each combination of parameters are conducted trials repetition five times so that the total search trial scheduling solutions on the program were as much as 405 times the experiment.

For parameter that is used, it can be seen from table 3.1. DOE is done using statistical software with a significance level of 5%. The results of the 405 experiments conducted can be found in appendix II. From these experiments, it is obtained by a combination of parameters that produce the smallest makespan output of 34,200 seconds, which is the combination with permutation factor 0.7, Crossover rate 0.5, Number of Population by 10 individuals, and the total iteration with 15 iterations.

4.4.2. Job, Machine, Time and Sequence Input

Besides data input parameters, data to be entered to the program is a sequence of jobs, processing time of each process and the target population. All input data must be in the form of a matrix. For a sequence matrix, each row states job and column states sequence of job. Each cell expressed machines that work on these sequences. Matrix in sequence and processing time are different. At the processing time matrix, each column states the machine, each row represents job and cell expressed time of the process. For target population matrix, each row states job, each column presents individual and each cell shows random number. From Table 4.5, it can be seen matrix of target population. If total individual is added, it just adds column and for adding job, it just adds row.

4.5. Differential Evolution Performance

After the concept and program already prepared, next step is calculating and searching for the smallest makespan for this case. In searching the smallest makespan from the job shop scheduling problem, it needs some experiments using DOE method. In section 4.5, it will be explained how to get the optimum parameter due to find the optimum solution, the result of the job shop problem case by using Differential Evolution and the time performance of Differential Evolution for Job Shop Scheduling Problem.

4.5.1. Parameter Analysis – IIDN Test

In the observation, there will be an error and it is called as residual. However residual in an observation should have requirement which are normally distributed, identical, and independent. Residual assumption test is always called as Identical, Independent and Distributed Normal (IIDN). In the normal probability plot, there are red dots around the straight line which means the residual is normally distributed. In the residual versus fits plot the red dots are not forming any patterns so it just spread randomly which means constant variance are identical. In the residual versus order plot the line and red dots are moving up and down, but it also does not form any significant pattern which means the residual data is independent.

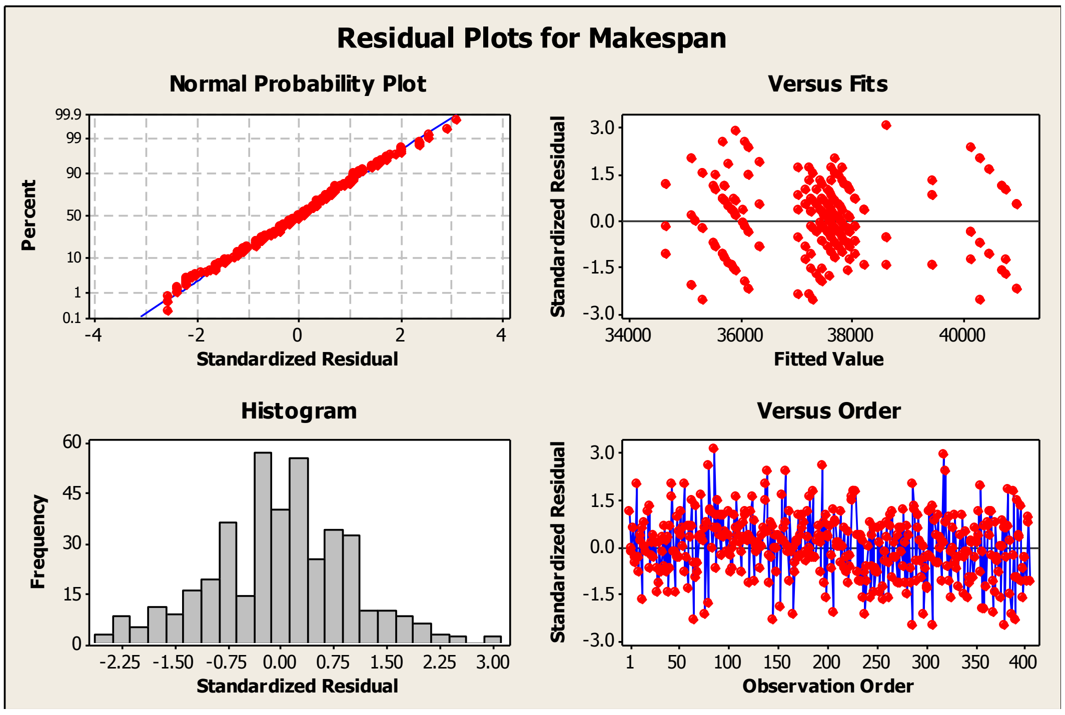


Figure 4.3 Residual Plots for Makespan

From the graph, it means that the distribution data is normal. The other test in section 4.5.1.1, 4.5.1.2 and 4.5.1.3 will show that the residual data is identical, independent and distributed normal (IIDN). Then this observation has fulfilled the requirement of residual.

4.5.1.1. Identical Test

H_0: the standardized residual data are in random order

H_1: the standardized residual data are not in random order

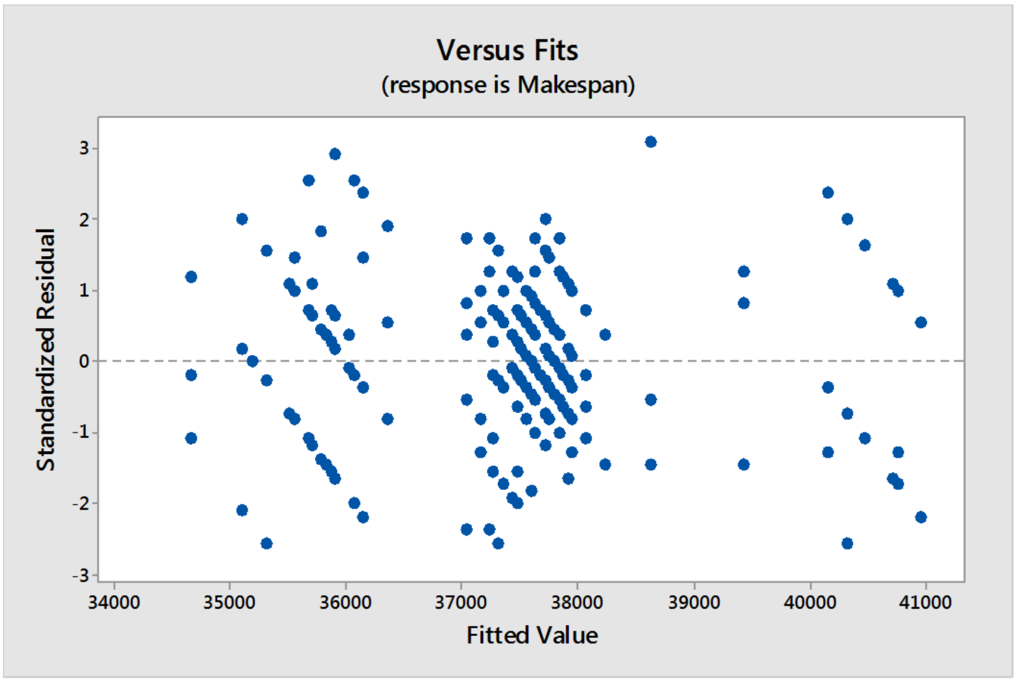
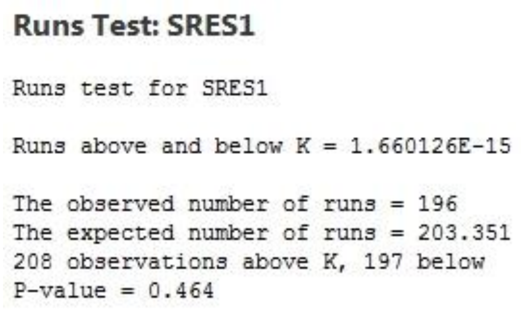


Figure 4.4 Fitted Value VS Standardized Residual Plot

The variance of the observations in each treatment should be equal. The constant variance assumption can be checked with Standardized Residual versus Fits plot. In this plot shown a random pattern of residuals on both sides of 0 and there is no significant pattern. It means the variance is constant.



Runs Test: SRES1

```
Runs test for SRES1

Runs above and below K = 1.660126E-15

The observed number of runs = 196
The expected number of runs = 203.351
208 observations above K, 197 below
P-value = 0.464
```

Figure 4.5 Runs Test

Other test to check the identical data is using runs test. If p-value is larger than alpha = 0.05 (5%), it means H_0 will be accepted. But if p-value is smaller than alpha, it means reject H_0. Since p-value of runs test is larger than alpha, the standardized residual data is in random order or it can be assumed as identical data.

4.5.1.2. Independent Test

H_0: the standardized residual data are independent

H_1: the standardized residual data are not independent

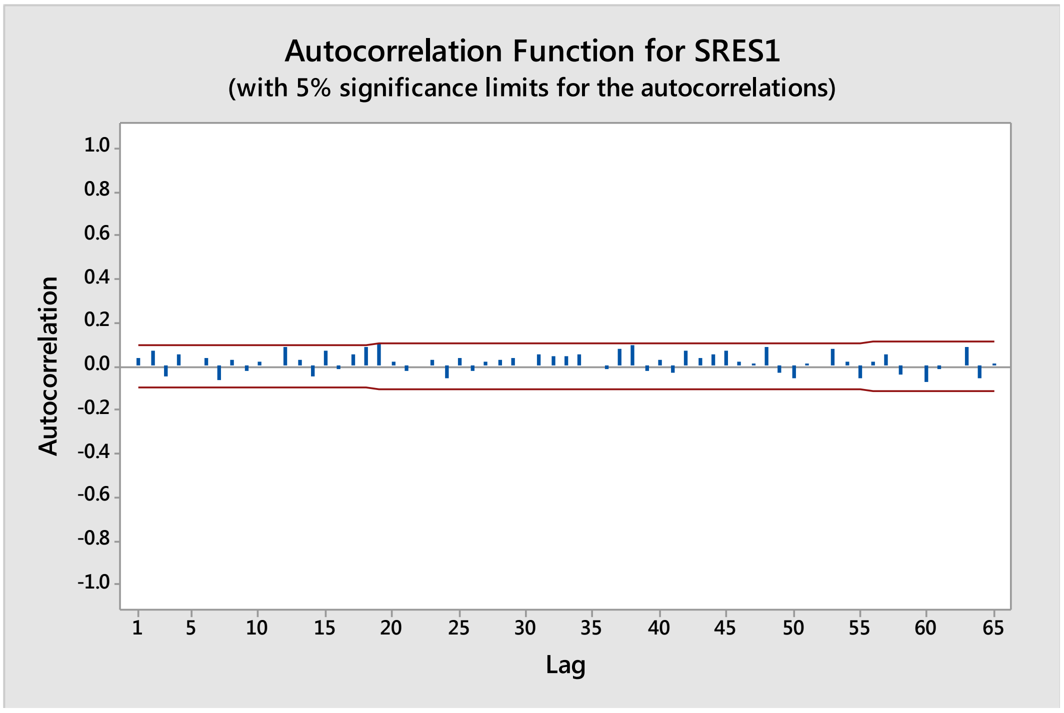


Figure 4.6 Autocorrelation Function for Standardized Residual Data

From figure above, it can be seen the autocorrelation function. Autocorrelation test can define the data is independent or dependent. The data is independent if there is no blue bar passed the limit red border. If there is blue lines pass the red border, the data is not independent. After conducting the autocorrelation test, it can be seen that there is no blue lines pass the red border. It means the standardized residual data is independent.

4.5.1.3. Normality Test

The residuals will be checked using Anderson Darling normality test and the hypothesis testing are as follows

H_0: the distribution of standardized residual is normal

H_1: the distribution of standardized residual is not normal

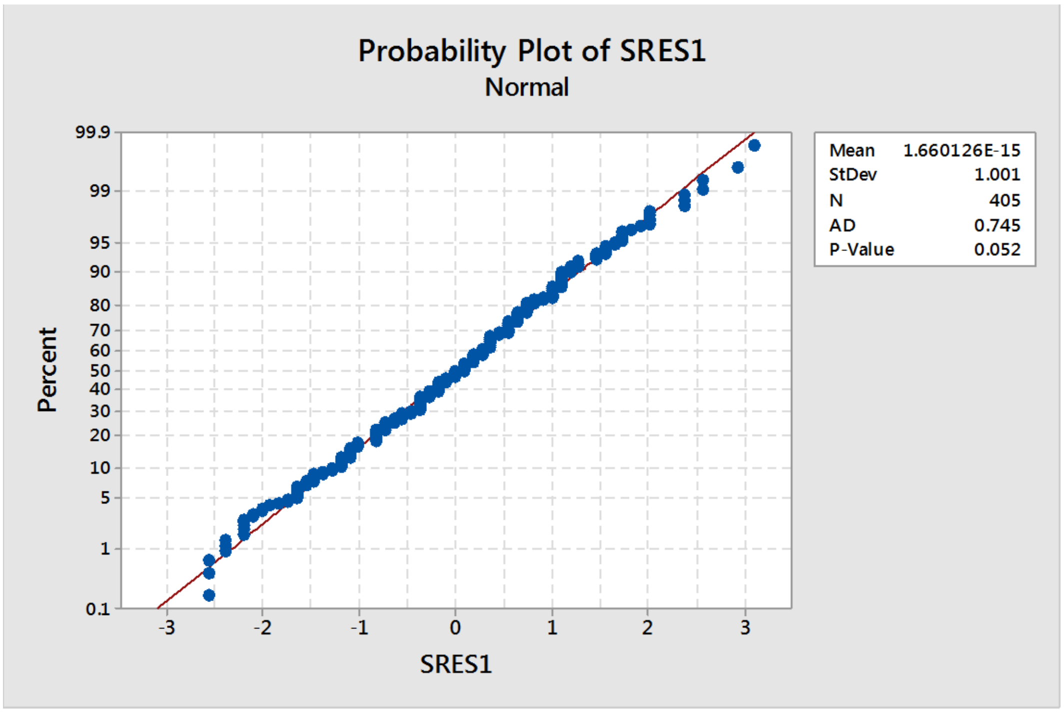
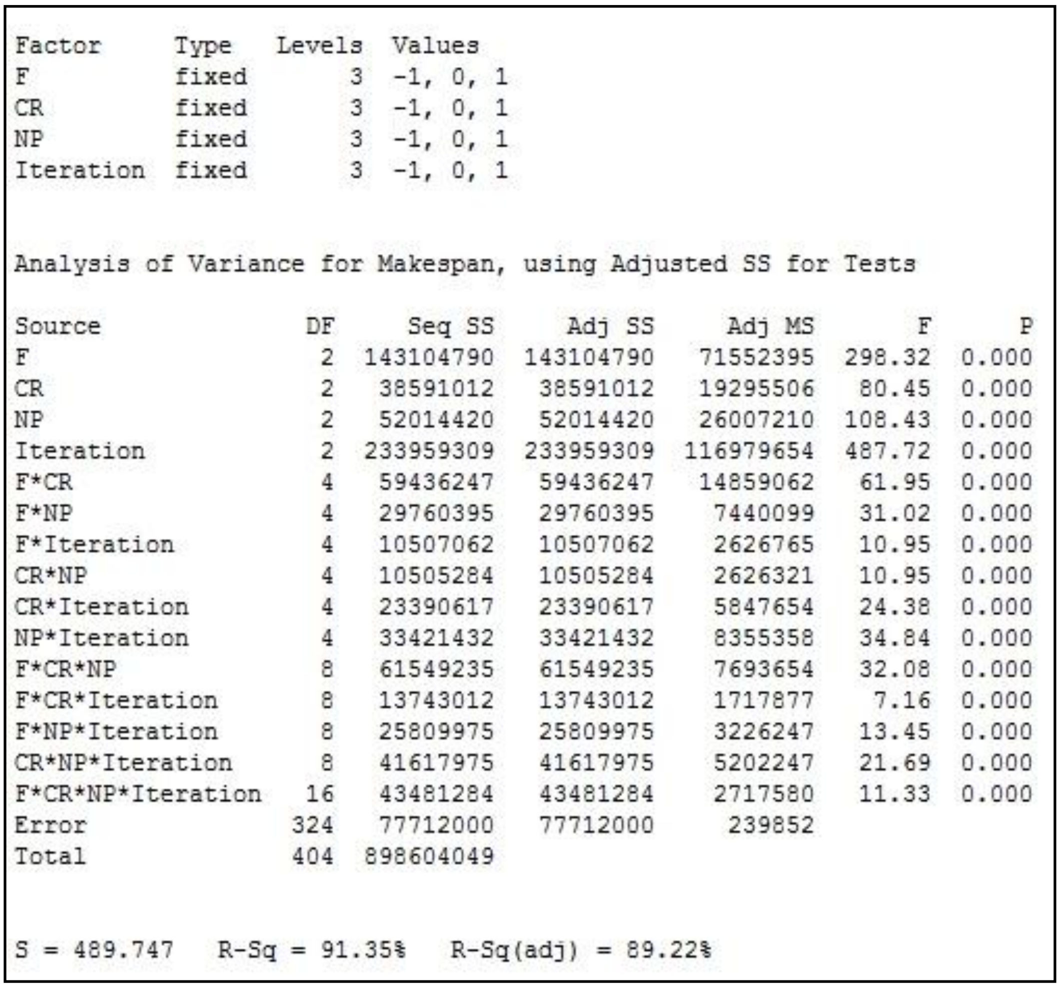


Figure 4.7 Normal Probability Plot

The residual data and standardized residual data should be normally distributed. This can be checked with a normal probability plot of standardized residuals. In small text box on the upper right, there is p-value which is 0.052. Because p-value is greater than level of significant (0.05) then do not reject null hypothesis. It means the distribution of residuals is normal. In addition, most standardized residual dots fall close to the straight line (see Figure 4.7).

4.5.2. ANOVA Result – Analyze Factorial Design

As mentioned earlier, this study is using a design of experiment to find the combination of parameters that generate a total output of makespan. Subsequently, 405 trials of 81 combinations of parameters, it can be used to see the influence of each parameter as well as the interaction between the parameters of the output produced. For that reason, Analysis of Variance (ANOVA) is conducted using statistical software. The effect is rated based on the value of significance (p-value) of each factor. The confidence level used is 95%, or α = 5% (0.05). If the p-value at a factor ≤ 0:05; then the relevant factors that significantly influence makespan, the following figure shows the parameters of the makespan.



```
Factor     Type   Levels  Values
F          fixed       3  -1, 0, 1
CR         fixed       3  -1, 0, 1
NP         fixed       3  -1, 0, 1
Iteration  fixed       3  -1, 0, 1


Analysis of Variance for Makespan, using Adjusted SS for Tests

Source              DF     Seq SS     Adj SS     Adj MS       F      P
F                    2  143104790  143104790   71552395  298.32  0.000
CR                   2   38591012   38591012   19295506   80.45  0.000
NP                   2   52014420   52014420   26007210  108.43  0.000
Iteration            2  233959309  233959309  116979654  487.72  0.000
F*CR                 4   59436247   59436247   14859062   61.95  0.000
F*NP                 4   29760395   29760395    7440099   31.02  0.000
F*Iteration          4   10507062   10507062    2626765   10.95  0.000
CR*NP                4   10505284   10505284    2626321   10.95  0.000
CR*Iteration         4   23390617   23390617    5847654   24.38  0.000
NP*Iteration         4   33421432   33421432    8355358   34.84  0.000
F*CR*NP              8   61549235   61549235    7693654   32.08  0.000
F*CR*Iteration       8   13743012   13743012    1717877    7.16  0.000
F*NP*Iteration       8   25809975   25809975    3226247   13.45  0.000
CR*NP*Iteration      8   41617975   41617975    5202247   21.69  0.000
F*CR*NP*Iteration   16   43481284   43481284    2717580   11.33  0.000
Error              324   77712000   77712000     239852
Total              404  898604049


S = 489.747   R-Sq = 91.35%   R-Sq(adj) = 89.22%
```

Figure 4.8 ANOVA Result of Parameter Setting

From the Figure 4.8 above, it can be seen that the four parameters that are used in DE algorithm provides a significant effect on total makespan. Each level of parameter has significant different makespan. This is indicated by a p-value of each parameter value under 0.05. Beside four main effects, there are some interactions which are 2-way interactions, 3-way interactions and 4-way interactions. All these interactions are also giving significant effect on total makespan.

Moreover, the effect parameters can also be assessed by looking at the value of F_{stat}. Values of F_{stat} illustrate the influence of a factor in output compared with the influence of other factors. Thus, based on the F_{stat} value can be known that the parameter that has the greatest impact is the mutation factor (F). It can be seen in figure above that value of r-square is around 91%. It means the correlation between each parameter is strong.

From Figure 4.9, permutation factor (F) and a cross-over rate (CR) have optimum values of 0.7 and 0.5 in this case. Meanwhile, other parameters such as the size of the population (NP) and the total iterations have a relationship which is inversely proportional to the total makespan produced. Makespan less precisely generated by the NP and the total iteration increases. Thus, the best combination of parameters obtained as follows:

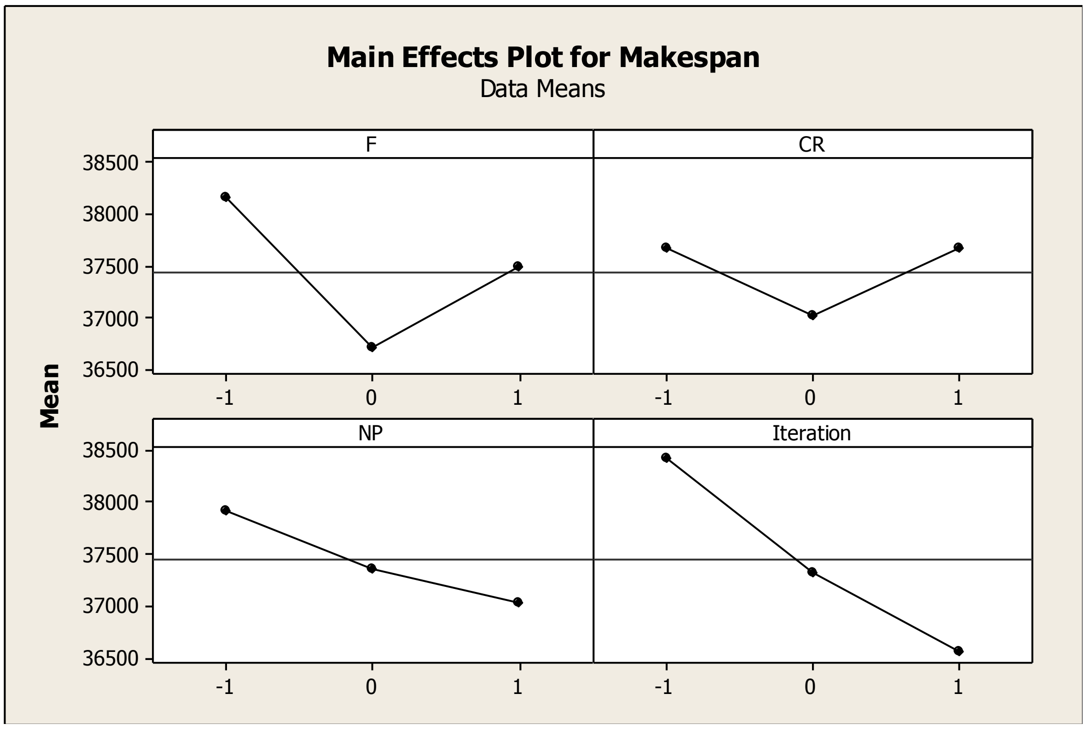


Figure 4.9 Main Effects Plot for Makespan

- Permutation factor (F) in the medium level

  In theory, the value of F should be above a certain level to prevent premature convergence. However, if the value of F is too large, the search for solutions will be sluggish. Large number of F can make big gap movement of the target vector into the mutant population. If the value of CR that is used is high, then most of the genes of the target population will be derived from the mutant population. Consequently, the resulting solution sequences trial population will not differ significantly from the resulting sequence of the target population. For a population with a very large size, it requires the small value of F because the increasing number of population size will make more individuals fill the search space. Therefore, it needs small movement due to make the searching of solution become more effective. Because of the size of the population used in this case is relatively small, then the value of F is used in accordance with the results of the DOE, that the value of F with a moderate level (0.7).

- Cross Over Rate (CR) in the medium level

  DOE results showed that the medium value of CR gives the best result. This is consistent with the theory, where the value of CR is small, it can increase probability of genes in trial populations derived from the target population. This makes the search for a solution to be sluggish due to the order of the trial population generated is not much different from the resulting sequence of the target population. CR with relatively large value will accelerate the convergence or the achievement of optimal solutions. However, there are conditions where the CR value is too high makes the DE is not tough. For example, if the value of F is used is too large, vector targets will move large enough in the mutant population. If the high value of CR is used, it will appear more random number falls below the CR so that most of the genes trial population derived from the mutant population. Thus, the resulting sequence of trial population is not much different from the sequence of the target population. This situation also makes the search for a solution to be slow. Therefore, lower CR value can make DE tougher. In accordance with the results of the DOE, CR value that will be used is 0.5.

- Number Population (NP) in the high level

  From DOE calculation, the size of a large population proved to produce total makespan smaller. The larger the population size that allows the emergence of order processing more varied job anyway. This prompted the search for a solution that is near optimal because of the many available solution options. At the DOE that has been done, the numbers on the size of the population used is very small with the aim to shorten the computation time. At the time of running a search program scheduling solution actual population sizes necessary to adjust in order to search more optimal solution.

- Total Iteration in the high level

  DOE results show that the large number of iterations have larger possibility of obtaining a nearly optimal scheduling solution. This is due to the increasing number of opportunities for the program to do permutations and crossing individuals with the best solutions generated by the previous iteration in order to produce individuals with a better solution again for the next iteration. This process will continue until the termination criteria specified. However, as well as the size of the population, the number of iterations will greatly affect the computation time. The number of iterations that too much can make the computation time becomes very long.

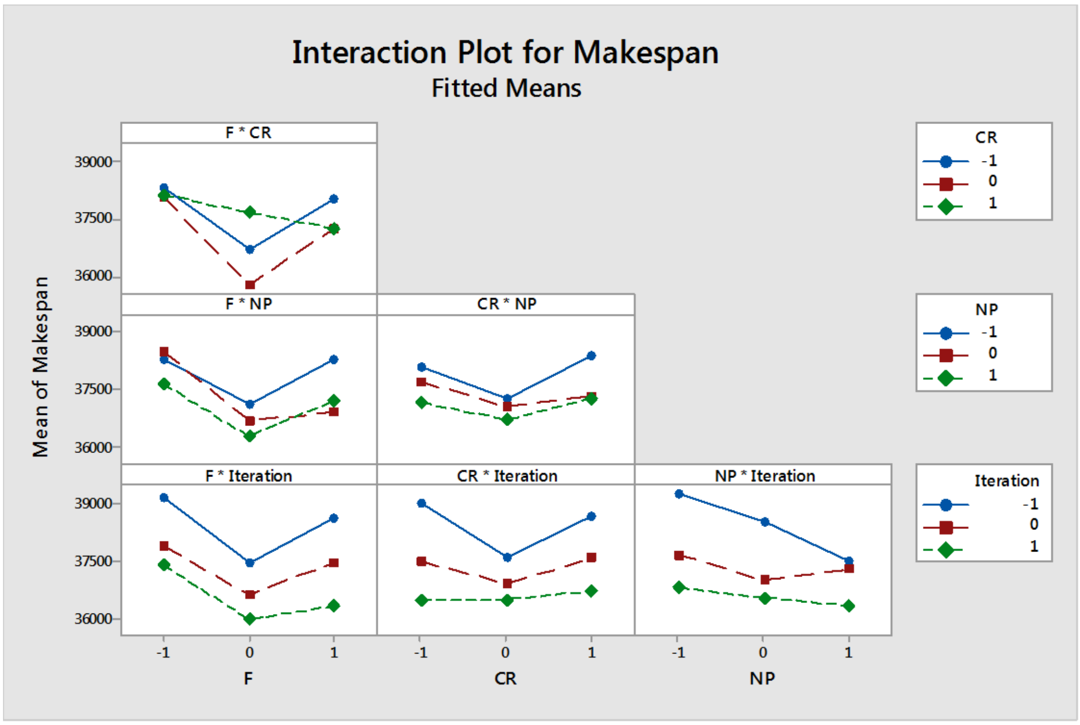


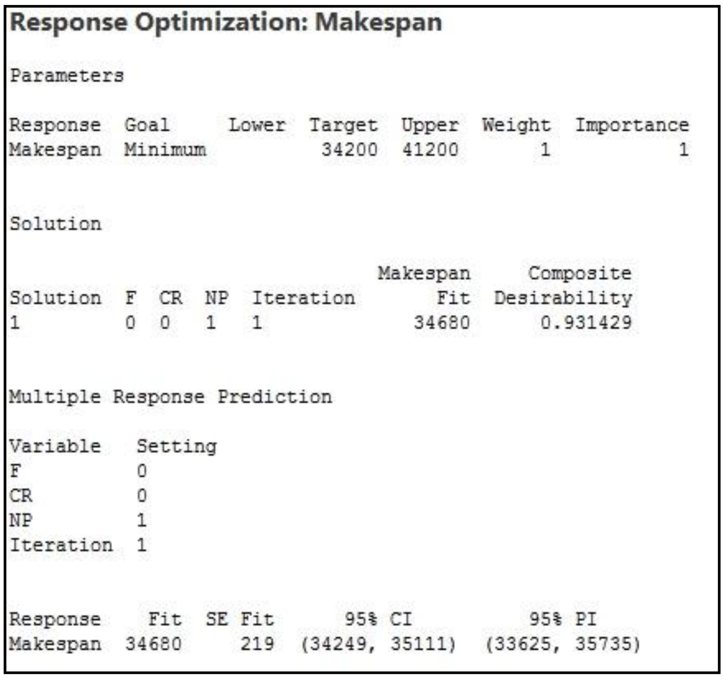
Figure 4.10 Interaction Plot for Makespan

This interaction plot for makespan show if there is interaction between two factors. There are 6 interactions and there will be discussed in this section. Permutation factor (F) and cross over rate (CR) has interaction because the line is going down for high level of CR and the other lines are going down and up. It means the optimum level between F and CR is in moderate level. For interaction between F and number of population (NP), CR and NP, F and iteration, these interactions have similar pattern each other. The pattern is going down then going up. It means the optimum makespan in moderate level for these interactions.

For interaction CR and Iteration, it shows the line is going up for high level of iteration but it is not significantly increasing. For other line, it looks like similar with interaction between F and NP. The significant different can be seen from NP and iteration interaction. The line is going down when the level increase. It means the optimum makespan is in high level for interaction NP and iteration.

4.5.3. Job Order Result by Using Differential Evolution Model

After finding the best parameters for Differential Evolution Model, it can be calculated the final result or the optimum solution of Job Shop Scheduling Problem by using Differential Evolution.



```
Response Optimization: Makespan

Parameters

Response  Goal     Lower  Target  Upper  Weight  Importance
Makespan  Minimum          34200  41200       1           1


Solution

                                  Makespan     Composite
Solution  F  CR  NP  Iteration         Fit  Desirability
1         0  0   1   1               34680      0.931429


Multiple Response Prediction

Variable   Setting
F          0
CR         0
NP         1
Iteration  1


Response    Fit  SE Fit       95% CI          95% PI
Makespan  34680     219  (34249, 35111)  (33625, 35735)
```

Figure 4.11 Response Optimization Summary

Figure 4.11 shows the calculation of response optimization of the model. From the response optimization below, it can be seen the optimum solution will be produced by using F in level 0 (moderate level = 0.7), CR in level 0 (moderate level = 0.5), NP in level 1 (high level = 10) and Iteration in level 1 (high level =15). In prediction, by using this combination of the parameters, it can produce makespan in average 34680 seconds and with composite desirability 93%.

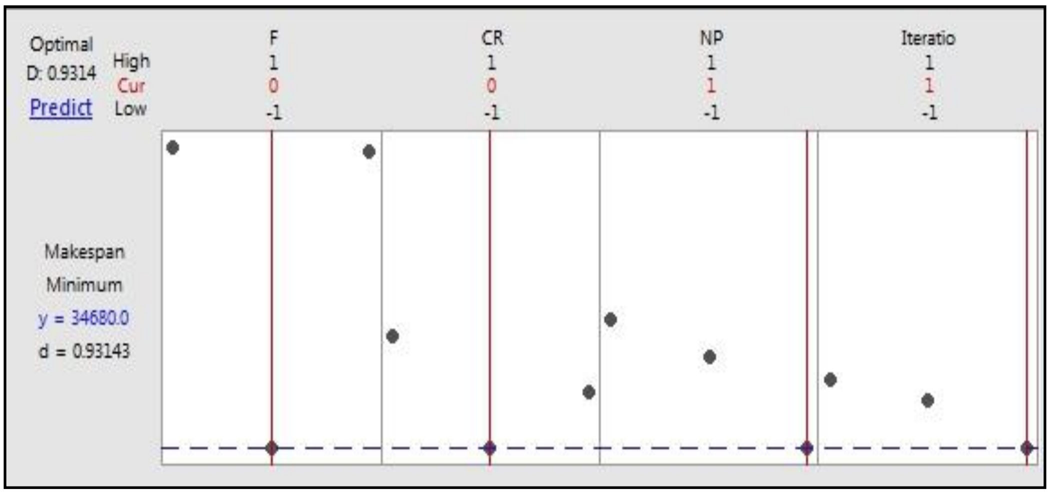


Figure 4.12 Response Optimization Graph

By finding the optimum parameter using statistical software, the total makespan is 34,200 seconds or 570 minutes with the order 2-3-5-1-4-6. Figure 4.2 shows the Gantt chart for the optimum solution of job shop scheduling problem. Actually, the statistical software also provides another alternative job order with similar makespan which is 34,200 seconds or 570 minutes. The job order is 3 – 2 – 5 – 1 – 4 – 6. The different between alternative 1 and 2 is the first and second job order.

Figure 4.13 show the gantt chart for alternative solution of job shop scheduling which has job order 3 – 2 – 5 – 1 – 4 – 6 and makespan same with the first alternative.

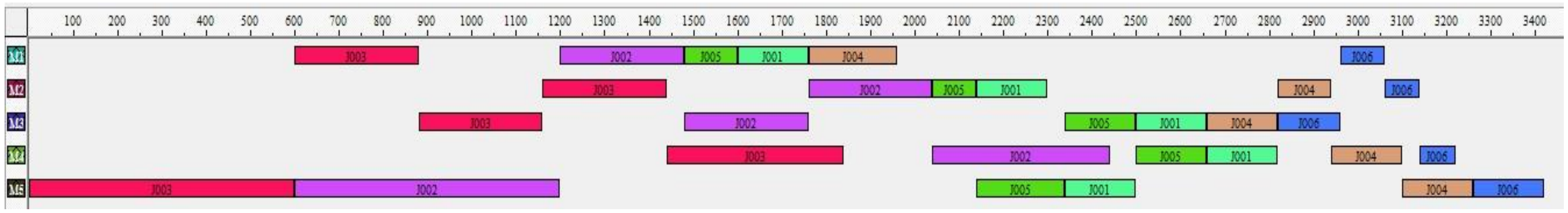


Figure 4.13 Gantt Chart for Alternative 1 Scheduling

From figure 4.13 and 4.14, the gantt charts can be compared. It can be seen the different only in first job order and second job order. For the rest job order is same. Job order for second alternative is 2 – 3 – 5 – 1 – 4 – 6.

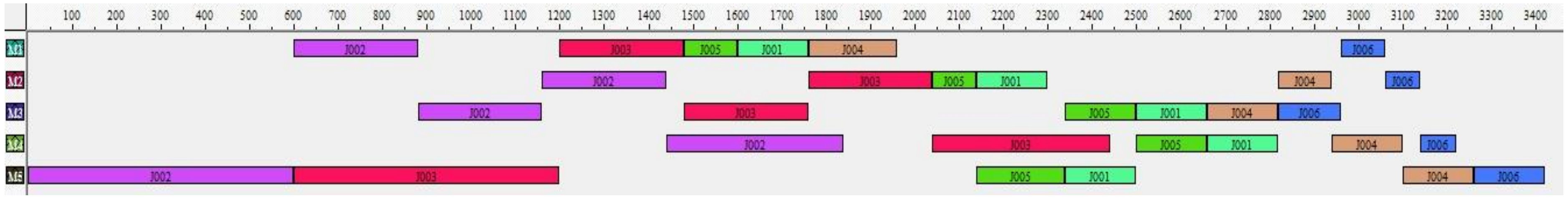


Figure 4.14 Gantt Chart for Alternative 2 Scheduling

4.5.4. Computation Time

In some journal, time consumption of calculating using Differential Evolution Algorithm is faster than other algorithms. For calculating this job shop scheduling problem (6 jobs and 5 machines) using optimum parameters, the computation time of getting optimum solution needs 7.61 seconds in average but it needs 18.46 seconds to finish all iterations process (15 iterations). Each program running results different computation time. The calculation of running time of the program can be seen in Table 4.17 below.

Table 4.17 Calculation of Computation Time on Several Runs

	Optimum Parameters (F = 0.7, CR = 0.5, NP = 10, Iteration = 15)
Run 1 (sec)	7.35 sec (iteration 6)
Run 2 (sec)	5.64 sec (iteration 4)
Run 3 (sec)	13.63 sec (iteration 12)
Run 4 (sec)	2.72 sec (iteration 2)
Run 5 (sec)	9.25 sec (iteration 9)
Average Time (sec)	7.61 seconds

There are some indicators that affect to computation time which are total of jobs, machines, number of population and total iteration. More value of each indicator can increase calculation time. In other side, other factor such as permutation factor (F) and cross over rate (CR) are set to be constant.

Table 4.18 Computation Time of DE Algorithm Model

N	**M**	**No of Population**	**Total Iteration**	**Calculation Time (seconds)**					**Average**
				1	**2**	**3**	**4**	**5**	
5	3	5	5	1.63	1.87	1.78	1.69	1.73	1.74
5	3	5	10	1.65	1.75	1.84	1.90	1.71	1.77
5	3	10	5	1.76	1.90	1.65	1.80	1.64	1.75
5	3	10	10	1.79	1.68	1.87	1.65	1.88	1.77
10	5	5	5	2.51	2.66	4.58	4.72	4.45	3.78
10	5	5	10	4.69	2.78	8.53	2.65	4.76	4.68
10	5	10	5	4.58	2.96	4.43	2.87	4.52	3.87
10	5	10	10	4.21	8.37	2.78	2.95	6.63	4.99

From table 4.18, it can be seen the variation of computation time to get optimum solution each run. More number of total job (N) and number of machine (M) can increase the running time of the program to find the optimum solution. Meanwhile number of population and total iteration are not significantly influencing calculation time for small number of total job and total machine.

Number of population and total iteration can affect computation time when the number of job and machine are large.

4.6. Comparison Differential Evolution and Genetic Algorithm Model

Comparison of Genetic Algorithm scheduling solution with the proposed scheduling algorithm based DE can be seen in the Table 4.19. In accordance with the objective function program models that have been made, scheduling solution obtained through the implementation of DE algorithm can reduce the makespan.

Table 4.19 Comparison Result for Job Shop Scheduling Problem

Solution using Differential Evolution Algorithm – DOE Method	**Solution using Genetic Algorithm**
Makespan : 34,200 seconds	Makespan : 36,000 seconds
Job Order : 2 – 3 – 5 – 1 – 4 – 6 3 – 2 – 5 – 1 – 4 – 6 (alternative)	Job Order : 2 – 3 – 4 – 6 – 1 – 5

In the Rendy (2016) ‘s Research, current scheduling of company produces makespan 38,220 seconds (637 minutes) while using Genetic algorithm produces makespan35,800 seconds (597 minutes). In this study, scheduling using Differential Evolution algorithm can produce makespan 34,200 seconds (570 minutes) with the same case. Through the application of DE algorithm, makespan reduced by 4.47% from scheduling Genetic algorithm while if compared to the current company's scheduling, makespan reduced by 10.52%. This value is obtained from the following calculation:

Makespan using company system	: 38,220 seconds (637 minutes)
Makespan using GA	: 35,800 seconds (597 minutes)
Makespan using DE	: 34,200 seconds (570 minutes)

Difference between GA and DE	= 35,800 – 34,200
	= 1,600 seconds
Difference between company system and DE	= 38,220 – 34,200
	= 4,020 seconds
Percentage between GA and DE	= (1,600/35,800) x 100%
	= 4.47%

Percentage between company system and DE = (4020/38220) x 100%

= 10.52%

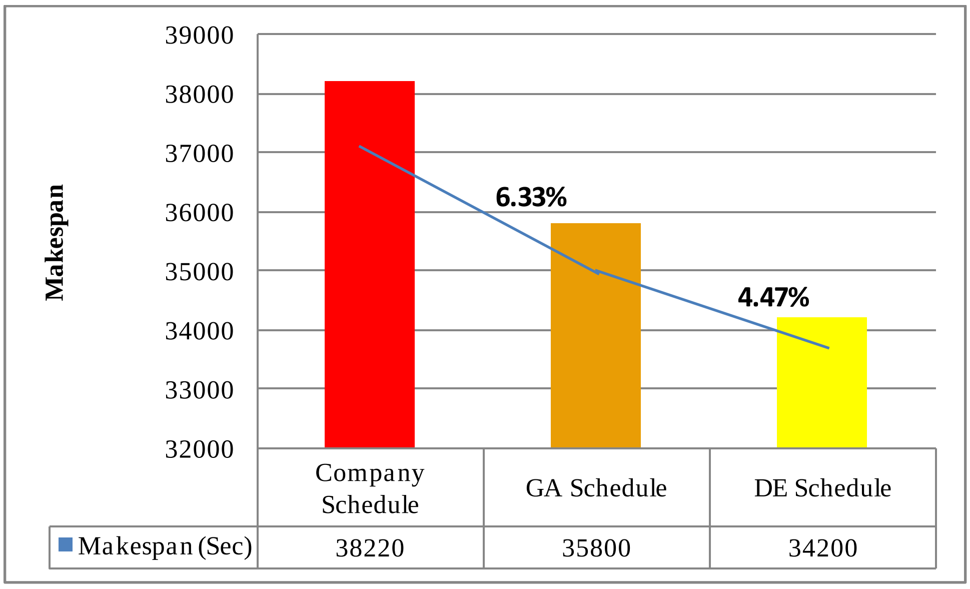


Figure 4.15 Makespan Comparison of Different Methods

By using Differential Evolution Algorithm, the production scheduling can be more effective and efficient. From the result above, it can be concluded that Differential Evolution model can give better solution than Genetic Algorithm model. Moreover, Differential Evolution method can give solution faster than Genetic Algorithm model. From Table 4.20, it can be seen the average computation time different n jobs and m machines.

Table 4.20 Comparison of Computation Time Calculation

Total Job (n)	Total Machine (m)	Number of Population	Total Iteration	Differential Evolution Model (seconds)	Genetic Algorithm Model (seconds) – (Rendy, 2016)	Differences (seconds)
5	3	5	5	1.74	1.78	0.04
5	3	5	10	1.77	3.04	1.27
5	3	10	5	1.75	2.91	1.16
5	3	10	10	1.77	4.76	2.99
10	5	5	5	3.78	4.00	0.22
10	5	5	10	4.68	7.16	2.48
10	5	10	5	3.87	7.19	3.32
10	5	10	10	4.99	12.95	7.96

Figure 4.16 shows that Genetic Algorithm takes more time to finish the calculation. The increase of total job and machine can the increase of running time that can be seen in Figure 4.16 and 4.17.

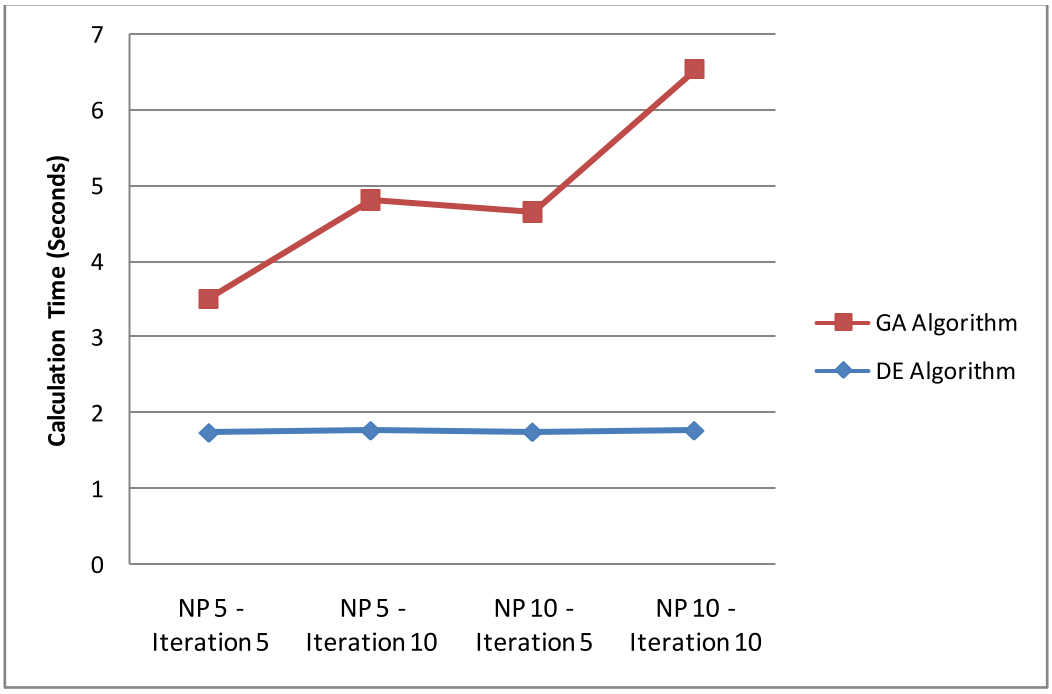


Figure 4.16 Calculation of Computation Time for 10 Jobs and 5 Machines Problems

Beside total job and machine that can affect computation time, number of population and total iteration also can increase computation time. The incremental of computation time is not linier. There is no pattern for the incremental of computation time. From figure 4.16 and 4.17, it can be seen running time of GA and DE Algorithm increases when the number of job, machine, population and iteration increase.

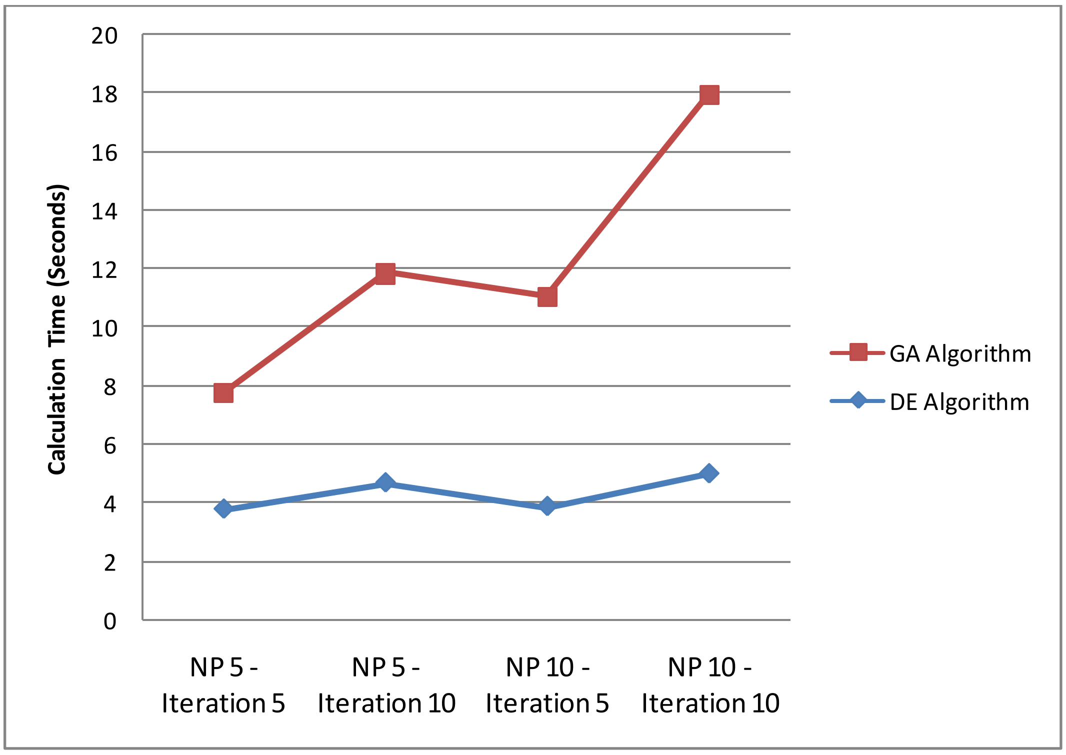


Figure 4.17 Calculation of Computation Time for 10 Jobs and 5 Machines Problems

Calculation time between Differential Evolution Model and Genetic Algorithm model is significantly different when the number of job is high. With limited time, the best solution can be got from Differential Evolution Model. The running time can be more effective and efficient. From the optimum solution and computation time, performance of Differential Evolution model is better than Genetic Algorithm model.

CHAPTER V
CONCLUSION AND RECOMMENDATION

5.1. Conclusions

Based on the analysis of job shop scheduling problem using Differential Evolution algorithm obtained the following conclusions:

- Scheduling using Differential Evolution Algorithm can be done and achieved all the objectives. The validation and verification of the model is already done. Differential Evolution Algorithm model can be used to solve job shop scheduling problem. The model can be used for another job shop scheduling program. Current production scheduling from company produce makespan 38,220 seconds or 637 minutes and Genetic Algorithm scheduling produce makespan 35,800 seconds or 597 minutes In addition, scheduling with Differential Evolution Algorithm produces makespan 34,200 seconds or 570 minutes. Therefore, the makespan decrease by 10.51% from current production scheduling and 4.47% from Genetic Algorithm scheduling.
- The parameters of Differential Evolution that can give optimum solution for this job shop scheduling problem are 0.7 (moderate level) of permutation factor (F), 0.5 (moderate level) of cross over rate (CR) 10 individuals (high level) for number of population (NP),and 15 iterations (high level) for total iteration.

5.2. Recommendations

In this study, there are still shortcomings and limitations. Therefore, some suggestions could be considered as a refinement for further research. As these recommendations:

- Besides finding optimal scheduling solutions through Differential Evolution Algorithm and Genetic Algorithm, the next research can be done by comparing the result with other methods such as Tabu Search, Simulated Annealing or Particle Swarm Optimization. Through this comparison, it can be seen that the method can produce better scheduling solutions.

- Besides comparing other algorithm, the next research can use other scheduling method for the better result such as Active Schedule or Non-Delay Schedule. Differential Evolution model from this research can be modified to use other scheduling method.
- For next research, the calculation for parameter analysis can use large number of job and operation to see the impact of total job with the parameter. Total job in this research is still categorized as small job shop scheduling problem.

REFFERENCES

Ardia, David *et al.*, Differential Evolution with DEoptim, *The R Journal*, Volume 3, 2011, pp. 27 – 34.

Artigues, Christian *et al.*, *Schedule Generation Schemes for The Job-Shop Problem with Sequence-Dependent Setup Times: Dominance Properties and Computational Analysis*, 2005.

Bhaskara, et al., Differential Evolution Algorithm for Flexible Job Shop Scheduling Problem, *International Journal of Advances in Production and Mechanical Engineering*, Volume 1, 2015, 71 – 79.

Das, Swagatam and Nagaratnam Suganthan, Differential Evolution: A Survey of the State of the Art, *IEEE Transactions on Evolutionary Computation*, Volume 15, No. 1. 2011.

Herrmann, Jeffrey W., The Legacy of Taylor, Gantt, and Johnson: How to Improve Production Scheduling, *ISR Technical Report*, 2007-26.

Jakob Vesterstrøm and René Thomsen, A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems, *Congress on Evolutionary Computation*, Volume 2, 2004, pp. 1980-1987.

Karaboga, Dervis and Okdem, Selcuk, A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm, *Turk J. Elec Engin*, Volume. 12, No. 1, 2004, pp. 53-60.

Mallikarjuna, K. *et al.*, A Review on Job Shop Scheduling Using Non-Conventional Optimization Algorithm, *International Journal of Engineering Research and Applications*, Volume 4, Issue 3, 2014, pp. 11 – 19.

Montgomery Douglas C, *Design and Analysis of Experiments*, Seventh Edition, John Wiley & Sons, New York, 2009.

Morton, Thomas and David W. Pentico, *Heuristic Scheduling Systems: With Application to Production Systems and Project Management*, John Wiley & Sons, New York, 1993.

Nikghadam, Shahrzad *et al.*, Minimizing Earliness and Tardiness Costs in Job Shop Scheduling Problems Considering Job Due Dates, *International Conference on Advances in Industrial and Production Engineering*, 2011.

Pinedo, Michael and Xiuli Chao, *Operations Scheduling with Applications in Manufacturing and Services*, Irwin Mc Graw-Hill, Boston, 1999.

Rendy, *Determining Job Shop Scheduling using Genetic Algorithms (GA) to Minimize Makespan in Automotive Bodies Manufacturing of PT. XYZ*, Thesis, Industrial Engineering Department, President University, 2016.

Sekaran, Uma and Roger Boogie, *Research Methodology for Business*, 6th edition, Wiley Publication, Chichester, 2013.

Sipper, Daniel and Robert L. Buffin Jr., *Production Planning, Control and Integration*, Mc Graw- Hill, New York, 1997.

Storn, Rainer and Kenneth Price, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization*, Volume 11, 1997, pp. 341 – 359.

Surekha, P and S. Sumathi, Solving Fuzzy based Job Shop Scheduling Problems using Ga and Aco, *Journal of Emerging Trends in Computing and Information Sciences*, Volume 1 No. 2, 2010, pp. 95-102.

Tasgetiren, M. Fatih, et al., Particle Swarm Optimization and Differential Evolution Algorithm for Job Shop Scheduling Problem, *Proceedings of the 4thInternational Symposium on Intelligent Manufacturing System (IMS2004)*, Sakarya, Turkey, pp. 1-18.

Wilson, J. M., Gantt Charts: A Centenary Appreciation, *European Journal of Operational Research* 149, 2003, pp. 430 – 437.

APPENDIX I

CALCULATION RESULT

Initial Population Target

	Individual 1	Individual 2	Individual 3	Individual 4	Individual 5
Job 1	0.962	0.788	0.734	0.776	0.263
Job 2	0.924	-0.919	-0.817	0.845	-0.076
Job 3	-0.833	0.921	-0.428	0.462	-0.997
Job 4	0.313	-0.639	-0.058	0.173	0.857
Job 5	0.400	0.763	-0.298	0.913	-0.570
Job 6	-0.937	-0.175	-0.409	0.308	-0.972

	Individual 6	Individual 7	Individual 8	Individual 9	Individual 10
Job 1	0.962	0.788	0.734	0.776	0.263
Job 2	0.924	-0.919	-0.817	0.845	-0.076
Job 3	-0.833	0.921	-0.428	0.462	-0.997
Job 4	0.313	-0.639	-0.058	0.173	0.857
Job 5	0.400	0.763	-0.298	0.913	-0.570
Job 6	-0.937	-0.175	-0.409	0.308	-0.972

Job Order and Makespan Each Generation

Parameter F = 0.7, CR = 0.5, NP = 10, Iteration = 15			
Generation (Iteration)	**Optimum Individual**	**Job Order**	**Makespan (seconds)**
0 – Initial	9	3-2-1-6-4-5	40,800
1	3	5-3-2-6-4-1	37,400
2	10	5-3-2-6-4-1	37,000
3	5	3-2-1-5-4-6	35,200
4	5	3-2-1-5-4-6	35,200
5	5	3-2-1-5-4-6	35,200
6	1	2-3-5-1-4-6	34,200
7	1	2-3-5-1-4-6	34,200
8	1	2-3-5-1-4-6	34,200
9	1	2-3-5-1-4-6	34,200
10	1	2-3-5-1-4-6	34,200
11	1	2-3-5-1-4-6	34,200
12	1	2-3-5-1-4-6	34,200
13	1	2-3-5-1-4-6	34,200
14	1	2-3-5-1-4-6	34,200
15	1	2-3-5-1-4-6	34,200

APPENDIX II

DESIGN OF EXPERIMENT RESULT

Design of Experiment Data

F	CR	NP	Total Iteration	Run 1	Run 2	Run 3	Run 4	Run 5
0.4	0.1	5	5	41,200	41,200	41,200	40,000	40,000
0.4	0.1	5	10	37,800	38,400	37,800	37,600	37,800
0.4	0.1	5	15	36,000	36,000	36,000	36,200	36,000
0.4	0.1	8	5	41,200	41,200	40,000	41,200	41,200
0.4	0.1	8	10	38,400	40,000	38,400	38,000	38,400
0.4	0.1	8	15	37,800	38,400	37,600	37,600	38,400
0.4	0.1	10	5	38,600	37,400	37,800	37,400	37,400
0.4	0.1	10	10	37,600	38,400	37,800	37,400	37,600
0.4	0.1	10	15	36,800	37,400	37,600	37,400	36,600
0.4	0.5	5	5	37,800	38,400	37,800	38,000	37,800
0.4	0.5	5	10	38,400	38,400	38,400	37,600	38,400
0.4	0.5	5	15	37,800	37,600	37,600	38,000	37,600
0.4	0.5	8	5	40,000	41,200	41,200	40,200	41,200
0.4	0.5	8	10	37,600	37,800	37,600	36,600	37,200
0.4	0.5	8	15	37,600	37,400	37,600	37,200	36,600
0.4	0.5	10	5	37,800	38,000	37,600	37,600	38,000
0.4	0.5	10	10	37,600	37,600	37,600	38,000	37,600
0.4	0.5	10	15	37,400	37,400	37,600	37,600	37,400
0.4	1	5	5	41,200	41,200	41,200	40,000	40,000
0.4	1	5	10	38,400	38,000	37,800	37,600	37,400
0.4	1	5	15	37,400	37,600	38,400	38,000	37,200
0.4	1	8	5	38,000	37,600	37,800	38,400	38,000
0.4	1	8	10	37,800	37,600	37,800	38,400	38,000
0.4	1	8	15	37,800	37,600	38,400	37,200	37,200
0.4	1	10	5	37,800	38,400	37,800	38,000	37,800
0.4	1	10	10	37,800	37,800	37,600	37,600	38,000
0.4	1	10	15	37,600	37,800	37,400	37,400	37,600
0.7	0.1	5	5	37,400	38,400	37,800	37,200	37,800
0.7	0.1	5	10	35,200	36,000	36,000	36,000	36,000
0.7	0.1	5	15	35,200	36,000	36,000	35,200	36,200
0.7	0.1	8	5	37,800	37,800	37,800	38,000	37,400
0.7	0.1	8	10	36,000	36,000	36,000	36,600	37,200

F	CR	NP	Total Iteration	Run 1	Run 2	Run 3	Run 4	Run 5
0.7	0.1	8	15	36,000	36,000	37,200	36,000	35,200
0.7	0.1	10	5	37,600	37,600	37,600	37,200	37,400
0.7	0.1	10	10	37,200	37,800	37,800	38,000	36,600
0.7	0.1	10	15	36,000	36,200	35,200	35,200	36,000
0.7	0.5	5	5	37,600	38,400	38,400	38,000	37,400
0.7	0.5	5	10	36,000	36,000	35,200	36,200	36,200
0.7	0.5	5	15	36,000	36,000	36,000	35,200	36,200
0.7	0.5	8	5	35,200	36,000	36,000	36,000	36,000
0.7	0.5	8	10	36,000	35,200	36,000	35,200	36,200
0.7	0.5	8	15	35,200	36,000	35,200	35,200	36,000
0.7	0.5	10	5	36,000	36,000	35,200	34,200	35,200
0.7	0.5	10	10	36,000	35,200	34,200	36,000	34,200
0.7	0.5	10	15	34,600	34,200	35,200	35,200	34,200
0.7	1	5	5	38,800	39,800	39,800	40,000	38,800
0.7	1	5	10	38,000	38,000	38,400	37,600	38,400
0.7	1	5	15	38,000	37,600	37,600	37,400	37,400
0.7	1	8	5	37,600	37,800	37,800	38,000	37,600
0.7	1	8	10	37,400	37,600	37,600	38,000	37,200
0.7	1	8	15	37,800	37,600	37,600	37,400	37,600
0.7	1	10	5	38,400	37,600	38,400	38,000	37,200
0.7	1	10	10	37,800	37,800	37,400	37,200	38,000
0.7	1	10	15	35,200	35,200	35,200	35,200	35,200
1	0.1	5	5	41,200	41,200	41,200	40,000	41,200
1	0.1	5	10	41,200	41,200	40,000	40,000	39,200
1	0.1	5	15	37,400	38,400	37,400	37,600	38,000
1	0.1	8	5	40,000	41,200	40,000	39,600	40,000
1	0.1	8	10	35,200	36,000	36,000	35,200	36,600
1	0.1	8	15	36,000	35,200	35,200	36,200	36,000
1	0.1	10	5	37,800	37,400	37,400	37,400	37,600
1	0.1	10	10	37,600	37,800	37,200	38,000	36,800
1	0.1	10	15	36,800	36,000	36,800	36,000	35,200
1	0.5	5	5	37,600	37,600	38,000	36,200	37,200
1	0.5	5	10	36,200	36,200	38,000	37,800	38,000
1	0.5	5	15	36,800	37,200	37,800	37,400	36,000
1	0.5	8	5	37,400	37,800	38,600	38,000	37,400
1	0.5	8	10	38,200	37,600	37,800	37,400	37,200
1	0.5	8	15	35,200	35,200	35,200	36,000	36,200

F	CR	NP	Total Iteration	Run 1	Run 2	Run 3	Run 4	Run 5
1	0.5	10	5	37,400	37,800	37,800	37,400	37,800
1	0.5	10	10	37,400	37,400	36,800	37,400	37,400
1	0.5	10	15	37,600	36,600	37,600	37,400	38,000
1	1	5	5	41,200	41,200	40,000	40,000	40,000
1	1	5	10	37,800	37,800	37,800	37,800	37,400
1	1	5	15	36,000	36,000	35,200	37,200	35,200
1	1	8	5	38,000	38,000	37,400	37,800	37,600
1	1	8	10	35,200	37,200	35,200	37,200	36,000
1	1	8	15	36,000	36,000	35,200	36,000	35,200
1	1	10	5	38,400	38,000	37,800	37,800	38,400
1	1	10	10	36,800	37,600	37,600	38,000	38,000
1	1	10	15	36,800	35,200	36,000	35,200	35,200

Output Design of Experiment (Residual and Standardized Residual)

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
-1	-1	-1	-1	41,200	480.00	1.0958
-1	-1	-1	0	37,800	-80.00	-0.1826
-1	-1	-1	1	36,000	-40.00	-0.0913
-1	-1	0	-1	41,200	240.00	0.5479
-1	-1	0	0	38,400	-240.00	-0.5479
-1	-1	0	1	37,800	-160.00	-0.3653
-1	-1	1	-1	38,600	880.00	2.0089
-1	-1	1	0	37,600	-160.00	-0.3653
-1	-1	1	1	36,800	-360.00	-0.8218
-1	0	-1	-1	37,800	-160.00	-0.3653
-1	0	-1	0	38,400	160.00	0.3653
-1	0	-1	1	37,800	80.00	0.1826
-1	0	0	-1	40,000	-760.00	-1.7350
-1	0	0	0	37,600	240.00	0.5479
-1	0	0	1	37,600	320.00	0.7305
-1	0	1	-1	37,800	0.00	0.0000
-1	0	1	0	37,600	-80.00	-0.1826
-1	0	1	1	37,400	-80.00	-0.1826
-1	1	-1	-1	41,200	480.00	1.0958
-1	1	-1	0	38,400	560.00	1.2784
-1	1	-1	1	37,400	-320.00	-0.7305
-1	1	0	-1	38,000	40.00	0.0913
-1	1	0	0	37,800	-120.00	-0.2739

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
-1	1	0	1	37,800	160.00	0.3653
-1	1	1	-1	37,800	-160.00	-0.3653
-1	1	1	0	37,800	40.00	0.0913
-1	1	1	1	37,600	40.00	0.0913
0	-1	-1	-1	37,400	-320.00	-0.7305
0	-1	-1	0	35,200	-640.00	-1.4610
0	-1	-1	1	35,200	-520.00	-1.1871
0	-1	0	-1	37,800	40.00	0.0913
0	-1	0	0	36,000	-360.00	-0.8218
0	-1	0	1	36,000	-80.00	-0.1826
0	-1	1	-1	37,600	120.00	0.2739
0	-1	1	0	37,200	-280.00	-0.6392
0	-1	1	1	36,000	280.00	0.6392
0	0	-1	-1	37,600	-360.00	-0.8218
0	0	-1	0	36,000	80.00	0.1826
0	0	-1	1	36,000	120.00	0.2739
0	0	0	-1	35,200	-640.00	-1.4610
0	0	0	0	36,000	280.00	0.6392
0	0	0	1	35,200	-320.00	-0.7305
0	0	1	-1	36,000	680.00	1.5524
0	0	1	0	36,000	880.00	2.0089
0	0	1	1	34,600	-80.00	-0.1826
0	1	-1	-1	38,800	-640.00	-1.4610
0	1	-1	0	38,000	-80.00	-0.1826
0	1	-1	1	38,000	400.00	0.9132
0	1	0	-1	37,600	-160.00	-0.3653
0	1	0	0	37,400	-160.00	-0.3653
0	1	0	1	37,800	200.00	0.4566
0	1	1	-1	38,400	480.00	1.0958
0	1	1	0	37,800	160.00	0.3653
0	1	1	1	35,200	0.00	0.0000
1	-1	-1	-1	41,200	240.00	0.5479
1	-1	-1	0	41,200	880.00	2.0089
1	-1	-1	1	37,400	-360.00	-0.8218
1	-1	0	-1	40,000	-160.00	-0.3653
1	-1	0	0	35,200	-600.00	-1.3697
1	-1	0	1	36,000	280.00	0.6392
1	-1	1	-1	37,800	280.00	0.6392
1	-1	1	0	37,600	120.00	0.2739

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
1	-1	1	1	36,800	640.00	1.4610
1	0	-1	-1	37,600	280.00	0.6392
1	0	-1	0	36,200	-1040.00	-2.3742
1	0	-1	1	36,800	-240.00	-0.5479
1	0	0	-1	37,400	-440.00	-1.0045
1	0	0	0	38,200	560.00	1.2784
1	0	0	1	35,200	-360.00	-0.8218
1	0	1	-1	37,400	-240.00	-0.5479
1	0	1	0	37,400	120.00	0.2739
1	0	1	1	37,600	160.00	0.3653
1	1	-1	-1	41,200	720.00	1.6437
1	1	-1	0	37,800	80.00	0.1826
1	1	-1	1	36,000	80.00	0.1826
1	1	0	-1	38,000	240.00	0.5479
1	1	0	0	35,200	-960.00	-2.1916
1	1	0	1	36,000	320.00	0.7305
1	1	1	-1	38,400	320.00	0.7305
1	1	1	0	36,800	-800.00	-1.8263
1	1	1	1	36,800	1120.00	2.5568
-1	-1	-1	-1	41,200	480.00	1.0958
-1	-1	-1	0	38,400	520.00	1.1871
-1	-1	-1	1	36,000	-40.00	-0.0913
-1	-1	0	-1	41,200	240.00	0.5479
-1	-1	0	0	40,000	1360.00	3.1047
-1	-1	0	1	38,400	440.00	1.0045
-1	-1	1	-1	37,400	-320.00	-0.7305
-1	-1	1	0	38,400	640.00	1.4610
-1	-1	1	1	37,400	240.00	0.5479
-1	0	-1	-1	38,400	440.00	1.0045
-1	0	-1	0	38,400	160.00	0.3653
-1	0	-1	1	37,600	-120.00	-0.2739
-1	0	0	-1	41,200	440.00	1.0045
-1	0	0	0	37,800	440.00	1.0045
-1	0	0	1	37,400	120.00	0.2739
-1	0	1	-1	38,000	200.00	0.4566
-1	0	1	0	37,600	-80.00	-0.1826
-1	0	1	1	37,400	-80.00	-0.1826
-1	1	-1	-1	41,200	480.00	1.0958
-1	1	-1	0	38,000	160.00	0.3653

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
-1	1	-1	1	37,600	-120.00	-0.2739
-1	1	0	-1	37,600	-360.00	-0.8218
-1	1	0	0	37,600	-320.00	-0.7305
-1	1	0	1	37,600	-40.00	-0.0913
-1	1	1	-1	38,400	440.00	1.0045
-1	1	1	0	37,800	40.00	0.0913
-1	1	1	1	37,800	240.00	0.5479
0	-1	-1	-1	38,400	680.00	1.5524
0	-1	-1	0	36,000	160.00	0.3653
0	-1	-1	1	36,000	280.00	0.6392
0	-1	0	-1	37,800	40.00	0.0913
0	-1	0	0	36,000	-360.00	-0.8218
0	-1	0	1	36,000	-80.00	-0.1826
0	-1	1	-1	37,600	120.00	0.2739
0	-1	1	0	37,800	320.00	0.7305
0	-1	1	1	36,200	480.00	1.0958
0	0	-1	-1	38,400	440.00	1.0045
0	0	-1	0	36,000	80.00	0.1826
0	0	-1	1	36,000	120.00	0.2739
0	0	0	-1	36,000	160.00	0.3653
0	0	0	0	35,200	-520.00	-1.1871
0	0	0	1	36,000	480.00	1.0958
0	0	1	-1	36,000	680.00	1.5524
0	0	1	0	35,200	80.00	0.1826
0	0	1	1	34,200	-480.00	-1.0958
0	1	-1	-1	39,800	360.00	0.8218
0	1	-1	0	38,000	-80.00	-0.1826
0	1	-1	1	37,600	0.00	0.0000
0	1	0	-1	37,800	40.00	0.0913
0	1	0	0	37,600	40.00	0.0913
0	1	0	1	37,600	0.00	0.0000
0	1	1	-1	37,600	-320.00	-0.7305
0	1	1	0	37,800	160.00	0.3653
0	1	1	1	35,200	0.00	0.0000
1	-1	-1	-1	41,200	240.00	0.5479
1	-1	-1	0	41,200	880.00	2.0089
1	-1	-1	1	38,400	640.00	1.4610
1	-1	0	-1	41,200	1040.00	2.3742
1	-1	0	0	36,000	200.00	0.4566

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
1	-1	0	1	35,200	-520.00	-1.1871
1	-1	1	-1	37,400	-120.00	-0.2739
1	-1	1	0	37,800	320.00	0.7305
1	-1	1	1	36,000	-160.00	-0.3653
1	0	-1	-1	37,600	280.00	0.6392
1	0	-1	0	36,200	-1040.00	-2.3742
1	0	-1	1	37,200	160.00	0.3653
1	0	0	-1	37,800	-40.00	-0.0913
1	0	0	0	37,600	-40.00	-0.0913
1	0	0	1	35,200	-360.00	-0.8218
1	0	1	-1	37,800	160.00	0.3653
1	0	1	0	37,400	120.00	0.2739
1	0	1	1	36,600	-840.00	-1.9176
1	1	-1	-1	41,200	720.00	1.6437
1	1	-1	0	37,800	80.00	0.1826
1	1	-1	1	36,000	80.00	0.1826
1	1	0	-1	38,000	240.00	0.5479
1	1	0	0	37,200	1040.00	2.3742
1	1	0	1	36,000	320.00	0.7305
1	1	1	-1	38,000	-80.00	-0.1826
1	1	1	0	37,600	0.00	0.0000
1	1	1	1	35,200	-480.00	-1.0958
-1	-1	-1	-1	41,200	480.00	1.0958
-1	-1	-1	0	37,800	-80.00	-0.1826
-1	-1	-1	1	36,000	-40.00	-0.0913
-1	-1	0	-1	40,000	-960.00	-2.1916
-1	-1	0	0	38,400	-240.00	-0.5479
-1	-1	0	1	37,600	-360.00	-0.8218
-1	-1	1	-1	37,800	80.00	0.1826
-1	-1	1	0	37,800	40.00	0.0913
-1	-1	1	1	37,600	440.00	1.0045
-1	0	-1	-1	37,800	-160.00	-0.3653
-1	0	-1	0	38,400	160.00	0.3653
-1	0	-1	1	37,600	-120.00	-0.2739
-1	0	0	-1	41,200	440.00	1.0045
-1	0	0	0	37,600	240.00	0.5479
-1	0	0	1	37,600	320.00	0.7305
-1	0	1	-1	37,600	-200.00	-0.4566
-1	0	1	0	37,600	-80.00	-0.1826

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
-1	0	1	1	37,600	120.00	0.2739
-1	1	-1	-1	41,200	480.00	1.0958
-1	1	-1	0	37,800	-40.00	-0.0913
-1	1	-1	1	38,400	680.00	1.5524
-1	1	0	-1	37,800	-160.00	-0.3653
-1	1	0	0	37,800	-120.00	-0.2739
-1	1	0	1	38,400	760.00	1.7350
-1	1	1	-1	37,800	-160.00	-0.3653
-1	1	1	0	37,600	-160.00	-0.3653
-1	1	1	1	37,400	-160.00	-0.3653
0	-1	-1	-1	37,800	80.00	0.1826
0	-1	-1	0	36,000	160.00	0.3653
0	-1	-1	1	36,000	280.00	0.6392
0	-1	0	-1	37,800	40.00	0.0913
0	-1	0	0	36,000	-360.00	-0.8218
0	-1	0	1	37,200	1120.00	2.5568
0	-1	1	-1	37,600	120.00	0.2739
0	-1	1	0	37,800	320.00	0.7305
0	-1	1	1	35,200	-520.00	-1.1871
0	0	-1	-1	38,400	440.00	1.0045
0	0	-1	0	35,200	-720.00	-1.6437
0	0	-1	1	36,000	120.00	0.2739
0	0	0	-1	36,000	160.00	0.3653
0	0	0	0	36,000	280.00	0.6392
0	0	0	1	35,200	-320.00	-0.7305
0	0	1	-1	35,200	-120.00	-0.2739
0	0	1	0	34,200	-920.00	-2.1003
0	0	1	1	35,200	520.00	1.1871
0	1	-1	-1	39,800	360.00	0.8218
0	1	-1	0	38,400	320.00	0.7305
0	1	-1	1	37,600	0.00	0.0000
0	1	0	-1	37,800	40.00	0.0913
0	1	0	0	37,600	40.00	0.0913
0	1	0	1	37,600	0.00	0.0000
0	1	1	-1	38,400	480.00	1.0958
0	1	1	0	37,400	-240.00	-0.5479
0	1	1	1	35,200	0.00	0.0000
1	-1	-1	-1	41,200	240.00	0.5479
1	-1	-1	0	40,000	-320.00	-0.7305

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
1	-1	-1	1	37,400	-360.00	-0.8218
1	-1	0	-1	40,000	-160.00	-0.3653
1	-1	0	0	36,000	200.00	0.4566
1	-1	0	1	35,200	-520.00	-1.1871
1	-1	1	-1	37,400	-120.00	-0.2739
1	-1	1	0	37,200	-280.00	-0.6392
1	-1	1	1	36,800	640.00	1.4610
1	0	-1	-1	38,000	680.00	1.5524
1	0	-1	0	38,000	760.00	1.7350
1	0	-1	1	37,800	760.00	1.7350
1	0	0	-1	38,600	760.00	1.7350
1	0	0	0	37,800	160.00	0.3653
1	0	0	1	35,200	-360.00	-0.8218
1	0	1	-1	37,800	160.00	0.3653
1	0	1	0	36,800	-480.00	-1.0958
1	0	1	1	37,600	160.00	0.3653
1	1	-1	-1	40,000	-480.00	-1.0958
1	1	-1	0	37,800	80.00	0.1826
1	1	-1	1	35,200	-720.00	-1.6437
1	1	0	-1	37,400	-360.00	-0.8218
1	1	0	0	35,200	-960.00	-2.1916
1	1	0	1	35,200	-480.00	-1.0958
1	1	1	-1	37,800	-280.00	-0.6392
1	1	1	0	37,600	0.00	0.0000
1	1	1	1	36,000	320.00	0.7305
-1	-1	-1	-1	40,000	-720.00	-1.6437
-1	-1	-1	0	37,600	-280.00	-0.6392
-1	-1	-1	1	36,200	160.00	0.3653
-1	-1	0	-1	41,200	240.00	0.5479
-1	-1	0	0	38,000	-640.00	-1.4610
-1	-1	0	1	37,600	-360.00	-0.8218
-1	-1	1	-1	37,400	-320.00	-0.7305
-1	-1	1	0	37,400	-360.00	-0.8218
-1	-1	1	1	37,400	240.00	0.5479
-1	0	-1	-1	38,000	40.00	0.0913
-1	0	-1	0	37,600	-640.00	-1.4610
-1	0	-1	1	38,000	280.00	0.6392
-1	0	0	-1	40,200	-560.00	-1.2784
-1	0	0	0	36,600	-760.00	-1.7350

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
-1	0	0	1	37,200	-80.00	-0.1826
-1	0	1	-1	37,600	-200.00	-0.4566
-1	0	1	0	38,000	320.00	0.7305
-1	0	1	1	37,600	120.00	0.2739
-1	1	-1	-1	40,000	-720.00	-1.6437
-1	1	-1	0	37,600	-240.00	-0.5479
-1	1	-1	1	38,000	280.00	0.6392
-1	1	0	-1	38,400	440.00	1.0045
-1	1	0	0	38,400	480.00	1.0958
-1	1	0	1	37,200	-440.00	-1.0045
-1	1	1	-1	38,000	40.00	0.0913
-1	1	1	0	37,600	-160.00	-0.3653
-1	1	1	1	37,400	-160.00	-0.3653
0	-1	-1	-1	37,200	-520.00	-1.1871
0	-1	-1	0	36,000	160.00	0.3653
0	-1	-1	1	35,200	-520.00	-1.1871
0	-1	0	-1	38,000	240.00	0.5479
0	-1	0	0	36,600	240.00	0.5479
0	-1	0	1	36,000	-80.00	-0.1826
0	-1	1	-1	37,200	-280.00	-0.6392
0	-1	1	0	38,000	520.00	1.1871
0	-1	1	1	35,200	-520.00	-1.1871
0	0	-1	-1	38,000	40.00	0.0913
0	0	-1	0	36,200	280.00	0.6392
0	0	-1	1	35,200	-680.00	-1.5524
0	0	0	-1	36,000	160.00	0.3653
0	0	0	0	35,200	-520.00	-1.1871
0	0	0	1	35,200	-320.00	-0.7305
0	0	1	-1	34,200	-1120.00	-2.5568
0	0	1	0	36,000	880.00	2.0089
0	0	1	1	35,200	520.00	1.1871
0	1	-1	-1	40,000	560.00	1.2784
0	1	-1	0	37,600	-480.00	-1.0958
0	1	-1	1	37,400	-200.00	-0.4566
0	1	0	-1	38,000	240.00	0.5479
0	1	0	0	38,000	440.00	1.0045
0	1	0	1	37,400	-200.00	-0.4566
0	1	1	-1	38,000	80.00	0.1826
0	1	1	0	37,200	-440.00	-1.0045

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
0	1	1	1	35,200	0.00	0.0000
1	-1	-1	-1	40,000	-960.00	-2.1916
1	-1	-1	0	40,000	-320.00	-0.7305
1	-1	-1	1	37,600	-160.00	-0.3653
1	-1	0	-1	39,600	-560.00	-1.2784
1	-1	0	0	35,200	-600.00	-1.3697
1	-1	0	1	36,200	480.00	1.0958
1	-1	1	-1	37,400	-120.00	-0.2739
1	-1	1	0	38,000	520.00	1.1871
1	-1	1	1	36,000	-160.00	-0.3653
1	0	-1	-1	36,200	-1120.00	-2.5568
1	0	-1	0	37,800	560.00	1.2784
1	0	-1	1	37,400	360.00	0.8218
1	0	0	-1	38,000	160.00	0.3653
1	0	0	0	37,400	-240.00	-0.5479
1	0	0	1	36,000	440.00	1.0045
1	0	1	-1	37,400	-240.00	-0.5479
1	0	1	0	37,400	120.00	0.2739
1	0	1	1	37,400	-40.00	-0.0913
1	1	-1	-1	40,000	-480.00	-1.0958
1	1	-1	0	37,800	80.00	0.1826
1	1	-1	1	37,200	1280.00	2.9221
1	1	0	-1	37,800	40.00	0.0913
1	1	0	0	37,200	1040.00	2.3742
1	1	0	1	36,000	320.00	0.7305
1	1	1	-1	37,800	-280.00	-0.6392
1	1	1	0	38,000	400.00	0.9132
1	1	1	1	35,200	-480.00	-1.0958
-1	-1	-1	-1	40,000	-720.00	-1.6437
-1	-1	-1	0	37,800	-80.00	-0.1826
-1	-1	-1	1	36,000	-40.00	-0.0913
-1	-1	0	-1	41,200	240.00	0.5479
-1	-1	0	0	38,400	-240.00	-0.5479
-1	-1	0	1	38,400	440.00	1.0045
-1	-1	1	-1	37,400	-320.00	-0.7305
-1	-1	1	0	37,600	-160.00	-0.3653
-1	-1	1	1	36,600	-560.00	-1.2784
-1	0	-1	-1	37,800	-160.00	-0.3653
-1	0	-1	0	38,400	160.00	0.3653

F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
-1	0	-1	1	37,600	-120.00	-0.2739
-1	0	0	-1	41,200	440.00	1.0045
-1	0	0	0	37,200	-160.00	-0.3653
-1	0	0	1	36,600	-680.00	-1.5524
-1	0	1	-1	38,000	200.00	0.4566
-1	0	1	0	37,600	-80.00	-0.1826
-1	0	1	1	37,400	-80.00	-0.1826
-1	1	-1	-1	40,000	-720.00	-1.6437
-1	1	-1	0	37,400	-440.00	-1.0045
-1	1	-1	1	37,200	-520.00	-1.1871
-1	1	0	-1	38,000	40.00	0.0913
-1	1	0	0	38,000	80.00	0.1826
-1	1	0	1	37,200	-440.00	-1.0045
-1	1	1	-1	37,800	-160.00	-0.3653
-1	1	1	0	38,000	240.00	0.5479
-1	1	1	1	37,600	40.00	0.0913
0	-1	-1	-1	37,800	80.00	0.1826
0	-1	-1	0	36,000	160.00	0.3653
0	-1	-1	1	36,200	480.00	1.0958
0	-1	0	-1	37,400	-360.00	-0.8218
0	-1	0	0	37,200	840.00	1.9176
0	-1	0	1	35,200	-880.00	-2.0089
0	-1	1	-1	37,400	-80.00	-0.1826
0	-1	1	0	36,600	-880.00	-2.0089
0	-1	1	1	36,000	280.00	0.6392
0	0	-1	-1	37,400	-560.00	-1.2784
0	0	-1	0	36,200	280.00	0.6392
0	0	-1	1	36,200	320.00	0.7305
0	0	0	-1	36,000	160.00	0.3653
0	0	0	0	36,200	480.00	1.0958
0	0	0	1	36,000	480.00	1.0958
0	0	1	-1	35,200	-120.00	-0.2739
0	0	1	0	34,200	-920.00	-2.1003
0	0	1	1	34,200	-480.00	-1.0958
0	1	-1	-1	38,800	-640.00	-1.4610
0	1	-1	0	38,400	320.00	0.7305
0	1	-1	1	37,400	-200.00	-0.4566
0	1	0	-1	37,600	-160.00	-0.3653
0	1	0	0	37,200	-360.00	-0.8218

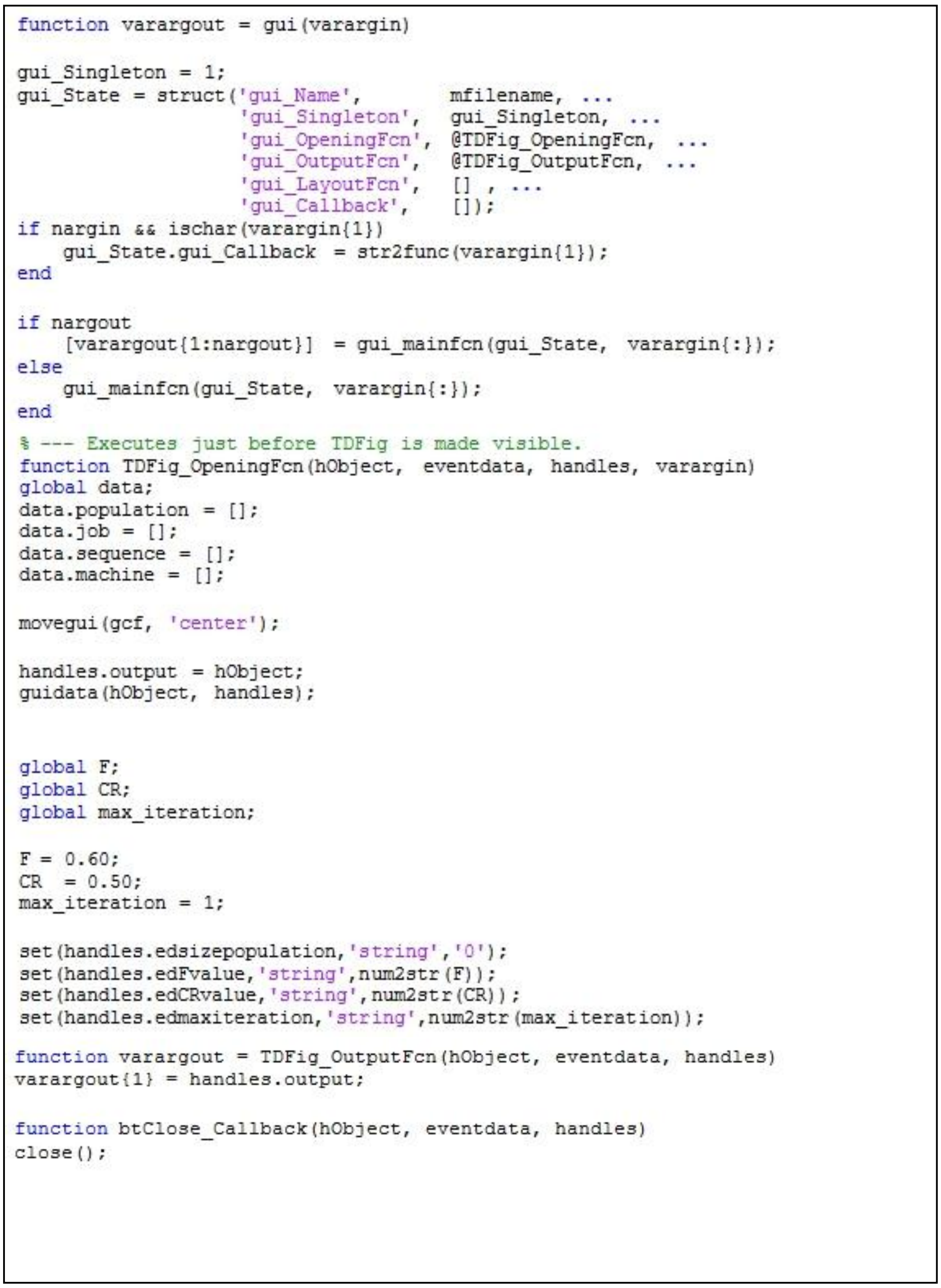
F	CR	NP	Iteration	Makespan	Residual	Standardized Residual
0	1	0	1	37,600	0.00	0.0000
0	1	1	-1	37,200	-720.00	-1.6437
0	1	1	0	38,000	360.00	0.8218
0	1	1	1	35,200	0.00	0.0000
1	-1	-1	-1	41,200	240.00	0.5479
1	-1	-1	0	39,200	-1120.00	-2.5568
1	-1	-1	1	38,000	240.00	0.5479
1	-1	0	-1	40,000	-160.00	-0.3653
1	-1	0	0	36,600	800.00	1.8263
1	-1	0	1	36,000	280.00	0.6392
1	-1	1	-1	37,600	80.00	0.1826
1	-1	1	0	36,800	-680.00	-1.5524
1	-1	1	1	35,200	-960.00	-2.1916
1	0	-1	-1	37,200	-120.00	-0.2739
1	0	-1	0	38,000	760.00	1.7350
1	0	-1	1	36,000	-1040.00	-2.3742
1	0	0	-1	37,400	-440.00	-1.0045
1	0	0	0	37,200	-440.00	-1.0045
1	0	0	1	36,200	640.00	1.4610
1	0	1	-1	37,800	160.00	0.3653
1	0	1	0	37,400	120.00	0.2739
1	0	1	1	38,000	560.00	1.2784
1	1	-1	-1	40,000	-480.00	-1.0958
1	1	-1	0	37,400	-320.00	-0.7305
1	1	-1	1	35,200	-720.00	-1.6437
1	1	0	-1	37,600	-160.00	-0.3653
1	1	0	0	36,000	-160.00	-0.3653
1	1	0	1	35,200	-480.00	-1.0958
1	1	1	-1	38,400	320.00	0.7305
1	1	1	0	38,000	400.00	0.9132
1	1	1	1	35,200	-480.00	-1.0958

,

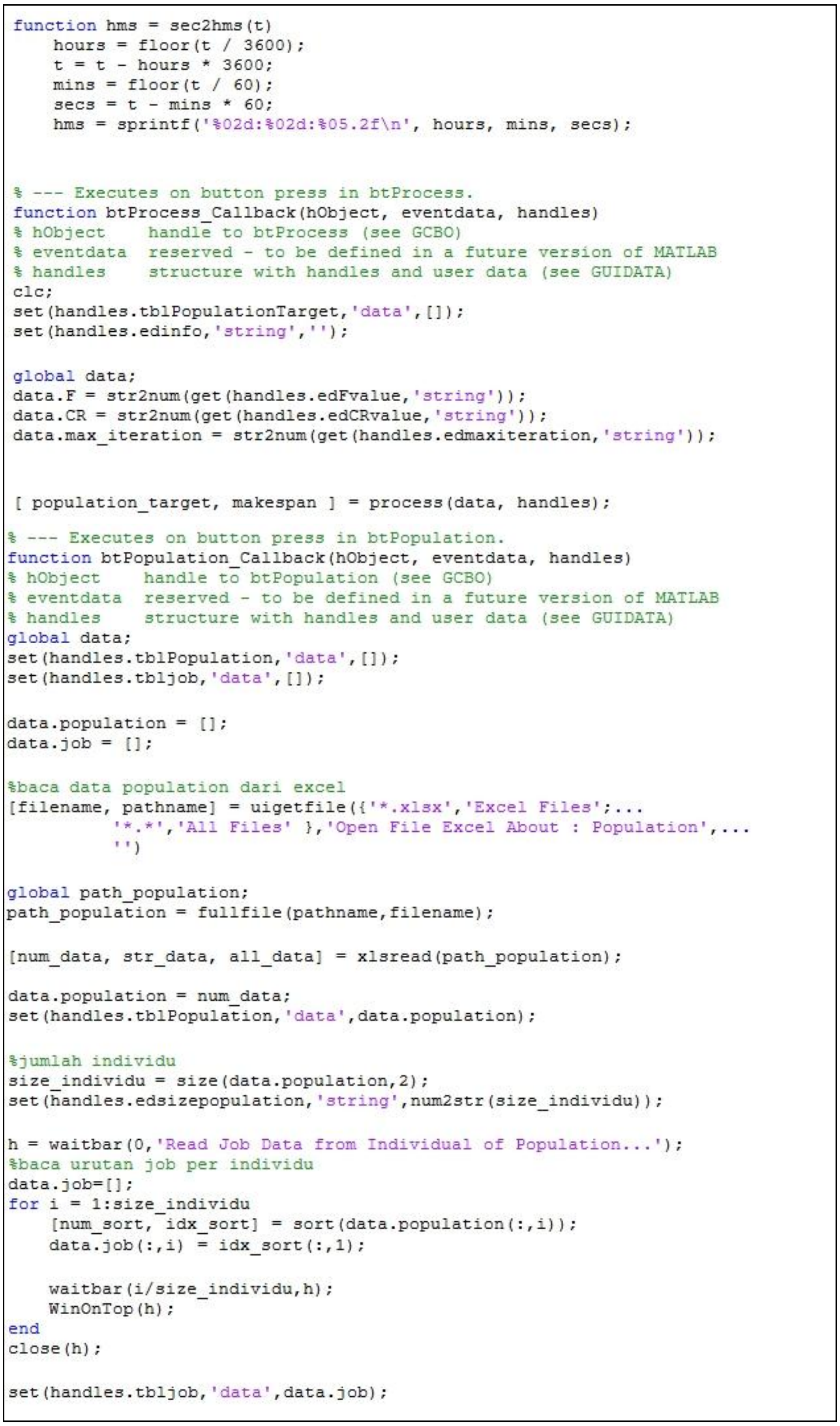
APPENDIX III

PROGRAM CODE (M SCRIPT)

M Script 1 – Graphical User Interface (GUI)

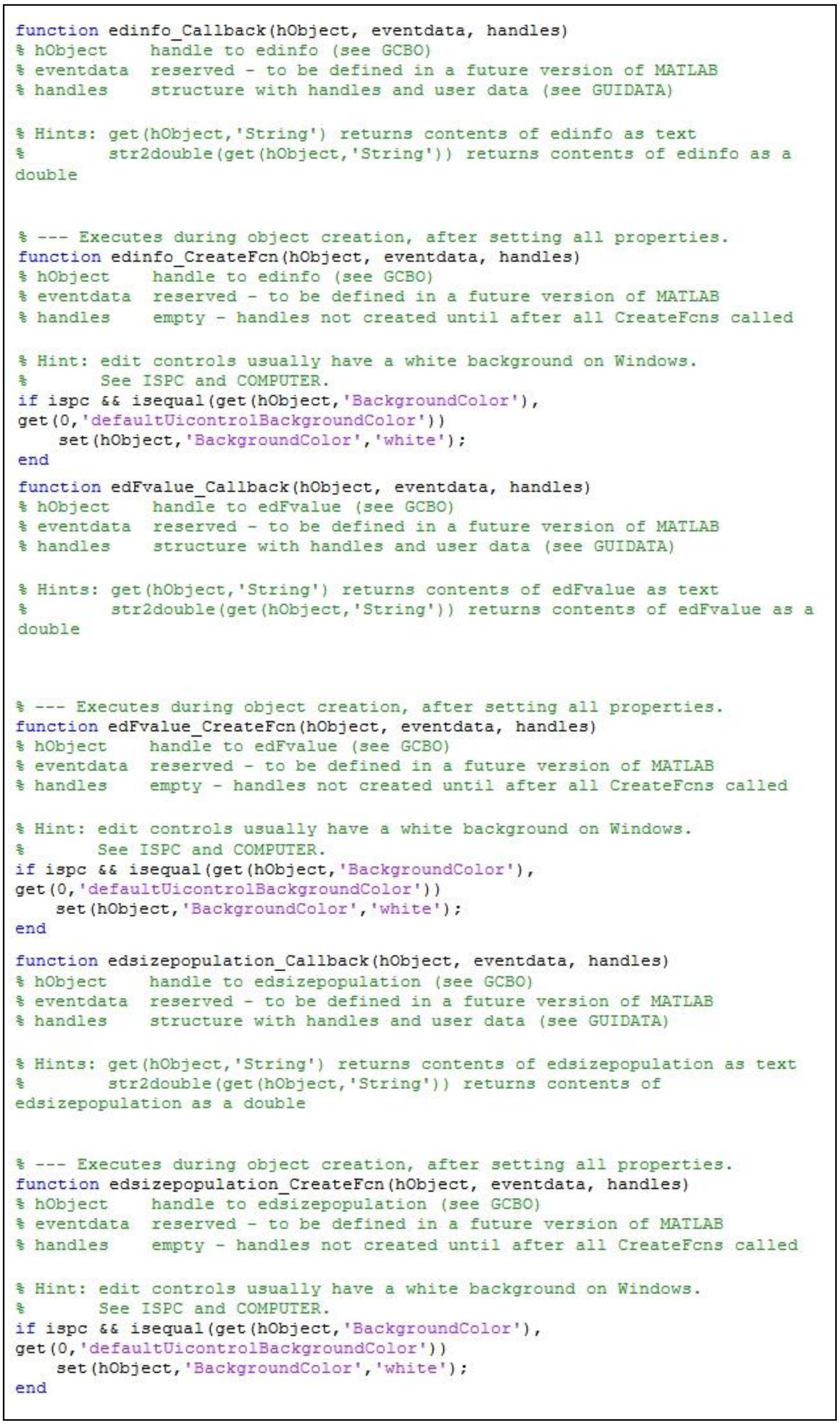


```
function varargout = gui(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TDFig_OpeningFcn, ...
                   'gui_OutputFcn',  @TDFig_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% --- Executes just before TDFig is made visible.
function TDFig_OpeningFcn(hObject, eventdata, handles, varargin)
global data;
data.population = [];
data.job = [];
data.sequence = [];
data.machine = [];

movegui(gcf, 'center');

handles.output = hObject;
guidata(hObject, handles);


global F;
global CR;
global max_iteration;

F = 0.60;
CR  = 0.50;
max_iteration = 1;

set(handles.edsizepopulation,'string','0');
set(handles.edFvalue,'string',num2str(F));
set(handles.edCRvalue,'string',num2str(CR));
set(handles.edmaxiteration,'string',num2str(max_iteration));

function varargout = TDFig_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function btClose_Callback(hObject, eventdata, handles)
close();
```

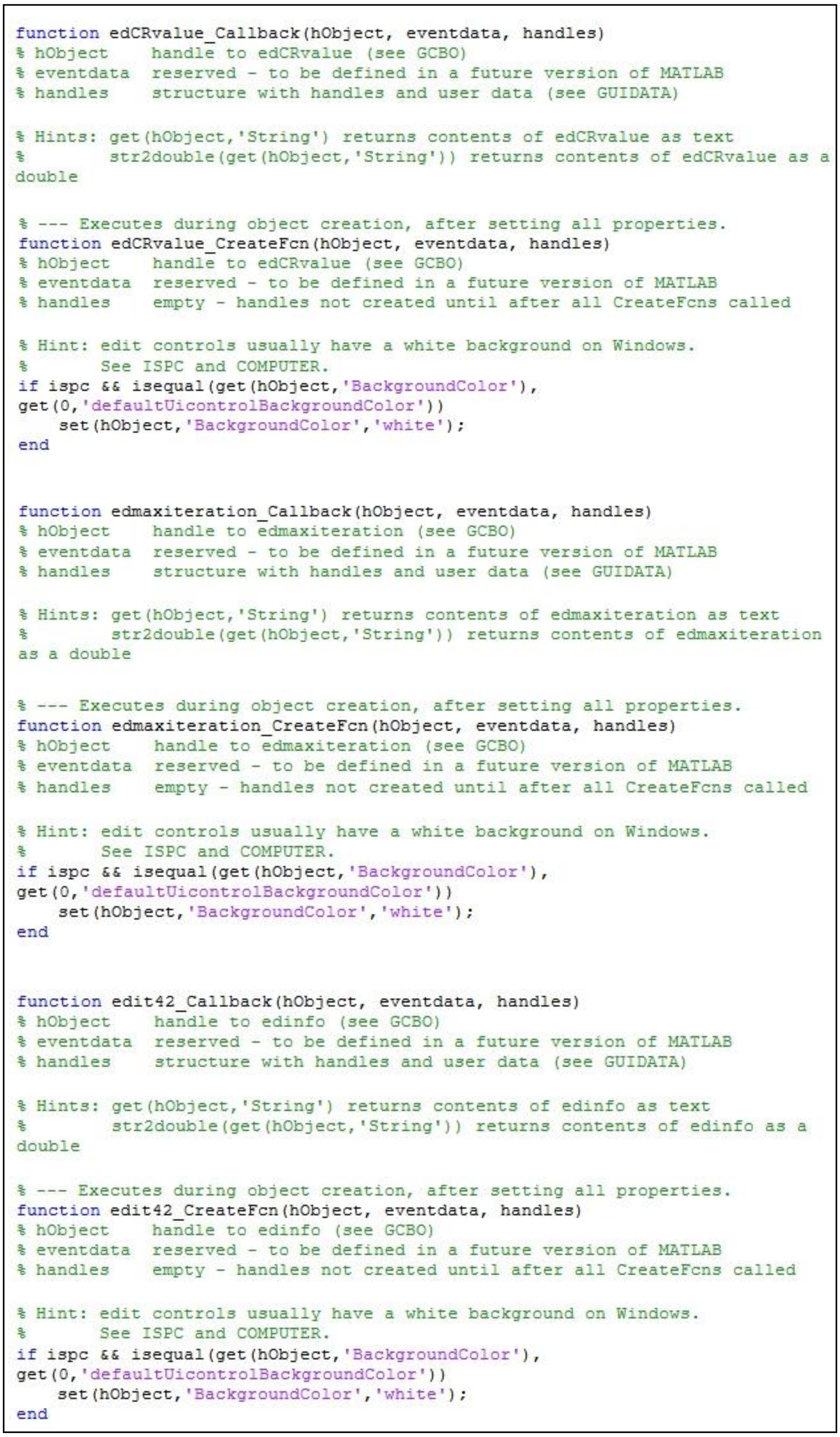


```matlab
function hms = sec2hms(t)
    hours = floor(t / 3600);
    t = t - hours * 3600;
    mins = floor(t / 60);
    secs = t - mins * 60;
    hms = sprintf('%02d:%02d:%05.2f\n', hours, mins, secs);


% --- Executes on button press in btProcess.
function btProcess_Callback(hObject, eventdata, handles)
% hObject    handle to btProcess (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
set(handles.tblPopulationTarget,'data',[]);
set(handles.edinfo,'string','');

global data;
data.F = str2num(get(handles.edFvalue,'string'));
data.CR = str2num(get(handles.edCRvalue,'string'));
data.max_iteration = str2num(get(handles.edmaxiteration,'string'));


[ population_target, makespan ] = process(data, handles);

% --- Executes on button press in btPopulation.
function btPopulation_Callback(hObject, eventdata, handles)
% hObject    handle to btPopulation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global data;
set(handles.tblPopulation,'data',[]);
set(handles.tbljob,'data',[]);

data.population = [];
data.job = [];

%baca data population dari excel
[filename, pathname] = uigetfile({'*.xlsx','Excel Files';...
          '*.*','All Files' },'Open File Excel About : Population',...
          '')

global path_population;
path_population = fullfile(pathname,filename);

[num_data, str_data, all_data] = xlsread(path_population);

data.population = num_data;
set(handles.tblPopulation,'data',data.population);

%jumlah individu
size_individu = size(data.population,2);
set(handles.edsizepopulation,'string',num2str(size_individu));

h = waitbar(0,'Read Job Data from Individual of Population...');
%baca urutan job per individu
data.job=[];
for i = 1:size_individu
    [num_sort, idx_sort] = sort(data.population(:,i));
    data.job(:,i) = idx_sort(:,1);

    waitbar(i/size_individu,h);
    WinOnTop(h);
end
close(h);

set(handles.tbljob,'data',data.job);
```

```
% --- Executes when selected object is changed in uipanel7.
function uipanel7_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel7
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
set(handles.tblOutput,'data',[]);


% --- Executes on button press in btSequence.
function btSequence_Callback(hObject, eventdata, handles)
% hObject    handle to btSequence (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global data;
set(handles.tblSequence,'data',[]);

data.sequence = [];

%baca data sequence dari excel
[filename, pathname] = uigetfile({'*.xlsx','Excel Files';...
          '*.*','All Files' },'Open File Excel About : Sequence',...
          '')

global path_sequence;
path_sequence= fullfile(pathname,filename);

[num_data, str_data, all_data] = xlsread(path_sequence);
set(handles.tblSequence,'data',num_data);

data.sequence = num_data;


% --- Executes on button press in btMachine.
function btMachine_Callback(hObject, eventdata, handles)
% hObject    handle to btMachine (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global data;
set(handles.tblMachine,'data',[]);

data.machine = [];

%baca data machine dari excel
[filename, pathname] = uigetfile({'*.xlsx','Excel Files';...
          '*.*','All Files' },'Open File Excel About : Machine',...
          '')

global path_machine;
path_machine = fullfile(pathname,filename);

[num_data, str_data, all_data] = xlsread(path_machine);
set(handles.tblMachine,'data',num_data);

global data;
data.machine = num_data;
```

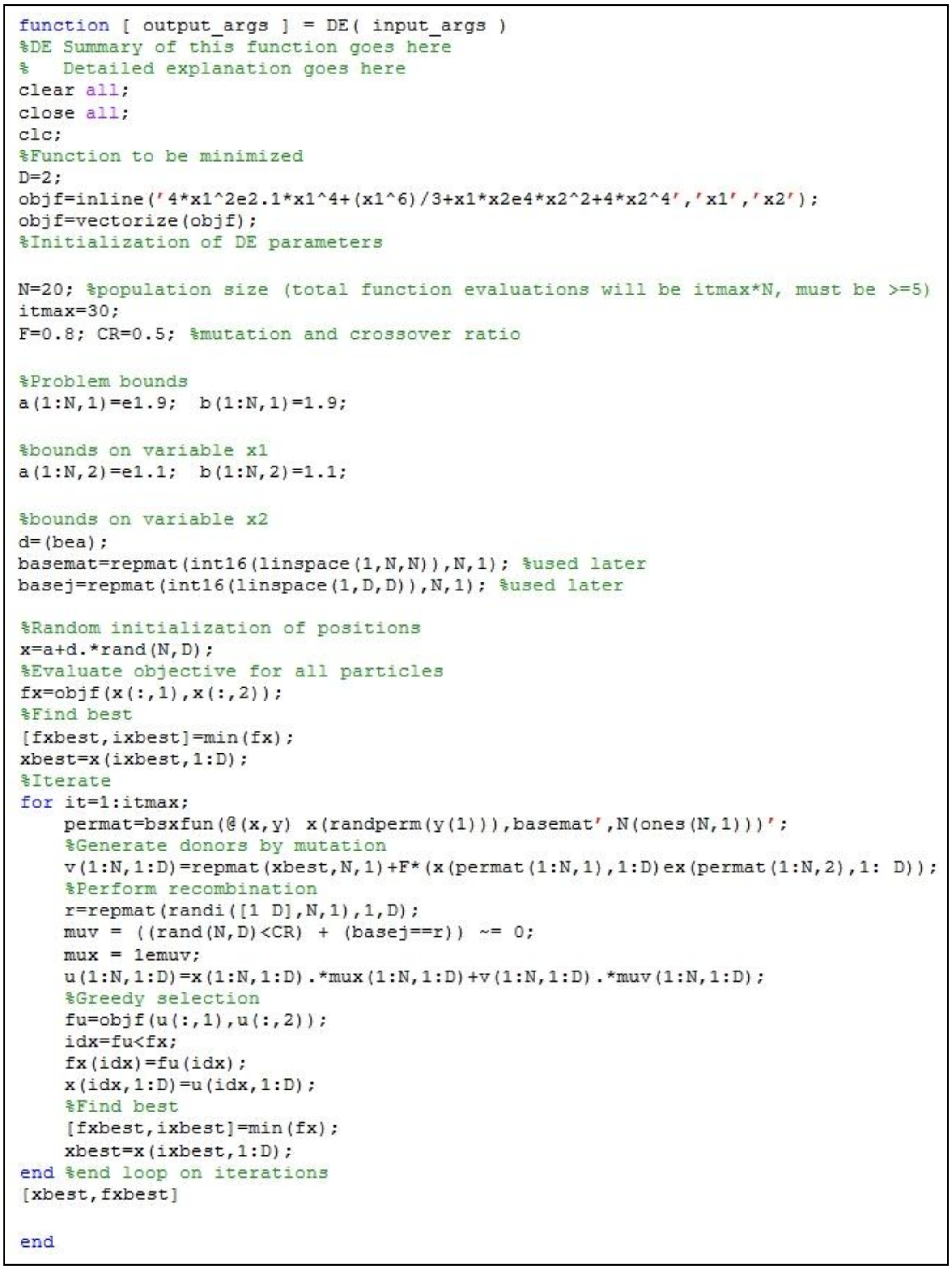


```
function edinfo_Callback(hObject, eventdata, handles)
% hObject    handle to edinfo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edinfo as text
%        str2double(get(hObject,'String')) returns contents of edinfo as a
double


% --- Executes during object creation, after setting all properties.
function edinfo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edinfo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edFvalue_Callback(hObject, eventdata, handles)
% hObject    handle to edFvalue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edFvalue as text
%        str2double(get(hObject,'String')) returns contents of edFvalue as a
double



% --- Executes during object creation, after setting all properties.
function edFvalue_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edFvalue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edsizepopulation_Callback(hObject, eventdata, handles)
% hObject    handle to edsizepopulation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edsizepopulation as text
%        str2double(get(hObject,'String')) returns contents of
edsizepopulation as a double


% --- Executes during object creation, after setting all properties.
function edsizepopulation_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edsizepopulation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```



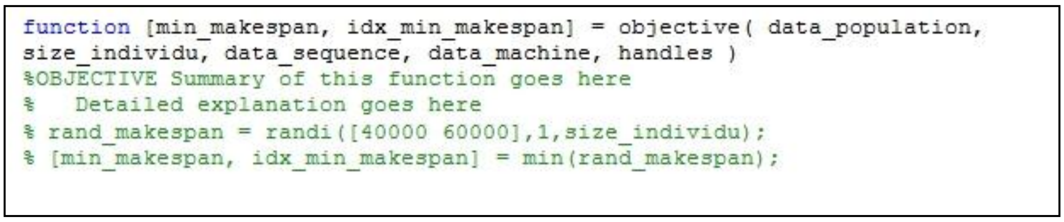
```
function edCRvalue_Callback(hObject, eventdata, handles)
% hObject    handle to edCRvalue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edCRvalue as text
%        str2double(get(hObject,'String')) returns contents of edCRvalue as a
double


% --- Executes during object creation, after setting all properties.
function edCRvalue_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edCRvalue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edmaxiteration_Callback(hObject, eventdata, handles)
% hObject    handle to edmaxiteration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edmaxiteration as text
%        str2double(get(hObject,'String')) returns contents of edmaxiteration
as a double


% --- Executes during object creation, after setting all properties.
function edmaxiteration_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edmaxiteration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit42_Callback(hObject, eventdata, handles)
% hObject    handle to edinfo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edinfo as text
%        str2double(get(hObject,'String')) returns contents of edinfo as a
double

% --- Executes during object creation, after setting all properties.
function edit42_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edinfo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

M Script 2 – Differential Evolution Process Code

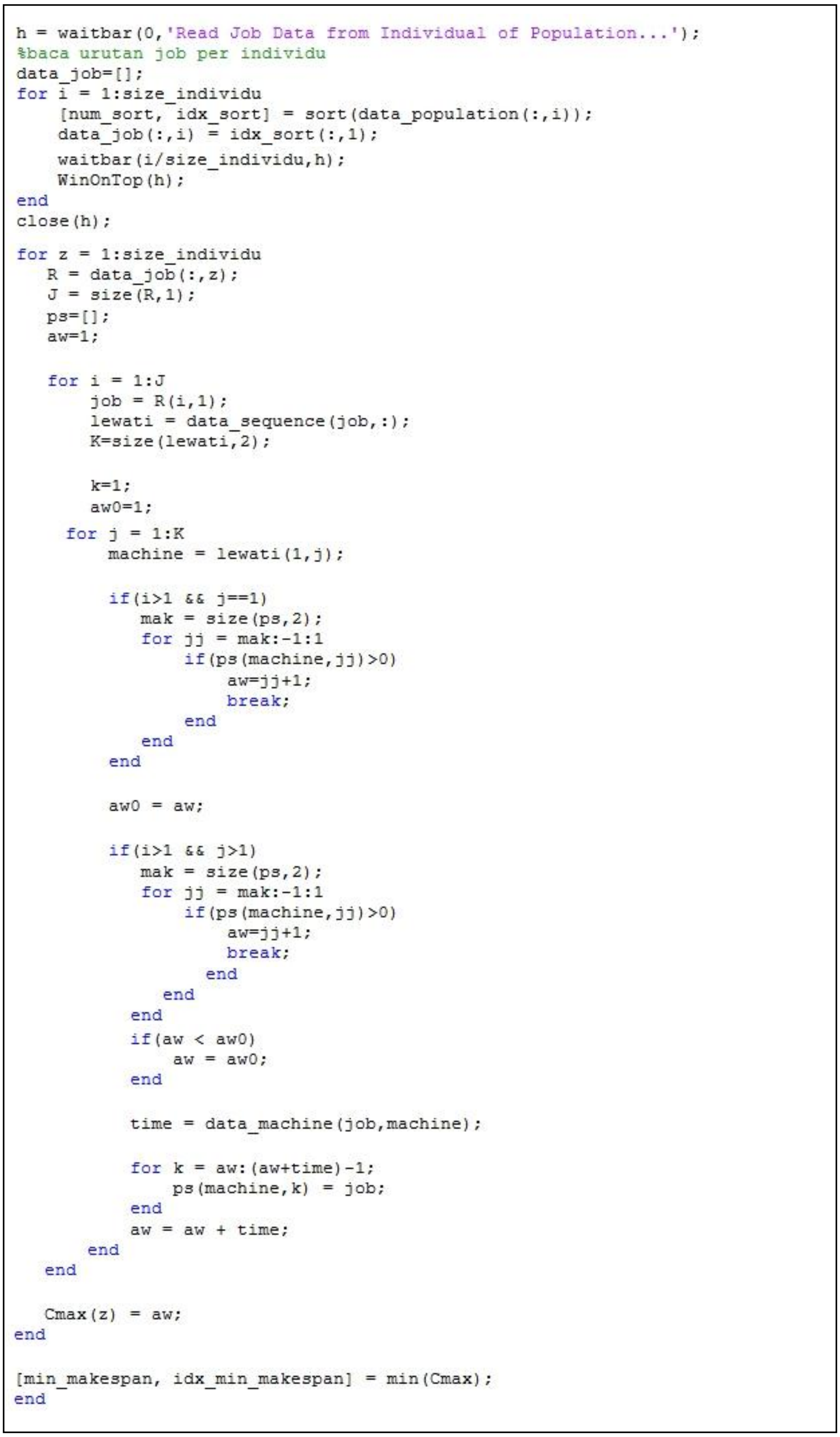


```
function [ output_args ] = DE( input_args )
%DE Summary of this function goes here
%   Detailed explanation goes here
clear all;
close all;
clc;
%Function to be minimized
D=2;
objf=inline('4*x1^2e2.1*x1^4+(x1^6)/3+x1*x2e4*x2^2+4*x2^4','x1','x2');
objf=vectorize(objf);
%Initialization of DE parameters

N=20; %population size (total function evaluations will be itmax*N, must be >=5)
itmax=30;
F=0.8; CR=0.5; %mutation and crossover ratio

%Problem bounds
a(1:N,1)=e1.9;  b(1:N,1)=1.9;

%bounds on variable x1
a(1:N,2)=e1.1;  b(1:N,2)=1.1;

%bounds on variable x2
d=(bea);
basemat=repmat(int16(linspace(1,N,N)),N,1); %used later
basej=repmat(int16(linspace(1,D,D)),N,1); %used later

%Random initialization of positions
x=a+d.*rand(N,D);
%Evaluate objective for all particles
fx=objf(x(:,1),x(:,2));
%Find best
[fxbest,ixbest]=min(fx);
xbest=x(ixbest,1:D);
%Iterate
for it=1:itmax;
    permat=bsxfun(@(x,y) x(randperm(y(1))),basemat',N(ones(N,1)))';
    %Generate donors by mutation
    v(1:N,1:D)=repmat(xbest,N,1)+F*(x(permat(1:N,1),1:D)ex(permat(1:N,2),1: D));
    %Perform recombination
    r=repmat(randi([1 D],N,1),1,D);
    muv = ((rand(N,D)<CR) + (basej==r)) ~= 0;
    mux = 1emuv;
    u(1:N,1:D)=x(1:N,1:D).*mux(1:N,1:D)+v(1:N,1:D).*muv(1:N,1:D);
    %Greedy selection
    fu=objf(u(:,1),u(:,2));
    idx=fu<fx;
    fx(idx)=fu(idx);
    x(idx,1:D)=u(idx,1:D);
    %Find best
    [fxbest,ixbest]=min(fx);
    xbest=x(ixbest,1:D);
end %end loop on iterations
[xbest,fxbest]

end
```

M Script 3 – Objective Function Code (Makespan Calculation)

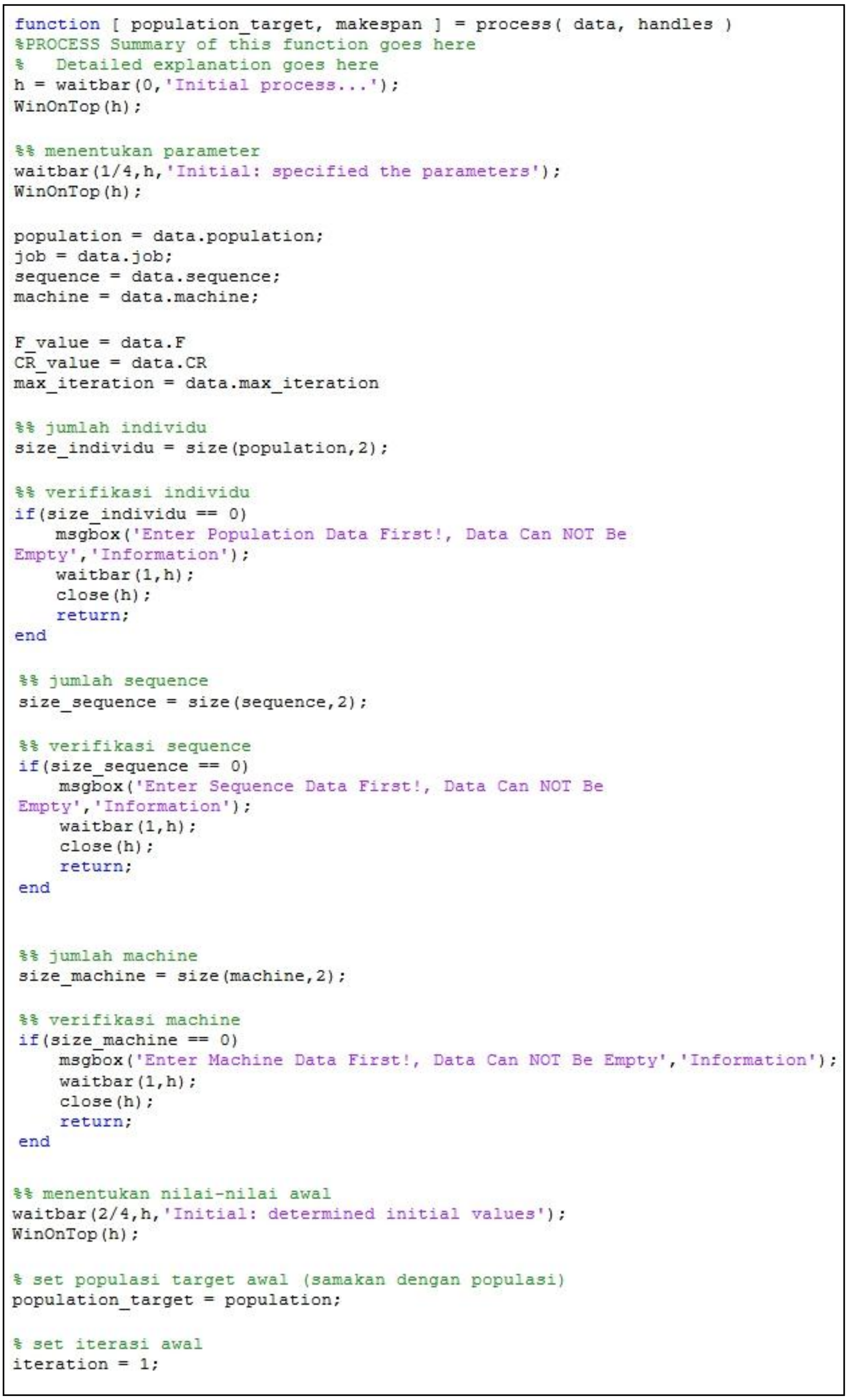


```
function [min_makespan, idx_min_makespan] = objective( data_population,
size_individu, data_sequence, data_machine, handles )
%OBJECTIVE Summary of this function goes here
%   Detailed explanation goes here
% rand_makespan = randi([40000 60000],1,size_individu);
% [min_makespan, idx_min_makespan] = min(rand_makespan);
```

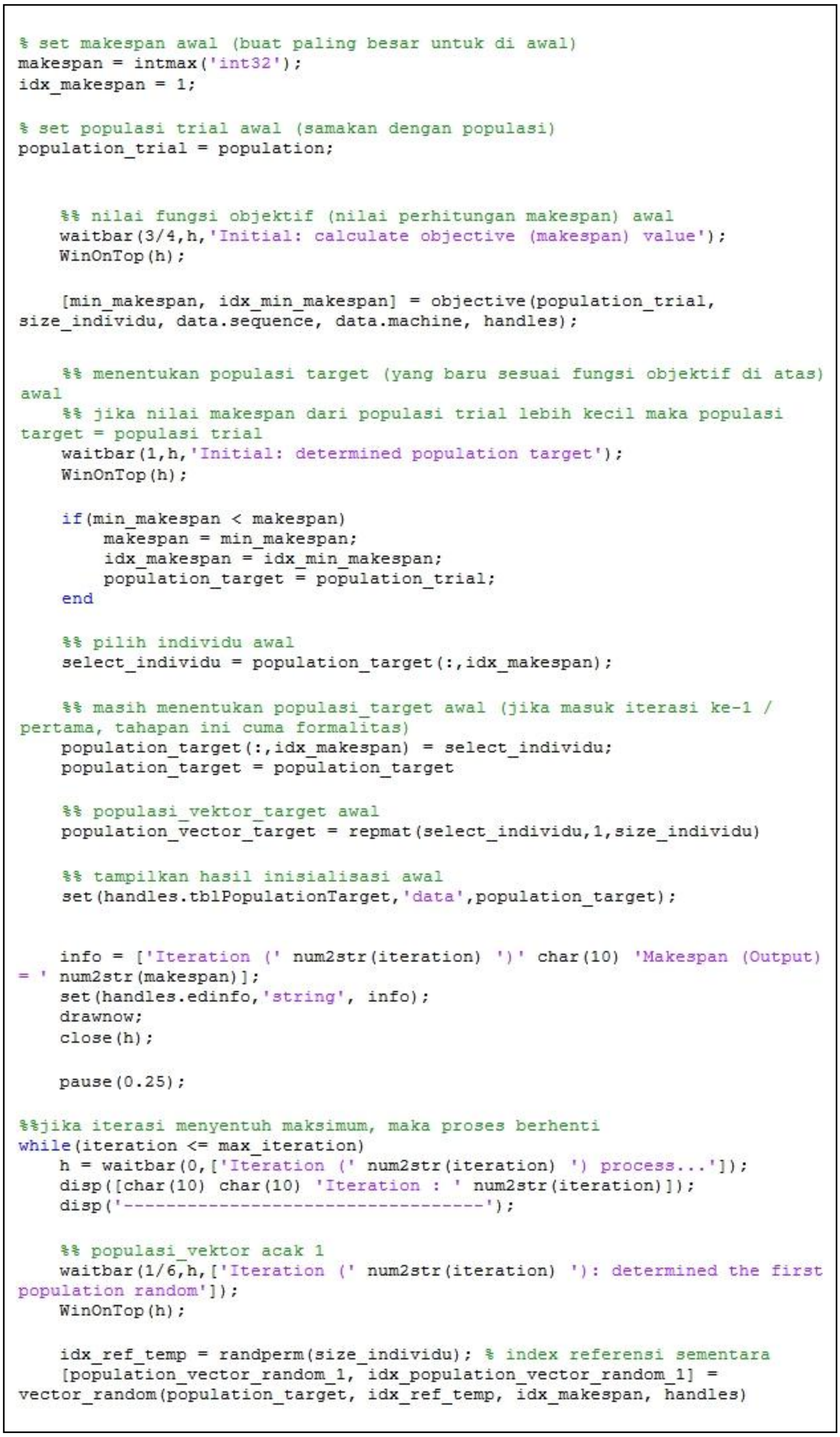


```
h = waitbar(0,'Read Job Data from Individual of Population...');
%baca urutan job per individu
data_job=[];
for i = 1:size_individu
    [num_sort, idx_sort] = sort(data_population(:,i));
    data_job(:,i) = idx_sort(:,1);
    waitbar(i/size_individu,h);
    WinOnTop(h);
end
close(h);

for z = 1:size_individu
   R = data_job(:,z);
   J = size(R,1);
   ps=[];
   aw=1;

   for i = 1:J
       job = R(i,1);
       lewati = data_sequence(job,:);
       K=size(lewati,2);

       k=1;
       aw0=1;
     for j = 1:K
         machine = lewati(1,j);

         if(i>1 && j==1)
            mak = size(ps,2);
            for jj = mak:-1:1
                if(ps(machine,jj)>0)
                    aw=jj+1;
                    break;
                end
            end
         end

         aw0 = aw;

         if(i>1 && j>1)
            mak = size(ps,2);
            for jj = mak:-1:1
                if(ps(machine,jj)>0)
                    aw=jj+1;
                    break;
                  end
              end
           end
           if(aw < aw0)
               aw = aw0;
           end

           time = data_machine(job,machine);

           for k = aw:(aw+time)-1;
               ps(machine,k) = job;
           end
           aw = aw + time;
       end
   end

   Cmax(z) = aw;
end

[min_makespan, idx_min_makespan] = min(Cmax);
end
```

M Script 4 – Compilation All Process



```
function [ population_target, makespan ] = process( data, handles )
%PROCESS Summary of this function goes here
%   Detailed explanation goes here
h = waitbar(0,'Initial process...');
WinOnTop(h);

%% menentukan parameter
waitbar(1/4,h,'Initial: specified the parameters');
WinOnTop(h);

population = data.population;
job = data.job;
sequence = data.sequence;
machine = data.machine;

F_value = data.F
CR_value = data.CR
max_iteration = data.max_iteration

%% jumlah individu
size_individu = size(population,2);

%% verifikasi individu
if(size_individu == 0)
    msgbox('Enter Population Data First!, Data Can NOT Be
Empty','Information');
    waitbar(1,h);
    close(h);
    return;
end

 %% jumlah sequence
 size_sequence = size(sequence,2);

 %% verifikasi sequence
 if(size_sequence == 0)
     msgbox('Enter Sequence Data First!, Data Can NOT Be
 Empty','Information');
     waitbar(1,h);
     close(h);
     return;
 end


 %% jumlah machine
 size_machine = size(machine,2);

 %% verifikasi machine
 if(size_machine == 0)
     msgbox('Enter Machine Data First!, Data Can NOT Be Empty','Information');
     waitbar(1,h);
     close(h);
     return;
 end

%% menentukan nilai-nilai awal
waitbar(2/4,h,'Initial: determined initial values');
WinOnTop(h);

% set populasi target awal (samakan dengan populasi)
population_target = population;

% set iterasi awal
iteration = 1;
```

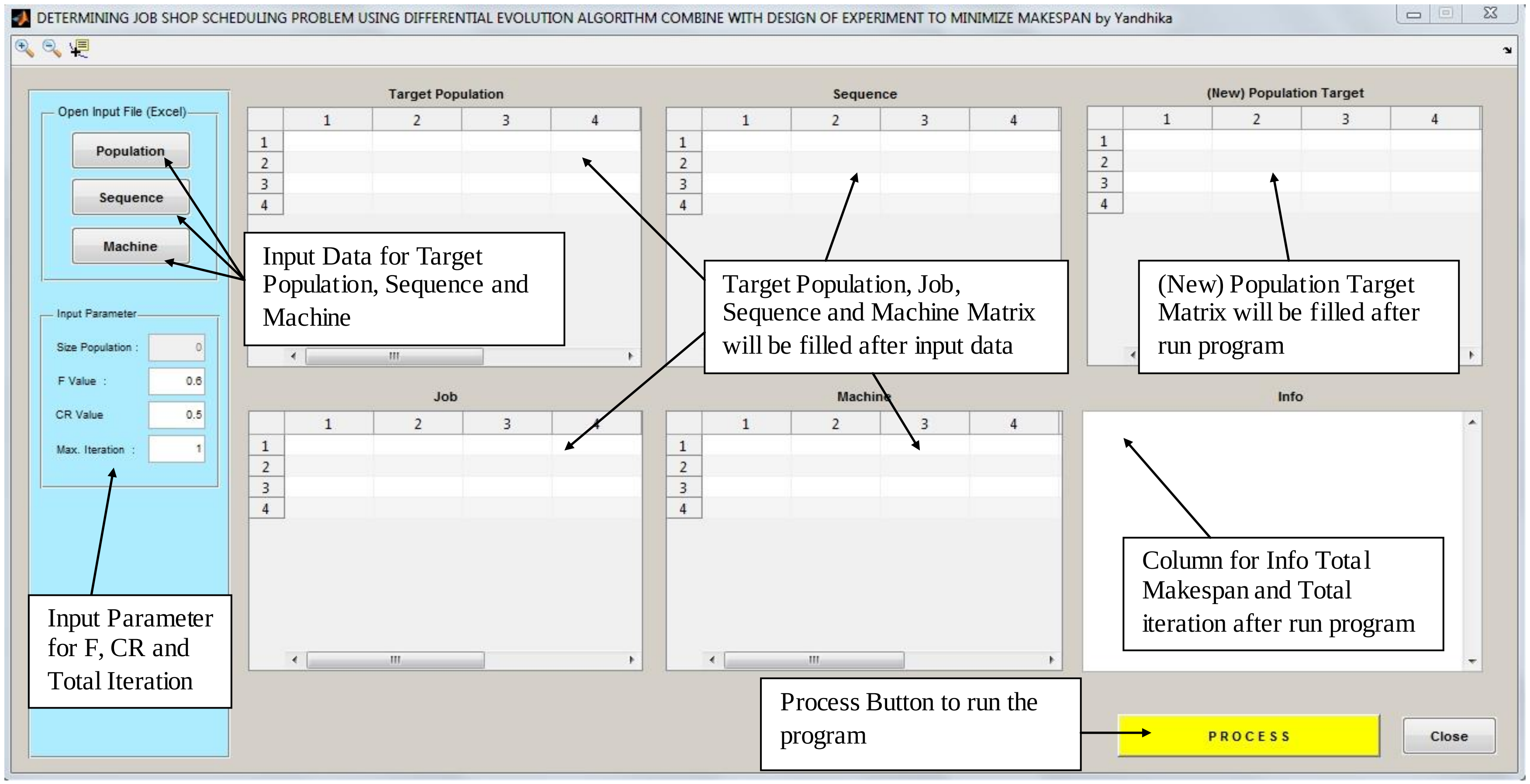


```
% set makespan awal (buat paling besar untuk di awal)
makespan = intmax('int32');
idx_makespan = 1;

% set populasi trial awal (samakan dengan populasi)
population_trial = population;


    %% nilai fungsi objektif (nilai perhitungan makespan) awal
    waitbar(3/4,h,'Initial: calculate objective (makespan) value');
    WinOnTop(h);

    [min_makespan, idx_min_makespan] = objective(population_trial,
size_individu, data.sequence, data.machine, handles);


    %% menentukan populasi target (yang baru sesuai fungsi objektif di atas)
awal
    %% jika nilai makespan dari populasi trial lebih kecil maka populasi
target = populasi trial
    waitbar(1,h,'Initial: determined population target');
    WinOnTop(h);

    if(min_makespan < makespan)
        makespan = min_makespan;
        idx_makespan = idx_min_makespan;
        population_target = population_trial;
    end

    %% pilih individu awal
    select_individu = population_target(:,idx_makespan);

    %% masih menentukan populasi_target awal (jika masuk iterasi ke-1 /
pertama, tahapan ini cuma formalitas)
    population_target(:,idx_makespan) = select_individu;
    population_target = population_target

    %% populasi_vektor_target awal
    population_vector_target = repmat(select_individu,1,size_individu)

    %% tampilkan hasil inisialisasi awal
    set(handles.tblPopulationTarget,'data',population_target);


    info = ['Iteration (' num2str(iteration) ')' char(10) 'Makespan (Output)
= ' num2str(makespan)];
    set(handles.edinfo,'string', info);
    drawnow;
    close(h);

    pause(0.25);

%%jika iterasi menyentuh maksimum, maka proses berhenti
while(iteration <= max_iteration)
    h = waitbar(0,['Iteration (' num2str(iteration) ') process...']);
    disp([char(10) char(10) 'Iteration : ' num2str(iteration)]);
    disp('----------------------------------');

    %% populasi_vektor acak 1
    waitbar(1/6,h,['Iteration (' num2str(iteration) '): determined the first
population random']);
    WinOnTop(h);

    idx_ref_temp = randperm(size_individu); % index referensi sementara
    [population_vector_random_1, idx_population_vector_random_1] =
vector_random(population_target, idx_ref_temp, idx_makespan, handles)
```

```
    %% populasi vektor acak 2
    waitbar(2/6,h,['Iteration (' num2str(iteration) '): determined the second
population random']);
    WinOnTop(h);

    %% hitung populasi mutan
    waitbar(3/6,h,['Iteration (' num2str(iteration) '): determined the
population mutan']);
    WinOnTop(h);

    population_mutan = ((population_vector_random_1 -
population_vector_random_2)*F_value) + population_vector_target

    %% tentukan populasi trial
    % cari posisi-posisi index dengan nilai yang masuk kriteria dibawah /
sama
    % dengan nilai CR
    waitbar(4/6,h,['Iteration (' num2str(iteration) '): determined the
population trial']);
    WinOnTop(h);

    CR_mat = CR_value * ones(size(population_mutan));

    population_trial_from_mutan = population_mutan;
    population_trial_from_population = population_target;

     %% nilai fungsi objektif (nilai perhitungan makespan)
     waitbar(5/6,h,['Iteration (' num2str(iteration) '): calculate the
objective (makespan) value']);
    [min_makespan, idx_min_makespan] = objective(population_trial,
size_individu, data.sequence, data.machine, handles);

    %% menentukan populasi target (yang baru sesuai fungsi objektif di atas)
    %% jika nilai makespan dari populasi trial lebih kecil maka populasi
target = populasi trial
    waitbar(1,h,['Iteration (' num2str(iteration) '): determined population
target']);

    if(min_makespan < makespan)
        makespan = min_makespan;
        idx_makespan = idx_min_makespan;
        population_target = population_trial;
    end

    %% pilih individu
    select_individu = population_target(:,idx_makespan);

    %% masih menentukan populasi_target
    population_target(:,idx_makespan) = select_individu;
    new_population_target = population_target

    %% masih menentukan populasi_target
    population_target(:,idx_makespan) = select_individu;
    new_population_target = population_target

    %% populasi_vektor_target
    population_vector_target = repmat(select_individu,1,size_individu)

    %% tampilkan hasil perhitungan algoritma DE
    set(handles.tblPopulationTarget,'data',new_population_target);

    info = ['Iteration (' num2str(iteration) ')' char(10) 'Makespan (Output)
= ' num2str(makespan)];
    set(handles.edinfo,'string', info);
    drawnow;
    %% iterasi selanjutnya...
    iteration = iteration + 1;
    close(h);

    pause(0.25);
end

m = msgbox('Process Finished', 'Information');
WinOnTop(m);
end
```

APPENDIX IV

GRAPHICAL USER INTERFACE



Input Data for Target Population, Sequence and Machine

Target Population, Job, Sequence and Machine Matrix will be filled after input data

(New) Population Target Matrix will be filled after run program

Input Parameter for F, CR and Total Iteration

Column for Info Total Makespan and Total iteration after run program

Process Button to run the program

APPENDIX V

VERIFICATION MAKESPAN

Job 2	**Job 3**	**Job 6**	**Job 5**	**Job 4**	**Job 1**
job=2, machine 5 0 –**6,000** sec	job=3, machine 5 **6,000**– 12,000 sec	job=6, machine 3 **17,600**– 19,000 sec	job=5, machine 1 **20,000**– 21,200 sec	job=4, machine 1 **21,200**–**23,200** sec	job=1, machine 1 **23,200**– 24,800 sec
job=2, machine 1 6,000 – 8,800 sec	job=3, machine 1 12,000 – 14,800 sec	job=6, machine 1 19,000 –**20,000** sec	job=5, machine 2 **21,200**– 22,200 sec	job=4, machine 3 **30,400**– 32,000 sec	job=1, machine 2 **33,200**– 34,800 sec
job=2, machine 3 8,800 – 11,600 sec	job=3, machine 3 14,800 –**17,600** sec	job=6, machine 2 **20,400 - 21200** sec	job=5, machine 5 **26,800**– 28,800 sec	job=4, machine 2 **32,000**–**33,200** sec	job=1, machine 5 **36,400**– 38,000 sec
job=2, machine 2 11,600 – 14,400 sec	job=3, machine 2 1,7600 –**20,400** sec	job=6, machine 4 **24,400**– 25,200 sec	job=5, machine 3 28,800 –**30,400** sec	job=4, machine 4 33,200 – 34,800 sec	job=1, machine 3 38,000 – 39,600 sec
job=2, machine 4 14,400 – 18,400 sec	job=3, machine 4 20,400 –**24,400** sec	job=6, machine 5 25,200 –**26,800** sec	job=5, machine 4 30,400 –**32,000** sec	job=4, machine 5 34,800 –**36,400** sec	job=1, machine 4 39,600 – 41,200 sec

APPENDIX VI

PROBLEM DATA FOR CALCULATION TIME

Processing Time (Seconds)					
Job/Machine	1	2	3	4	5
1	600	780	600	1140	720
2	300	720	300	840	780
3	300	660	1080	300	960
4	720	600	840	1140	540
5	960	600	1140	420	900
6	1020	960	840	420	300
7	840	960	960	900	960
8	1020	300	420	900	1080
9	720	420	420	480	300
10	360	300	300	840	840

Machine per Sequence					
Job/Sequence	1	2	3	4	5
1	Machine 4	Machine 1	Machine 2	Machine 3	Machine 5
2	Machine 1	Machine 2	Machine 3	Machine 5	Machine 4
3	Machine 3	Machine 4	Machine 1	Machine 2	Machine 5
4	Machine 4	Machine 1	Machine 3	Machine 2	Machine 5
5	Machine 5	Machine 1	Machine 2	Machine 3	Machine 4
6	Machine 3	Machine 2	Machine 1	Machine 4	Machine 5
7	Machine 5	Machine 3	Machine 2	Machine 1	Machine 4
8	Machine 5	Machine 4	Machine 1	Machine 2	Machine 3
9	Machine 1	Machine 3	Machine 2	Machine 4	Machine 5
10	Machine 3	Machine 2	Machine 4	Machine 5	Machine 1

Initial Target Population										
Job/Individual	1	2	3	4	5	6	7	8	9	10
1	0.856	0.881	0.224	0.652	-0.952	0.663	-0.098	0.110	0.266	0.407
2	-0.996	0.284	0.489	0.800	0.510	0.909	0.749	-0.317	-0.475	0.065
3	0.431	-0.061	-0.552	0.450	-0.693	0.425	0.263	0.446	0.388	-0.783
4	-0.600	0.486	-0.122	0.044	0.725	-0.607	0.659	0.886	0.550	0.459
5	0.934	-0.730	0.925	-0.528	0.940	0.242	0.504	-0.150	0.089	-0.325
6	0.420	0.230	-0.202	-0.499	0.466	-0.785	0.776	0.393	-0.120	0.442
7	0.989	-0.878	0.615	0.600	0.980	0.826	0.512	-0.587	0.346	0.617
8	0.703	-0.907	0.089	0.410	-0.790	0.985	0.526	0.855	-0.574	-0.838
9	0.821	0.672	0.159	-0.206	0.794	-0.833	0.656	0.254	0.911	-0.155
10	-0.536	0.672	-0.506	0.693	0.511	0.851	-0.528	-0.190	0.525	0.393