APPLICATION OF GENETIC ALGORITHM METHODS TO OPTIMIZE FLOWSHOP SEQUENCING PROBLEM

MOHD FADIL BIN MD SAIRI

UNIVERSITI MALAYSIA PAHANG

APPLICATION OF GENETIC ALGORITHM METHODS TO OPTIMIZE FLOWSHOP SEQUENCING PROBLEM

MOHD FADIL BIN MD SAIRI

A report submitted in partial fulfilment of the requirements
for the award of the degree of
Bachelor of Mechanical Engineering

Faculty of Mechanical Engineering
UNIVERSITI MALAYSIA PAHANG

NOVEMBER 2008

SUPERVISOR'S DECLARATION

We hereby declare that we have checked this project and in our opinion this project is satisfactory in terms of scope and quality for the award of the degree of Bachelor of Mechanical Engineering.

Signature.....

Name of Supervisor: Ms Noraini Bte Mohd Razali
Position: Lecturer
Date: .....

Signature.....
Name of Panel: Dr. Thet Thet Mon
Position: Lecturer
Date: .....

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.

Signature.....

Name: Mohd Fadil B Md Sairi

ID Number: MA05029

Date: .....

To my beloved Father and Mother

Md Sairi Bin Minhad@Menhad
Suchiah Bte Sathi

ACKNOWLEDGEMENT

First of all I am grateful to ALLAH S.W.T for blessing me in my final year project (PSM) with success in achieving my objectives to complete this project.

Secondly I want to thank to all my family members for supporting me and giving moral support in completing my project and also throughout my study in UMP or formerly known as KUKTEM. I also would like to thank my supervisor Pn. Noraini Bt. Razali and En. Fadzil Faisae Bin AB Rashid for guiding and supervising me started for my final year project 1 and final year project 2. They have been very helpful in helping me to finish my final year project. I am really appreciate every single advises they give in correcting my mistake. I am also thanks them for willing to help me with my final year project. The credit also goes to Mr. CH Chong for giving me chance to collect data from his company Nexus Electronic Snd. Bhb. and give me new experience on how the real company was running. With all the cooperation I obtain from Nexus Electronic Snd. Bhd. I could finish my final tear project.

Last but not least I want to thank my entire friend that keep on supporting me and encouraging me in completing my project. A tone of thanks I wish to all and may Allah bless you.

ABSTRACT

Application of genetic algorithm method to optimize flow shop sequencing problem as the title of this project is the study about the method used in order to optimize flow shop sequencing problem. Genetic algorithm method was one of the methods that were widely used in solving optimization problem. Genetic algorithm method is methods that follow the natural concept. Flow shop sequencing problem or also known as assembly line problem that normally faced by production company. This project will define the application of genetic algorithm method in solving flow shop sequencing problem in details and evaluate the strength and weakness of genetic algorithm method in order to optimize the optimization problem. This project will focusing on the method used to solve an optimization problem, the limitation of the method used and the results of solving flow shop sequencing problem using genetic algorithm method. At the end of this project, we can see the performance of genetic algorithm method in solving flow shop sequencing problem and types of flow shop sequencing problems that can be solve through genetic algorithm method. Limitation of genetic algorithm method also can be shown at the end of this project.

ABSTRAK

Penggunaan kaedah ginetik algoritma dalam mencari nilai optimum bagi masalah garis pemasangan sebagai tajuk projek ini adalah kajian tentang cara yang digunakan untuk mencari jawapan yang terbaik bagi masalah bahagian pemasangan. Kaedah ginetik algoritma adalah salah satu cara yang teleh digunakan secara meluas untuk menyelesaikan masalah bagi mendapatkan nilai yang optimum. Kaedah ginetik algoritma merupakan cara penyelesaian mengikut keadaan semulajadi. Masalah bahagian pemasangan pula selalunya dihadapi oleh syarikat-syarikat pembuatan. Projek ini akan menjelaskan berkenaan aplikasi ginetik algoritma dalam menyelesaikan masalah bahagian pemasangan secara terperinci dan mengkaji kekuatan serta kelemahan ginetik algoritma dalam mencari jawapan yang optimum. Secara keseluruhannya projek ini akan membincangkan dengan lebih mendalam berkenaan prestasi ginetik algoritma dalam menyelesaikan masalah bahagian pemasangan. Projek ini akan memfokuskan kepada cara yang digunakan untuk mendapatkan nilai optimum, mencari kelemahan kaedah ginetik algoritma dalam mendapatkan nilai optimum dan hasil penyelesaian masalah bahagian pemasangan. Pada pengakhiran projek ini, kita akan dapat melihat prestasi kaedah ginetik algoritma dalam mencari nilai optimum bagi masalah mendapatkan nilai optimum dalam bahagian pemasangan dan jenis-jenis masalah bahagian pemasangan yang dapat diselesakan menggunakanginetik algoritma. Kelemahan-kelemahan kaedah ginetik algoritma juga dapat dilihat pada pengakhiran projek ini.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

m	Number of machine
n	Number of job
F	Flow shop
c_{max}	Maximum flow time

LIST OF ABBREVIATIONS

GA	Genetic Algorithm
FSP	Flow Shop Sequencing Problem
TSP	Travelling Salesman Problem
CPU	Central processing unit
RAM	Random-access memory
OQC	Outgoing quality control
QC	Quality control

CHAPTER 1

INTRODUCTION

1.1 RESEARCH BACKGROUND

A flow shop is characterized by unidirectional flow of work with a variety of jobs being processed sequentially in a one –pass manner. A job shop, in the other hand, involves processing of several machines without any series structure. In the past 40 years extensive research has been done on both flow shop and job shop problems. A job shop is thus a collection of operation to be performed on an item or unit with relevent technological constraints. Often the operation must be done on all jobs in the same order. The machines are assumed to be set up in a series and such a processing enviroment is referred to as flow shop (Baker, 1974). In many manufacturing and assembly facilities a number operations need to be done on every job. Flow shop sequencing problems (FSP) has been very well studied in the field of combinatorial optimization. A combinatorial optimization problem is either maximization problem or minimization problem with an associated set of instances (Stutzle 1998). Normally, it is the problem faced by the manager in business operation. As a manager, they need to make a decision to each activity that will make profit to the company.

Flow shop sequencing problems is similar to traveling saleseman problems (TSP). The first paper on flow shop sequencing problem was published by Johnson in year 1954 (Colin, 1995). The flow shop sequencing problems is generally described as follows. There are *m* machines and *n* jobs, each job consists of m operations, and each operation requires a different machine. *n* jobs have to be processed in the same sequence on *m* machines. The majority of the research effort

during the past 30 years has been devoted to pure deterministic flow shop sequencing problem. Such a problem is usually labeled as $n/m/F/c_{max}$, which means *n*-number of job/*m*-number of machine/*F*-flow shop/ c_{max}-maximum flow time (Colin 1995).

Traveling salesman problems (TSP) is one of the most widely studied problems in combinatorial optimization (Chatterjee *et. al,* 1995). The idea of TSP is to find a tour of a given number of cities, visiting each city exactly once and returning to the starting city where the length of this tour is minimized. Flow shop sequencing problems is similar to TSP where the no of cities represent the no of machines and length of tour represent time taken to produce a certain product.

1.2 PROBLEM STATEMENTS

Modern production factories, which want to obtain high profits, usually maximize their productivity. The latter goal can be achieved, among others, by optimal or almost optimal scheduling of jobs in production process. In this thesis, it explains about the method used to solve the optimization problems (GA method). The objective function is to minimize maximum completion time (makespan).

In sequencing problem, it can be divided into two categories that are job shop and assembly line or flow shop. Job shop was sequencing problem that produce a product in a small quantity based on the demand while the assembly line was the sequencing problem that produce the product in a large quantity. This thesis will focus on assembly line problems.

Genetic algorithm method was one of the tools used to solve optimization problems. Each of the method such as tabu search, neural networks and genetic algorithms should have the limitation for solving optimization problems. Therefore this thesis will focus on finding the limitation of genetic algorithm toolbox in solving optimization problems.

1.3 RESEARCH OBJECTIVES

The research objectives are:

- Apply matlab (genetic algorithm tool) to determine the sequence of jobs in order to minimize the maximum flow time which is called makespan.
- To determine the limitation of genetic algorithm toolbox in solving flow shop sequencing problems.

1.4 RESEARCH SCOPES

Genetic algorithms (GAs) are intelligent random search strategies which have been used successfully to find near optimal solutions to many complex problems. Implementation of GA in solving problems often overlooks certain information available in a particular problem. Making use of this information may need modification of the coding of the search space and of the operators constituting GA. This is a problem specific task. This thesis hopes to address this issue with regard to solving the permutation flow shop problem. From the time onwards, this will refer as the flow shop problem, since it consider only the permutation flow shops. A permutation flow shop is a job processing facility which consists of several machines and several jobs to be processed on the machines.. The questions pose in order to implement the improved search heuristic can be extended easily to a host of scheduling problems with single and multi-objective optimization criteria.

Flow shops are useful tools in modeling manufacturing processes. In a permutation flow shop all jobs follow the same machine or processing order. Flow shop refers to the fact that job processing is not interrupted once started. Our objective is to find a sequence for the jobs so that the makespan or the completion time is minimized. It is well known that this is a difficult problem to solve in a reasonable amount of time.

1.5 RESEARCH METHODOLOGY

This research is conducted under three main steps. The first step is the literature review. In literature review, the previous method that used to solve flow shop sequencing problems are modeled and simulated to ensure the algorithm are working as reported in scientific literatures. Then, the existing algorithms limitations are identified.

From the previous limitations method, a new sequencing problem is developed as the purpose solution to optimize the flow shop sequencing problem. In order to prove that genetic algorithm was one of the methods that could solve flow shop sequencing problem, various kind of problem will be perform. The results of the performance will be analyst as the final step of this research.

Numerical experiment of flow shop sequencing problems is performed using the following setup:

- MATLAB Version 7.0
- Acer aspire 4520 notebook
- 1.7 Gigahertz of CPU
- 1.5 Gigabyte of RAM

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter explain the flow shop sequencing problems in general. This chapter also contain about previous research on flow shop sequencing problem. There was also an explanation to the genetic algorithm (GA) method and the steps that we need to follow to solve the problem using GA method. In the end of this chapter, it discuss about the method used (GA method) to solve the flow shop sequencing problems.

2.2 FLOW SHOP SEQUENCING PROBLEM

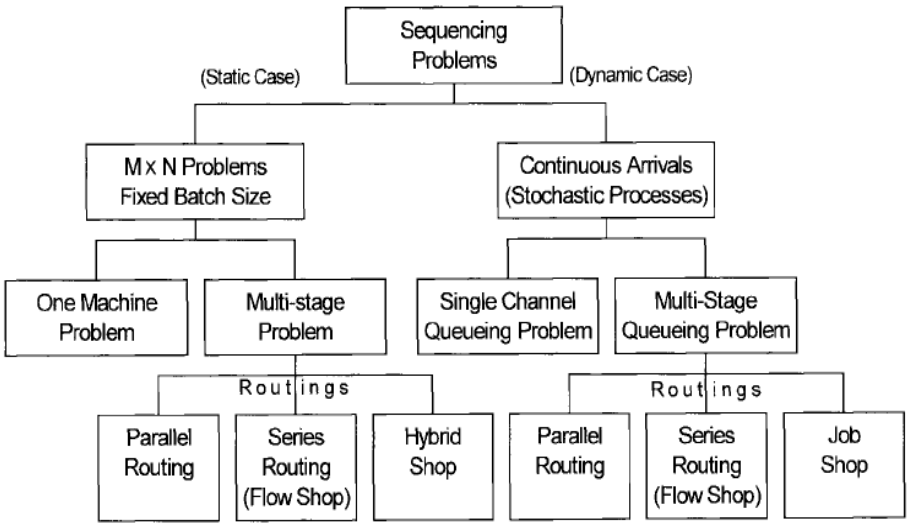

Figure 2.1 Classification of sequencing problem

Source taken from

J. E. Day and M. P. Hottenstein, "Review of Sequencing Research,"Nav. Res. Logist. Quart., vol. 17, pp. 11–39, 1970.

Figure 2.1 show the classification of sequencing problem. Flow shop sequencing problems has been very well studied in the field of combinatorial optimization. The idea of flow shop sequencing problems have begin since year 1954 when Johnson published the first paper of flow shop sequencing problem. Since then, this problem has held the attention of many researchers. The flow shop sequencing problems are generally described as follows. There are *m* machines and *n* jobs, each job consists of *m* operations, and each operation requires a different machine. The flow shop sequencing problem may be classified into three following categories:

- Deterministic flow shop problem
- Stochastic flow shop problem
- Fuzzy flow shop problem

Deterministic problems assume that fixed processing times of jobs are known. In the stochastic cases, processing times vary according to chosen probability distribution. In fuzzy decision context, a fuzzy due date is assigned to each to

represent the grade of satisfaction of decision makers for the completion time of the job.

2.2.1 Sequencing flow shop objective.

The flow shop sequencing problem involves the determination of the order of processing jobs over machines to meet a desired objective while all jobs have the same machine sequence. Normally most of the desired objectives have been devoted to makespan minimization and the flow time minimization. Practically, by minimization the makespan will lead to the minimization of the total production run, while minimization of flow time leads to stable or even utilization of resources, rapid turn-around of jobs and the minimization of in-process inventory. These clearly show that both objectives can be used to reduce cost significantly (Gupta and Dudek 1971). In sequencing flow shop the problems depends on the number of machine, number of jobs, the flow shop and the maximum flow time that was fixed by the company. Therefore, there is various kind of problem can be made up.

2.3 TRAVELLING SALESMAN PROBLEM (TSP).

Travelling salesman problem (TSP) is one of the most widely studied in combinatorial optimization (Chatterjee *et. al,* 1995). The idea of TSP is to find a tour of a given number of cities, visiting each city exactly once and returning to the starting city where the length of this tour is minimized. Flow shop sequencing problems (FSP) or also known as assembly line problems actually followed the concept of TSP. There was only a few different between TSP and FSP problems.

For TSP, the problems focus on finding the shortest length in order to minimize the time taken for the whole city visited. While for FSP, the problems focus on finding the best sequence in assembly line in order to minimize the time taken to produce a product. The different for TSP and FSP only on the problems that want to be solve.

TSP represent by the number of city visited while for TSP, the problems represent by the number of machine used in the assembly line. There is also one major different between TSP and FSP. The assembly line problems actually end at the last process or machine and it is different with TSP where the length was taken until the salesman return to the starting point.

2.4 GENETIC ALGORITHM.

Genetic Algorithm (GA) is an optimization technique, based on natural evolution. It was introduced by John Holland in 1975 (Othman 2002). This technique copied the biological theory, where the concept of "*survival of the fittest"* exits. GA provides a method of searching which does not need to explore every possible solution in the feasible region to obtain a good result (Othman 2002).

In nature, the fittest individuals are most likely to survive and mate. Therefore the next generation should be fitter and healthier because they were bred from healthy parents. This same idea is applied to a problem by first "guessing" solution and then combining the fittest solutions to create a new generation. The genetic algorithm consist five steps, that is:

1. Encoding
2. Evaluation
3. Crossover
4. Mutation
5. Decoding

2.4.1 Encoding

A suitable encoding is found for the solution to a problem so that each possible solution has a unique encoding and the encoding is some form of string. Many possible solution need to be encoded to create a population. The traditional way to represent a solution is with a string of zeroes (0) and ones (1). However genetic algorithms are not restricted to this encoding (Chatterjee *et.al* 1996).

2.4.2 Evaluation

The fitness of each individual in the population is then computed; this is how well the individual fits the problem and whether it is near the optimum compared to other individuals in the population. This fitness is used to find the individual's probability of crossover. Evaluation function is used to decide how good a chromosome is. This function is also known as objective function (Bryant 2001).

2.4.3 Crossover

Crossover is where the two individuals are recombined to create new individuals which are copied into the new generation. Not every chromosome is used in crossover. The evaluation function gives each chromosome a 'score' which is used to decide the chromosome's probability of crossover. The chromosome is chosen to crossover randomly and the chromosomes with the highest scores are more likely to be chosen.

2.4.4 Mutation

Mutation, which is rare in nature, represents a change in gene. It may lead to a significant improvement in fitness, but more often has rather more harmful results. Mutation is used to avoid getting trapped in a local optimum. The chromosome is naturally near the local optimum and very far from the global optimum (possible solution) due to the randomness process. Some individuals are chosen randomly to be mutated and then a mutation point is randomly chosen. Mutation causes the character in the corresponding position of the string changed (Negnevitsky 2002).

2.4.5 Decoding

On all the four processes are done, a new generation has been formed and the process is repeated until some stopping criteria have been reached. At this point the individual who is closest to the optimum is decoded and the process is complete (Bryant 2000).

2.5 CHROMOSOME REPRESENTATION IN GA.

In genetic algorithms, each individual that is a member of the population represents a potential solution to the problem. This solution information is coded in the associated chromosome of that individual. A chromosome is a string of gene positions, where each gene position holds an allele value that constitutes a part of the solution to the problem. Allele value at a gene position represents an element from a finite alphabet. This alphabet depends on the nature of the problem. There are number of possible chromosome representations, due to vast variety of problem types. However there are two representation types which are most commonly used: binary representation and permutation representation (Bryant 2000). In genetic algorithms method, binary representation method is the most commonly used in chromosome representation (Bjarnadottir 2004).

2.5.1 Binary Representation

In binary representation, the finite alphabet domain which allele at each gene position takes its value from is the set {0, 1}. An example of problem to minimize the function of $= 3x^3 + 4x^2 - 7x + 1$ over the integers in set {0, 1, …, 15}. The binary number for the integer set was represented in table figure 2.1. The possible solution for the problem are obviously just numbers, so the representation is simple the binary form of each number. For example, the binary representation of 8 and 14 are 1000 and 1110 respectively. For initial chromosomes (1000 and 1110) which represent value 8 and 14, the evaluation are perform by calculating the fitness function above.

$$f(8) = 3(8^3) + 4(8^2) - 7(8) + 1$$
$$= 1737$$
$$f(14) = 3(14^3) + 4(14^2) - 7(14) + 1$$
$$= 8919$$

Obviously 8 is better solution than 14 (since $f(8)$ is lower than $f(14)$) and would therefore have a lower fitness. The initial chromosomes then are being re-generated by using simple crossover.

$$P_1 = 1000$$

$P_2 = 1110$

Then the crossover point is randomly chosen. In this example, the crossover point is after the second gene.

$P_1 = 10|00$

$P_2 = 11|10$

The genes are switched after the crossover point and give the new offspring as follow:

$P_1 = 1010 = 10$

$P_2 = 1100 = 12$

Therefore the new offspring are 10 and 12. This number will be evaluated as above. These procedures are continuous to evaluate and re-generate new offspring until a termination criteria is satisfied (Bjarnadottir 2004).

Table 2.1: Binary number representation

Real number	Binary number
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0110
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

2.6 SIMPLE PROBLEM

An explanation through an example as shown below was presented in order to get better understanding in genetic algorithm (GA) method. Let try to find the maximum value of the function ($12x - x^2$). Thus, chromosome can be build with only 4 genes. As stated in the project scope, suppose that the crossover probability p_c equal 0.7, mutation probability p_m equals 0.001 and the chromosome population N is 6.

The next step is to calculate the fitness of each individual chromosome. We take 5, 3, 10, 7, 1 and 9 as the possible answer to the problem. The results are shown in table 2.2. In order to improve it, the initial population is modified by using selection, crossover and mutation, the genetic operators.

In natural selection, only the fittest species can survive, breed, and thereby pass their genes on to the next generation. GAs uses the similar approach, but unlike nature, the size of the chromosome remains unchanged from generation to the next.

Table 2.2: The initial randomly generated population of chromosomes

Chromosome Label	Chromosome string	Decoded integer	Chromosome fitness	Fitness ratio (%)
X1	0101	5	35	22.6
X2	0011	3	27	17.4
X3	1101	10	20	13.0
X4	0111	7	35	22.6
X5	1001	9	27	17.4
X6	0001	1	11	7.0
	Total	30	155	100

The values of chromosome fitness in the third column in table 2.2 were determine by change the value of x in the function $(12x - x^2)$ with decoded integer for each chromosome. The fitness ratio in the last column in table 2.2 determines the chromosome's chance to be selected for mating. The values of the fitness ratio were determined by dividing the value of the chromosome fitness for each chromosome with the total value of chromosome fitness. Thus, the chromosome X1 and X4 stands a higher change to be selected while chromosome X6 has a very low probability to be selected. As a result, the chromosome with average fitness improves from one generation to the next.

One of the most commonly used chromosome selection techniques is the roulette wheel selection (Davis, 1991). Figure 2.2 illustrates the roulette wheel as an example. As shown in figure 2.1, each chromosome is given a slice of a circular roulette wheel. The area of the slice within the wheel equal to the fitness ratio value (see table 2.2). To select the chromosome for mating, a random number is generated in interval [0, 100]. It is like spinning a roulette wheel where each chromosome has a

segment on the wheel based on the fitness ratio. The roulette wheel is spun, and when the arrow comes to rest on the segments, the corresponding chromosome is selected for mating. The roulette would be spun six times because we have an initial population of six chromosomes.

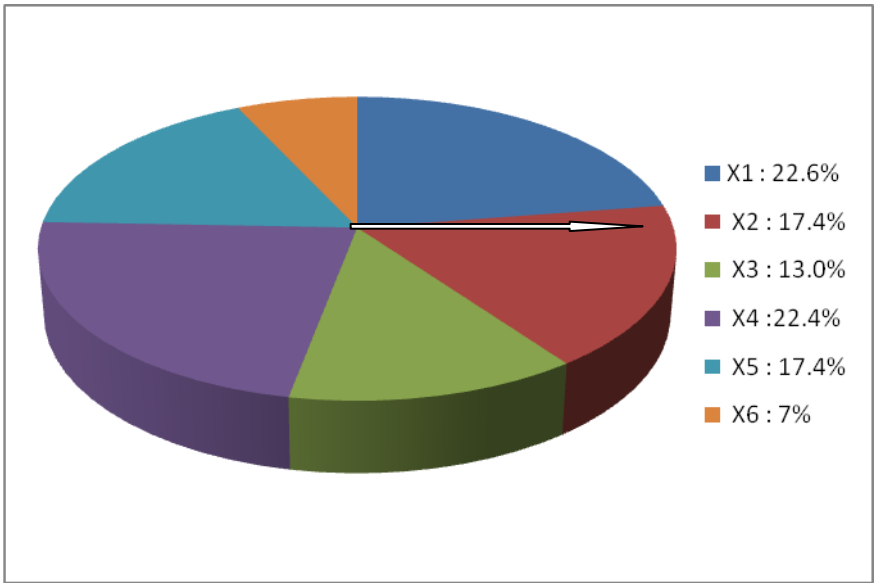

Figure 2.2: Roulette wheel

Once a pair or parents is selected, the crossover operator is applied. First the crossover operator randomly chooses a crossover point where two parents chromosomes 'break' and then exchanges the chromosome part after that point. As a result two new offspring are created. The process will continue with the other two pairs of parent chromosomes. Finally, six new offspring will create. A value of 0.7 for the crossover probability generally produces good results.

A mutation process will take over after the crossover finish. The mutation operator flips a randomly selected gene in a chromosome. Mutation can occur at any gene in a chromosome with some probability. The mutation probability is quite small in nature, and is keep quite low for gas, typically in the range 0.001 to 0.01 as stated before in project scope.

The process from evaluation of the fitness function until the mutation process will be repeated several times. The numbers of cycles depend on the performance of the computer. Genetic algorithm (GA) method does not provide an exact solution to the problem. It will produce several solutions near to the exact solution. The final answer to the problem depends on the objective that has been set and it is up to us to choose the suitable solution based on the objective.

2.7 GENETIC ALGORITHM IN SEQUENCING FLOW SHOP

In genetic algorithm method, the encoding, crossover and mutation process will give different forms. The different forms of crossover and mutation process in genetic algorithm method can be combined to give various genetic algorithms that can be used to solve the flow shop sequencing problem.

The first step in applying GA to a particular problem is to convert the feasible solutions of that problem into a string type structure called chromosome. In order to find the optimal solution of a problem, standard GA starts from a set of assumed or randomly generated solution (chromosome) called initial population and evolve different but better set of solution (chromosome) over sequence of generation. In each generation the objective function (fitness measuring criterion) determines the suitability of each chromosome and based on the values, some of them are selected for reproduction. The number of copies reproduced by an individual parent is expected to be directly proportional to its fitness value, thereby embodying the natural selection procedure, to some extent. The procedure thus selects the better (highly fitted) chromosomes while the worse chromosome are eliminated. Genetic operators such as crossover and mutation are applied to these (reproduced) chromosomes and new chromosomes (offspring) are generated. These new chromosomes constitute the next generation. This iteration continues until some termination criterion is satisfied (Srikanth and Saxena 2004).

The fitness function plays an important rule in genetic algorithms because it is used to decide the quality of a particular chromosome. Generally, different types of

problems use different fitness functions. There are five assumptions for this kind of problem to be modeled in general. The assumption as follow:

- The operation processing times on the machines are known, fixed and some of them may be zero if some job is not processed on a machine.
- Set-up times are included in the processing times and they are independent of the job position in the sequence of jobs,
- At a time, every job is processed on only one machine and every machine processed only one job.
- The job operations on the machine may not be preempted.

In general there are four steps to be followed when implementing genetic algorithm (GA) for flow shop sequencing problem FSP).

1. = initialization
2. = representation
3. = evaluation and selection
4. = generating new offspring

2.7.1 Initialization

Initialization step consists of establishing initial chromosomes and parameter setting such as probability of crossover and mutation. The number of generation also is set here as termination criteria for algorithm. The traditional method of generating an initial population is to randomly sample the search space. By initializing the first population with entirely random solutions, the GA receives a complete freedom to evolve any solutions that will fulfill the problem specification. The GA can create novel and potentially unconventional solution (Kim *et. Al* 1996).

2.7.2 Representation

Representation plays an important role in searching optimal solution. It is very important in GA because it will affect the entire results of algorithm. Representation means how the chromosome is encoded. It is important to ensure that the chromosome represents the actual problem that needs to be simulated. An

important issue that effects the chromosome representation is how fast it can be repaired and comes out with feasible chromosome (Rekiek 2001). Apart from benefits of speeding up execution time, which is the main reason heuristic method are used, fast excess to repair chromosome can heavily affects speed of generation to optimal solution. (Rekiek 2001). As stated by Othman (2002), it os difficult to achieve optimal solution if the chromosome representation is not suitable. A poorly designed chromosome representation can be useless and should be abandoned in favor of other steps in genetic algorithm (Rekiek 2001, Othman 2002).

2.7.3 Evaluation and selection

Evaluation of chromosome is performed by applying a fitness function to each chromosome. It measured the quality of chromosomes. The fitness values then influence the selection of chromosomes to be re-generated. When selection is performed, the best individual is eventually selected to completely take over the population. Selection of chromosomes is the guidance for GA to generate better solution (Carter 2004).

2.7.4 Generation of new offspring

The selection mechanism does not introduce any new chromosome for consideration. It just copied some solutions to form an intermediate population. The second step of evaluation cycle is recombination, which will introduce new chromosome into population. This is done by genetic operators: crossover and mutation (Rekiek 2001).

The simple version of this operator inherits individuals as they are. The most popular one is the crossover where two individuals are selected and are mated (crossed-over) in order to produce offspring (Zhang and Yassine 2004). Information is extracted from the parents and is used to create offspring. The aim of crossover is to produce new solutions in region of search space where successful ones have already been found. Mutation introduces random changes to chromosomes. It is a mechanism that has only a small change of occurring. This does not mean that the

mutation operator is useless. Indeed, in absence of diversity, the crossover would again fall into the trap of non-representative sampling, because the progeny would be identical to the parents. The mutation also creates new points containing some of the gene from parent chromosome (Moon *et. al* 2002).

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

This chapter is focused on the methodology process. Before the problem can be solve using genetic algorithm method, there will be steps that must be follow. In this chapter, all the important steps to solve the problem are stated. The flow of the project from the beginning until the end of the project is also stated.

3.2 PROJECT FLOW CHART

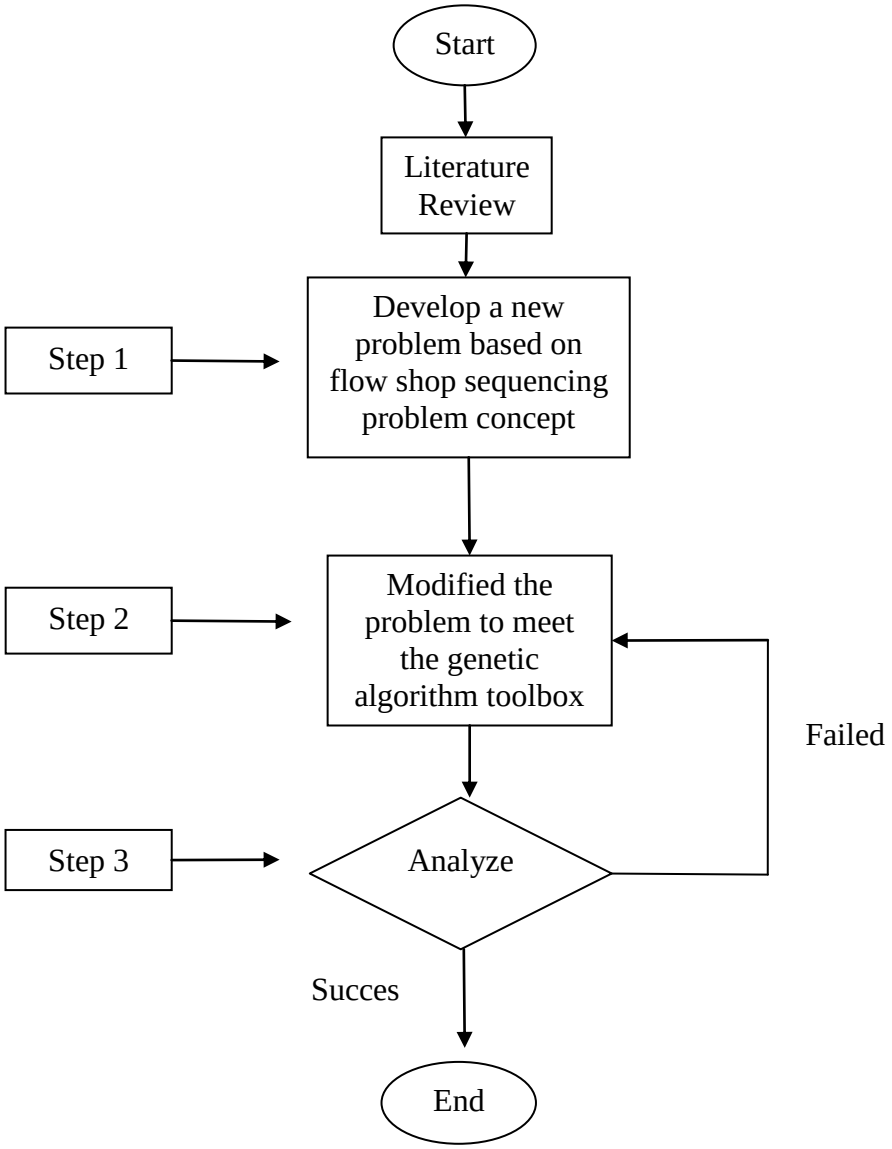

Figure 3.1 Overall process flow chart

3.3 PROBLEM IDENTIFICATION (step 1)

A problem related to sequencing flow shop will be identified before a further movement can be made. During this process, data collection will be made from the real situation such as the data from the company or collecting data from the thesis

that have been studied in literature review. The data taken then will be evaluate, in order to used the genetic algorithm method.

3.4 GENETIC ALGORITHM METHOD (step 2)

In genetic algorithm method, there are five steps that must be followed before final solution can be evaluated. The steps in genetic algorithm are as stated below:

1. Encoding
2. Evaluation
3. Crossover
4. Mutation
5. Decoding

3.4.1 Encoding process

The data that have been collected will go through the encoding process before it can be solve. Encoding process was the most important part in genetic algorithm method. This process was also the most difficult process in genetic algorithm. In encoding process we need to convert the real problem that is the data collected into computer language.

3.4.2 Evaluation process

The fitness of each individual in the population is then computed. In evaluation process the evolution function is used to decide how good a chromosome that represents a possible solution to the problem. This function is also known as objective function (Bryant 2001)

3.4.3 Crossover process

The crossover process is where the two individuals are recombined to create new individuals which are copied into the new generation. The crossover probability was set in the range of 0.5 to 0.7 and normally 0.7 will be selected as the crossover

probability. The crossover process will provide new offspring that represents a better solution to the problem.

3.4.4 Mutation process

The chromosome is naturally near the local optimum and very far from the global optimum (possible solution) due to the randomness process. Therefore, mutation process will take place to prevent the chromosome near the local optimum. The probability for mutation process was set in the range of 0.001 to 0.01.

3.4.5 Decoding process

The process will be repeated from steps one to step four several times based on the performance the computer before decoding process take place. Decoding process was the final process in genetic algorithm method. In this process, the possible solution will be converted from computer language into a better understanding language (real situation). The possible solution will be selected from the solutions given from the genetic algorithm method based on the objective of the project.

3.5 GENETIC ALGORITHM METHOD FLOW CHART (step 3)

Figure 3.2 in the next page represent the overall computer process for genetic algorithm method:

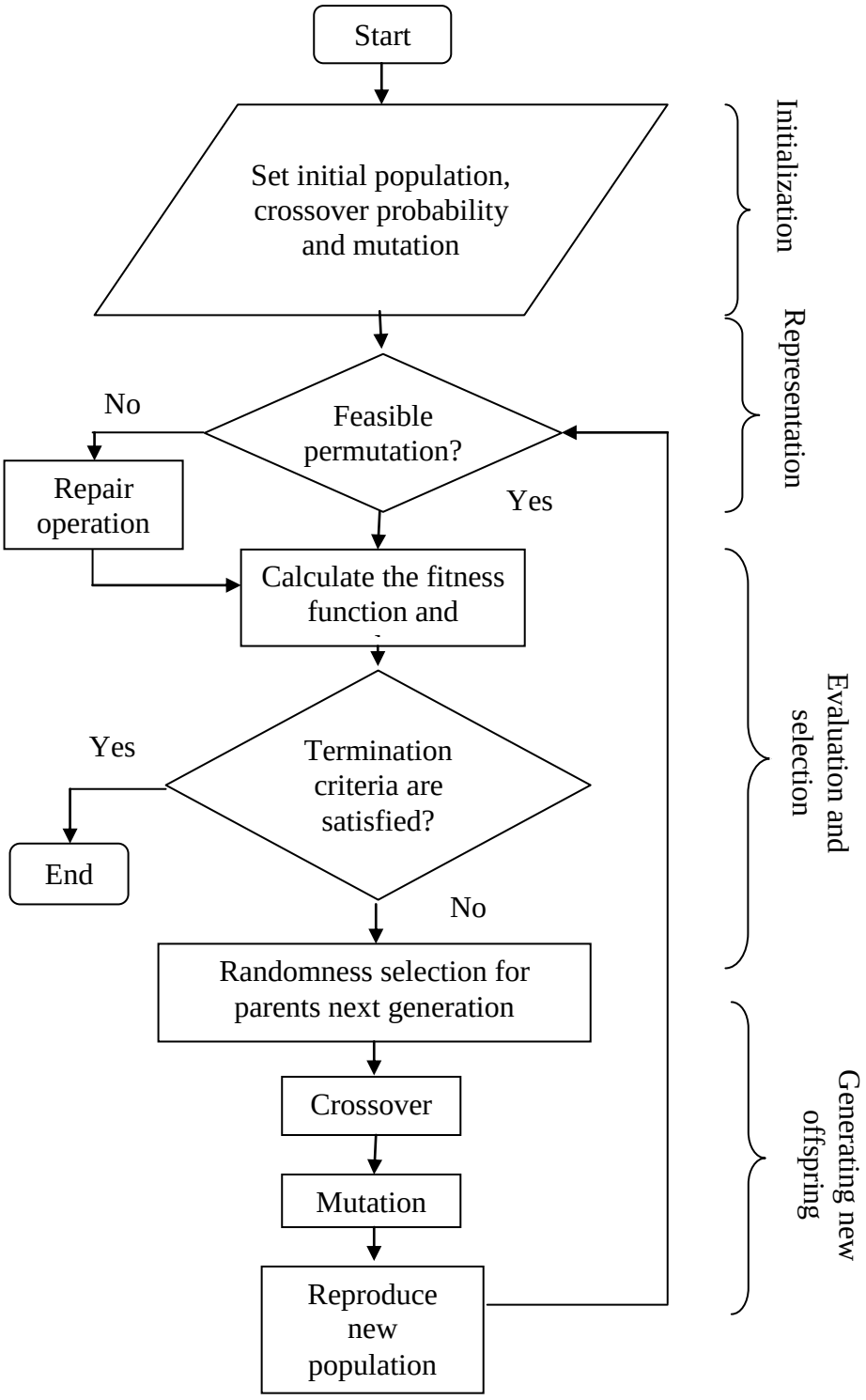

Figure 3.2 Genetic Algorithm Procedures for FSP

CHAPTER 4

NUMERICAL EXPERIMENT

4.1 INTRODUCTION

Chapter 4 is dealing with numerical experiment of flow shop sequencing problem. This chapter started with experiment setup, finding the real case study and then numerical experiment results. Genetic algorithm toolbox in math lab programming is used to run the experiment. Genetic algorithm toolbox was design to solve optimization problems without writing the programming for each problem that want to solve.

4.2 GENETIC ALGORITHM AND DIRECT SEARCH TOOLBOX

Genetic Algorithm and Direct Search Toolbox extends the optimization capabilities in MATLAB and Optimization Toolbox with tools for using genetic algorithms, simulated annealing, and direct search. These algorithms can be used for problems that are difficult to solve with traditional optimization techniques, including problems that are not well defined or are difficult to model mathematically. It also can be used when computation of the objective function is discontinuous, highly nonlinear, stochastic, or has unreliable or undefined derivatives. Genetic Algorithm and Direct Search Toolbox complements other optimization methods to help find good starting points. Traditional optimization techniques then can be used to refine the solution.

Toolbox functions, accessible through a graphical user interface (GUI) or the MATLAB command line, are written in the open MATLAB language. This means

that the algorithms can be inspected, modify the source code, and created custom functions by own self. The Genetic Algorithm Tool is a graphical user interface that enables to use the genetic algorithm without working at the command line.

4.2.1 Genetic Algorithm Toolbox

The genetic algorithms solve optimization problems by mimicking the principles of biological evolution, repeatedly modifying a population of individual points using rules modeled on gene combinations in biological reproduction. Due to its random nature, the genetic algorithm improves chances of finding a global solution.

The Genetic Algorithm and Direct Search Toolbox provide the following standard algorithm options.

Table 4.1: Overview of genetic algorithm toolbox

Step	Algorithm Option
Creation	Uniform
Fitness scaling	Rank-based, proportional, top (truncation), linear scaling, shift
Selection	Roulette, stochastic uniform selection (SUS), tournament, uniform
Crossover	Arithmetic, heuristic, intermediate, scattered, single-point, two-point
Mutation	Adaptive feasible, Gaussian, uniform
Plotting	Best fitness, best individual, distance among individuals, expectation of individuals, range, diversity of population, selection index, stopping conditions

Genetic algorithm method using genetic algorithm toolbox provide user to key in the data based on the objective of the problems. All the data that need to be key-in in the toolbox was stated as below:

- Population size

- Crossover fraction
- Mutation fraction
- Fitness function

4.2.1.1 Population size

The population size was the part where the user can set how large the process of solving the problems before the user gets the exact solution. It determines how many round of process solving using genetic algorithm technique starting from crossover until mutation process. The greater number of population size will provide the solution that almost the exact answer. As stated before in chapter 2, genetic algorithm method cannot give an exact solution to the problem.

Genetic algorithm methods only provide the solution that almost the exact solution. To be able to get the solution that nearer the exact solution, a greater number of population sizes were suggested. Normally the user will used 100 as a population size to solve the problem.

4.2.1.2 Crossover fraction

Crossover fraction was the percentage that definite which chromosomes are selected to mate each other to produce new generation of chromosome. The chromosome that was produce contains genes or in other words the possible answer of the problems. The greater number of crossover fraction will increase the percentage of the chromosome to be selected. Based on the previous researcher that is Moon, 0.6 was set as the crossover fraction.

4.2.1.3 Mutation fraction

Crossover fraction was the percentage that the user set for the new generation of chromosome to be selected for mutation process. The new generation of chromosome that was produced from crossover process will continue the genetic algorithm technique with mutation process. The mutation process will modified the

selected chromosome in order to determine the nearest answer with the exact solution. Normally the percentage of the chromosome to be selected for mutation process is small. Therefore, the user should set the small number for the mutation fraction. 0.2 was set as the mutation fraction to solve the problems in this thesis based on the previous researcher Moon.

4.2.1.4 Fitness function

The fitness function represents the problem that the users want to solve. Therefore, it was the most important part before the problem can be solve. In fitness function section, it converts the problem into mathematical equation. It is depends on the user how to convert the problem into the mathematical equation.

For this section, the creativity of the user to convert the problem was the most important things. If the user cannot convert the problems into mathematical equation, the process of solving problem cannot continue.

4.2.2 Displaying, Monitoring, and Outputting Results

The toolbox includes a number of plotting functions for visualizing the optimization problems. These visualizations give live feedback about the problem solving, enabling the user to make modifications while executing. Specific plotting functions are provided for both the genetic algorithm and direct search algorithms. The plotting functions include function value, score histogram, genealogy, and fitness value, mesh size, and function evaluations. Multiple plots also can be showing altogether in a graph or select specific plots for closer examination.

The genetic algorithm toolbox is provided in matlab software version 7.0. The toolbox can be called out by typing ‘gatool’ in mathlab workspace. An example of genetic algorithm toolbox was given in figure 4.1

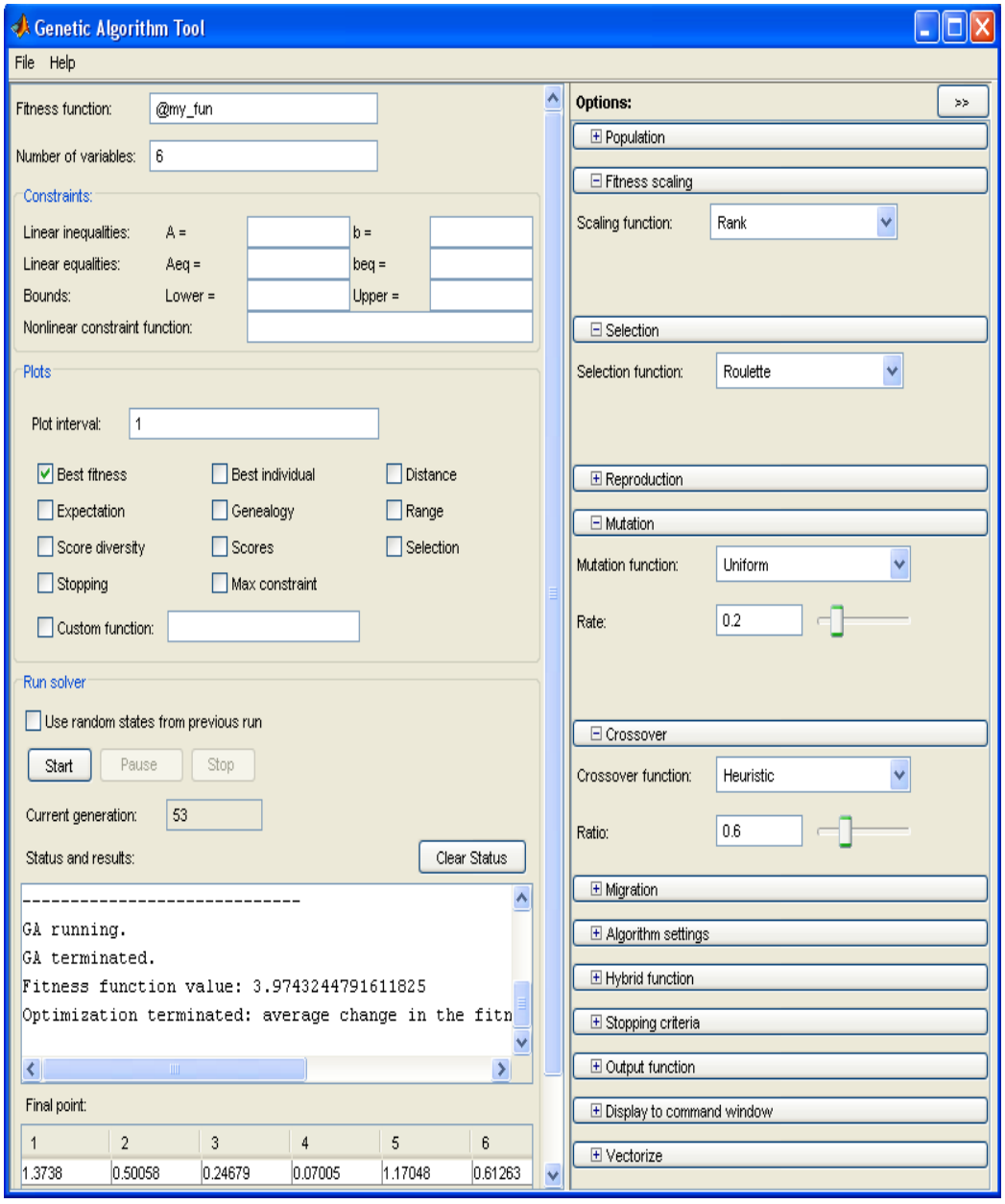

Figure 4.1: Genetic algorithm toolbox

4.3 CASE STUDY I: NEXUS ELECTRONIC SDN. BHD.

The case study I was taken from the real situation of optimization problem. The data was taken from an electronic company, Nexus Electronic Snd Bhd , Jaya Gading, Kuantan. The general manager of the company was Mr. CH Chong and technical director was Mr. Ismail b Md Dom. Nexus Electronic Sdn Bhd produces

various types of transformer and toroid. They produce 5000-6000 transformers and 1000-1200 toroid per day with 115 workers. The picture of the transformer and toroid were given in figure 4.2 and figure 4.3 All the product that was produce was then send to Nemic-Lamdar that acts as nexus main customer. The objective of case study I is to find the fastest time to produce their product therefore the company can maximize their production per day.

.

Figure 4.2: Transformer

Figure 4.3: Toroid

The precedence diagram for toroid production was same with the precedence diagram for transformer production. Therefore the precedence diagram in figure 4.4 below represents precedence for both products

.

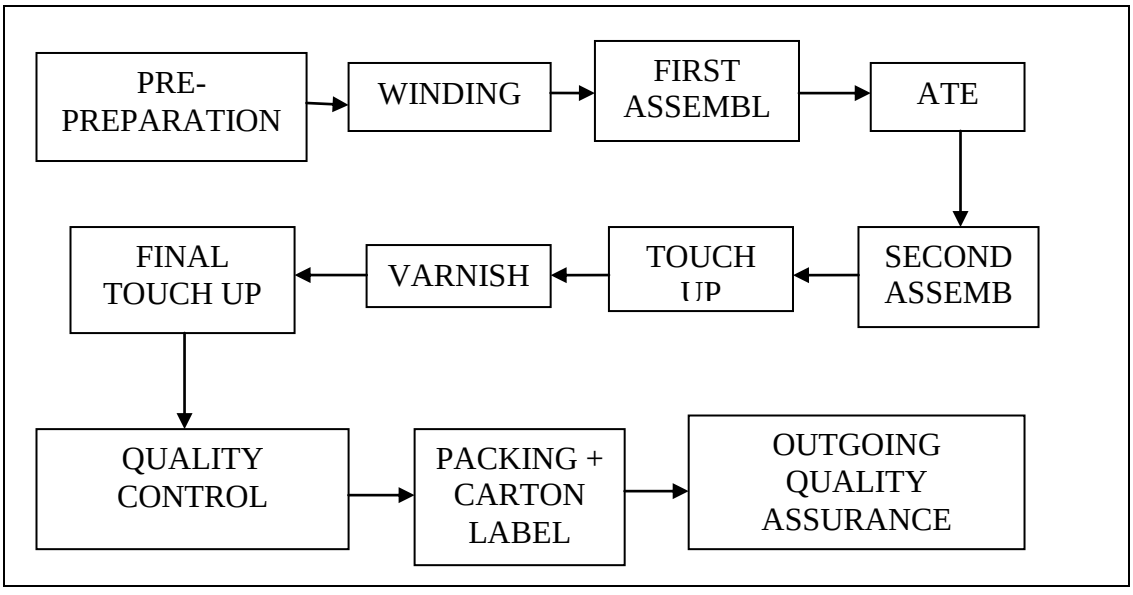

Figure 4.4: Precedence constraint for case study I: Nexus Electronic Sdn Bhd

The transition time for case study I: Nexus Electronic Sdn Bhd is given in table 4.2 for transformer production and table 4.3 for toroid production.

Table 4.2: Processing and transition time for transformer production

TASK	OQA	WINDING	FIRST ASSENBLY	ATE	SECOND ASSEMBLY	TOUCH UP	VARNISH	FINAL TOUCH UP	QC	PACKING	PRE-PREPARATION
PRE-PREPARATOIN	15 (s)	-	-	-	-	-	-	-	-	-	-
WINDING	-	830 (s)	-	-	-	-	-	-	-	-	-
FIRST ASSENBLY	-	-	63 (s)	-	-	-	-	-	-	-	-
ATE	-	-	-	20 (s)	-	-	-	-	-	-	-
SECOND ASSEMBLY	-	-	-	-	45 (s)	-	-	-	-	-	-
TOUCH UP	-	-	-	-	-	65 (s)	-	-	-	-	-
VARNISH	-	-	-	-	-	-	2700 (s)	-	-	-	-
FINAL TOUCH UP	-	-	-	-	-	-	-	50 (s)	-	-	-
QC	-	-	-	-	-	-	-	-	30 (s)	-	-
PACKING	-	-	-	-	-	-	-	-	-	8 (s)	-
OQA	-	-	-	-	-	-	-	-	-	-	45 (s)

Table 4.3: Processing and transition time for toroid production

TASK	OQA	WINDING	FIRST ASSENBLY	ATE	SECOND ASSEMBLY	TOUCH UP	VARNISH	FINAL TOUCH UP	QC	PACKING	PRE- PREPARATION
PRE-PREPARATOIN	15 (s)	-	-	-	-	-	-	-	-	-	-
WINDING	-	850 (s)	-	-	-	-	-	-	-	-	-
FIRST ASSENBLY	-	-	50 (s)	-	-	-	-	-	-	-	-
ATE	-	-	-	14(s)	-	-	-	-	-	-	-
SECOND ASSEMBLY	-	-	-	-	45 (s)	-	-	-	-	-	-
TOUCH UP	-	-	-	-	-	65 (s)	-	-	-	-	-
VARNISH	-	-	-	-	-	-	2700 (s)	-	-	-	-
FINAL TOUCH UP	-	-	-	-	-	-	-	50 (s)	-	-	-
QC	-	-	-	-	-	-	-	-	30 (s)	-	-
PACKING	-	-	-	-	-	-	-	-	-	8 (s)	-
OQA	-	-	-	-	-	-	-	-	-	-	45 (s)

4.3.1 Numerical Experiment Results for Case Study I: Nexus Electronic Sdn Bhd

There are no results to display for case study I: Nexus Electronic Sdn Bhd .

4.3.2 Analysis of Numerical Experiment Results for Case Study I: Nexus Electronic Sdn Bhd

There are no results to display for case study I: Nexus Electronic Sdn Bhd because the limitation of genetic algorithm method using genetic algorithm toolbox. Each optimization method will have the limitation in solving the problems. The same reason goes to genetic algorithm method using genetic algorithm tool

.

Because of the limitation for each method developed various type of solving method has been proposed. In the field of optimization problem, various methods have been developed such as neural network, tabu search and genetic algorithms. All of the optimization method will have the limitation and method used is depends on type of problem to optimize.

From the precedence diagram in case study I: Nexus Electronic Sdn Bhd it showed that each job must be done by following the sequence given. As an example, to be able to do winding process pre-preparation process must be done first. It is same goes to ATE process that must complete the winding process before it can be perform. To be able to produce a complete transformer or toroid, it must follow the sequence from pre-preparation until outgoing quality assurance by following the sequence given as shown in figure 4.1.

From case study I: Nexus Electronic Sdn Bhd , it clearly showed the limitation of genetic algorithm method using genetic algorithm toolbox. As it is shown in case study I: Nexus Electronic Sdn Bhd , the problem that consists of process that must perform by following the sequence cannot be solving in genetic algorithm toolbox.

4.4 CASE STUDY II: PROCESS FLOW SHOP WITHOUT PRECEDENCE CONSTRAINT PROBLEM (TSP CONCEPT).

Case study II was focused on the process flow shop without precedence constraint. Flow shop sequencing or assembly line problems without precedence constraint were similar with traveling salesman problem (TSP). As stated in chapter 2, TSP concept deal with the distance take between the city visited but for assembly line it deals with the time taken between each machine. A simple problem was illustrated in figure 4.5 to represent the process flow shop without precedence constraint problem.

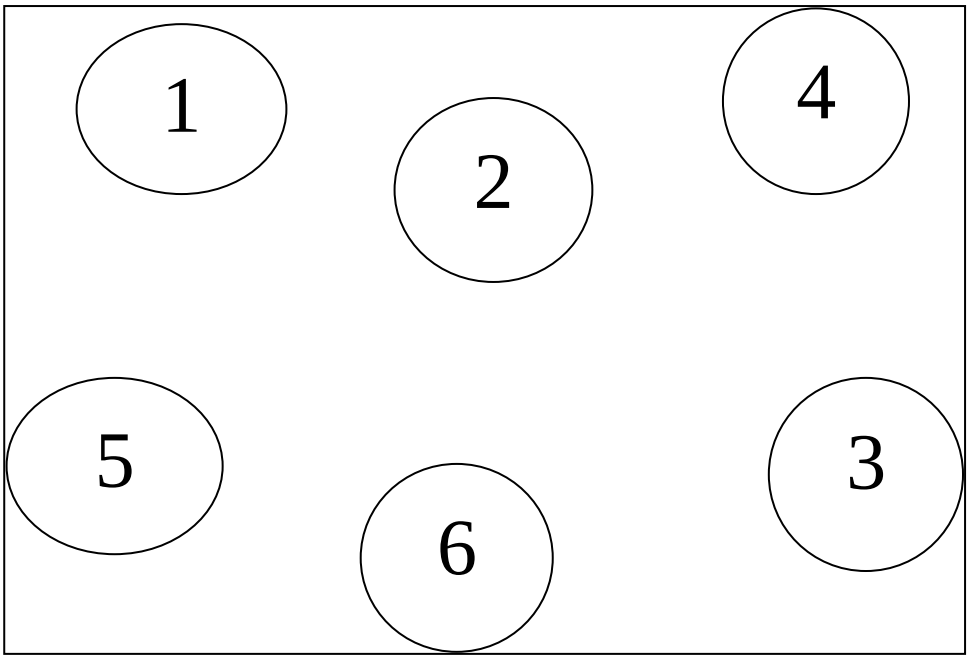

Figure 4.5: Process flow shop without precedence constraint

Figure 4.5 represent the process flow shop without precedence constraint problem. The number 1, 2, 3, 4, 5 and 6 as shown in figure 4.5 represent the number of machine used in an assembly line problem. The problem was not restricted to start at a certain machine and end at certain machine because it was the problem without precedence constraint. Therefore, the process of solving the problem can be starting at any machine as long as all the machines were used. As an example, the assembly line process can start at machine number 5, 4, 1, 6, 3 and end at machine number 2.

The main objective of this problem was to determine the best sequence that can produce the minimum transition time. By minimizing the transition time, the number of product can be increase in a certain time. The time taken for transition time from each machine was assumed as constant in order for the problem to be able to solve using genetic algorithm toolbox.

4.4.1 Numerical Experiment Results for Case Study II: Process flow shop without precedence constraint problem

The results for case study II are shown in table 4.4, figure 4.6 and figure 4.7.

Table 4.4 Comparison table for the possible solution

No. of machine	**Possible solution**				
	Weight 1	**Weight 2**	**Weight 3**	**Weight 4**	**Weight 5**
1	0.13154	0.17859	0.17853	0.1317	0.53953
2	0.23991	0.1014	0.32356	0.00371	0.00569
3	0.2313	0.32285	0.02599	0.57006	0.14117
4	0.02233	0.07679	0.32633	0.28427	0.22039
5	0.06098	0.27845	0.1134	0.27721	0.19877
6	0.012672	0.1247	0.19187	0.40851	0.22841
Best fitness	0.8128	1.0828	1.1587	1.6755	1.334

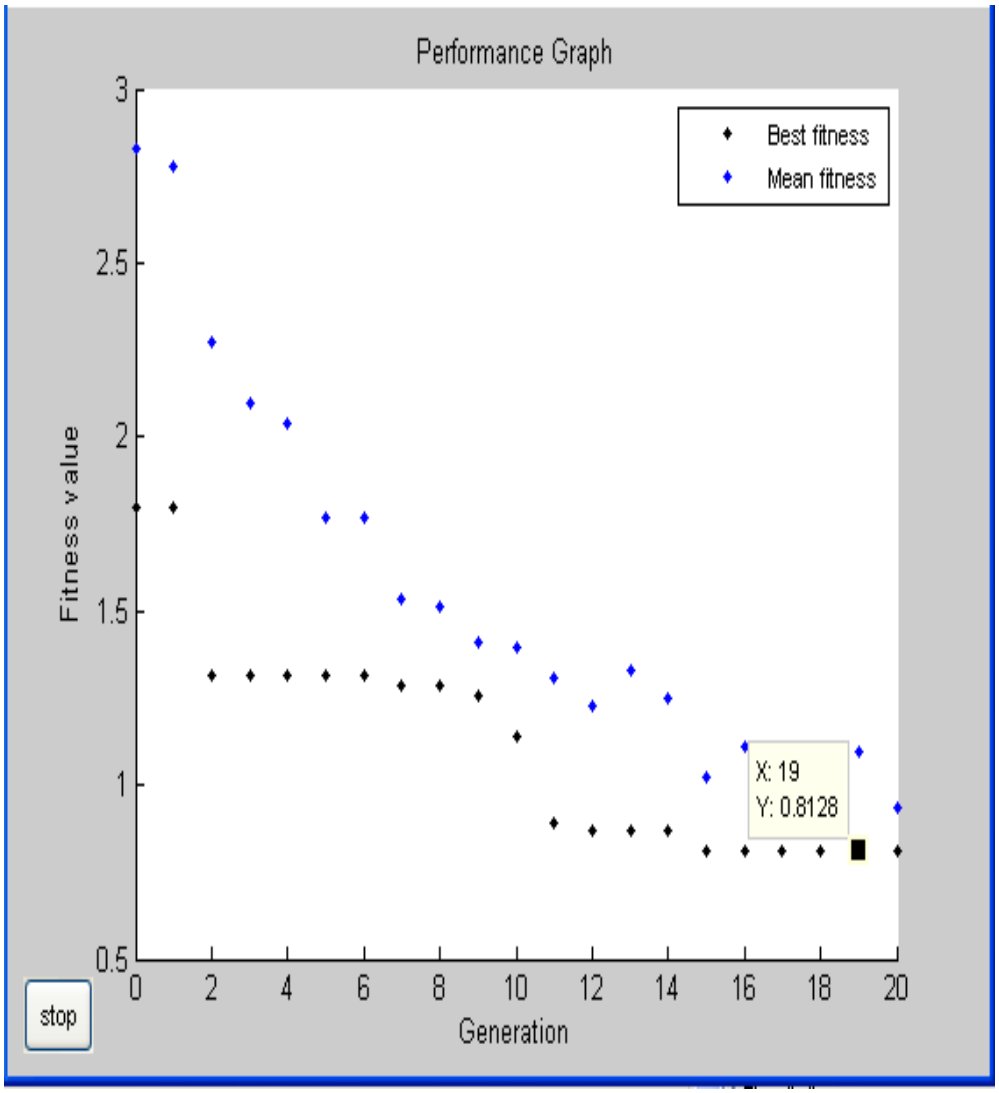

Figure 4.6: Performance graph

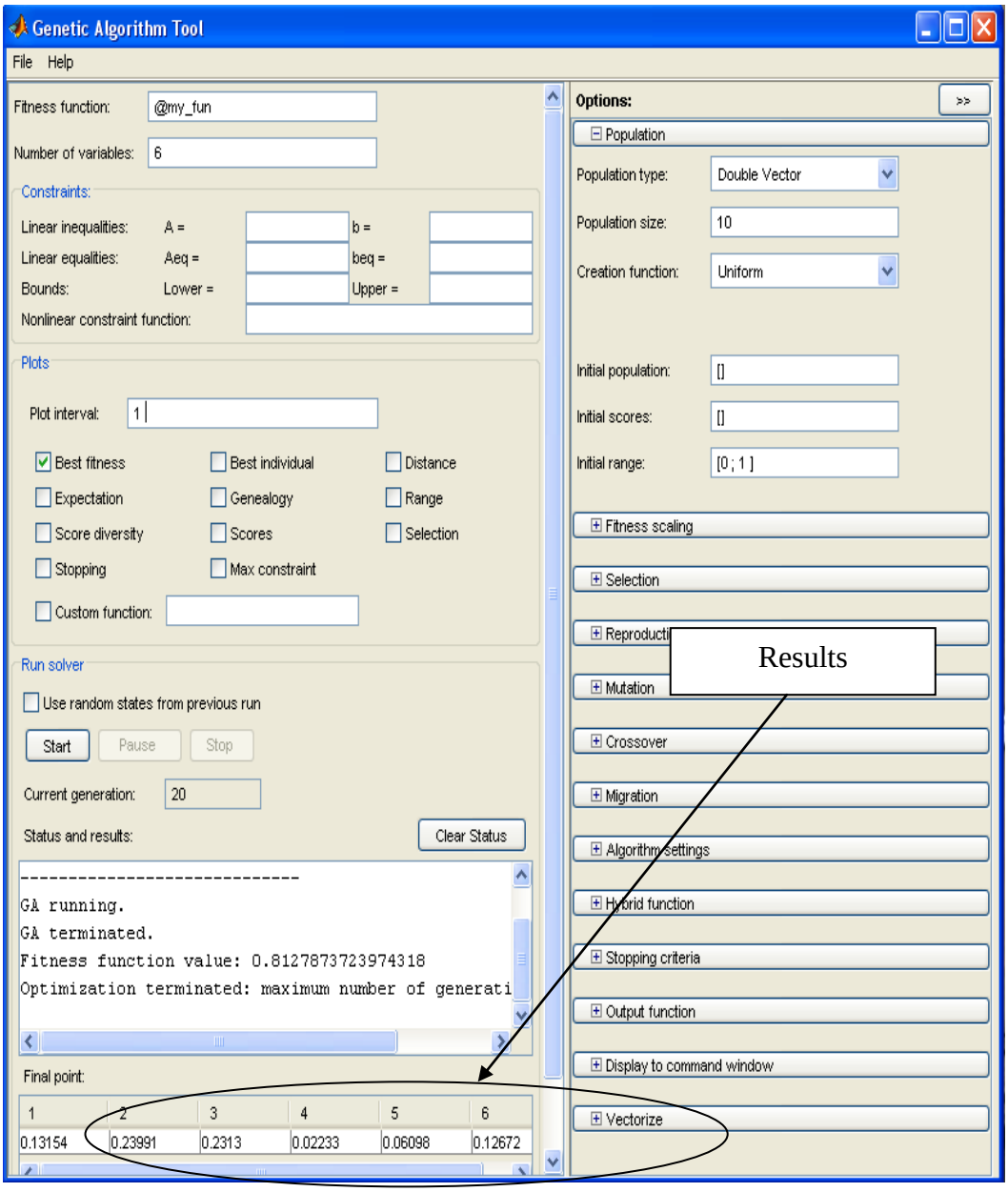

Figure 4.7: Results after running genetic algorithm toolbox

4.4.2 Analysis of Numerical Experiment Results for Case Study II: Process Flow Shop without precedence constraint problem

From the problem given, there were several possible sequences that can be constructing. In order to give a clear view on how the best sequence was selected, comparison between five possible sequences have been made. Table 4.4 shows the comparison for the possible solution that can minimize the transition time. The

transition time for each machine was assumed as constant. Therefore, all the numbers that shown in table 4.5 in the possible solution column represented as weight for transition time for each machine starting from machine number 1 until machine number 6.

Five different kind of analysis has been done using genetic algorithm toolbox and the result was as shown in table 4.5. From the results given in table 4.5, there were five different sequences that can be constructed based on the results. The sequences were constructed based on the orientation from the lowest number of weight until the highest number of weight. This is because by performing the fastest process first, the production time can be minimizing by reducing the wasting time from the longest process.

The sequence for weight 1 was 6, 4, 5, 1, 3, and 2. For weight 2, the sequence was 4, 2, 6, 1, 5, and 3. It was followed by weight 3 with the sequence 3, 5, 1, 6, 2, and 4. For weight 4 and weight 5 the sequence was 2, 1, 5, 4, 6, 3 and 2, 3, 5, 4, 6, 1.

The best sequence was determined by selecting the lowest fitness function value. Therefore, weight 1 was selected as the best answer for the case study II with 0.8128 as fitness value as shown in table 4.4. The fitness function was calculated by adding all the weight for each machine. As a result of the analysis, 6, 4, 5, 1, 3, and 2 sequence was selected as the best sequence that can minimize the transition time.

The performance graph for the best sequence that is the sequence for weight 1 was shown in figure 4.6. The value of the fitness function can be determined from the graph given in figure 4.6. The value of the fitness function was selected by selecting the value at the graph that has been constant. The graph has been constant starting from the 15^{th} generation until 20^{th} generation as shown in figure 4.6. Therefore the best fitness value was 0.8128. The value for the weight for each machine can be collected from the genetic algorithm toolbox after the analysis has been done. The weight value for machine 1 was 0.13154 followed by machine 2, 0.23991, machine 3, 0.2313, machine 4, 0.02233, machine 5, 0.06098 and machine 6, 0.12672. All the value was shown in figure 4.7.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

This chapter summarizes and concludes the research which related to the research objectives. Then, several recommendations for further research works are presented.

5.1 RESEARCH SUMMARY

In this research, few problems of genetic algorithm method and flow shop sequencing problem or assembly line problem was highlighted. One of the problems is how to apply genetic algorithm method. Genetic algorithm method was an optimization method that can solve variable kind of optimization problems. There are two different ways on how to apply genetic algorithm method in order to solve combinatorial optimization problems.

The first way to apply genetic algorithm method is by using the genetic algorithm toolbox where this research was focus on. Genetic algorithm toolbox was designed to help the user find the solution in the simple way. However the genetic algorithm toolbox has a limitation on what types of problems that can be solve. One of these research objectives is to determine the limitation of genetic algorithm toolbox. This research has successfully found the limitation of genetic algorithm toolbox.

The second ways of applying genetic algorithm method is by constructing new algorithm or writing programming based on assembly line problem. Various kind of assembly line problem such as assembly line with precedence constraint

problem or an assembly line without precedence constraint problem can be solving. The way on how to write the programming depends on the creativity by following a certain command.

Flow shop sequencing problems (FSP) also know as an assembly line problem. FSP problems have been well studied in the field of combinatorial optimization. The problems were an optimization problems that normally faced by the company that produced product based on an assembly production. There were various kinds of ways to solve optimization problems such as tabu search, neural networks and genetic algorithm.

In order to determine the limitation of genetic algorithm toolbox, the real data was collected from the real case study that is in case study I. Numerical experiment was performed based the data collected using genetic algorithm toolbox. As a result of the numerical experiment, it shows that the genetic algorithm toolbox cannot be used in order to solve real case study.

The real case study shows the problem with precedence constraint with processing and transition time on each machine used. This kind of optimization problem cannot be solving using genetic algorithm toolbox because of the genetic algorithm toolbox limitation. Therefore, case study II was constructed in order to see the performance of genetic algorithm toolbox. The result for genetic algorithm toolbox performance was given in chapter 4.

5.2 RESEARCH CONCLUSIONS

In relation to the research objectives;

1. The research had successfully determined the limitation of genetic algorithm toolbox. There was two limitation of genetic algorithm toolbox:
    a. Genetic algorithm toolbox only can solve flow shop sequencing problem without precedence constraint
    b. The transition time from each machine must be assumed as constant in order to solve the problem using genetic algorithm toolbox.

2. The genetic algorithm toolbox had successfully being implemented to the modified assembly line sequencing problem.

From this research, the significant contribution is that the genetic algorithm toolbox is enabling to learn genetic algorithm method to solve modification of flow shop sequencing problem in the easier way before learn to solve flow shop sequencing problem by typing the programming

.

5.3 RECOMMENDATIONS

Several recommendation for future study on solving flow shop sequencing problem or also known as an assembly line problem by using genetic algorithm method are proposed. The first recommendation is genetic algorithm toolbox only suitable to solve the assembly line without precedence constraint. The transition time also need to be assuming as constant in order for the problem to be solving using genetic algorithm toolbox.

Another recommendation is learn to write the programming in order to solve variable type of assembly line problems. Learning to write programming will give a huge benefit. Various types of assembly line problems such as problems with precedence diagram with transition time can be solve. It only depends on creativity to manipulate the programming command to meet the problem.

REFERENCES

Arnold Reisman, Ashok Kumar, and Jaideep Motwani. 1997. Flow shop Scheduling/Sequencing Research: A Statistical Review of the Literature, 1952–1994. IEEE transactions on engineering management. 44(3): 316-329

D. L. Santos, J. L. Hunsucker and D.E. Deal. 1995. An evaluation of sequencing heuristic in flow shops with multiple processors. *Computer Industrial engineering.* 30(4): 681-692

Jose M. Framinan, Rainer Leisten and Rafael Ruiz-Usano. (2002). Efficient heuristics for flow shop sequencing with the objectives of makespan and flow time minimization. *European Journal of Operational Research*

Kylie Bryant. 2000. Genetic Algorithms and the Traveling Salesman Problem. Department of mathematic, Harvy Mudd College

Michael Negnevitsky. 2004. *Artificial Intelligence – A guide to intelligence systems.* Addison-Wesley

Mitsuo Gen. 1997. *Genetic Algorithms and engineering design.* New York: John Wiley and Sons, Inc.

Paulo M. Franca, Gilberto Tin Jr. and Luciana Buriol. The no-wait flow shop problem with sequence dependent setup time and release dates. *European Journal of Operational Research*

Quan-Ke Pan, M. Fatih Tasgetiren and Yun-Chia Liang. 2008. A discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem. *Computers & Operations Research.* 35: 2807 – 2839

Srikanth K. Iyer and Barkha Saxena. (2003). Improved genetic algorithm for the permutation flow shop scheduling problem. *European Journal of Operational Research*

APPENDIX A

Gantt chart

Gantt chart for FYP 1

Project activities	**Weeks**													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Literature study														
Identify problem statement														
Define objective and scope of study														
Detailed methodology														
Proposal preparation														
Presentation preparation														
FYP 1 presentation														

Gantt chart for FYP 2

Project activities	**Weeks**															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Identify new problem																
Analyze the problem																
Report writing																
Presentation preparation																
FYP 2 preparation																
Supervisor and panel approval																
Report submission																

APPENDIX B

Example of Genetic Algorithm toolbox problem solving

(Rastrigin's function)

To get better understanding about genetic algorithm method using genetic algorithm tool, an equation called rastrigin's function is used as case study II. Rastrigin's function **($Ras\ (x) = 20 + x_1^2 + x_2^2 - 10\ (cos\ 2\pi x_1 + cos\ 2\pi x_2)$)** is often used to test the genetic algorithm. Therefore it will surely help to get better understanding about genetic algorithm method using genetic algorithm tool. The objective of case study II is to find the minimum of Rastrigin's function using genetic algorithm method using genetic algorithm tool.

Rastrigin's function is often used to test the genetic algorithm, because its many local minima make it difficult for standard, gradient-based methods to find the global minimum

As proposed by Moon, the parameters are as follows.

- Total number of generation, ngener = 20
- Population size, P = 10
- Probability of crossover, $P_c = 0.6$
- Probability of mutation, $P_m = 0.2$

Numerical Experiment Results for Case Study II: Rastrigin's function

$Ras\ (x) = 20 + x_1^2 + x_2^2 - 10\ (cos\ 2\pi x_1 + cos\ 2\pi x_2$

The results for Case Study II is shown in figure 4.2.

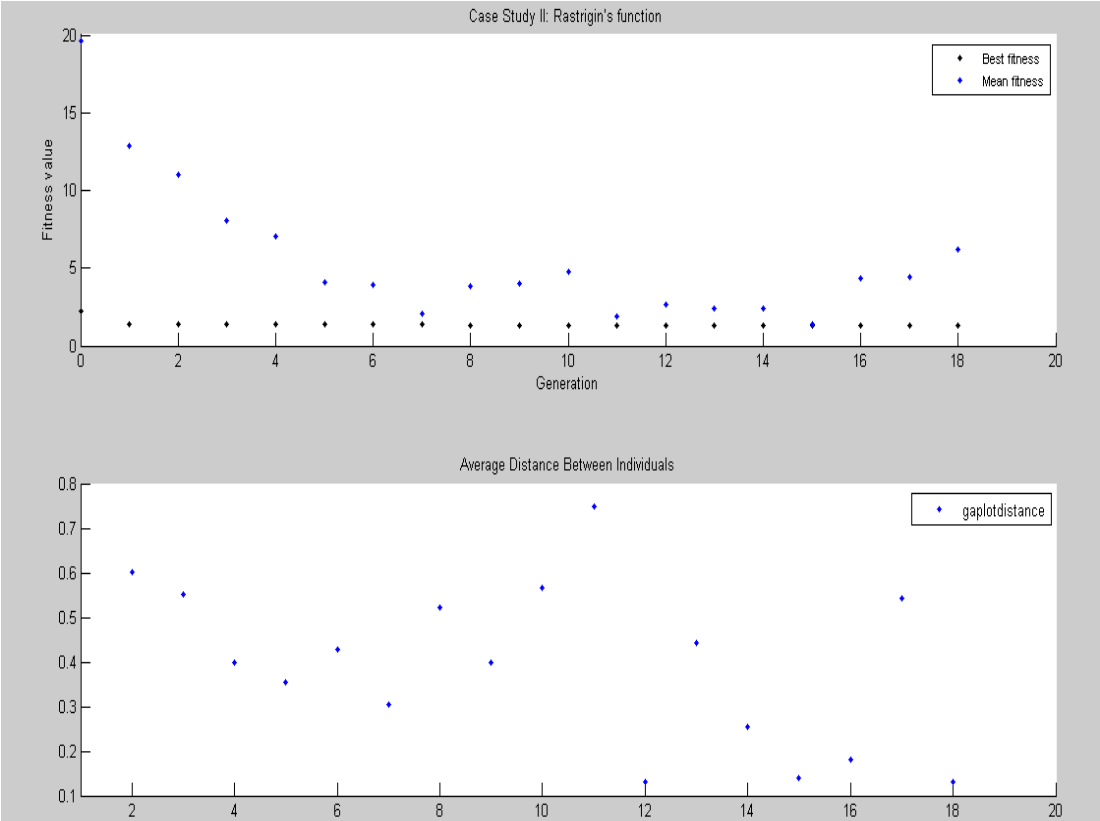

Figure 4.2: Plot of Fitness Value versus Generation and The distance Between Individuals for Case Study II: Rastrigin's function

Analysis of Numerical Experiment Results for Case Study II: Rastrigin's function

The best fitness value for case study Case Study II: Rastrigin's function is 0.90244 with the mean fitness 2.8372 as shown in the figure 4.2 above. The final point for this case study was 0.0185 and 0.06531 that stands for the value of case x_1 and x_2.

APPENDIX C

M-file for case study II

M-file 1

```
1) function y= my_fun(x)
2) y =x(1)+x(2)+x(3)+x(4)+x(5)+x(6);
```

M-file 2

```
1) function y= my_fun(x)
2) y =x(2)+x(3)+x(4)+x(5)+x(6)+x(1);
```

M-file 3

```
1) function y= my_fun(x)
2) y =x(3)+x(4)+x(5)+x(6)+x(1)+x(2);
```

M-file 4

```
1) function y= my_fun(x)
2) y =x(4)+x(5)+x(6)+x(1)+x(2)+x(3);
```

M-file 5

```
1) function y= my_fun(x)
2) y =x(5)+x(6)+x(1)+x(2)+x(3)+x(4);
```

APPENDIX D

Example of programming code

Main program

```
function GA_FSP
clear

disp('=====================================================')
disp('Genetic algorithm for Flow shop sequencing problem with precedence')
disp('=====================================================')

%pause
tic;
num=6; %no of machine

disp('')
fprintf(1,'num=%.0f;  no of machine\n',num);
disp('')

rand('seed',1.4929e+009);
machine_location=(rand(num,2));

figure
plot(machine_location(:,1),machine_location(:,2),'.r','markersize',25)
hold on

for i=1:num;
   text(machine_location(i,1)+0.02,machine_location(i,2),sprintf('%g',i));
end

%disp('Hit any key to plot all available connection between the machine.')
%pause

for n=1:num
   for i=1:num
      plot([machine_location(n,1) machine_location(i,1)],[machine_location(n,2)
machine_location(i,2)])
   end
end

%data of procwssing time (in this case it is time)
processing_time=[0 7 5 6 10 9
   7 0 14 6 10 8
   5 14 0 16 16 10
   6 6 16 0 10 6
   10 10 16 10 0 12
   9 8 10 6 12 0];
```

```
nind=10;  % size of chromosome population
ngenes=num;  % Number og genes in a chromosome
Pc=0.6;   % Crossover probability
Pm=0.2;   % Mutation probability
ngener=20; % Number of generatons
n_show=5;  % Number of generations between showing the progress

disp('')
fprintf(1,'nind=%.0f;  size of the chromosome population\n',nind);
fprintf(1,'Pc=%.1f;  Crossover probability\n',Pc);
fprintf(1,'Pm=%.3f;  Mutation probability\n',Pm);
fprintf(1,'ngener=%.0f;  Number of generation\n',ngener);
fprintf(1,'n_show=%.0f;  Number of generation between showing the
progress\n',n_show);
disp('')

fprintf(1,'Hit any key to generate a population of %.0f chromosomes.\n',nind);

chrom=[];

for k=1:nind
   chrom(k,:)=randperm(20);
end
'chrom';

rout=[chrom chrom(:,1)];

for f=1:nind

   brout=rout(f,:);
   routerepair5;

   routtemp(f,:)=newroute;
end
'routtemp';
rout=[routtemp routtemp(:,1)];

% Calcutate the chromosome fitness
ObjV=evalObjFun(rout,city_distance,nind,ngenes);
best=min(ObjV);
ave=mean(ObjV);

[a b]=min(ObjV);
chrom(b,:)
rout(b,:)

figure('name','The best rout found in the iniial population');
plot(machine_location(:,1),machine_location(:,2),'.r','markersize',25)
```

```
title([The total processing time:',num2str(a)'];
hold on

for i=1:ngenes;
   text(machine_location(i,1)+0.02,machine_location(i,2),sprintf('%g',i));
    plot([machine_location(rout(b,i),1)
machine_location(rout(b,(i+1)),1)],[machine_location(rout(b,i),2)
      machine_location(rout(b,(i+1)),2)])
end
hold

disp('')

ncross=0;
ccount=0;
for m=1:(ngener/n_show)
   for i=1:n_show

      % Fitness evaluation
      fitness=(1./ObjV)';

      % Roulette wheel selection
      numsel=round(nind*0.9); % The number of chromosomes to be selected for
reproduction
      cumfit=repmat(cumsum(fitness),1,numsel);
      change=repmat(rand(1,numsel),nind,1)*cumfit(nind,1);
      [selind,j]=find(change<cumfit & change>=[zero(1,numsel);cumfit(1:nind-1,:)]);
      newchrom=chrom(selind,:);

      % Crossover
      points=round(rand(floor(numsel/2),1).*(ngenes-1))+1;

      points=[points round(rand(floor(numsel/2),1).*(ngenes-1))+1];
      points=sort((points*(rand(1)<Pc)),2);

      for j=1:length(points(:,1))

         swap_sect=newchrom(2*j-1*j,points(j,1)+1:points(j,2));
         remain_sect=newchrom(2*j-1*j,:);
         Pa=remain_sect(1,:);
         Pb=remain_sect(2,:);

         if rand(1)<Pc
            Cross12;
         end
         c1=Pa;
         c2=Pb;

         remain_sect=[c1;c2];
```

```
    newchrom;
   newchrom(2*j-1:2*j)=[remain_sect(1:2,:)];

   remain_sect=[];
end

% Mutation

for i=1:numsel
   if rand(1)<Pm
      P=newchrom(i,:);

      J=20;

      sel=randint(1,2,[1,20]);
      PP=P;

         PP(sel(1))=P(sel(2));
         PP(sel(2))=P(sel(1));
         newchrom(i,:)=P;
   end
end

% Creating a new population of chromosomes

if nind-numsel, % Preserving a part of the parent chromosome population
   [ans,Index]=sort(fitness);

   chrom=[chrom(Index(numsel+1:nind),:);newchrom];
else % Replacing the entire parent chromosome population with a new one
   chrom=newchrom;
end

% Fitness calculation
rout=[chrom chrom(:,1)];

routtemp=[];

for f=1:nind
   brout=rout(f,:);
   routerepair5;
   routtemp(f,:)=[newroute];

end
routtemp;
rout=[routtemp routtemp(:,1)];

ObjV=evalObjFun(rout,processing_time,nind,ngenes);
```

```
        best=[best min(ObjV)];
        ave=[ave mean(ObjV)];
        ccount=ccount+1
        time(ccount)=toc;
        aan(ccount)=min(ObjV);
    end

    [a b]=min(ObjV);
    rout(b,:)

    % Plotting the best rout found in the current population
    figure('name','The best round found in the current population');
    plot(machine_location(:,1),machine_location(:,2),'.r','markersize',25)
    title(['Generation#',num2str(m*n_show),' The total distance:',num2str(a)]);
    hold on

    for i=1:ngenes;
        text(machine_location(i,1)+0.02,machine_location(i,2),sprintf('%g',i));
        plot([machine_location(rout(b,i),1)
machine_location(rout(b,(i+1)),1)],[machine_location(rout(b,i),2)
            machine_location(rout(b,(i+1)),2)])
    end
    pause(0.2);
    hold
    newchrom=[];
end

disp('')
%disp('Hit any key to display the performance graph.')
%pause

figure('name','Performance graph');
plot(0:ngener,best,o:ngener,ave);
legend('Best','Average',0);
title(['Pc=',num2str(Pc),',Pm=',num2str(Pm)]);
xlabel('Generation');
ylabel('Time taken')

figure('name','Performance graph');
plot(0:ngener,best);
legend('Best',0);
title(['Pc=',num2str(Pc),',Pm=',num2str(Pm)]);
xlabel('Generation');
ylabel('Time taken')

plot(time,aan)

function ObjV=evalObjFun(rout,machine_distance,nind,ngenes)
```

```
path=0; ObjV=[];
for k=1:nind
   for i=1:ngenes
      path=path+machine_distance(rout(k,i),rout(k,(i+1)));
      if i==ngenes
         path=path+machine_distance(rout(k,i),rout(k,(i+1)));
      end
   end
   ObjV(k)=path;path=0;
end
```