SCHEDULING IN ROBOTIC CELLS WITH TWO AND THREE MACHINES

BY

TAO LUAN

A thesis submitted to the Faculty of Graduate studies
In Partial Fulfilment of the Requirements for the Degree of

MASTER OF SCIENCE

Department of Mechanical and Manufacturing Engineering
University of Manitoba
Winnipeg, Manitoba

© May, 2005

THE UNIVESITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

SCHEDULING IN ROBOTIC CELLS WITH TWO AND THREE MACHINES

BY

TAO LUAN

A Thesis/Practicum submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science

TAO LUAN © 2005

Permission has been granted to the Library of the University of Manitoba to lend or sell copies of this thesis/practicum, to the national Library of Canada to microfilm this thesis and to lend of sell copies of the film, and to University Microfilm Inc. to publish an abstract of this thesis/practicum.

The author reserves other publication rights, and neither this thesis/practicum nor extensive extracts from it may be printed otherwise reproduced without the author's written permission.

Acknowledgments

I would like to thank Dr. Q. Peng, my advisor, whose lectures interest me in this work. He has given me lots of help and instruction during the work. Without his knowledge, ideas, and patience, this work cannot be finished.

I am appreciative of Prof. Dr. S. Balakrishnan's patience, advice, and encouragement. He has spent a lot of time reviewing this thesis and provided many helpful comments. I am also grateful to Dr. K. Snelgrove for his review and suggestions.

I take this opportunity to express my gratitude to Chulho Chung, Chunseng Yu, Muhammad Ahsan, and Xiumei Kang. They help me and share their knowledge with me in the work.

I should appreciate my family as well. They support and encourage me throughout my study. Without them, I would never be able to complete this work.

Abstract

For part schedules in robotic cells with two and three machines, identical and different part types are discussed. The objective of this work is to find the optimal cycle time for such cells.

For identical parts, different sequences of robot moves generate different cycle time in robotic cells with two or three machines. To find the optimal cycle time, a simulation tool is used, which proves to be efficient for analyzing the effects of various parameters and the implications of various alternatives.

For different parts, both the sequences of part input and the sequences of robot moves result in different cycle time. To find a solution of such problems, which are characterized by a fixed path, a genetic algorithm is proposed. The algorithm employs natural chromosome representation. A point cut crossover operator is developed for the operation of chromosome crossover. A mutation operator is also designed to select two genes randomly in a chromosome and to exchange their positions. The optimal part sequence and cycle time are obtained through processing genetic evaluation and improving the solution performance. Examples are presented, which demonstrate that the genetic algorithm can solve the problems easily.

Table of Contents

Acknowledgments i

Abstract ii

Table of Contents iii

List of Figures vi

List of Tables viii

Chapter **Page**

1. Introduction 1
   1.1 Background 1
   1.2 Objectives 4
   1.3 The structure of the thesis 6
2. Literature review 8
   2.1 Identical part type in a robotic cell 8
   2.2 Different part types in a robotic cell 12
   2.3 Summary 16
3. Robot moving cycles in the cells with two or three machines 19
   3.1 Sequences of robot moves in a robotic cell with two machines 20
      3.1.1 Alternative 1 20
      3.1.2 Alternative 2 21
   3.2 Sequences of robot moves in a robotic cell with three machines 23
      3.2.1 Alternatives 1 and 3 23
      3.2.2 Alternatives 4 and 5 26
      3.2.3 Alternatives 2 and 6 29
   3.3 Discussion of these sequences of robot moves 35
4. Simulation of identical part type problems in robotic cells with two and three machines 37
   4.1 Program description 37

4.1.1 Overall program 38
4.1.2 Description of the simulation program 45
4.2 Examples and analysis of results 49
4.2.1 Examinations of the robotic cell program with two machines 49
4.2.2 Examinations of the robotic cell program with three machines 51
4.3 Conclusions of the chapter 53
5. Scheduling different part types in robotic cells with two and three machines 55
5.1 Outline of Genetic Algorithms 55
5.2 The Implementation of Genetic algorithms 57
5.2.1 Encoding scheme 57
5.2.2 Initial population 58
5.2.3 Fitness evaluation function 58
5.2.4 Parent selection 58
5.2.5 Genetic operators 59
5.2.5.1 One point crossover 59
5.2.5.2 Mutation-Arbitrary two parts change 60
5.2.6 Evolution Strategy 61
5.3 Overall algorithm 61
5.4 The GA program 62
5.4.1 Initial population function 66
5.4.2 Objective function 66
5.4.3 Selection function 67
5.4.4 Crossover function 67
5.4.5 Mutation function 67
5.4.6 Selecting new population function 68
5.5 Testing of the GA program 68
5.6 Experiments and results 72
5.6.1 Different part types scheduling in the robotic cell with two machines 72
5.6.2 Different part types scheduling in the robotic cell with three machines 74
6. Conclusions and Future work 79
References 82

APPENDIX: Publications 86

1. Q. Peng and T. Luan, "INTERNET-BASED TEACHING AND LEARNING FOR THE ROBOT CELL DESIGN", The E-Commerce – 2003 Workshop on E-Commerce and Education, 24 July 2003, Hong Kong, PP.74-79.

2. T. Luan and Q. Peng, "Genetic Algorithm-based Parts Scheduling in a Two-machine Robotic Cell", Flexible Automation and Intelligent Manufacturing, FAIM2004, Toronto, Canada, PP.974-980.

List of Figures

Figure **Page**

3.1 Alternative 1 for a robotic cell with two machines 21
3.2 Alternative 2 for a robotic cell with two machines 21
3.3 Alternative 1 for a robotic cell with three machines 23
3.4 Alternative 3 for a robotic cell with three machines 24
3.5 Alternative 4 for a robotic cell with three machines 26
3.6 Alternative 5 for a robotic cell with three machines 27
3.7 Alternative 2 for a robotic cell with three machines 29
3.8 Alternative 6 for a robotic cell with three machines 32
4.1 A graphical user interface for a robotic cell with two machines 38
4.2 A graphical user interface for a robotic cell with three machines 39
4.3 The sequences of robot moves in a robotic cell with two machines 40
4.4 The states that cycle time begins 41
4.5 The sequences of robot moves in a robotic cell with three machines 42
4.6 The states that cycle time begins for alternative 1 and 2 43
4.7 The states that cycle time begins for alternative 3 and 4 43
4.8 The states that cycle time begins alternatives 5 and 6 44
4.9 The flow chart of the program 45
5.1 The representation of a chromosome 57
5.2 One point cut crossover 60
5.3 Arbitrary two parts exchange mutation 61
5.4 A graphical user interface for different part types in a robotic cell with two machines 63
5.5 A graphical user interface for different part types in a robotic cell with three machines 63
5.6 The flow chart of the program 65
5.7 The cycle time obtained by using GA program for a robotic cell with two machines 69

5.8 The cycle time obtained by using GA program for a robotic cell with three machines 71
5.9 The converging process of alternative 2 for a robotic cell with two machines 73
5.10 T_2 cycle time when Pc=0.5 and Pm=0.5 of alternative 2 for a robotic cell with two machines 74
5.11 Optimal parts sequence and cycle time of alternative 2 for a robotic cell with two machines 74
5.12 The converging process of alternative 2 for a robotic cell with three machines 75
5.13 The converging process of alternative 3 for a robotic cell with three machines 76
5.14 The converging process of alternative 4 for a robotic cell with three machines 76
5.15 The converging process of alternative 5 for a robotic cell with three machines 77
5.16 The converging process of alternative 6 for a robotic cell with three machines 77
5.15 The optimal part sequences and cycle time for a robotic cell with three machines 78

List of Tables

Table **Page**

4.1 Input data of the robotic cell with two machines 50
4.2 Calculated cycle time of the robotic cell with two machines 50
4.3 Simulated cycle time of the robotic cell with two machines 50
4.4 Input data of the robotic cell with three machines 51
4.5 Calculated cycle time of the robotic cell with three machines 51
4.6 Simulated cycle time of the robotic cell with three machines 52
5.1 Input testing data for a robotic cell with two machines 69
5.2 Input testing data for a robotic cell with three machines 70
5.3 The part number, type and processing time for a robotic cell with two machines 72
5.4 The part number, type and processing time for a robotic cell with three machines 75

Chapter 1 Introduction

1.1 Background

Manufacturing has moved towards a highly automated and computer integrated flexible process as a result of developments in advanced computer and information technology. A general flexible manufacturing system (FMS) has been defined as an automated system composed of computer-controlled machining centers and automated material-handling devices. Computer-controlled machining centers are used to produce parts. The automated material-handling devices are used to transfer parts between machines/ processes. Machines in an FMS are usually equipped with necessary cutting tools and capable of performing different operations [1].

An FMS often works at four different levels: i) factory or enterprise level, ii) shop or line level, iii) cell level, and iv) component level [2]. At the cell level, the production flexibility and efficiency of an FMS depend on a set of multi-purpose machines and automated transport/material handling systems. The multi-purpose machines are capable of performing different operations with minimum set-up time between successive operations. The automated transport/material handling systems enable parts to move between machines. A network of supervisory computers directs and coordinates the simultaneous processing of many parts.

In a flexible manufacturing cell, industrial robots are widely used and the cell is called robotic cell. A robotic cell may have one or several robots and several machines. The

machines are arranged within the reach of a robot. The robot performs all material handling functions including loading and unloading parts among the machines. In an FMS unit or a robotic cell, machines are normally computer numeral controlled (CNC) or automated. Typical machining processes include milling, drilling, shaping, grinding, and cutting.

There are many applications of robotic cells in industry, which include the manufacture of semiconductors, glass products, cosmetics, printed circuit boards, and building products [3]. The efficient use of robotic cells increases production rates, which results in enhanced manufacturing competitiveness. Robotic cells in industry are typically designed for repetitive operations [4]. In robotic cells, there are usually several machines arranged within the reach of a robot. The robot loads and unloads parts among machines in robotic cells. For example, a robot can be used to pick up a part from an input buffer, load and unload a lathe machine, and then transfer it to a drill machine, and finally discharge it in an output buffer.

In an FMS, the type of material handling devices determines the layout of machines for processing a part or a family of parts. If a handling robot is used in a robotic cell, the machines are usually placed in a circle [5]. If input and output devices are arranged along a semicircle, the robot is normally located in the center of the semicircle and transports parts between machines and input/output devices. This type of cell is referred to as a robot-centered cell. If the robot is on a rail and can move along the rail in a cell to load and unload parts on the machines, the robotic cell is called a mobile-robot cell. Some

applications combine the above two layouts: the robot's arm rotates on its base, and the robot moves linearly along a track. This is also referred to as a mobile robot cell [6]. A commonly used cell layout is robot-centered. The robotic cells consist of m machines arranged in a circular layout and served by a single central robot.

Robotic cells with one robot have been widely used in manufacturing companies. A real-world example of a two-machine robot-centered cell is presented by Xerox Corporation, Rochester, NY, which has three robotic cells. Each cell is served by a dual-gripper 2105 B Unimate robot and consists of two CNC lathes, to produce a family of products called "duplicator fuser rollers" [7]. Multiple CNC machines and robot programs facilitate fast changeover from one product to another of the family of roller designs [8]. Another example is a robotic cell with three machines used in a production line of machine castings [9]. The robotic cell is used for truck differential assemblies where a robot transfers heavy mechanical parts between large machines. The system contains a conveyor belt for the incoming parts and another belt for the outgoing parts. In this particular system, the robot is not able to traverse the conveyor. Therefore, the movement of the robot, from output to input, has to traverse the entire cell.

There have been many studies on the robot-centered cells. In such robotic cells, the machines have no input or output buffers, each machine can process one part at a time, and the part waits on the machines until the robot picks it up. After processing a part, the machine is blocked until the robot removes the part. This research was focused on the sequences of robot moves and on scheduling parts. Efficient sequences of robot moves

and favorable schedules of parts in a robotic cell can greatly increase productivity for manufacturing industries.

The objective of this study is to find the optimal sequence of robot moves and parts to maximize production rate.

1.2 Objectives

The research discusses identical and different part type problems in robotic cells with two and three machines. The objective of the research is to find

- The robot sequences in robotic cells with two and three machines to optimize the cycle time for the identical part type problems.
- The part schedule at input buffer that maximizes the long-run average production rate for a certain robot movement sequence in robotic cells with two and three machines, given a family of parts with a minimum part set (MPS).

In robotic cells, special fixtures and jigs are often changed at regular intervals to match the production rates of different products. For identical products or a family of products, the same fixtures can be used [5]. In this work, the robot sequence problems of the movement of identical and different part types are analyzed, and scheduling a cell with two and three machines is considered. There are no intermediate buffers between machines. The operation procedure of the robot-centered cell is assumed as follows:

(1). Each part will go through all elements of the cell in the order IN, M_1, M_2, OUT for a robotic cell with two machines and IN, M_1, M_2, M_3, OUT for a robotic cell with three machines. IN and OUT represent input buffer and output buffer. M_1, M_2, and M_3 represent Machines 1, 2, and 3, respectively.

(2). Each machine processes only one part at a time.

(3). The set-up time is included in the processing time.

(4). The time of each pickup, loading, unloading, and drop-off is the same.

(5). The robot travel time between any 2 adjacent positions is the same.

For the identical part type problems, the sequence of different robot movements affects the production rate. The theoretical methods have been provided by Sethi et al [10]. Sethi's formulation is used to find the optimal sequence of robot movements. A simulation tool has been developed for identical part type problems. It provides a quick visual simulation for understanding the process in order to search for the optimal solution. By comparing the simulation result with the result calculated by using Sethi's method, the effectiveness of the tool developed can be verified.

For the different part type problems, we focus our attention on the input sequence of parts. Different input sequences generate different cycle time. The research is to find the maximum throughput rate or the minimum cycle time. The problems belong to the NP-hard problem (non-deterministic polynomial time hard) [11]. NP-hard is defined as

follows. There are N problems. If a polynomial time can solve one of the N problems and the polynomial time can be translated to solve any other problems by using an approach, the problems are NP-hard problems. Different approaches for the problem have been proposed, such as branch-and-bound techniques [12]. However, these techniques are not very effective for large-sized or even medium-sized part problems [13].

A genetic algorithm is proposed to find a solution to the problem that is characterized by a fixed path. The fixed path limits the sequences of robot moves in a robotic cell with two or three machines. Therefore, the problem belongs to part input permutation. The proposed algorithm employs the natural chromosome representation of genetic algorithm. One point cut crossover operator is selected for chromosome crossover operation. A mutation operator is designed to select two genes randomly in a chromosome and to exchange their positions. The objective function of the maximum production rate is used to determine the fitness of each chromosome. The optimal part sequence and cycle time are obtained through processing genetic evaluation in order to improve the solution. Examples are presented and their results are analyzed.

1.3 The structure of this thesis

This thesis is organized as follows. A literature review is presented in Chapter 2. The sequences of robot moves in robotic cells with two and three machines are introduced in Chapter 3. A simulation tool is developed in Chapter 4 to simulate the robotic cells with

two and three machines for identical part type. Genetic algorithms are designed in Chapter 5, to solve the problems of different parts. Conclusions and future work are described in Chapter 6.

Chapter 2 Literature review

Robots are being extensively used in flexible manufacturing systems. A major application of robots is in robotic cells. Since robots in such cells usually perform a repeated operation, the optimal scheduling of robot moves is very important. Most research has focused on a single robotic cell, which consists of an input device, several machines, an output device, and a robot. The machines perform different processes on each part. The robot transports parts from machines to I/O (Input/Output) devices, or between machines. It is assumed in this research that there are no buffers for intermediate storage in the robotic cell. All parts must be either in the I/O device, on one of the machines, or held by the robot. Based on these assumptions, the research has aimed at obtaining the minimum cycle time for processing a set of parts that may all be identical or have different processing requirements. The common goal is to maximize productivity. For identical parts, the objective is to determine the sequence of robot moves for maximum part throughput. For parts with varying processing times, the objective is to decide the optimum sequence of robot moves and part schedule to maximize the part throughput. Therefore, the research aim can be summarized as determination of the order of different parts entering the cell, and the sequence of robot moves.

2.1 Identical part type in a robotic cell

For identical part types, the work undertaken by different researchers is to decide sequences of robot moves in one-unit cycle. A one-unit cycle is a sequence of robot moves in which each machine is loaded and unloaded exactly once. Without loss of generality, a robotic cell with m machines was considered by Crama and Klundert [14].

In such a robotic cell, there are exactly $m!$ one-unit cycles. Activity A_i ($i = 0,...,m,$) consists of the following sequence of robot moves: unload M_i; travel from M_i to M_{i+1}; and load M_{i+1}. Each one-unit cycle defines a permutation of the activities starting with A_0. Different permutations represent different cycles. Computing the optimal one-unit cycle is equivalent to computing the optimal permutation of the activities. For a robotic cell with two machines, there are 2 (2!) one-unit cycles. The permutations are $\{A_0, A_1, A_2\}$ and $\{A_0, A_2, A_1\}$. For a robotic cell with three machines, there are 6 (3!) one-unit cycles. The permutations are $\{A_0, A_1, A_2, A_3\}$, $\{A_0, A_2, A_1, A_3\}$, $\{A_0, A_1, A_3, A_2\}$, $\{A_0, A_3, A_1, A_2\}$, $\{A_0, A_2, A_3, A_1\}$, and $\{A_0, A_3, A_2, A_1\}$. Therefore, the sequences of robot moves in a robotic cell can be easily obtained. The objective is to find a sequence of robot moves that maximizes the throughput rate or minimizes the cycle time.

Sethi et al [10] discussed sequences of robot moves in robotic cells with two and three machines for identical part type based on known robot travel time and the time to pickup, load, unload, and drop parts. The objective is to find the optimal sequence of robot moves in robotic cells with two and three machines.

For the robotic cell with two machines, there are only two different cycles (S_1 and S_2). For cycle S_1, the robot loads machine one, waits for the processing to complete, transfers the part to machine two, waits until processing is complete before delivering the part to the output station. For cycle S_2, the initial state of the two different cycles is that the robot has just completed loading it on machine M_2. The robot begins to move from this state. The cycle with a shorter cycle time is what is desired.

Sethi et al [10] also addressed the problem of finding the optimal sequence of robot moves in a robotic cell with three machines. They identified six different cycles ($S_1, S_2, S_3, S_4, S_5,$ and S_6), among which four cycles ($S_1, S_3, S_4,$ and S_5) are very simple, since their cycle time can be formulated easily. The other two (S_2 and S_6) are complicated, whose cycle time can be formulated under assumption of the cell being in steady state. The formulations show that the four (S_1, S_3, S_5 and S_6) of six cycles dominates the other two. The optimal cycle time comes from the four dominated cycles.

Crama et al [14] investigated identical part type problems where the number of machines is arbitrary. If there are m machines in a robotic cell, there are exactly $m!$ permutation of activities. The search for the optimal cycle is equivalent to finding the optimal permutation of robot moves starting with A_0. The robot moves can be either pyramidal permutation or no pyramidal permutation. The number of the pyramidal permutations is 2^{m-1}. It has been proved that there is an optimal cycle in the pyramidal permutations. If the number of machines is fixed, the optimal cycle time can be obtained (in polynomial time) from the pyramidal permutations.

Logendran et al [6] studied the sequence of robot moves in a robotic cell with two machines. The time spent in a robot cycle was defined as: ε_1 for picking up a part from the input buffer I; ε_2 for loading the part on machine M_1; ε_3 for unloading the part from machine M_1; ε_4 for loading the part on machine M_2; ε_5 for unloading the part from machine M_2; and finally ε_6 for dropping the part at the output buffer O. To simulate the real motion of a robot in the cell, the robot moving from one position to another was considered as three different phases, namely acceleration, followed by constant velocity,

and finally deceleration to rest at the point of destination. The robot application is improved by considering the robot's acceleration and deceleration. For instance, the robot travel time from I to M_2 directly would be slightly (γ units) less than the travel time from I to M_1 and then from M_1 to M_2, since the latter requires an additional acceleration and a deceleration phase compared to the former. Due to the same reason, the robot travel time of a move directly from I to O would be 2γ units less than the travel time from I to M_1, then from M_1 to M_2, and finally from M_2 to O. The time needed for each pickup, load, unload, and drop operation performed by the robot was assumed to be different. The cycle time of S_1 and S_2 is obtained simply by adding the event times in a polynomial time. Three types of robotic cell layouts were considered, i.e., robot-centered cell, mobile robot cell, and in-line robot cell.

Agentis [15] considered a situation of no-wait constraint. In other words, as soon as a machine completes a part, the robot will immediately move it to the next machine. No-wait processing is typical in the steel industry and some electroplating processes. In fact, the "machines" are treatment furnaces and the "parts" are some materials that are processed in the "machines", and the "robot" is a crane for moving the materials around the workshop. The problems in a robotic cell with two and three machines were discussed. For two machines, the problem is very simple. After a part is loaded on the machine 2, there are two cycles (S_1 and S_2) available. Cycle S_1 is always available. If cycle S_2 is available, the robot moves along cycle S_2. Cycle S_2 is better than cycle S_1, since two parts enter the cell. For three machines, the problem is complex. Due to no-wait constraint, there is no steady state case to consider. The robot begins at input buffer for each cycle. When a part is processed on a machine, the input time of next part must

satisfy the constraint that the empty robot must reach the machine before completion of the processing of the part on the machine. The expressions were given by the author. The optimal cycle can be computed.

2.2 Different part types in a robotic cell

For different part types, the robotic cell is required to repetitively produce families of parts. The parts have similar design features, functions, materials, and processing requirements. Thus, the families of parts are produced by sharing the same fixtures and cutting tools [16]. In a robotic cell, the entire system is under the supervision of a programmable controller. Multiple CNC (Computer numerical control) and robot programs facilitate fast changeover from one part to another of a family. Therefore, different parts can be produced in a robotic cell as identical parts. There have been a few studies about different part types in a robotic cell, aimed to maximize the throughput rate of the cell. For the objective, two decisions have to be made simultaneously. One is the sequence of robot moves. The other is the scheduling of parts.

Due to the trend towards just-in-time manufacturing, a minimum part set (MPS) is required to decide cycle production schedules. The MPS has been introduced by Hall et al [11], which is the smallest possible set of parts. For example, there are three types of parts: A, B, and C. The total production target is 700 parts of type A; 300 parts of type B; and 400 parts of type C. If an MPS consists of 14 parts made of seven A, three B, and four C, the cycle time is defined as the necessary time period to complete processing one MPS. Therefore, the objective is to minimize the time to produce one MPS in a cyclic production environment.

The study of robot moves and scheduling parts in a robotic cell was first introduced by Sethi et al [10]. The optimal part schedule with different part types was considered against a given fixed sequence of robot moves in a robotic cell with two machines. The time of picking up a part from an input buffer, loading a part on a machine, unloading a part from a machine, and dropping a part into an output buffer is assumed to be same for different parts. A polynomial time algorithm for scheduling different part types can be derived. In this case, there are two fixed cycles, i.e., S_1 and S_2. For cycle S_1, the optimal solution is decided by the parameters of machine processing time only, and not by the part-input schedule. For cycle S_2, given the parts schedule, the equation of the cycle time can be formulated by using polynomial time algorithm. The equation can be interpreted as a traveling salesman problem (TSP). Sethi used Gilmore and Gomory algorithm [6] to solve the problem. Gilmore and Gomory algorithm first sort and order the parts, and then a graph is used to find optimal parts schedule. According to the optimal cycle time, the sequence of robot moves and the part schedule can be decided.

Logendran at al [6] analyzed different part types in a mobile robotic cell, an in-line robotic cell, and a robot-centered cell with two machines. Different times were assumed for loading and unloading of different parts on different machines. Gilmore and Gomory's algorithm [6] can be used to find the optimal permutation of parts. The minimized cycle time is then obtained.

Aneja et al [17] also studied a robotic cell with two machines that process different part types. The cycle time is reduced by using both sequences S_1 and S_2 in the repetitive production of MPSs. In a robotic cell with two machines, there are two sequences of robot move cycles, S_1 and S_2. Cycles S_1 and S_2 were introduced in detail by Sethi et al

[10]. There is a common state where the sequence of robot moves can be switched to both S_1 and S_2. The state is that the robot has just finished loading a part on machine 2. In the state, the robot has two possible motions:

(a) Wait and unload the part from machine 2 (as in cycle S_1);

(b) Move back to input buffer, then pick up the next part at input buffer (as in cycle S_2).

For a given part scheduling at an input buffer, after the robot has just loaded a part on machine 2, a decision has to be made if cycle S_1 or S_2 is used for next cycle. An expression for the decision can be obtained by using a polynomial time algorithm. The Gilmore and Gomory's algorithm was used to find the optimal solution and cycles from the expression. By using a combination of different cycles in a robotic cell with two machines, the cycle time can be improved.

The different part type's problem in a mobile-robot cell with three machines was studied by Hall et al [18]. There are six potentially optimal one-unit robot move cycles ($S_1, S_2\ S_3, S_4, S_5$, and S_6) to be considered in the mobile-robot cell with three machines. The production of parts begins from an initial state. The initial state is that the three machines are empty and the robot is at input buffer and ready to pick up a part. The optimal part scheduling problems associated with four of the six cycles (S_1, S_3, S_4, and S_5) are easily solved. The way to solve them is similar to the two cycles (S_1 and S_2) in a robotic cell with two machines. Given a fixed sequence of robot moves, a polynomial algorithm can be used to formulate an expression for the cycle time. The expression finally can be formulated as a TSP [18] (traveling salesman problem). The algorithm of Gilmore and Gomory was used to find the optimal cycle time. However, in the other two

cycles (S_2 and S_6), the part scheduling problems are very complex because initial steps must be considered. The authors found first the solutions for four of the six cycles. When machines' processing time is much longer than the robot move time, the two cycles (S_2 and S_6) are dominated by the other four. Thus the problem of part scheduling in a robotic cell with three machines can still be solved efficiently.

Hall et al [11] analyzed the steady state in a robotic cell with three machines. A steady state is defined as one when the waiting time of the robot is same for each part on each machine. In robot move cycles S_1, S_3, S_4, and S_5, the steady state is that the robot has just loaded a part on a machine and the other machines are all empty. Therefore, it is easy for the cell to converge to a steady state. However, for S_2 and S_6, several cycles are need to reach a steady state. Two different ways to initialize the cell are described by Hall et al [18]. Based on the initialization, Hall et al [11] discussed on how to converge to a steady state. They found that the steady state is bounded by the cell data and gave a guide to initialize S_2 and S_6. No experiments were conducted to support the guide. The general part-sequencing problem, which is not restricted to any robot move cycle in a robotic cell with three machines, is still NP-complete. Both NP and NP hard problem is called NP-complete.

Agentis [15] and Agentis et al [19] concentrated their research on a no-wait robotic cell. A no-wait robotic cell with two machines for different parts was analyzed. Given part schedule, the expression for cycle time can be formulated. The analysis is similar to that of identical part types in a no-wait robotic cell. The expression can be formulated as a TSP. The Gilmore and Gomory's algorithm was used to solve the TSP. The problem of minimizing the cycle time was solved. Agentis et al [19] considered the problem of parts

scheduling for different parts in a no-wait robotic cell with three machines. The feasibility was discussed when the robot move cycle was fixed. The problem of a no-wait robotic cell with three machines is still open.

The above different part type problems in robotic cells belong to scheduling problems. Scheduling is one of the most important tasks in manufacturing systems. Many different approaches to scheduling have been proposed [20]. Since the approaches are complex, heuristic rules are used. A genetic algorithm (GA) is a feasible method to solve the problem. It has been used to plan assembly sequences [21], to optimize a shop floor process with multiple machines within the limits of the due date of products [22], to schedule a flexible manufacturing cell with setup constraint [23], to format part families and machine-component groups [24], to consider maximizing productivity of blocked shop floor operation without buffers [25], and to schedule a "no-wait" shop floor with multiple machines [26]. The Genetic algorithm has proven efficient for many complex computations.

2.3 Summary

Research on a robotic cell with a single robot has been reported in the literatures. Since the number of machines in a robotic cell is usually no more than four [12], most of the previous literature focused on a robotic cell with two machines and three machines. The objective was to maximize the cell throughput, which is decided by the sequence of robot move and the scheduling of different parts. In a robot cell with two machines or three machines, total cycle number of the robot move is small, and is able to be found by

permutation. The above literature review reveals that many researchers concentrated on the problems of identical part types and different part types in a robot cell.

For identical part types, Sethi et al [10] solved the problem of finding optimal cycles of sequences of robot move in a robotic cell with two and three machines. The solution was based on the assumption that the robotic cell was in a steady state. The steady state in a robotic cell with three machines was analysed by Hall et al [11]. There are several cycles required for S_2 and S_6 to reach a steady state. There were no details about how the cell reached a steady and how many cycles were needful.

For different part types, based on the minimum part set (MPS), the objective of most research was to determine the optimal part scheduling when the sequence of robot moves is fixed. The parts scheduling in a robot cell is a complex process. The scheduling of S_2 and S_6 in a robotic cell with three machines is NP-complete. A polynomial algorithm has been used to formulate the cycle time based on a given sequence of robot moves. The algorithm proposed by Gilmore and Gomory has been used when the cycle time is formulated as a TSP. It is tedious and complex to formulate a cycle time in a TSP using a mathematical method. Gilmore and Gomory's algorithm is a graphical format to renumber the part permutations. But the method is restricted to part size.

In recent years, genetic algorithm (GA) has been applied for schedule problems [27]. GA was developed by John Holland in 1970s [28]. Genetic algorithms have the features of flexible, global and stochastic searching mechanism. GA has been proven efficient for many complex problems, which belong to combinational optimization. This research also investigates a robotic cell with two machines and three machines for different parts by using GA. The objective is to find the optimal schedule of different parts and robot

moves. A polynomial time algorithm is used to formulate the cycle time for a given sequence of robot moves. A genetic algorithm is used to find the optimal part schedule based on the given sequence of robot moves. Genetic algorithm is expected to improve the efficiency for scheduling different parts.

Chapter 3 Robot moving cycles in the cells with two or three machines

This Chapter presents the details of problem formulation for robotic cells. Two and three machine robotic cell for processing of parts is considered. The original description and calculation of robot working cycles in robotic cells were reported by Sethi [10]. Detailed simulation procedures of the robot moves are investigated in this Chapter in order to devise a clear sequence plan of the robot moves in robotic cells.

In order to describe parts processing in a robotic cell, the number of parts produced in a single MPS (minimum part set) is assumed as n. In a robotic cell with two machines, there are two sequences of robot moves (S_1 and S_2). In a robotic cell with three machines, there are six sequences of robot moves (S_1, S_2, S_3, S_4, S_5, and S_6). For a given permutation of n parts, the robot moving cycle time can be formulated according to respective sequences of robot moves in robotic cells with two and three machines. The formulations are presented in Sections 3.1 and 3.2, respectively. To formulate the cycle time, the following notations are used in a robotic cell with two or three machines.

M_1, M_2, and M_3 - Machines in the robotic cell.

ε - Time used by robot to pick up, load, unload, or drop off a part.

δ - Robot moving time between any adjacent locations.

$P_{\sigma(1)}$, $P_{\sigma(2)}$, ... , $P_{\sigma(n)}$ - A schedule of a given permutation σ of n parts.

$a_{\sigma(i)}$, $b_{\sigma(i)}$, and $c_{\sigma(i)}$ - Processing time of a part $P_{\sigma(i)}$ on M_1, M_2, and M_3, respectively. $i = 1,2,...,n$.

$w_{1\sigma(i)}$, $w_{2\sigma(i)}$, and $w_{3\sigma(i)}$ -Waiting time of the robot before unloading part $p_{\sigma(i)}$ at M_1, M_2, and M_3, respectively. $i = 1,2,...,n$.

It is assumed that the cell is empty at the initiation of the process and the robot is at the input buffer ready to pick up a part before the production of the n different parts begins. Thus, if the robot moves in a sequence, the cell must be initialized corresponding to the sequence. An initialization is a transient production phase, which has no impact on the long-run throughput rate associated with the sequence. For repetitive production process, one MPS is followed by another MPS. Thus, the processing time $a_{\sigma(n+1)}$ equals $a_{\sigma(1)}$, and $a_{\sigma(n+2)}$ equals $a_{\sigma(2)}$. It is similar for $b_{\sigma(n+1)}$, $b_{\sigma(n+2)}$, $c_{\sigma(n+1)}$, and $c_{\sigma(n+2)}$. The waiting time $w_{i\sigma(n+1)}$ equals $w_{i\sigma(1)}$, $i = 1,2,3$.

3.1 Sequences of robot moves in a robotic cell with two machines

There are two sequences of robot moves in a robotic cell with two machines: alternative 1 and alternative 2. The procedures of alternatives 1 and 2 are shown in Figures 3.1 and 3.2 respectively. The part processing begins when the machines are unloaded. In alternative 2, there must be an initial step. The initial step will be described in Section 3.1.2. After the initialization, the first cycle begins. The cycle time of alternatives 1 and 2 is formulated in Sections 3.1.1 and 3.1.2, respectively.

3.1.1 Alternative 1

The robot picks up a part $P_{\sigma(1)}$ at input buffer I, moves to M_1, loads M_1 with the part $P_{\sigma(1)}$, waits at M_1 until the part $P_{\sigma(1)}$ has been processed, unloads the part $P_{\sigma(1)}$ from M_1,

moves to M_2, loads the part $P_{\sigma(1)}$ on M_2, waits at M_2 until the part $P_{\sigma(1)}$ has been processed, unloads the part $P_{\sigma(1)}$ from M_2, moves to output buffer O, drops the part $P_{\sigma(1)}$ at O, and moves back to I. Then the robot picks up the next part $P_{\sigma(2)}$ to start next cycle. The interval time between $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ can be obtained as follows:

$$T_{1\sigma(i)\sigma(i+1)} = \varepsilon + \delta + \varepsilon + a_{\sigma(i)} + \varepsilon + \delta + \varepsilon + b_{\sigma(i)} + \varepsilon + \delta + \varepsilon + 3\delta$$

$$= 6\varepsilon + 6\delta + a_{\sigma(i)} + b_{\sigma(i)} \qquad (3.1.1.1)$$

The cycle time of Alternative 1 for n different parts can be expressed as:

$$T_1 = \sum_{i=1}^{n} T_{1\sigma(i)\sigma(i+1)} \qquad (3.1.1.2)$$

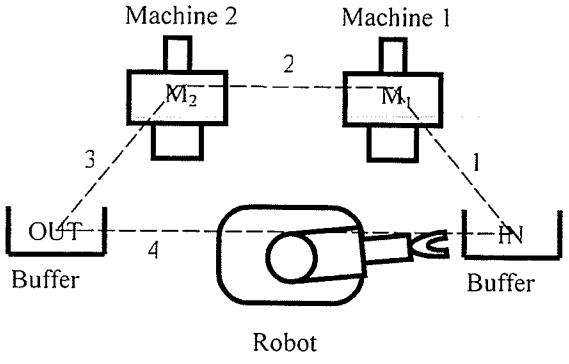


Figure 3.1 Alternative 1 for a robotic cell with two machines

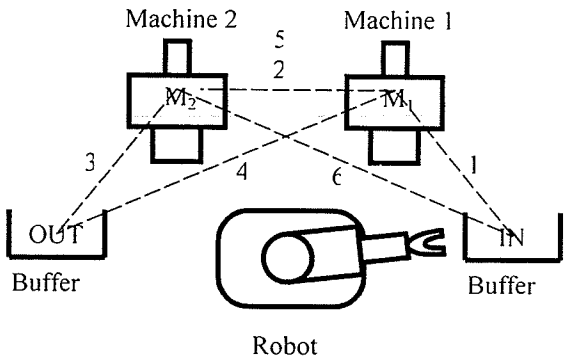


Figure 3.2 Alternative 2 for a robotic cell with two machines

3.1.2 Alternative 2

The robot picks up a part $p_{\sigma(1)}$ at input buffer I, moves to M_1, loads the part $p_{\sigma(1)}$ on M_1, waits at M_1 until the part $p_{\sigma(1)}$ has been processed, unloads the part $p_{\sigma(1)}$, moves to M_2, loads the part $p_{\sigma(1)}$ on M_2, moves back to input buffer I and ends the initial cycle.

The next part $p_{\sigma(2)}$ will enter the cell from input buffer. The robot picks up a part $p_{\sigma(2)}$ at I, moves to M_1, loads $p_{\sigma(2)}$ on M_1, moves to M_2, waits at M_2 until the part $p_{\sigma(1)}$ which is on M_2 has been processed if necessary, unloads the part $p_{\sigma(1)}$, moves to output buffer O, drops the part $p_{\sigma(1)}$ at O, moves to M_1, wait at M_1 until the part $p_{\sigma(2)}$ has been processed if necessary, unloads $p_{\sigma(2)}$, moves to M_2, loads $p_{\sigma(2)}$ on M_2, and moves back to I. The procedure is shown in Figure 3.2. The interval time between $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ can be obtained as follows:

$$T_{2\sigma(i)\sigma(i+1)} = \varepsilon + \delta + \varepsilon + 2\delta + w_{1\sigma(i+1)} + \varepsilon + \delta + \varepsilon + 2\delta + \varepsilon + \delta + \varepsilon + \delta + w_{2\sigma(i)}$$

$$= 8\delta + 6\varepsilon + w_{2\sigma(i)} + w_{1\sigma(i+1)} \qquad (3.1.1.3)$$

For $w_{2\sigma(i)}$, the robot moves back to I after loading $p_{\sigma(i)}$ on M_2, picks up a part $p_{\sigma(i+1)}$ at I, moves to M_1, loads $p_{\sigma(i+1)}$ on M_1, moves to M_2, and waits at M_2 until the part $p_{\sigma(i)}$ on M_2 has been processed if necessary. The interval time of the circle equals $b_{\sigma(i)}$. The equation is obtained as follows.

$$2\delta + \varepsilon + \delta + \varepsilon + \delta + w_{2\sigma(i)} = b_{\sigma(i)}$$

$$w_{2\sigma(i)} = \max\{0, b_{\sigma(i)} - 4\delta - 2\varepsilon\} \qquad (3.1.1.4)$$

For $w_{1\sigma(i+1)}$, the robot moves to M_2 after loading $p_{\sigma(i+1)}$ on M_1, waits at M_2 until the part $p_{\sigma(i)}$ on M_2 has been processed if necessary, unloads the part $p_{\sigma(i)}$, moves to output buffer O, drops the part $p_{\sigma(i)}$ at O, moves to M_1, and waits at M_1 until the part $p_{\sigma(i+1)}$ has been processed if necessary. The interval time of the circle equals $a_{\sigma(i+1)}$. The equation is obtained as follows.

$$\delta + w_{2\sigma(i)} + \varepsilon + \delta + \varepsilon + 2\delta + w_{1\sigma(i+1)} = a_{\sigma(i+1)}$$

$$w_{1\sigma(i+1)} = \max\{0, \alpha_{\sigma(i+1)} - w_{2\sigma(i)} - 4\delta - 2\varepsilon\}$$

$$T_{2\sigma(i)\sigma(i+1)} = 8\delta + 6\varepsilon + \max\{0, b_{\sigma(i+1)} - 4\delta - 2\varepsilon\} + \max\{0, a_{\sigma(i+1)} - 4\delta - 2\varepsilon - \max\{0, b_{\sigma(i)} - 4\delta - 2\varepsilon\}\} \quad (3.1.1.5)$$

The cycle time of Alternative 2 for n different parts can be expressed as:

$$T_2 = \sum_{i=1}^{n} T_{2\sigma(i)\sigma(i+1)} \quad (3.1.1.6)$$

3.2 Sequences of robot moves in a robotic cell with three machines

There are six robot move sequences for a robotic cell with three machines. They are Alternatives 1, 2, 3, 4, 5, and 6. Usually the production of parts begins when the machines are unloaded and the robot is at input buffer I ready to pick up a part. Similar as the two-machine cell, there must be an initial step for Alternatives 2, 3, 4, 5, and 6, respectively. They are illustrated in Sections 3.2.1 through 3.2.3.

3.2.1 Alternatives 1 and 3

The procedure of Alternative 1 is shown in Figure 3.3.

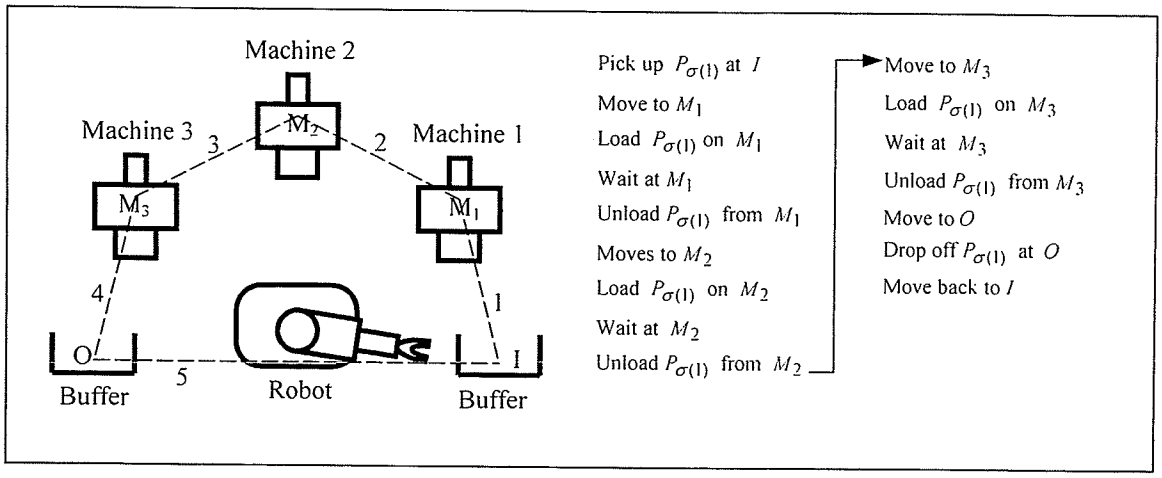


Figure 3.3 Alternative 1 for a robotic cell with three machines

The interval time between $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ can be obtained as follows:

$$T_{1\sigma(i)\sigma(i+1)} = \varepsilon+\delta+\varepsilon+a_{\sigma(i)}+\varepsilon+\delta+\varepsilon+b_{\sigma(i)}+\varepsilon+\delta+\varepsilon+c_{\sigma(i)}+\varepsilon+\delta+\varepsilon+4\delta$$

$$= 8\varepsilon+8\delta+a_{\sigma(i)}+b_{\sigma(i)}+c_{\sigma(i)} \qquad (3.2.1.1)$$

$$T_1 = \sum_{i=1}^{n} T_{1\sigma(i)\sigma(i+1)} \qquad (3.2.1.2)$$

For Alternative 3, the robot picks up a part $p_{\sigma(1)}$ at input buffer I, moves to M_1, loads the part $p_{\sigma(1)}$ on M_1, waits at M_1 until the part $p_{\sigma(1)}$ has been processed, unloads the part $p_{\sigma(1)}$, moves to M_2, loads the part $p_{\sigma(1)}$ on M_2, waits at M_2 until the part $p_{\sigma(1)}$ has been processed, unloads the part $p_{\sigma(1)}$, moves to M_3, loads the part $p_{\sigma(1)}$ on M_3, and moves back to input buffer I to end the initial cycle.

After the initial cycle, the next part $p_{\sigma(2)}$ will enter the cell from input buffer. The procedure is shown in Figure 3.4.

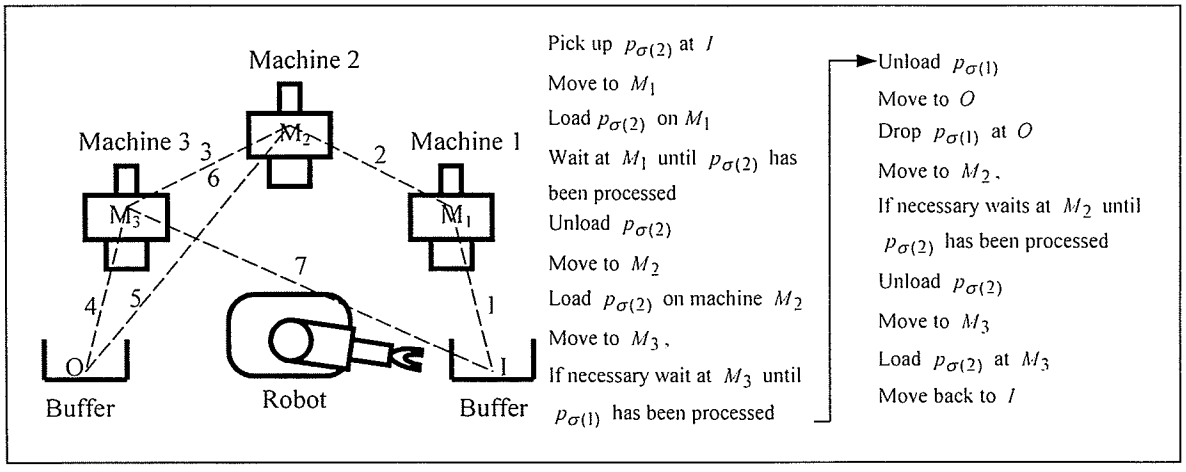


Figure 3.4 Alternative 3 for a robotic cell with three machines

The interval time between $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ can be obtained as follows:

$$T_{3\sigma(i)\sigma(i+1)} = \varepsilon+\delta+\varepsilon+a_{\sigma(i+1)}+\varepsilon+\delta+\varepsilon+\delta+w_{3\sigma(i)}+\varepsilon+\delta+\varepsilon+2\delta+w_{2\sigma(i+1)}+\varepsilon+\delta+\varepsilon+3\delta$$
$$= 8\varepsilon+10\delta+a_{\sigma(i+1)}+w_{3\sigma(i)}+w_{2\sigma(i+1)} \qquad (3.2.1.3)$$

For $w_{3\sigma(i)}$, the robot moves to I after loading $p_{\sigma(i)}$ at M_3, picks up a part $p_{\sigma(i+1)}$ at I, moves to M_1, loads $p_{\sigma(i+1)}$ on M_1, waits at M_1 until $p_{\sigma(i+1)}$ has been processed, unloads $p_{\sigma(i+1)}$, moves to M_2, loads $p_{\sigma(i+1)}$ on M_2, moves to M_3, and waits at M_3 until $p_{\sigma(i)}$ has been processed if necessary. The interval time of the circle equals $c_{\sigma(i)}$. The equation is obtained as follows.

$$3\delta+\varepsilon+\delta+\varepsilon+a_{\sigma(i+1)}+\varepsilon+\delta+\varepsilon+\delta+w_{3\sigma(i)}=c_{\sigma(i)}$$

$$w_{3\sigma(i)} = \max\{0, c_{\sigma(i)}-a_{\sigma(i+1)}-4\varepsilon-6\delta\} \qquad (3.2.1.4)$$

For $w_{2\sigma(i+1)}$, the robot moves to M_3 after loading $p_{\sigma(i+1)}$ on M_2, waits at M_3 until $p_{\sigma(i)}$ has been processed if necessary, unloads $p_{\sigma(i)}$, moves to O, drops $p_{\sigma(i)}$ at O, moves to M_2, and waits at M_2 until $p_{\sigma(i+1)}$ has been processed if necessary. The interval time of the circle equals $b_{\sigma(i+1)}$. The equation is obtained as follows.

$$\delta+w_{3\sigma(i)}+\varepsilon+\delta+\varepsilon+2\delta+w_{2\sigma(i+1)}=b_{\sigma(i+1)}$$

$$w_{2\sigma(i+1)} = \max\{0, b_{\sigma(i+1)}-w_{3\sigma(i)}-2\varepsilon-4\delta\}$$

$$T_{3\sigma(i)\sigma(i+1)} = 8\varepsilon+10\delta+a_{\sigma(i+1)}+\max\{0, c_{\sigma(i)}-a_{\sigma(i+1)}-4\varepsilon-6\delta\}$$
$$+\max\{0, b_{\sigma(i+1)}-2\varepsilon-4\delta-\max\{0, c_{\sigma(i)}-a_{\sigma(i+1)}-4\varepsilon-6\delta\}\} \qquad (3.2.1.5)$$

$$T_3 = \sum_{i=1}^{n} T_{3\sigma(i)\sigma(i+1)} \qquad (3.2.1.6)$$

3.2.2 Alternatives 4 and 5

For Alternative 4, the initial cycle is the same as that of Alterative 3. After the initial cycle, the next part $p_{\sigma(2)}$ will enter the cell from input buffer. The procedure is shown in Figure 3.5.

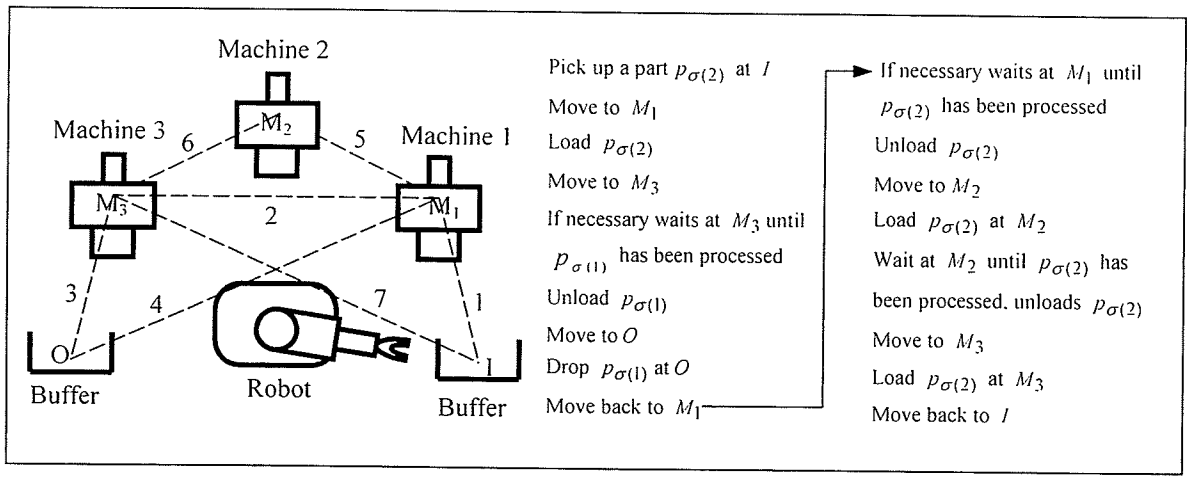


Figure 3.5 Alternative 4 for a robotic cell with three machines

The interval time between $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ can be obtained as follows:

$$T_{4\sigma(i)\sigma(i+1)} = \varepsilon+\delta+\varepsilon+2\delta+w_{3\sigma(i)}+\varepsilon+\delta+\varepsilon+3\delta+w_{1\sigma(i+1)}+\varepsilon+\delta+\varepsilon+b_{\sigma(i+1)}+\varepsilon+\delta+\varepsilon+3\delta$$
$$= 12\delta+8\varepsilon+b_{\sigma(i+1)}+w_{1\sigma(i+1)}+w_{3\sigma(i)} \qquad (3.2.2.1)$$

For $w_{3\sigma(i)}$, the robot moves back to I after loading $p_{\sigma(i)}$ at M_3, picks up a part $p_{\sigma(i+1)}$ at I, moves to M_1, loads $p_{\sigma(i+1)}$ on M_1, moves to M_3, and waits at M_3 until $p_{\sigma(i)}$ has been processed if necessary. The interval time of the circle equals $c_{\sigma(i)}$. The equation is obtained as follows.

$$3\delta+\varepsilon+\delta+\varepsilon+2\delta+w_{3\sigma(i)} = c_{\sigma(i)}$$

$$w_{3\sigma(i)} = \max\{0, c_{\sigma(i)}-6\delta-2\varepsilon\} \qquad (3.2.2.2)$$

For $w_{1\sigma(i+1)}$, the robot moves to M_3 after loading $p_{\sigma(i+1)}$ on M_1, waits at M_3 until $p_{\sigma(i)}$ has been processed if necessary, unloads $p_{\sigma(i)}$, moves to O, drops $p_{\sigma(i)}$ at O, moves back to M_1, and waits at M_1 until $p_{\sigma(i+1)}$ has been processed if necessary. The interval time of the circle equals $a_{\sigma(i+1)}$. The equation is obtained as follows.

$$2\delta + w_{3\sigma(i)} + \varepsilon + \delta + \varepsilon + 3\delta + w_{1\sigma(i+1)} = a_{\sigma(i+1)}$$

$$w_{1\sigma(i+1)} = \max\{0, a_{\sigma(i+1)} - w_{3\sigma(i)} - 6\delta - 2\varepsilon\}$$

$$T_{4\sigma(i)\sigma(i+1)} = 12\delta + 8\varepsilon + b_{\sigma(i+1)} + \max\{0, c_{\sigma(i)} - 6\delta - 2\varepsilon\}$$
$$+ \max\{0, a_{\sigma(i+1)} - 6\delta - 2\varepsilon - \max\{0, c_{\sigma(i)} - 6\delta - 2\varepsilon\}\} \quad (3.2.2.3)$$

$$T_4 = \sum_{i=1}^{n} T_{4\sigma(i)\sigma(i+1)} \quad (3.2.2.4)$$

For Alternative 5, the robot picks up a part $p_{\sigma(1)}$ at input buffer I, moves to M_1, loads the part $p_{\sigma(1)}$ on M_1, waits at M_1 until the part $p_{\sigma(1)}$ has been processed, unloads the part $p_{\sigma(1)}$, moves to M_2, loads the part $p_{\sigma(1)}$ on M_2, and moves back to input buffer I to end the initial cycle. Then the next part $p_{\sigma(2)}$ will enter the cell from input buffer. The procedure is shown in Figure 3.6.

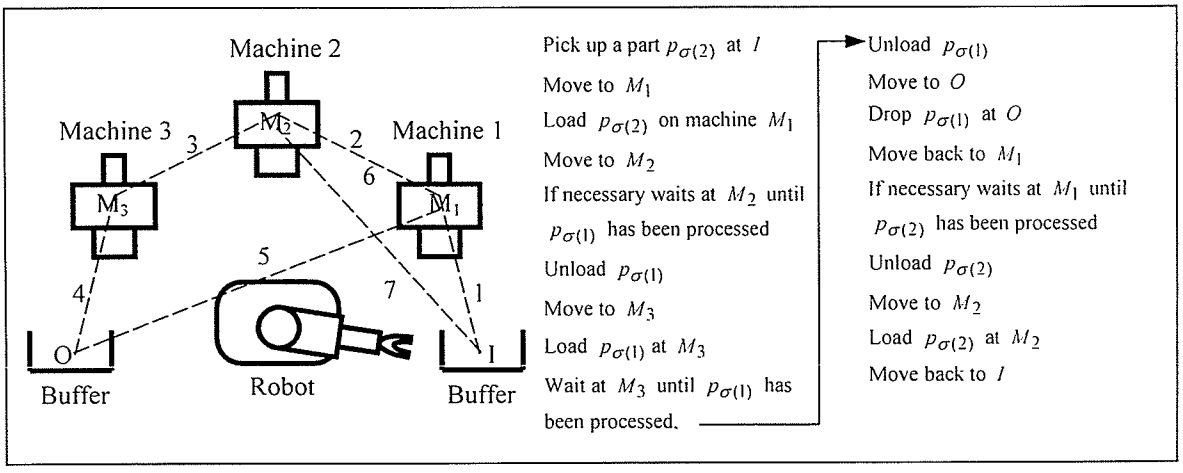


Figure 3.6 Alternative 5 for a robotic cell with three machines

The interval time between $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ can be obtained as follows:

$$T_{5\sigma(i)\sigma(i+1)} = \varepsilon+\delta+\varepsilon+\delta+w_{2\sigma(i)}+\varepsilon+\delta+\varepsilon+c_{\sigma(i)}+\varepsilon+\delta+\varepsilon+3\delta+w_{1\sigma(i+1)}+\varepsilon+\delta+\varepsilon+2\delta$$

$$= 10\delta+8\varepsilon+w_{2\sigma(i)}+c_{\sigma(i)}+w_{1\sigma(i+1)} \qquad (3.2.2.5)$$

For $w_{2\sigma(i)}$, the robot moves back to I after loading $p_{\sigma(i)}$ at M_2, picks up a part $p_{\sigma(i+1)}$ at I, moves to M_1, loads $p_{\sigma(i+1)}$ on M_1, moves to M_2, and waits at M_2 until $p_{\sigma(i)}$ has been processed if necessary. The interval time of the circle equals $b_{\sigma(i)}$. The equation is obtained as follows.

$$2\delta+\varepsilon+\delta+\varepsilon+\delta+w_{2\sigma(i)} = b_{\sigma(i)}$$

$$w_{2\sigma(i)} = \max\{0, b_{\sigma(i)}-4\delta-2\varepsilon\} \qquad (3.2.2.6)$$

For $w_{1\sigma(i+1)}$, the robot moves to M_2 after loading $p_{\sigma(i+1)}$ on M_1, waits at M_2 until $p_{\sigma(i)}$ has been processed if necessary, unloads $p_{\sigma(i)}$, moves to M_3, loads $p_{\sigma(i)}$ at M_3, waits at M_3 until $p_{\sigma(i)}$ has been processed, unloads $p_{\sigma(i)}$, moves to O, drops $p_{\sigma(i)}$ at O, moves back to M_1, and waits at M_1 until $p_{\sigma(i+1)}$ has been processed if necessary. The interval time of the circle equals $a_{\sigma(i+1)}$. The equation is obtained as follows.

$$\delta+w_{2\sigma(i)}+\varepsilon+\delta+\varepsilon+c_{\sigma(i)}+\varepsilon+\delta+\varepsilon+3\delta+w_{1\sigma(i+1)} = a_{\sigma(i+1)}$$

$$w_{1\sigma(i+1)} = \max\{0, a_{\sigma(i+1)}-c_{\sigma(i)}-w_{2\sigma(i)}-6\delta-4\varepsilon\}$$

$$T_{5\sigma(i)\sigma(i+1)} = 10\delta+8\varepsilon+c_{\sigma(i)}+\max\{0, b_{\sigma(i)}-4\delta-2\varepsilon\}$$

$$+\max\{0, a_{\sigma(i+1)}-c_{\sigma(i)}-6\delta-4\varepsilon-\max\{0, b_{\sigma(i)}-4\delta-2\varepsilon\}\} \qquad (3.2.2.7)$$

$$T_5 = \sum_{i=1}^{n} T_{5\sigma(i)\sigma(i+1)} \qquad (3.2.2.8)$$

3.2.3 Alternatives 2 and 6

For Alternative 2, the robot picks up a part $p_{\sigma(1)}$ at input buffer I, moves to M_1, loads the part $p_{\sigma(1)}$ on M_1, waits at M_1 until the part $p_{\sigma(1)}$ has been processed, unloads the part $p_{\sigma(1)}$, moves to M_2, loads the part $p_{\sigma(1)}$ on M_2, moves back to input buffer I, and ends the initial cycle. Then next part $p_{\sigma(2)}$ enters the cell from input buffer. The procedure is shown in Figure 3.7.

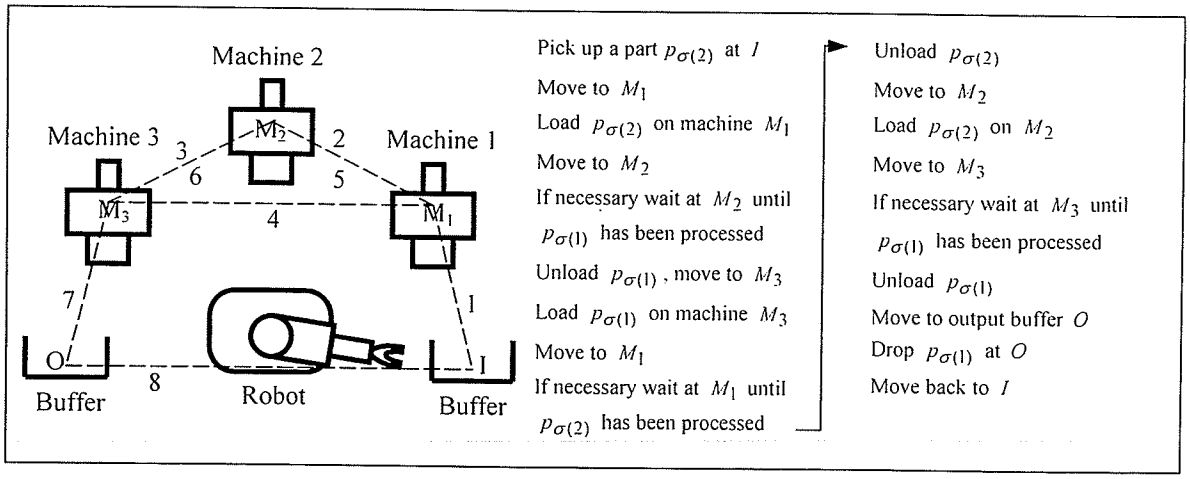


Figure 3.7 Alternative 2 for a robotic cell with three machines

The interval time between $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ can be obtained as follows.

$$T_{2\sigma(i)\sigma(i+1)} = \varepsilon+\delta+\varepsilon+\delta+w_{2\sigma(i)}+\varepsilon+\delta+\varepsilon+2\delta+w_{1\sigma(i+1)}+\varepsilon+\delta+\varepsilon+\delta+w_{3\sigma(i)}+\varepsilon+\delta+\varepsilon+4\delta$$

$$=8\varepsilon+12\delta+w_{1\sigma(i+1)}+w_{2\sigma(i)}+w_{3\sigma(i)} \qquad (3.2.3.1)$$

For $w_{1\sigma(i+1)}$, after a part $p_{\sigma(i+1)}$ is loaded on M_1, the robot moves to M_2, waits at M_2 until $p_{\sigma(i)}$ has been processed if necessary, unloads $p_{\sigma(i)}$, moves to M_3, loads $p_{\sigma(i)}$ on M_3, moves back to M_1, and waits at M_1 until $p_{\sigma(i+1)}$ has been processed if necessary. The interval time of the circle equals $a_{\sigma(i+1)}$. The equation is obtained as follows.

$$\delta + w_{2\sigma(i)} + \varepsilon + \delta + \varepsilon + 2\delta + w_{1\sigma(i+1)} = a_{\sigma(i+1)}$$

$$w_{1\sigma(i+1)} = \max\{0, a_{\sigma(i+1)} - 2\varepsilon - 4\delta - w_{2\sigma(i)}\} \tag{3.2.3.2}$$

For $w_{2\sigma(i)}$, after a part $p_{\sigma(i)}$ is loaded on M_2, the robot moves to M_3, waits at M_3 until $p_{\sigma(i-1)}$ has been processed if necessary, unloads $p_{\sigma(i-1)}$, moves to output buffer O, drops $p_{\sigma(i-1)}$ at O, moves to I, picks up $p_{\sigma(i+1)}$ at I, moves to M_1, loads $p_{\sigma(i+1)}$ on M_1, moves to M_2, and waits at M_2 until $p_{\sigma(1)}$ has been processed if necessary. Assume the cycle time is $b_{\sigma(i)}$, which can be calculated as follows.

$$\delta + w_{3\sigma(i-1)} + \varepsilon + \delta + \varepsilon + 4\delta + \varepsilon + \delta + \varepsilon + \delta + w_{2\sigma(i)} = b_{\sigma(i)}$$

$$w_{2\sigma(i)} = \max\{0, b_{\sigma(i)} - 4\varepsilon - 8\delta - w_{3\sigma(i-1)}\} \tag{3.2.3.3}$$

For $w_{3\sigma(i)}$, after a part $p_{\sigma(i)}$ is loaded on M_3, the robot moves to M_1, waits at M_1 until $p_{\sigma(i+1)}$ has been processed if necessary, unloads $p_{\sigma(i+1)}$, moves to M_2, loads $p_{\sigma(i+1)}$ on M_2, moves to M_3, and waits at M_3 until $p_{\sigma(i)}$ has been processed if necessary. Assume the cycle time is $c_{\sigma(i)}$, which can be calculated as follows.

$$2\delta + w_{1\sigma(i+1)} + \varepsilon + \delta + \varepsilon + \delta + w_{3\sigma(i)} = c_{\sigma(i)}$$

$$w_{3\sigma(i)} = \max\{0, c_{\sigma(i)} - 2\varepsilon - 4\delta - w_{1\sigma(i+1)}\}$$

Substitute $w_{1\sigma(i+1)}$ and $w_{2\sigma(i)}$ into the expression of $T_{2\sigma(i)\sigma(i+1)}$,

$$T_{2\sigma(i)\sigma(i+1)} = 8\varepsilon + 12\delta + \max\{0, a_{\sigma(i+1)} - 2\varepsilon - 4\delta - \max\{0, b_{\sigma(i)} - 4\varepsilon - 8\delta - w_{3\sigma(i-1)}\}\}$$
$$+ \max\{0, b_{\sigma(i)} - 4\varepsilon - 8\delta - w_{3\sigma(i-1)}\} + w_{3\sigma(i)} \tag{3.2.3.4}$$

Where

$$w_{3\sigma(i)} = \max\{0, c_{\sigma(i)} - 2\varepsilon - 4\delta - \max\{0, a_{\sigma(i+1)} - 2\varepsilon - 4\delta - \max\{0, b_{\sigma(i)} - 4\varepsilon - 8\delta - w_{3\sigma(i-1)}\}\}\} \quad (3.2.3.5)$$

$$T_2 = \sum_{i=1}^{n} T_{2\sigma(i)\sigma(i+1)} \quad (3.2.3.6)$$

Because of the recursive expression for $w_{3\sigma(i)}$, the cycle time T_2 is affected by $w_{3\sigma(i)}$. The initial cycle has to be considered to obtain T_2. The following describes how to obtain the first $w_{3\sigma(1)}$.

Based on the initial step, after the part $p_{\sigma(1)}$ has been loaded at M_2, the robot moves back to I, picks up the part $p_{\sigma(2)}$, moves to M_1, loads $p_{\sigma(2)}$ on M_1, moves to M_2, and waits at M_2 until $p_{\sigma(1)}$ has been processed if necessary. Assume the cycle time is $b_{\sigma(1)}$,

$$2\delta + \varepsilon + \delta + \varepsilon + \delta + w_{2\sigma(1)} = b_{\sigma(1)}$$

$$w_{2\sigma(1)} = \max\{0, b_{\sigma(1)} - 4\delta - 2\varepsilon\} \quad (3.2.3.7)$$

If $w_{2\sigma(1)}$ is obtained, $w_{1\sigma(2)}$ can be got,

$$w_{1\sigma(2)} = \max\{0, a_{\sigma(2)} - 2\varepsilon - 4\delta - w_{2\sigma(1)}\}$$

$$= \max\{0, a_{\sigma(2)} - 2\varepsilon - 4\delta - \max\{0, b_{\sigma(1)} - 4\delta - 2\varepsilon\}\}$$

So

$$w_{3\sigma(1)} = \max\{0, c_{\sigma(1)} - 2\varepsilon - 4\delta - w_{1\sigma(2)}\}$$

$$= \max\{0, c_{\sigma(1)} - 2\varepsilon - 4\delta - \max\{0, a_{\sigma(2)} - 2\varepsilon - 4\delta - \max\{0, b_{\sigma(1)} - 4\delta - 2\varepsilon\}\}\} \quad (3.2.3.8)$$

Once $w_{3\sigma(1)}$ is got, the cycle time T_2 can be obtained.

For Alternative 6, the robot picks up a part $p_{\sigma(1)}$ at input buffer I, moves to M_1, loads $p_{\sigma(1)}$ on M_1, waits at M_1, unloads $p_{\sigma(1)}$ from M_1, moves to M_2, loads $p_{\sigma(1)}$ on M_2, waits at M_2, unloads $p_{\sigma(1)}$ from M_2, moves to M_3, loads $p_{\sigma(1)}$ on M_3, moves back to I, picks up part $p_{\sigma(2)}$ at I, moves to M_1, loads $p_{\sigma(2)}$ on M_1, waits at M_1, unloads $p_{\sigma(2)}$ from M_1, moves to M_2, loads $p_{\sigma(2)}$ on M_2, and moves back to I. The initial cycle is finished. The next part $p_{\sigma(3)}$ will enter the cell from input buffer. The procedure is shown in Figure 3.8.

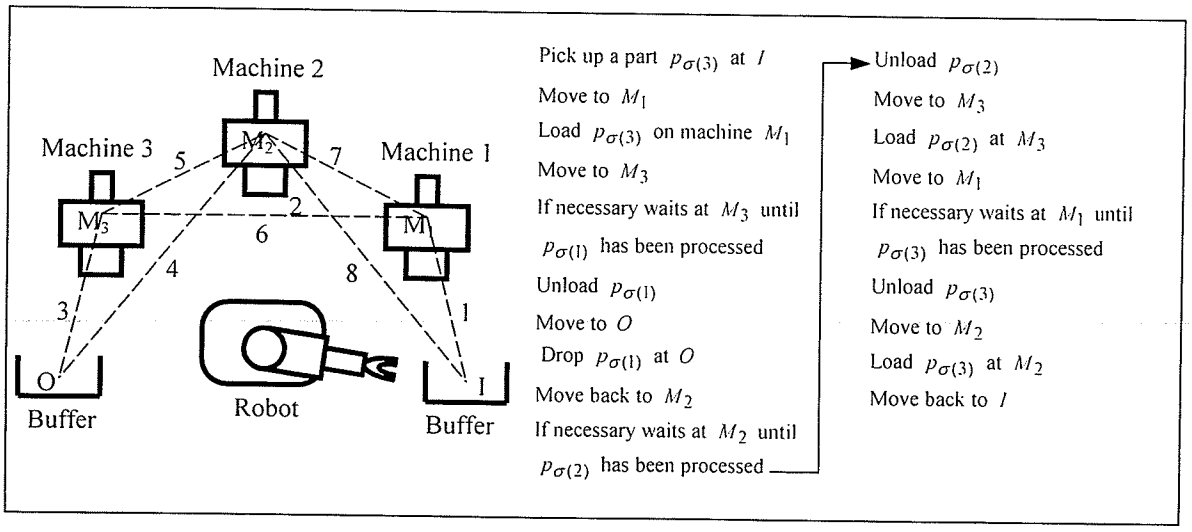


Figure 3.8 Alternative 6 for a robotic cell with three machines

The interval time between $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ can be obtained as follows.

$$T_{6\sigma(i)\sigma(i+1)} = \varepsilon+\delta+\varepsilon+2\delta+w_{3\sigma(i)}+\varepsilon+\delta+\varepsilon+2\delta+w_{2\sigma(i+1)}+\varepsilon+\delta+\varepsilon+2\delta+w_{1\sigma(i+2)}+\varepsilon+\delta+\varepsilon+2\delta$$

$$= 8\varepsilon+12\delta+w_{3\sigma(i)}+w_{2\sigma(i+1)}+w_{1\sigma(i+2)} \qquad (3.2.3.9)$$

For $w_{3\sigma(i)}$, after a part $p_{\sigma(i)}$ is loaded on M_3, the robot moves back to M_1, waits at M_1 until $p_{\sigma(i+1)}$ has been processed if necessary, unloads $p_{\sigma(i+1)}$, moves to M_2, loads $p_{\sigma(i+1)}$ at M_2, moves to I, picks up a part $p_{\sigma(i+2)}$ at I, moves to M_1, loads $p_{\sigma(i+2)}$ on M_1, moves to M_3, and waits at M_3 until $p_{\sigma(i)}$ has been processed if necessary. If interval time of the cycle equals $c_{\sigma(i)}$, the equation can be obtained as follows.

$$2\delta + w_{1\sigma(i+1)} + \varepsilon + \delta + \varepsilon + 2\delta + \varepsilon + \delta + \varepsilon + 2\delta + w_{3\sigma(i)} = c_{\sigma(i)}$$

$$4\varepsilon + 8\delta + w_{1\sigma(i+1)} + w_{3\sigma(i)} = c_{\sigma(i)}$$

$$w_{3\sigma(i)} = \max\{0, c_{\sigma(i)} - 4\varepsilon - 8\delta - w_{1\sigma(i+1)}\} \qquad (3.2.3.10)$$

For $w_{2\sigma(i+1)}$, after a part $p_{\sigma(i+1)}$ is loaded on M_2, the robot moves back to I, picks up a part $p_{\sigma(i+2)}$ at I, moves to M_1, loads $p_{\sigma(i+2)}$ on M_1, moves to M_3, waits at M_3 until $p_{\sigma(i)}$ has been processed if necessary, unloads $p_{\sigma(i)}$, moves to O, drops $p_{\sigma(i)}$ at O, moves to M_2, and waits at M_2 until $p_{\sigma(i+1)}$ has been processed if necessary. If interval time of the cycle equals $b_{\sigma(i+1)}$, the equation can be obtained as follows.

$$2\delta + \varepsilon + \delta + \varepsilon + 2\delta + w_{3\sigma(i)} + \varepsilon + \delta + \varepsilon + 2\delta + w_{2\sigma(i+1)} = b_{\sigma(i+1)}$$

$$4\varepsilon + 8\delta + w_{3\sigma(i)} + w_{2\sigma(i+1)} = b_{\sigma(i+1)}$$

$$w_{2\sigma(i+1)} = \max\{0, b_{\sigma(i+1)} - 4\varepsilon - 8\delta - w_{3\sigma(i)}\}$$

$$= \max\{0, b_{\sigma(i+1)} - 4\varepsilon - 8\delta - \max\{0, c_{\sigma(i)} - 4\varepsilon - 8\delta - w_{1\sigma(i+1)}\}\} \qquad (3.2.3.11)$$

For $w_{1\sigma(i+1)}$, after a part $p_{\sigma(i+2)}$ is loaded on M_1, the robot moves to M_3, waits at M_3 until $p_{\sigma(i)}$ has been processed if necessary, unloads $p_{\sigma(i)}$, moves to O, drops $p_{\sigma(i)}$ at O, moves to M_2, waits at M_2 until $p_{\sigma(i+1)}$ has been processed if necessary, unloads $p_{\sigma(i+1)}$,

moves to M_3, loads $p_{\sigma(i+1)}$ at M_3, moves to M_1, and waits at M_1 until $p_{\sigma(i+2)}$ has been processed if necessary. If interval time of the cycle equals $a_{\sigma(i+2)}$, the equation can be obtained as follows.

$$2\delta + w_{3\sigma(i)} + \varepsilon + \delta + \varepsilon + 2\delta + w_{2\sigma(i+1)} + \varepsilon + \delta + \varepsilon + 2\delta + w_{1\sigma(i+2)} = a_{\sigma(i+2)}$$

$$4\varepsilon + 8\delta + w_{3\sigma(i)} + w_{2\sigma(i+1)} + w_{1\sigma(i+2)} = a_{\sigma(i+2)}$$

$$w_{1\sigma(i+2)} = \max\{0, a_{\sigma(i+2)} - 4\varepsilon - 8\delta - w_{3\sigma(i)} - w_{2\sigma(i+1)}\}$$

Substitute $w_{2\sigma(i+1)}$, $w_{3\sigma(i)}$ into $T_{6\sigma(i)\sigma(i+1)}$

$$T_{6\sigma(i)\sigma(i+1)} = 8\varepsilon + 12\delta + w_{3\sigma(i)} + w_{2\sigma(i+1)} + w_{1\sigma(i+2)}$$

$$= 8\varepsilon + 12\delta + \max\{0, c_{\sigma(i)} - 4\varepsilon - 8\delta - w_{1\sigma(i+1)}\}$$

$$+ \max\{0, b_{\sigma(i+1)} - 4\varepsilon - 8\delta - \max\{0, c_{\sigma(i)} - 4\varepsilon - 8\delta - w_{1\sigma(i+1)}\}\} + w_{1\sigma(i+2)} \qquad (3.2.3.12)$$

Where

$$w_{1\sigma(i+2)} = \max\{0, a_{\sigma(i+2)} - 4\varepsilon - 8\delta - w_{3\sigma(i)} - w_{2\sigma(i+1)}\}$$

$$= \max\{0, a_{\sigma(i+2)} - 4\varepsilon - 8\delta - \max\{0, c_{\sigma(i)} - 4\varepsilon - 8\delta - w_{1\sigma(i+1)}\}$$

$$- \max\{0, b_{\sigma(i+1)} - 4\varepsilon - 8\delta - \max\{0, c_{\sigma(i)} - 4\varepsilon - 8\delta - w_{1\sigma(i+1)}\}\}\} \qquad (3.2.3.13)$$

$$T_6 = \sum_{i=1}^{n} T_{6\sigma(i)\sigma(i+1)} \qquad (3.2.3.14)$$

Similar to Alternative 2, the computing of $w_{1\sigma(i+2)}$ is also a recursive expression. To obtain T_6, the initial cycle time has to be known and $w_{1\sigma(3)}$ has to be obtained. $w_{1\sigma(3)}$ can be found as follows.

Based on the initial cycle, $w_{1\sigma(1)}$, $w_{2\sigma(1)}$, and $w_{3\sigma(1)}$ can be obtained.

$$w_{1\sigma(1)} = a_{\sigma(1)}$$

$w_{2\sigma(1)} = b_{\sigma(1)}$

For $w_{3\sigma(1)}$, after $p_{\sigma(1)}$ has been loaded at M_3, the robot moves back to I, picks up part $p_{\sigma(2)}$ at I, moves to M_1, loads $p_{\sigma(2)}$ on M_1, waits at M_1, unloads $p_{\sigma(2)}$ from M_1, moves to M_2, loads $p_{\sigma(2)}$ on M_2, moves back to I, picks up part $p_{\sigma(3)}$ at I, moves to M_1, loads $p_{\sigma(3)}$ on M_1, moves to M_3, and waits at M_3 until $p_{\sigma(1)}$ has been processed if necessary. If interval time of the cycle equals $c_{\sigma(1)}$, the equation is obtained as follows.

$$3\delta + \varepsilon + \delta + \varepsilon + a_{\sigma(2)} + \varepsilon + \delta + \varepsilon + 2\delta + \varepsilon + \delta + \varepsilon + 2\delta + w_{3\sigma(1)} = c_{\sigma(1)}$$

$$w_{3\sigma(1)} = \max\{0, c_{\sigma(1)} - a_{\sigma(2)} - 6\varepsilon - 10\delta\}$$

If $w_{3\sigma(1)}$ is obtained, $w_{2\sigma(2)}$ can be got:

$$w_{2\sigma(2)} = \max\{0, b_{\sigma(2)} - 4\varepsilon - 8\delta - w_{3\sigma(1)}\}$$

Substitute $w_{2\sigma(2)}$ and $w_{3\sigma(1)}$, $w_{1\sigma(3)}$ is:

$$w_{1\sigma(3)} = \max\{0, a_{\sigma(3)} - 4\varepsilon - 8\delta - w_{3\sigma(1)} - w_{2\sigma(2)}\} \qquad (3.2.3.15)$$

Once $w_{1\sigma(3)}$ is obtained, the cycle time T_6 can be obtained.

3.3 Discussion of these sequences of robot moves

Part processing in a robotic cell with two or three machines has been described. In a two-machine robotic cell, the cycle time of Alternatives 1 and 2 is simple. In Alternative 1, the cycle is in a constant state as the robot moves. In Alternative 2, as the robot finishes loading a part on Machine 2, the cell is in a stable state. In a three-machine robotic cell, the cycle time of Alternatives 1, 3, 4, and 5 is easily obtained. Alternative 1 is similar to that of a robotic cell with two machines. Alternatives 3 and 4 are in a stable state as the

robot just finishes loading a part on Machine 3. In Alternative 5, the cell is in a stable state as the robot finishes loading a part on Machine 1. Therefore, the cycle time is easily obtained in these four robot moves in a robotic cell with three machines. However, the cycle time of T_2 and T_6 demonstrates that Alternatives 2 and 6 are intractable. To obtain T_2 and T_6, the initial cycle must be considered. The cell may need several cycles to converge to a stable state. Then T_2 and T_6 can settle to stable values.

The expressions of cycle time in a robotic cell with two machines or a robotic cell with three machines can also be applied to the case for identical part types. In this case, the number of parts n is 1. The processing time of parts on the same machine is the same. a, b, and c represent the processing time of a part on M_1, M_2, and M_3, respectively. Therefore, the identical part type is a special case of different part types. Because all parts are the same type, the problem of scheduling parts vanishes. The objective of scheduling becomes finding the optimal cycle time based on sequences of robot moves.

In the next chapter, a simulation tool is developed for identical part types in a robotic cell with two machines or a robotic cell with three machines. It will provide an easy understanding tool for the cell to reach a stable state while we are searching for the optimal solution. The solutions will be compared with those suggested by Sethi [10].

Chapter 4 Simulation of identical part type problems in robotic cells with two and three machines

Simulation is a widely used approach in the evaluation of manufacturing systems. Its applications include: teaching and training [29], product design [30, 31], process planning [32, 33], and manufacturing processing [34, 35]. One of the greatest advantages of simulation is that time can be compressed or expanded in the model. Manufacturing systems can be better understood with the help of simulation models. Once a model is developed, it provides an efficient tool to analyze the effects of various parameters and to study the implementation of various alternatives.

In this Chapter, a simulation tool with visual animation capabilities is developed to determine the sequences of robot moves that maximize the long-run average production rate in a robotic cell with two or three machines. The simulation tool can be used to design a robotic cell. It can also be used to examine the theoretical methods provided by Sethi et al [10].

4.1 Program description

The simulation program is written in Java language. Java is an ideal language for Internet applications. Since the software code is compiled on the Java platform, it can be run on any computer. There are two different kinds of software codes: applets and applications. In order that the simulation can be run on Web browsers, an Applet on HTML is created and embedded into the Web browsers by using the Applet class of Java. The thread classes are used in the program, which involve using objects to implement a runnable

interface. The runnable interface is activated by using the run() method in order that the simulation can be paused or stopped at anytime during program execution. In practice, from a user viewpoint, a process on a local computer must be started by using either a Web browser or a Java Applet view tool.

4.1.1 Overall program

The simulation program has a graphic interface. After the program starts, a graphical user interface (GUI) appears. This is the initial GUI. Figure 4.1 shows a GUI for a robotic cell with two machines.

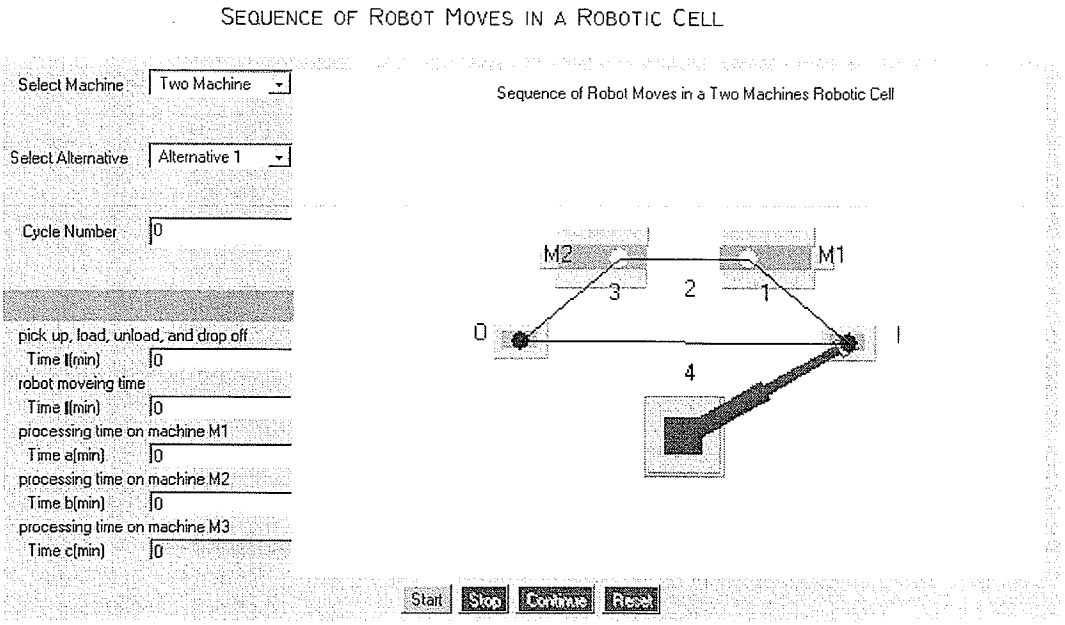


Figure 4.1 A graphical user interface for a robotic cell with two machines

The GUI has two parts: a control panel and a display area. The control panel includes multiple choices, textfields, and buttons. The multiple choice of "Select Machine" is used

to choose the robotic cell environment with two machines or the robotic cell environment with three machines. If a user selects 'three machines', the GUI of a robotic cell with three machines is displayed, as shown in Figure 4.2.

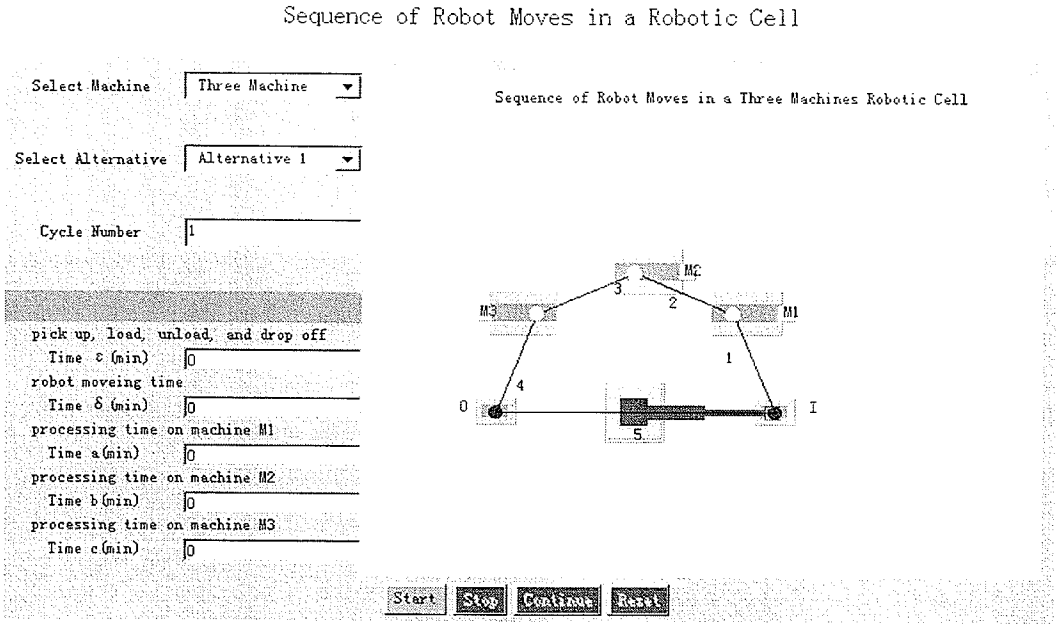
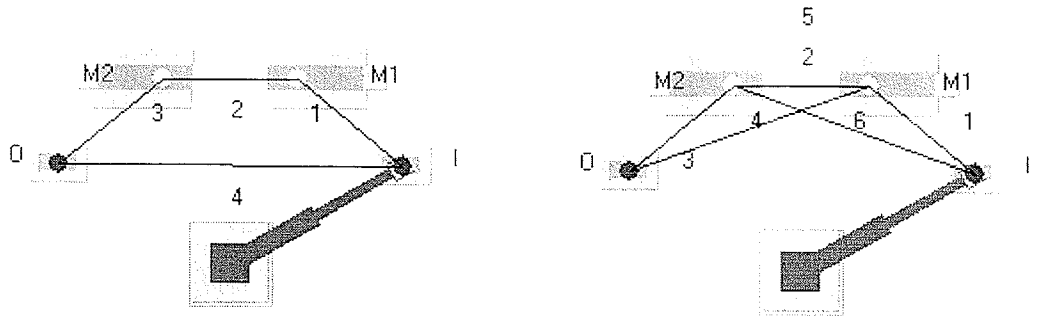


Figure 4.2 A graphical user interface for a robotic cell with three machines

Using these two graphical interfaces, individual sequence of robot moves predefined for a robotic cell with two machines or three machines can be simulated in the center of the display area by selecting the multiple choice of "Select Alternative". In the robotic cell environment with two machines, the multiple choice of "Select Alternative" has two alternatives: 1 and 2. If Alternative 1 is chosen in "Select Alternative", the pattern of Alternative 1 is shown in the display area. If Alternative 2 is chosen, the pattern of Alternative 1 will be changed into that of Alternative 2. The two alternatives are shown in Figure 4.3.



Alternative1 Alternative 2

Figure 4.3 The sequences of robot moves in a robotic cell with two machines

In the GUI of the robotic cell with two machines, the user is allowed to input various time elements: ε, δ, a, and b in the "Textfield". The cycle number decides how many cycles the simulation will run. There are two sequences of robot moves, Alternatives 1 and 2, in the robotic cell with two machines. The research objective is to find the one with the minimal cycle time. After Alternatives 1 and 2 are simulated, the minimal cycle time can be obtained. For each alternative, the part production begins from the state when the cell is empty. For Alternative 1, there is no initial step. The cycle time associated with Alternative 1 can be accumulated until the robot moves back to input buffer I. For Alternative 2, there is an initial step. The initial step has been introduced in Chapter 3. After the initial step finishes, the cell is in a steady state. The simulation of the process begins in that state. The cycle time begins to accumulate until the robot returns to input buffer I. The state at which the system begins to calculate the cycle time is shown in Figure 4.4.

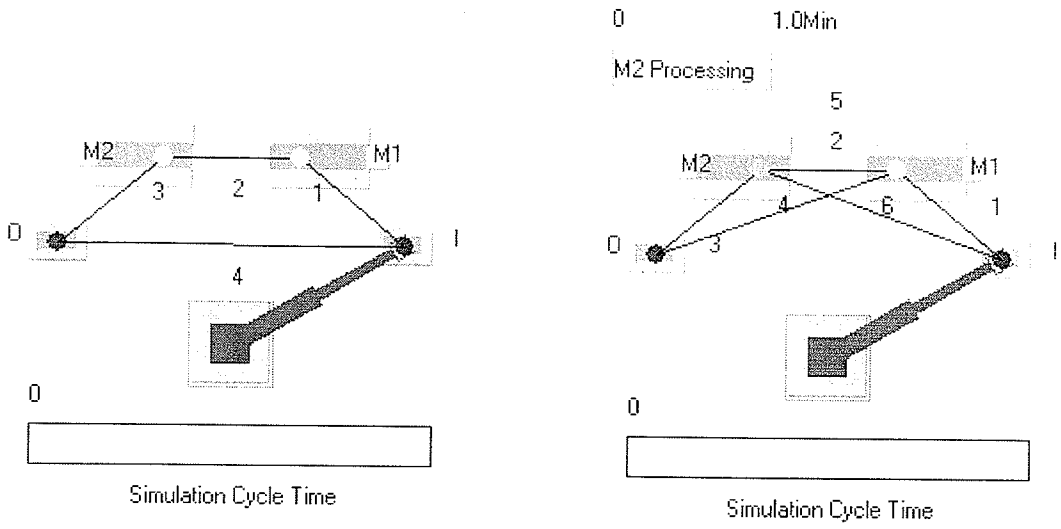
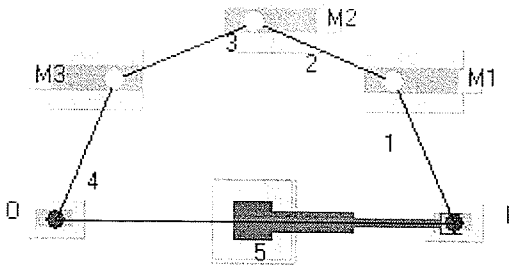


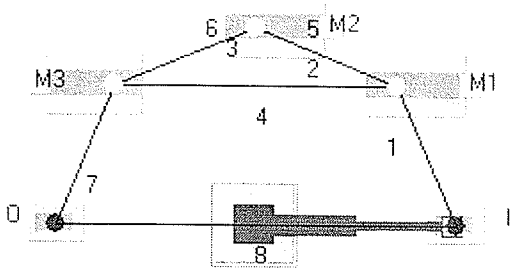
Figure 4.4 The states that cycle time begins

In a robotic cell environment with three machines, there are six alternatives available in the "Select Alternative". Alternatives 1 through 6 shown in Figure 4.5 respectively, can be displayed by selecting Alternative 1, 2, 3, 4, 5, or 6 in "Select Alternative". The purpose of "Select Alternative" is to determine the sequence of robot moves.

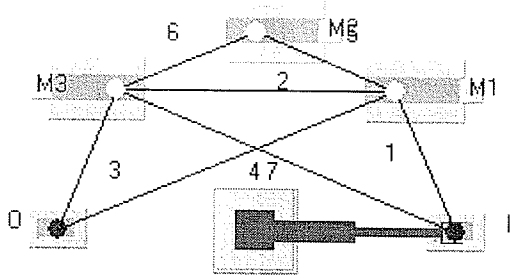
In the GUI for the robotic cell with three machines, the user is allowed to input elements ε, δ, a, b and c in the "Textfield". The cycle number decides how many cycles the simulation will run. In the robotic cell with three machines, there are six sequences of robot moves, Alternatives 1 through 6. For each of them, the production of parts begins from the state that the cell is empty.



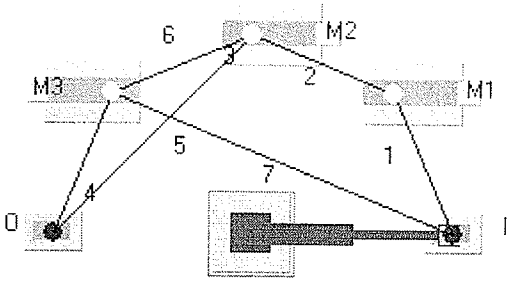
Alternative 1



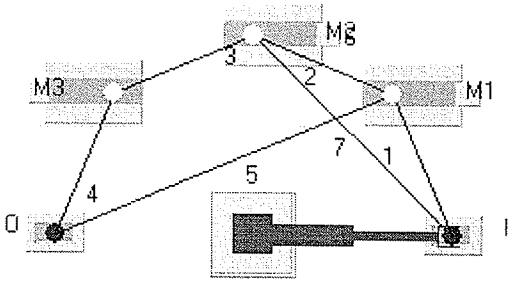
Alternative 2



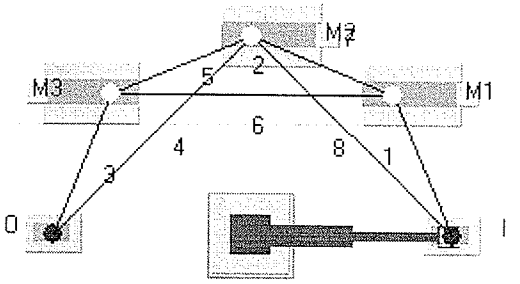
Alternative 3



Alternative 4



Alternative 5



Alternative 6

Figure 4.5 The sequences of robot moves in a robotic cell with three machines

For Alternative 1, there is no initial step. For Alternative 2, after the initial step finishes, the cycle time begins to accumulate until the robot moves back to input buffer I. The cell state after the initial step finishes is shown in Figure 4.6.

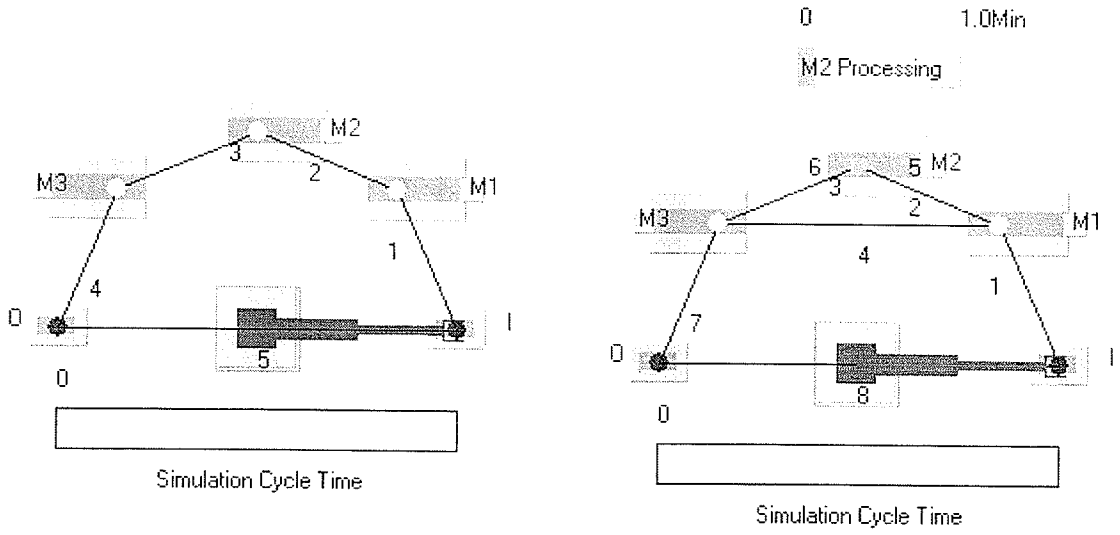


Figure 4.6 The states that cycle time begins for Alternatives 1 and 2

For Alternative 3, after the initial step finishes, a part is processed on Machine 3 and the robot moves back to the input buffer. The cycle time begins to accumulate from this state. For Alternative 4, after the initial step finishes, the cycle time begins to accumulate in the same state. The states are shown in Figure 4.7.

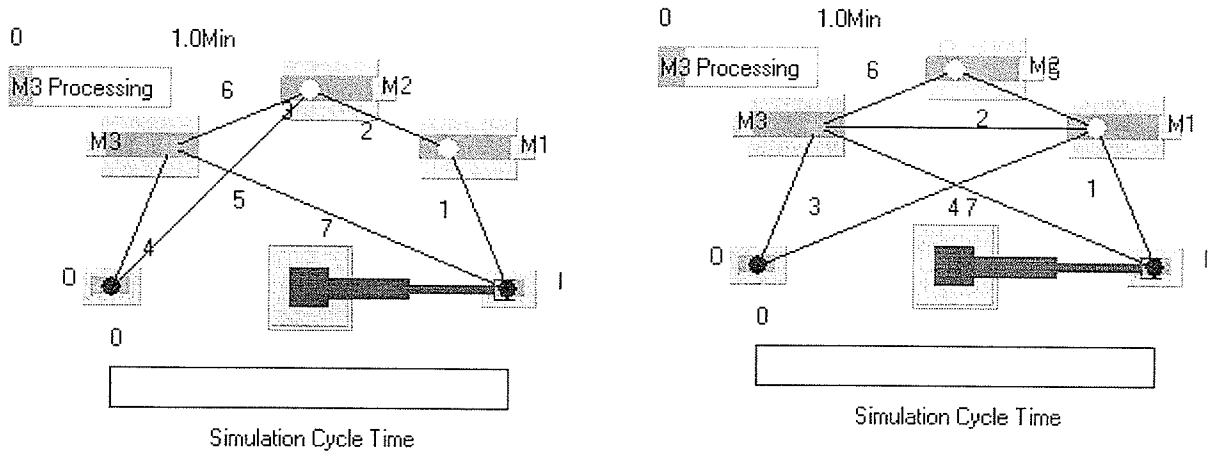


Figure 4.7 The states that cycle time begins for Alternatives 3 and 4

For Alternative 5, after the initial step finishes, the cell will be in a status with a part on Machine 2 and the robot at the input buffer. The cycle time begins to calculate until the robot returns to input buffer I. For Alternative 6, after the initial step finishes, the cell has a part on Machine 2 and another part on Machine 3. From this state, the cycle time begins to calculate until the robot moves back to input buffer I. They are shown in Figure 4.8.

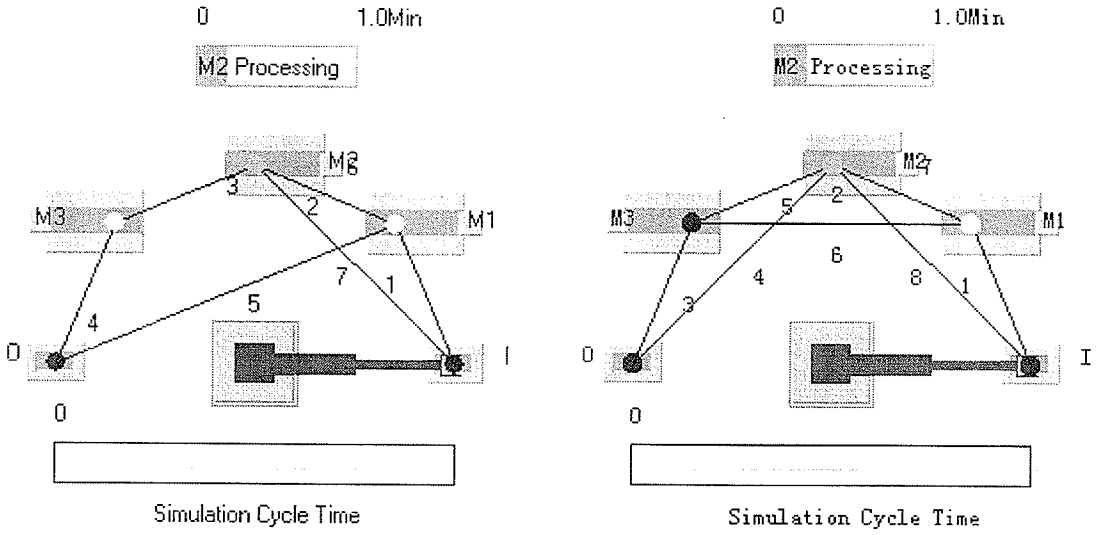


Figure 4.8 The states that cycle time begins for Alternatives 5 and 6

After the initial step, the cell simulation begins. The user can observe the performance of the sequence of robot moves as well as the cell state. The simulation cycle time is displayed in "Simulation Cycle Time" box. The time is accumulated step by step. The cycle time calculated by using Sethi's method appears on the right side of the display area. Comparison between simulation cycle time and calculated cycle time can help users to understand the different approaches.

4.1.2 Description of the simulation program

Java program language is used to write the simulation tool. The following procedure is used to develop the simulation tool in order to meet the design objectives. The programming flow chart is shown in Figure 4.9.

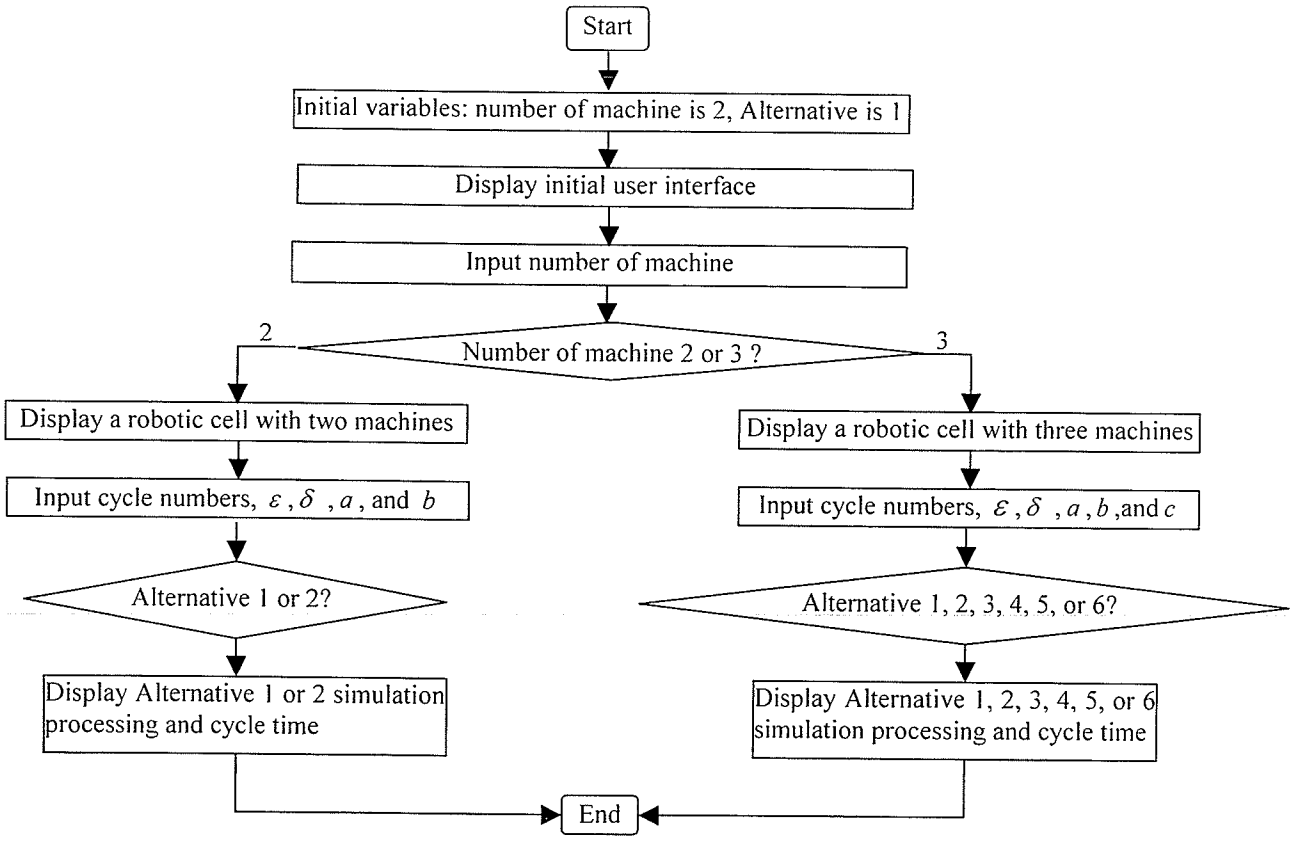


Figure 4.9 The flow chart of the program

The steps of the program are as follows:

Step 1: after the applet is loaded, the system first executes the following functions: init(), initial variables; initForm(), to initialize the user interface frame; and paint(), to output the result of the program on screen.

Step 2: collect the chosen number of machines and "Select Alternative" to decide cell environments and sequence of robot moves.

Step 3: collect data, such as machine processing time, robot moving time, from 'textfields'.

Step 4: display simulation process and the result of cycle time calculated by using Sethi's method and simulation time. Find optimal cycle time from Alternatives 1 and 2 for the robotic cell with two machines or from Alternatives 1 through 6 for the robotic cell with three machines.

In the program, four classes are used: Applet1.java, Drawline.java, Drawmachine.java, and TfTimeParameter. TfTimeParameter is an internal class of Applet1. It is used to control the keyboard input. The methods of both Drawmachine and Drawline classes are used to create the cell layout and the sequence of robot moves for each case, such as Alternative 1 in the robotic cell with two machines. Applet1 is a main class, in which the program runs by using objects that implement the Runnable interface. The Runnable interface provides the method of run (), which can be run by using a thread instant. If a pause or a stop is required at anytime while the program is running, it will just stop the thread instant. If the program needs reset as the program pauses, stops, or runs, it will just kill the thread instant. The methods, which are included in the Applet1.java class, are described as follows.

The methods of init(), initFirm(), initPaint() are used to create initial interface. init() is first called by the browser when the applet is first loaded and the program system is initialized. It is used to define the initial value of variables. initForm() initialazes the user interface frame. initPaint() defines the display after the applet is first loaded. It is called

by the method of paint(). After the applet is loaded, the system first executes the method of init(), initForm(), and finally paint(). The function of paint() is used to output the results of the program on screen. "ItemStateChanged(ItemEvent e)" is used to change interface and show the robotic cell with two macines or the three machines as well as related alternatves.

The methods of CalOut2(Graphics g) and SelectOptimal2(Graphics g) are used to display the time of Alternatives 1 or 2 for two machines and to select the shorter calculated cycle time from Alternatives 1 and 2, respectively. Displaying Alternatives 1 or 2 depends on users' choice. The methods of CalOut3(Graphics g) and SelectOptimal3(Graphics g) are used to display the time, and to select the shortest calculated cycle time from Alternatives 1 through 6 for the robotic cell with three machines, respectively.

The functions of Calculate2(Graphics g) is used to calculate the cycle time of Alternatives 1 and 2 for the robotic cell with two machines by using Sethi's method. The function of Calculate3(graphics g) is used to calculate the cycle time of Alternatives 1 through 6 for the robotic cell with three machines by using Sethi's method. In the function of Calculate2(Graphics g), the cycle time formulated by Sethi [10] is $T_1 = 6\varepsilon + 6\delta + a + b$ and $T_2 = 4\varepsilon + 4\delta + \max\{2\varepsilon + 4\delta, a, b\}$. In the function of Calculate3(graphics g), the cycle time proposed by Sethi's [10] is :

$T_1 = 2\alpha + a + b + c$;

$T_2 = \alpha + \max\{\beta, b, \beta/2 + a, \beta/2 + c, (a+b+c)/2\}$;

$T_3 = \alpha + \max\{c, \alpha + a + 2\delta, \alpha/2 + a + b\}$; $T_4 = \alpha + \max\{\beta + b, \alpha/2 + a + b, \alpha/2 + b + c\}$;

$T_5 = \alpha + \max\{a, \alpha + c + 2\delta, \alpha/2 + b + c\}$; $T_6 = \alpha + \max\{\beta, a, b, c\}$;

Where: $\alpha = 4\delta + 4\varepsilon$ and $\beta = 8\delta + 4\varepsilon$

In the simulation program, drawRobot(Graphics g, int ang) draws the position of robot in the cell. robotMove(Graphics g, Graphics g1, int an1, int an2) animates the robot moving from angle1 position to angle2 position. The machines are arranged in a circle so that the robot arm can rotate like a clock hand, but in any direction. The process of loading, unloading, picking up, and dropping off are implemented in the functions of robotLoad(Graphics g, Graphics g1), robotUnload(Graphics g, Graphics g1), robotPick(Graphics g, Graphics g1), and robotDrop(Graphics g, Graphics g1). The function of decideParts(Graphics g, Graphics g1) is used to check the status of parts in the machines. If the color of a part is green, the part is processing in the machine. If it is blue, the process has been completed and the part is waiting for unloading. If it is white, the machine is empty.

The animating processing of each alternative is completed in paint21(Graphics g, graphics g1) and paint22(Graphics g, graphics g1) for the robotic cell with two machines. The following functions are used to amimate the robotic cell with three machines: paint31(Graphics g, graphics g1), paint32(Graphics g, graphics g1), paint33(Graphics g, graphics g1), paint34(Graphics g, graphics g1), paint35(Graphics g, graphics g1), and paint36(Graphics g, graphics g1). All of them have a similar character. In each of these functions, a cycle number is used to determine how many times the cycle is executed during the simulation. After the cycle number is reached, the program terminates and the simulated cycle time is displayed. The simulated time is obtained by using simulation techniques. In fact, the simulation technique is relatively simple and easy. It is used to simulate each step of robot moves and to accumulate the simulated cycle time by adding the time of each step up.

The function actionPerformed (ActionEvent e) is used to run the program. When button events happen on the interface, the system automatically executes the program. The input data is collected. Thread instant is started. Once the thread instant is started, the run() is called. Based on the chosen alternative, the corresponding animating process will be run. During the animating process of an alternative, the button events of stop and pause can also be accepted. Both buttons can kill the Thread instant to stop or pause the program. The button of reset is used to reset the program.

4.2. Examples and analysis of results

The following examples are used to illustrate the simulation approach. The examples involve a robotic cell with two machines and a robotic cell with three machines. Tables 4.1 and 4.4 show the input data for the robotic cells with two and three machines, respectively. Based on the input data, the solution of the simulation cycle time and the calculated cycle time for both cells can be obtained. The calculated and the simulated cycle time for the robotic cell with two machines are shown in Tables 4.2 and 4.3, respectively. The calculated and the simulated cycle time for the robotic cell with three machines are shown in Tables 4.5 and 4.6, respectively. By comparing the simulated and the calculated cycle time, the sequence of robot moves and the optimal cycle time can be decided.

4.2.1 Examinations of the robotic cell with two machines program

The performance of the simulation for the robotic cell with two machines based on given robot sequences of Alternatives 1 and 2 is tested. The input data are given in Table 4.1.

Table 4.1: Input data of the robotic cell with two machines (time unit: minutes)

Testing	1	2	3	4	5	6	7	8
The picking up, loading, unloading, and dropping off time: ε	0.2	0.2	0.4	0.1	0.5	0.5	1.0	0.3
The robot moving time: δ	0.1	0.3	0.2	0.3	0.2	0.2	0.4	0.4
The processing time of machine 1: a	13	8	7	12	17	5	15	4
The processing time of machine 2: b	6	9	15	5	5	16	12	10

Based on the eight groups of testing data, the cycle time of T_1 and T_2 is obtained, as shown in Tables 4.2 and 4.3. Table 4.2 shows the calculated cycle time by using Sethi's method. Table 4.3 shows the simulated cycle time. c_1 represents the first cycle. c_2 represents the second cycle.

Table 4.2 Calculated cycle time of the robotic cell with two machines (time unit: minutes)

T_1	20.8	20	25.6	19.4	26.2	25.2	35.4	18.2
T_2	14.2	11	17.4	13.6	19.8	18.8	20.6	12.8

Table 4.3 Simulated cycle time of the robotic cell with two machines (time unit: minutes)

T_1	c_1	20.8	20	25.6	19.4	26.2	25.2	35.4	18.2
	c_2	20.8	20	25.6	19.4	26.2	25.2	35.4	18.2
T_2	c_1	14.2	11	17.4	13.6	19.8	18.8	20.6	12.8
	c_2	14.2	11	17.4	13.6	19.8	18.8	20.6	12.8

The results show that the simulated cycle time has the same values as theoretical solutions calculated by using Sethi's method. They also illustrate that the robotic cell is in a steady state after the initial step has been finished. The simulation tool provides an easy

procedure to obtain the optimal cycle time and consequently the best sequence of robot movement. It is an efficient tool to show users how the parts are processed in a robotic cell with two machines.

4.2.2 Examinations of the robotic cell with three machines program

Eight groups of input data were used, see Table 4.4. Based on these input data, the cycle time for Alternatives 1 through 6 can be obtained.

Table 4.4 Input data of the robotic cell with three machines (time unit: minutes)

Testing	1	2	3	4	5	6	7	8
The picking up, loading, unloading, and dropping time: ε	0.2	0.2	0.1	0.1	0.5	0.5	0.6	1.0
The robot moving time: δ	0.2	0.3	0.2	0.3	0.2	0.2	0.4	0.4
The processing time of machine 1: a	12	10	5	10	10	5	5	14
The processing time of machine 2: b	7	7	4	5	5	14	12	15
The processing time of machine 3: c	9	4	10	12	4	4	14	10

Table 4.5 Calculated cycle time of the robotic cell with three machines(time unit: minutes)

Testing	1	2	3	4	5	6	7	8
T_1	31.2	25.0	21.4	30.2	24.6	28.6	39.0	50.2
T_2	15.6	13.6	12.2	15.1	14.6	16.8	20.8	25.1
T_3	21.4	20	11.2	17.4	19.2	23.2	23	37.4
T_4	21.4	20	15.8	19.4	19.2	23.2	32	37.4
T_5	18.4	14	15.8	19.4	13.2	22.2	32	33.4
T_6	13.6	12	11.2	13.6	12.8	16.8	18	20.6

Table 4.6 Simulated cycle time of the robotic cell with three machines (time unit: minutes)

Testing		1	2	3	4	5	6	7	8
T_1	c_1	31.2	25.0	21.4	30.2	24.6	28.6	39.0	50.2
	c_2	31.2	25.0	21.4	30.2	24.6	28.6	39.0	50.2
	c_3	31.2	25.0	21.4	30.2	24.6	28.6	39.0	50.2
	c_4	31.2	25.0	21.4	30.2	24.6	28.6	39.0	50.2
T_2	c_1	17.6	13.6	15.2	18.6	14.6	20.8	30	30.6
	c_2	14.8	13.6	12.2	15.0	14.6	16.8	20.8	23.2
	c_3	16.4	13.6	12.2	15.2	14.6	16.8	20.8	27.0
	c_4	14.8	13.6	12.2	15.0	14.6	16.8	20.8	23.2
T_3	c_1	21.4	20	11.2	17.4	19.2	23.2	23	37.4
	c_2	21.4	20	11.2	17.4	19.2	23.2	23	37.4
	c_3	21.4	20	11.2	17.4	19.2	23.2	23	37.4
	c_4	21.4	20	11.2	17.4	19.2	23.2	23	37.4
T_4	c_1	21.4	20	15.8	19.4	19.2	23.2	32	37.4
	c_2	21.4	20	15.8	19.4	19.2	23.2	32	37.4
	c_3	21.4	20	15.8	19.4	19.2	23.2	32	37.4
	c_4	21.4	20	15.8	19.4	19.2	23.2	32	37.4
T_5	c_1	18.4	14	15.8	19.4	13.2	22.2	32	33.4
	c_2	18.4	14	15.8	19.4	13.2	22.2	32	33.4
	c_3	18.4	14	15.8	19.4	13.2	22.2	32	33.4
	c_4	18.4	14	15.8	19.4	13.2	22.2	32	33.4
T_6	c_1	13.6	12	6.2	11.6	12.8	16.8	16.0	20.6
	c_2	13.6	12	10.6	11.6	12.8	16.8	18	20.6
	c_3	13.6	12	11.2	11.6	12.8	16.8	18	20.6
	c_4	13.6	12	11.2	11.6	12.8	16.8	18	20.6

Based on the eight groups of input data, the cycle time of T_1, T_2, T_3, T_4, T_5, and T_6 can be obtained. Table 4.5 shows the result obtained by using the theoretical solution discussed by Sethi's. Table 4.6 shows the result of simulation. In Table 4.6, $c_i, i = 1, 2, 3, 4$ represents the cycle number.

Based on the results, the cycle time of Alternatives 1, 3, 4, and 5 is in a steady state after the initial step has finished. They have the same value as the cycle time calculated by using Sethi's method. For some cases of Alternatives 2 and 6, after the initial step has been finished, however, the cycle time varies. The value of cycle time can reach a steady state after several cycles. Sometimes, two cycles are necessary to reach the steady state. Once the cycle time is in a steady state, the cycle time can be obtained.

In some cases of Alternatives 2 and 6, the cycle time can reach a steady state after two cycles. The cycle time is different from that obtained by using Sethi's method. This is because Sethi's method was established on the hypothesis that the robotic cell is in a steady state. According to the solution provided by the simulation, the average time of two cycles equals the cycle time calculated by using Sethi's method. Therefore, the simulation tool proves Sethi's hypothesis. The simulations of Alternatives 2 and 6 are a bit more complex compared to the other four cases, because of the recursive expressions of waiting time, which has been described in Chapter 3.

4.3 Conclusions of the chapter

A computer program as a simulation tool for the robotic cell has been developed. This simulation tool has been used to implement the theoretical solutions discussed by Sethi, and to find optimal cycle time for a robotic cell with two or three machines. Previous

research reported by Sethi [8] assumed that the cell was in a steady state. However, the author did not mention how the cell reached a steady state, and also did not consider the fact that several cycles are needed before the robot movement of Alternatives 2 and 6 in a robotic cell with three machines can attain a steady state.

The simulation tool can be employed to illustrate the procedure of part processing in a robotic cell with two or three machines. In fact, the production of parts begins from the state that the cell is empty and the robot is at the input buffer ready to pick up a part. By simulating the procedure of part processing, it was observed that all the results of processes are in a steady state in the robotic cells with two or three machines. The steady state is that the waiting time of robot for each part on each machine is the same.

In the robotic cell with two machines, there are two robot movement cycles. After the initial step has finished, the robotic cell is in a steady state. In the robotic cell with three machines, there are six robot movement cycles. For the robot movement cycles of Alternatives 1, 3, 4, and 5, after the initial step has finished, the robotic cell is in a steady state. For Alternatives 2 and 6, however, several cycles are required to reach a steady state.

The simulation tool has proved to be fast and reliable for simulating robotic cells with two or three machines. It is an assistant tool to analyze the part processing in a robotic cell. It allows users to test different hypotheses. It gives users opportunities to practice using their knowledge. By using the simulation tool, the optimal cycle time of robotic cells can be obtained.

Chapter 5 Scheduling different part types in robotic cells with two and three machines

Identical part type problems in a robotic cell with two and three machines have been studied in Chapter 4. A simulation tool has been developed to solve the problems. In this chapter, scheduling different part types for a specific sequence of robot moves in a robotic cell with two or three machines is considered. The production process of a number of MPS (minimum part set) has been described in Chapter 3. Analytical expressions for optimal cycle time as well as the part sequence were obtained. Therefore, if the part sequence of an MPS is known, the cycle time of the particular sequence of robot moves in a robotic cell with two or three machines can be obtained. To use theses analytical expressions, we need to know the optimal MPS part sequence, which is a scheduling problem.

For the particular sequence of robot moves in a robotic cell with two or three machines, optimal scheduling of different part types belongs to combinatorial optimization, which is NP-hard. Some of the problems can be interpreted as the total distance of the traveling salesman problem (TSP), while the others are very difficult to be interpreted as TSP. Genetic algorithms utilizing randomization as well as direct smart search to seek the global optimum, are more suitable for the problems.

5.1 Outline of Genetic Algorithms

Genetic algorithms are a result of observing the principle of natural genetics and the theory of evolution. The genetic operations mimic the hereditary process of genes, i.e.,

creating new offspring from each generation. The evolution operation mimics the process of Darwinian evolution to create population from generation to generation. Genetic algorithms are different from conventional search techniques, and are stochastic search techniques based on the mechanism of natural selection and natural genetics. They start with an initial set of random solutions called initial population. The population consists of individuals. Each individual in the population is called a chromosome, which represents a solution to the problem. A chromosome is a string of symbols.

After the initial population is established, the chromosomes begin to evolve from one generation to another generation. In general, the population size is conveniently kept constant in each generation. During the process of evolving generation after generation, the chromosomes are evaluated by using some measures of fitness. Some better chromosomes, which are called parents, are selected to generate new chromosomes, which are called children or offspring. The new chromosomes are formed by using a crossover or mutation operator. After the new chromosomes are generated, the population consist of previous chromosomes and new chromosomes. To keep population size, only the better chromosomes are selected and the worse chromosomes are allowed to die.

By this way, a new generation is created. The chromosomes of new generation consist of surviving chromosomes of the previous generation and some new chromosomes created by the previous generation. It is clear that the new generation is stronger than the old one. The new generation is used to produce its next generation. After several generations, the algorithms converge to the best chromosome, which results in the optimal solution to the problem.

5.2 The Implementation of Genetic Algorithms

In order to find the optimal cycle time and part schedule for any given sequence of the robot moves in a robotic cell with two or three machines, the genetic algorithm should be designed in accordance with the problem. The genetic algorithm consists of encoding scheme, initial population, fitness evaluation function, parent selection, genetic operators, and evolution strategy, which are illustrated as follows.

5.2.1 Encoding scheme

Encoding scheme is used to encode a solution of the problem into a chromosome. Each chromosome should represent a part schedule. Each gene in a chromosome represents a part. Encoding scheme is key for genetic algorithm to solve the problem. Because the part schedule of a robotic cell belongs to a permutation schedule problem, the permutation of parts in an input buffer is designed to represent chromosomes for the problem. Each chromosome is unique. Figure 5.1 shows a chromosome with eight parts.

1	5	3	4	2	6	7	8

Figure 5.1 The representation of a chromosome

The part schedule in Figure 5.1 will be Part 1, Part 5, Part 3, Part 4, Part 2, Part 6, Part 7, and Part 8. Therefore, if there are n parts to produce in a robotic cell with two machines or three machines, each gene in a chromosome represents a part between 1 and n.

5.2.2 Initial population

Initial population is a group of individuals or chromosomes. After the initial population size and the chromosome representation are defined, the initial population is generated randomly. Each individual of the initial population is a part schedule. To use genetic algorithm, generating the initial population is the first step towards the part-scheduling solution for a robotic cell with two machines or three machines.

5.2.3 Fitness evaluation function

Fitness evaluation function is used to measure the fitness value of a particular individual. By evaluating each individual of a population, the fitness value of shorter cycle time can be obtained. For the part-scheduling problem of a robotic cell with two machines or three machines, the inverse of cycle time of each alternative is used to determine the fitness for each individual, respectively. The fitness evaluation function can be designed as follows:

$eval(individual) = 1/T$; T is the cycle time of an alternative. (5.1)

5.2.4 Parent selection

Parent selection is an important operation in genetic algorithms because better parents can produce better children. In this work, a roulette wheel approach is used to select parent individuals from the population to perform genetic operators. The approach is stochastic. Each individual occupies a `slice of a pie'. A random number is generated. If the number is between the cumulative probabilities of q_k and q_{k+1}, the individual k is selected. In this way, the strongest individual with the highest probability is selected by

spinning the roulette wheel. The roulette wheel approach includes the following steps [13].

(1). Calculate each fitness value for each individual.

$$eval(individual_k),\quad k = 1,2,..,pop_size \tag{5.2}$$

(2). Calculate the total fitness for the population

$$F = \sum_{k=1}^{pop_size} eval(individual_k) \tag{5.3}$$

(3). Calculate selection probability p_k for each individual.

$$p_k = \frac{eval(individual_k)}{F},\quad k = 1,2,...,pop_size \tag{5.4}$$

(4). Calculate cumulative probability q_k for each individual.

$$q_k = \sum_{j=1}^{k} p_j,\quad k = 1,2,...,pop_size \tag{5.5}$$

Where k is the number of individuals in the population.

5.2.5 Genetic operators

The genetic operators, which are used to yield children in each generation, consist of crossover and mutation. In this work, two genetic operators are introduced to deal with the sequence of robot moves for different part types in robotic cell with two or three machines. They are one point cut crossover and arbitrary two parts exchange mutation [36].

5.2.5.1 One point cut crossover

Crossover is the main genetic operator. Two parents are selected each time and generate

one child by combining both chromosomes' features. There are several types of crossover, such as one-point crossover and two-point crossover. A simple way to achieve crossover is one point cut crossover, which is shown in Figure 5.2.

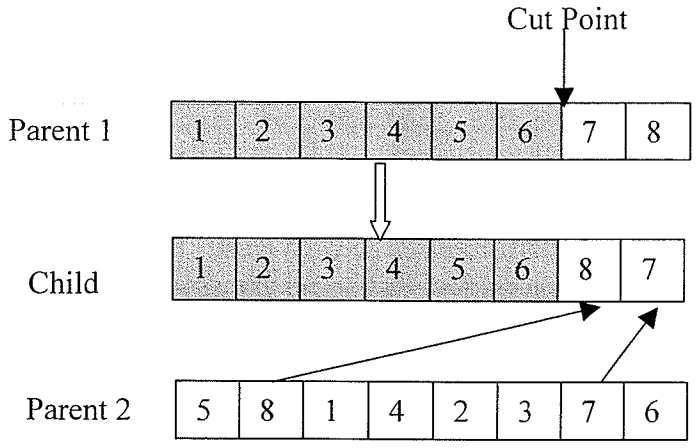


Figure 5.2 One point cut crossover

Figure 5.2 shows that one cut point is randomly generated. Parent 1 is divided into two parts by the cut point. The child inherits the left part of Parent 1. The other parts of the child come from Parent 2, in the same order as that of Parent 2.

5.2.5.2 Mutation-Arbitrary two parts exchange

Mutation, which is performed by random changing gene positions within a chromosome, is the other operator that is used to explore for the best chromosomes. Mutation can cause small changes in the population, to prevent local convergence. There are several types of mutation operators. In this work, arbitrary two parts exchange mutation [36] is designed to perform the mutation operation, as illustrated in Figure 5.3.

Two randomly selected parts in the parent of Figure 5.3 exchange their positions to generate a child.

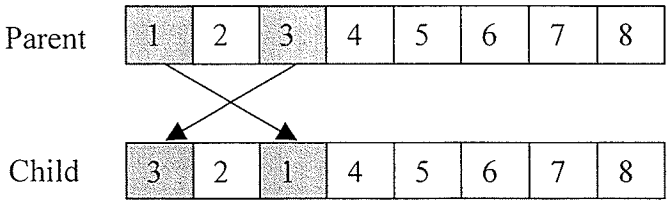


Figure 5.3 Arbitrary two parts exchange mutation

5.2.6 Evolution Strategy

Evolution strategy is used to select chromosomes with higher fitness characteristics and to eliminate the weaker chromosomes. For the genetic algorithms, after genetic operations, the total population increases due to the newly created children. To keep the population size constant, a new population and children for the next generation is selected from the old population, based on evolution strategy. After selection, the new population containing only the best or the better individuals, is better than the old one. In this way, the population becomes better and better from generation to generation. The optimal solutions are obtained after several generations.

5.3 Overall algorithm

A genetic algorithm is conducted by combining one point cut crossover and arbitrary parts exchange mutation. It is described as follows:

Input: the number of parts; the robot travel time; pickup, loading, unloading, and dropping time; the processing time of each part on Machines 1 and 2; population size; and the probability of crossover and mutation.

Output: cycle time and part schedule.

Beginning:

Step 1: Randomly generate the initial population.

Step 2: If the final criterion is not satisfied, go to Step 3. Otherwise output the result.

Step 3: Generate a random number.

Step 4: If the number satisfies the probability of crossover, two parents are selected from the initial population by the roulette wheel approach in order to generate one child. Otherwise one parent is selected from initial population by the roulette wheel approach in order to generate one child.

Step 5: If the number of children is not satisfied, go back to Step 3. Otherwise go to Step 6.

Step 6: Use fitness function to select the best population size and individuals from the initial population and the children.

Step 7: Go back to Step 2.

5.4 The GA program

Java language is used to write the genetic algorithm program. Two programs are designed to carry out genetic algorithms for the part sequences of robotic cells with two and three machines respectively. A graphic window allows inputting the size of population, the size of children, the probability of mutation, generation, the number of parts, pickup, loading, unloading, dropping time, and the robot moving time. The part sequences and cycle time are displayed on the window. The converging process of each alternative is output to a text file. Figures 5.4 and 5.5 show the operation windows for different part types in robotic cells with two and three machines respectively. The programs based on genetic

algorithm provide several functions to create initial population, to evaluate fitness, to select parent, to perform genetic operations, and to select new population.

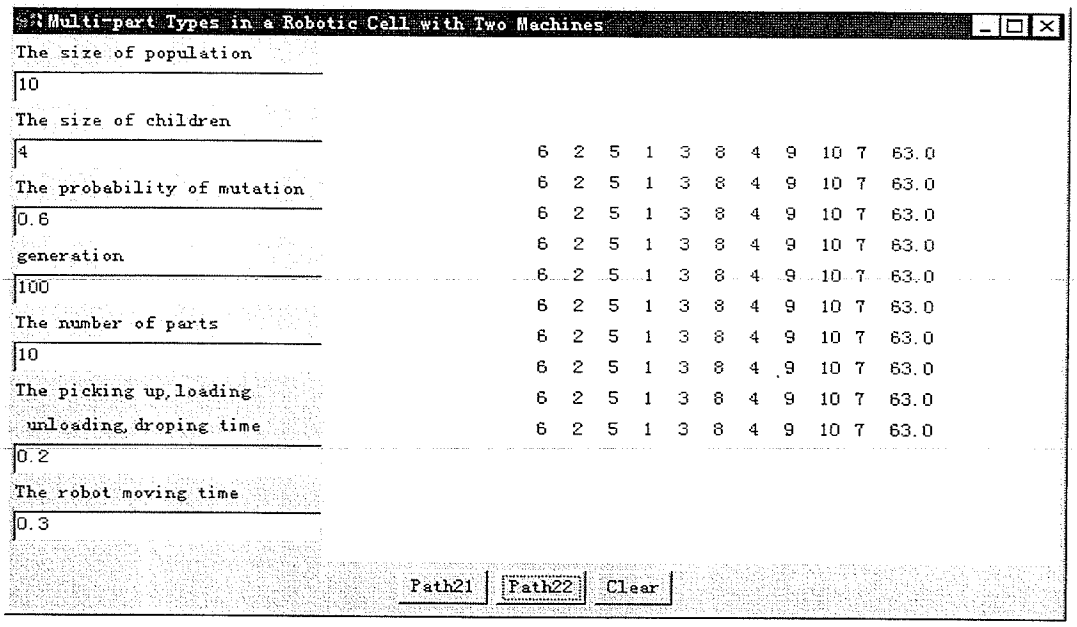


Figure 5.4 A graphical user interface for different part types in a robotic cell with two machines

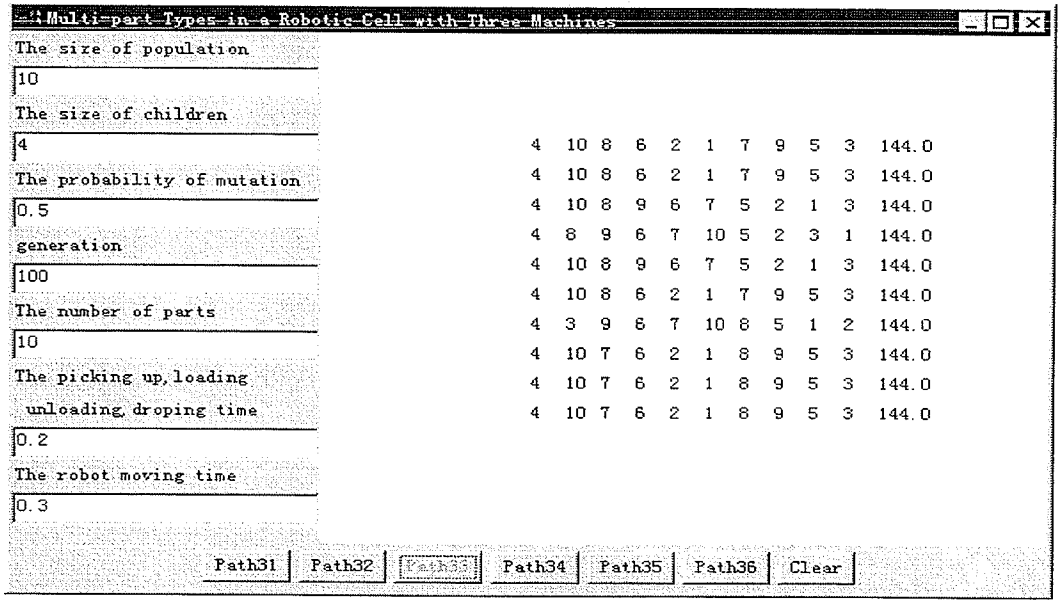


Figure 5.5 A graphical user interface for different part types in a robotic cell with three machines

Two text files are used to input part processing time for cells with two and three machines respectively. For the robotic cell with two machines, the part processing time input is the processing time of different parts on Machines 1 and 2. For the robotic cell with three machines, the part processing time input is the processing time of different parts on Machines 1, 2, and 3. The solution of alternative 2 for the robotic cell with two machines is output to T22 text file and the part sequence is displayed in the window. For the robotic cell with three machines, the solutions of alternatives 2 through 6 are output to T32, T33, T34, T35, and T36, respectively. Both programs use similar programming processes. Therefore, only the program design of the robotic cell with three machines is introduced here. The program flow chart is shown in Figure 5.6.

As shown in Figure 5.6, after the program has been loaded and all data has been input, the program waits for an action event. The action event function is ActionPerformed(ActionEvent e). The function of ActionPerformed (ActionEvent e) calls the function of alternative i(Graphics g1), $i = 1, 2, 3, 4, 5,$ and 6 according to the buttons selected. The cycle time of alternative 1 is not affected by part sequence (see Chapter 3). Therefore, the cycle time can be calculated with a polynomial time. However, the cycle time of alternatives 2 through 6 is affected by part sequence. The genetic algorithm is applied to alternatives 2 through 6. The following functions are designed to write the program.

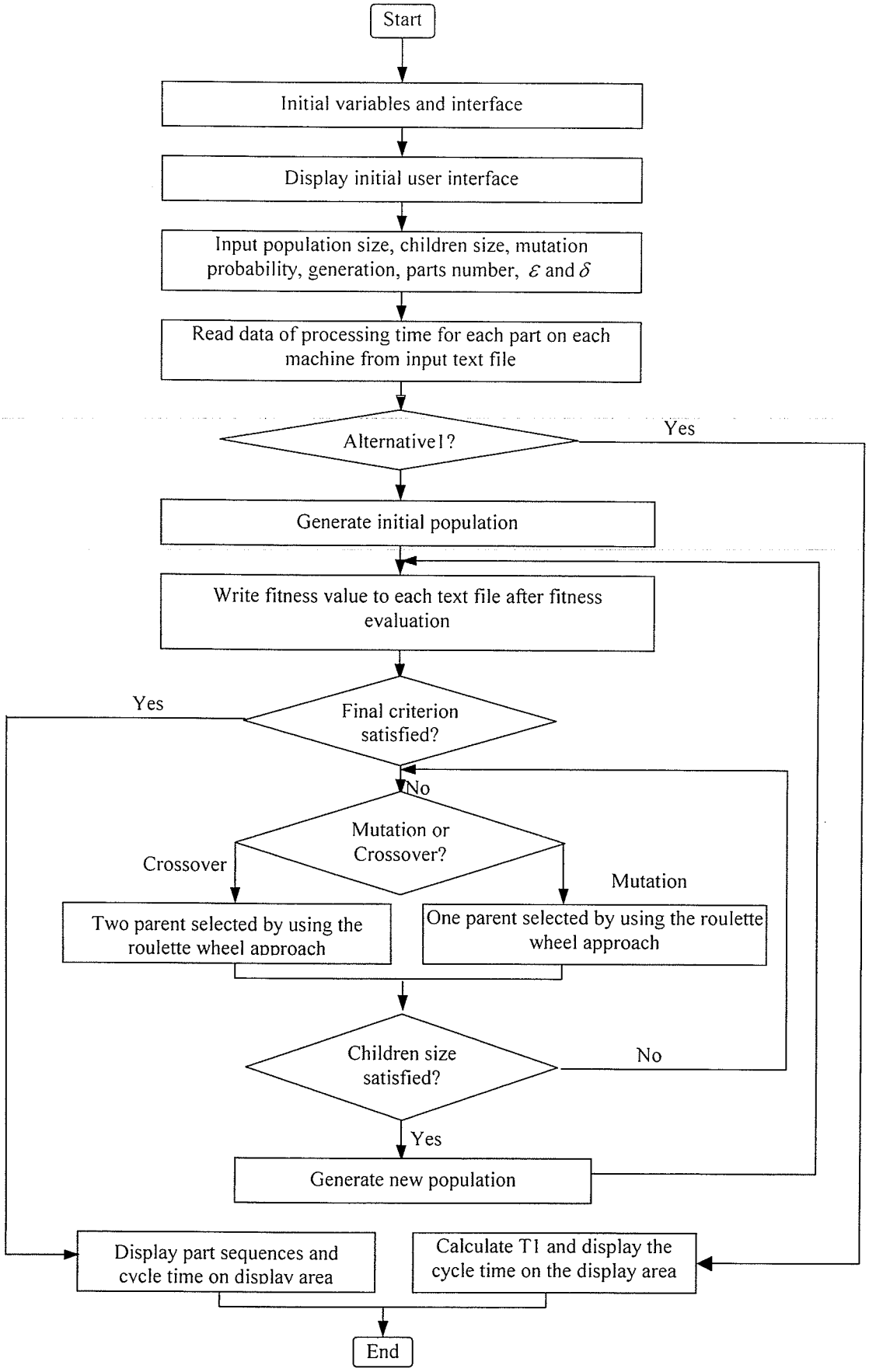


Figure 5.6 The flow chart of the program

5.4.1 Initial population function

Initial population function is indicated by int [][] RepRandom(). The value returned is an integer multi-dimensional array. Each row of the multi-dimensional array represents a chromosome. The length of chromosome is the number of parts. The number of rows is population size. The method of a random generator is used to create numbers randomly. The range of the number is between 0 and 1. For example, if the number of parts is five, the array of a row may be $\{0.5,0.9,0.2,0.1,0.8\}$ and the order is $\{1,2,3,4,5\}$. Bubble sort method is used to sort the array in descending order. The array will become $\{0.9,0.8,0.5,0.2,0.1,0.8\}$. The corresponding part sequence is $\{2,5,1,3,4\}$. This is the chromosome of initial population. In this way, the initial population is generated.

5.4.2 Objective function

Three_M_Path i(int[][] chromosome), $i=2$, 3, 4, 5, and 6, are the objective functions for alternatives 2 through 6 respectively. The parameters of the function are a multi-dimensional array of chromosomes. Given part sequences (chromosomes), their cycle time can be calculated. For alternatives 3, 4, and 5, the cycle time is easy to obtain using the formulations presented in Chapter 3, because the robotic cell is in a steady condition after the initial step. For alternatives 2 and 6, the robotic cell cannot reach a steady state after the initial step. But a steady state can be reached after several cycles of robot moves, which has been proved in Chapter 4. The steady state was analyzed by Hall et al [37]. For alternative 2, if the number of parts is even, a single MPS defines a steady state. Otherwise, two MPS may be required. For alternative 6, a single MPS is needed to define

a steady state. The results of Hall et al [37] are used to program Three_M_Path2(int[][] chromosome) and Three_M_Path6(int[][] chromosome).

5.4.3 Selection function

Function selection$i$(int select number), $i=2$, 3, 4, 5, and 6, is used to select parents to prepare the operation of crossover or mutation for alternatives 2 through 6 respectively. A roulette wheel approach is used for the selection. The select number is used to decide parent number. Only one parent is needed for mutation, while two parents are required for crossover.

5.4.4 Crossover function

Crossover function is crossover(int[] parent1, int[] parent2). The parameters of the crossover function are two parents, an array of Parent 1 and an array of Parent 2. The value of return is an array of child. The method of random() generates a random number between 0 and 1. The number of a cut point equals the integer of parts number times the random number. The array of Parent 1 is separated into two parts by the cut-point number. The left of the array of Parent 1 is copied to the left of the child array. The right of the array of the child comes from Parent 2.

5.4.5 Mutation function

Mutation function is mutation (int[] mutationChromosome). The parameter of the mutations function is an array of mutationChromosome. The value of return is an array of mutationChromosome. The method of random() generates two random numbers between

0 and 1. Two numbers of Points 1 and 2 equal the two random numbers times parts numbers respectively. The two parts at Points 1 and 2 in the array are exchanged. The array of mutationChromsome is obtained. The value of return is the array of a child.

5.4.6 Selecting new population function

The newpopulation i (int[][] parent,int[][] children), $i = 2$, 3, 4, 5, and 6, is used to conduct in the selection of new population for alternatives 2 through 6 respectively. The parameters of selecting new population function are the multi-dimensional arrays of old population and of children. The value of return is a multi-dimensional array of the new population. In the function, a multi-dimensional array of total population is used to copy old population and children. Each chromosome of total population is evaluated using fitness function. According to the value of the evaluation, all the chromosomes are arranged from high fitness to low fitness. The multi-dimensional array of new population, whose size equals that of initial population, consists of selected chromosomes with higher fitness.

5.5 Testing of the GA program

In this section, a simple method is used to test the GA program for a robotic cell with two or three machines respectively. The method assumes that ten identical parts are processed in a robotic cell. Due to the same processing time of these parts on the same machine, different part sequences for each alternative have the same cycle time. The cycle time can be obtained by using GA program. For identical parts, one-part cycle time can be obtained by using the simulation tool. If the cycle time obtained by using GA program

equals ten times as long as the cycle time obtained by using simulation tool, the GA program is verified. In the robotic cell with two machines, the input data are given in Table 5.1.

Table 5.1 Input testing data for a robotic cell with two machines (time unit: minutes)

The picking up, loading, unloading, and dropping off time: ε	The robot moving time: δ	The processing time of machine 1: a	The processing time of machine 2: b
0.4	0.2	7	15

Based on these input data, the cycle time for alternatives 1 and 2 by using the simulation tool is 25.6 and 17.4 minutes respectively. The cycle time for alternatives 1 and 2 by using GA program is 256 and 174 minutes respectively. The cycle time of alternative 2 is show in Figure 5.7.

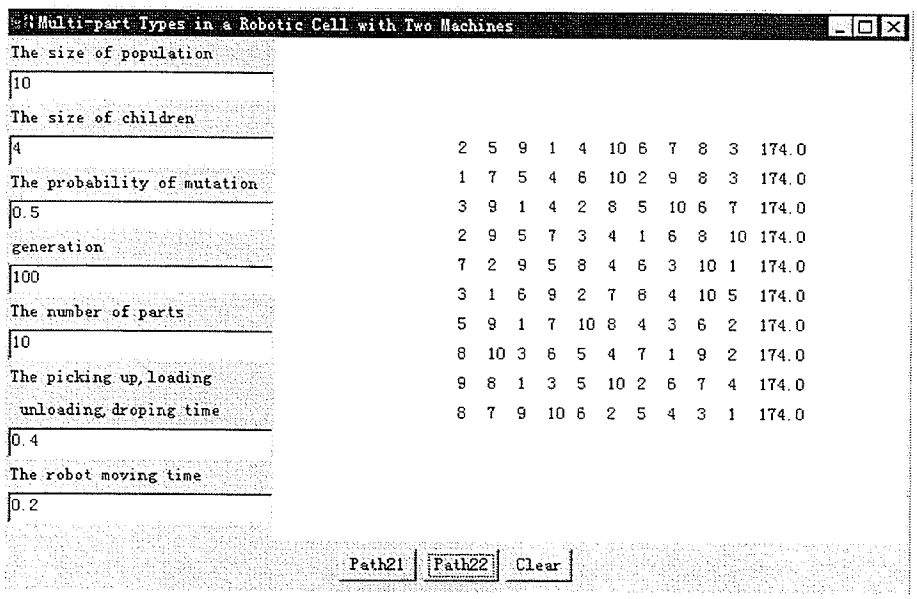


Figure 5.7 The cycle time obtained by using GA program for a robotic cell with two machines

In the robotic cell with three machines, the input testing data are given in Table 5.2.

Table 5.2 Input testing data for a robotic cell with three machines (time unit: minutes)

The picking up, loading, unloading, and dropping off time: ε	The robot moving time: δ	The processing time of machine 1: a	The processing time of machine 2: b	The processing time of machine 3: c
0.5	0.2	10	5	4

According to the input test data for a robotic cell with three machines, the cycle time of alternative 1 through 6 obtained by using the simulation tool are 24.6, 14.6, 19.2, 19.2, 13.2, and 12.8 minutes, respectively. The cycle time for alternative 1 obtained by using GA program is 246 minutes. The cycle time for alternative 2 through 6 is shown in Figure 5.8

The results show that the cycle time obtained by using GA program is ten times as long as that obtained by using the simulation method. It proves that the cycle time of a number of identical parts obtained by using GA program is correct.

3	8	2	10	9	4	7	1	5	6	146.0
1	7	2	4	6	5	10	3	8	9	146.0
7	2	10	5	1	8	3	4	6	9	146.0
9	5	2	3	10	4	6	8	1	7	146.0
5	9	7	4	6	3	2	1	10	8	146.0
1	8	9	6	4	10	5	2	3	7	146.0
10	9	4	8	7	3	2	5	1	6	146.0
2	4	8	3	7	10	6	5	9	1	146.0
4	9	2	8	10	6	7	1	5	3	146.0
2	10	4	9	7	5	8	1	6	3	146.0

Alternative 2

10	1	4	3	7	8	9	5	6	2	192.0
9	8	6	5	1	3	7	10	4	2	192.0
6	10	1	4	2	7	9	5	3	8	192.0
8	4	7	3	1	6	9	5	10	2	192.0
6	2	5	8	10	3	9	4	7	1	192.0
1	7	3	4	6	8	10	2	9	5	192.0
10	5	3	2	8	4	9	7	6	1	192.0
2	7	9	6	5	10	4	3	8	1	192.0
2	1	3	8	5	7	4	9	6	10	192.0
1	10	8	5	2	9	4	6	3	7	192.0

Alternative 3

5	4	8	9	7	1	2	3	6	10	192.0
3	10	8	4	9	6	1	5	7	2	192.0
3	5	8	1	10	2	7	6	9	4	192.0
3	2	6	7	8	9	1	4	10	5	192.0
3	10	7	9	2	8	5	1	6	4	192.0
7	9	1	10	8	4	6	5	2	3	192.0
10	3	2	7	6	4	9	1	8	5	192.0
4	10	9	3	7	1	2	8	5	6	192.0
10	5	2	7	6	8	3	9	1	4	192.0
8	6	7	1	3	5	4	2	10	9	192.0

Alternative 4

10	2	5	4	3	7	9	6	1	8	132.0
9	5	2	10	3	6	8	7	4	1	132.0
6	1	7	2	10	3	9	5	8	4	132.0
7	8	5	2	9	6	3	10	1	4	132.0
9	8	3	10	6	1	7	5	4	2	132.0
2	8	7	9	6	4	5	10	1	3	132.0
3	5	8	10	2	1	6	4	7	9	132.0
10	4	5	8	9	2	6	3	1	7	132.0
10	4	2	5	3	8	1	7	9	6	132.0
3	5	7	1	2	9	10	8	6	4	132.0

Alternative 5

3	1	7	4	8	6	2	5	10	9	128.0
10	3	8	5	4	6	1	9	7	2	128.0
5	8	9	6	7	1	3	10	4	2	128.0
1	8	7	5	3	6	4	10	9	2	128.0
9	3	5	2	7	1	10	4	8	6	128.0
6	5	10	9	4	8	7	2	1	3	128.0
6	7	8	5	4	9	3	10	1	2	128.0
9	10	6	3	5	1	7	2	4	8	128.0
6	5	10	2	8	7	9	3	1	4	128.0
3	5	4	10	1	9	2	7	8	6	128.0

Alternative 6

Figure 5.8 The cycle time obtained by using GA program for a robotic cell with three machines

5.6 Experiments and results

The genetic algorithm is programmed with Java language. Two examples are designed to illustrate the part-scheduling problems. One is for a robotic cell with two machines. The other is for a robotic cell with three machines.

5.6.1 Different part types scheduling in the robotic cell with two machines

For the robotic cell with two machines, an example with ten parts is presented. The minimal production ratio (MPR)[5] is $r = (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8) = (1, 2, 1, 1, 1, 2, 1, 1)$. Robot travel time δ is 0.3 minutes. Pickup, loading, unloading, and dropping time ε is 0.2 minutes. The processing time on Machines 1 and 2 is given in Table 5.3.

Table 5.3 The part number, type and processing time for a robotic cell with two machines

Part number	1	2	3	4	5	6	7	8	9	10
Part Type	P1	P2	P2	P3	P4	P5	P6	P6	P7	P8
M1 (min)	5.0	3.0	3.0	4.0	4.5	4.5	6.0	6.0	2.0	4.0
M2 (min)	3.0	4.5	4.5	3.0	5.0	3.0	4.0	4.0	3.5	6.0

The objective is to find the optimal solution of part schedule for alternatives 1 and 2. By comparing alternatives 1 and 2, the optimal cycle time can be obtained.

For alternative 1, it is clear that any part schedule at input buffer does not affect cycle time T_1 (see Section 3.1.1 of Chapter 3). Cycle time T_1 depends on ε, δ, as well as processing time on each machine for each part. T_1 is independent of part schedule at input buffer. Cycle time T_1 is 90.75 minutes for this example.

For alternative 2, four sets of operation rates are used. They are [crossover, mutation]=[0.2, 0.8], [0.4, 0.6], [0.6, 0.4], and [0.8, 0.2], respectively. Based on the four groups of operation rates and the population size of 10 individuals, the change of cycle time from generation to generation is shown in Figure 5.9.

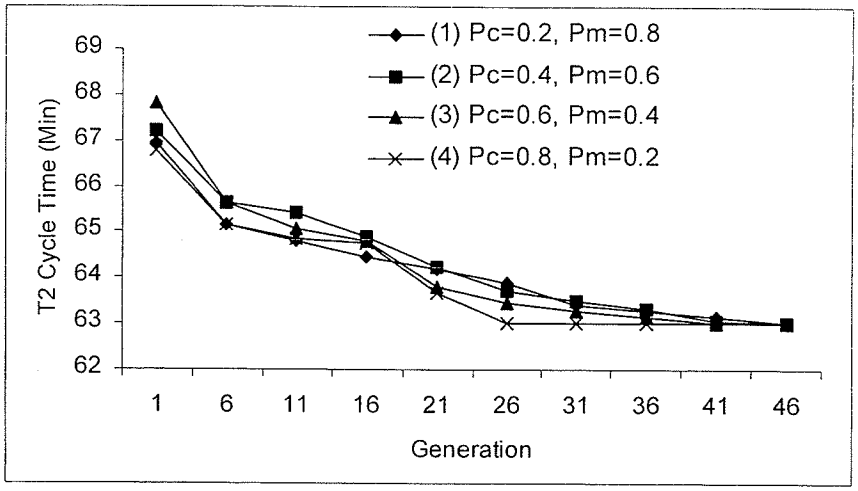


Figure 5.9 The converging process of alternative 2 for a robotic cell with two machines

Figure 5.9 shows that cycle time is different for different operation rates. This means that cycle time is affected by part schedule. For different sets of operation rates, cycle time converges to the same value after 45 generations. The four sets of operation rates are used to prevent local optimization. The final solution is globally optimum because the cycle time of the different sets of operation rates converges on the same value.

The optimal cycle time of alternatives 1 and 2 is 90.75 and 63.0 minutes respectively. Therefore, the optimal sequence is alternative 2.

Alternative 2 is tested with crossover 0.5 and mutation 0.5. The converging process is shown in Figure 5.10. The part-processing sequence obtained is shown in Figure 5.11. Each row on the left side of the vertical line in Figure 5.11 represents a part-processing

sequence. Each row on the right side of the line represents the cycle time of each schedule. All the optimal cycle time is 63.0 minutes while the part-processing sequences are not the same. There are several solutions for part schedule, and each sequence satisfies the optimal cycle time.

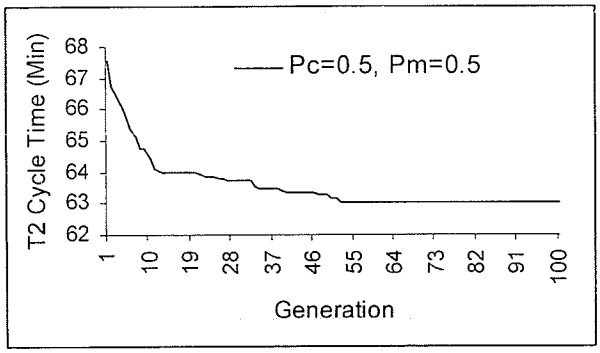


Figure 5.10 T_2 cycle time when Pc=0.5 and Pm=0.5 of alternative 2 for a robotic cell with two machines

9	5	7	10	8	4	2	6	3	1	63.0
7	1	3	6	9	10	8	4	2	5	63.0
9	5	7	10	8	4	3	6	2	1	63.0
9	5	7	10	8	4	3	6	2	1	63.0
9	5	7	10	8	4	2	6	3	1	63.0
7	6	3	1	9	10	8	4	2	5	63.0
9	5	7	10	8	4	3	6	2	1	63.0
9	5	7	10	8	4	3	6	2	1	63.0
7	6	3	1	9	10	8	4	2	5	63.0
9	5	7	10	8	4	2	6	3	1	63.0

Figure 5.11 Optimal part sequence and cycle time of alternative 2 for a robotic cell with two machines

5.6.2 Different part types scheduling in the robotic cell with three machines

An example with ten parts is used to illustrate the part schedule problem in the robotic cell with three machines. The minimal production ratio is the same as that used in the robotic cell with two machines. Assume robot travel time δ is 0.3 minutes, and pickup, loading, unloading, and dropping time ε is 0.2 minutes. The processing time on Machines 1, 2, and 3 is listed in Table 5.4.

For alternative 1, it is similar to the case in a robotic cell with two machines (see Section 3.2.1 of Chapter 3). The cycle time T_1 depends on ε, δ, as well as processing time of each part on each machine. T_1 is independent of part schedule at input buffer. Cycle time T_1 is 241 minutes.

Table5.4The part number, type and processing time for a robotic cell with three machines

Part number	1	2	3	4	5	6	7	8	9	10
Part type	P1	P2	P2	P3	P4	P5	P6	P6	P7	P8
M1 (min)	6.0	4.0	4.0	5.0	7.0	9.0	8.0	8.0	9.0	6.5
M2 (min)	3.0	5.0	5.0	4.0	6.0	5.0	3.5	3.5	6.5	6.0
M3 (min)	5.0	4.0	4.0	8.0	9.0	6.0	13.0	13.0	14.0	11.0

For alternatives 2 through 6, four sets of operation rates, which are [crossover, mutation]=[0.2, 0.8], [0.4, 0.6], [0.6, 0.4], and [0.8, 0.2], are used to obtain the optimal cycle time for each alternative respectively. By comparing each alternative, including alternative 1, the optimal cycle time of the robotic cell with three machines can be obtained. Based on the four groups of operation rates and the population size of 10 individuals, the converging processes of the cycle time of alternatives 2 through 6 are shown in Figures 5.12 through 5.16 respectively.

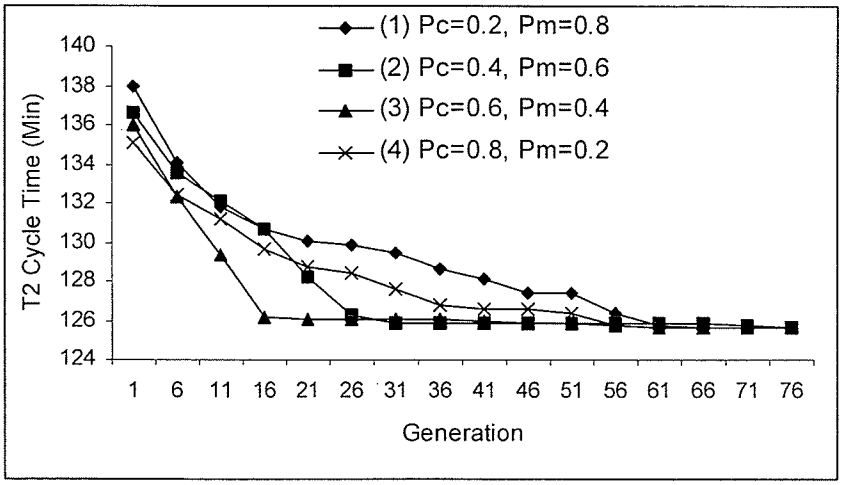


Figure 5.12 The converging process of alternative 2 for a robotic cell with three machines

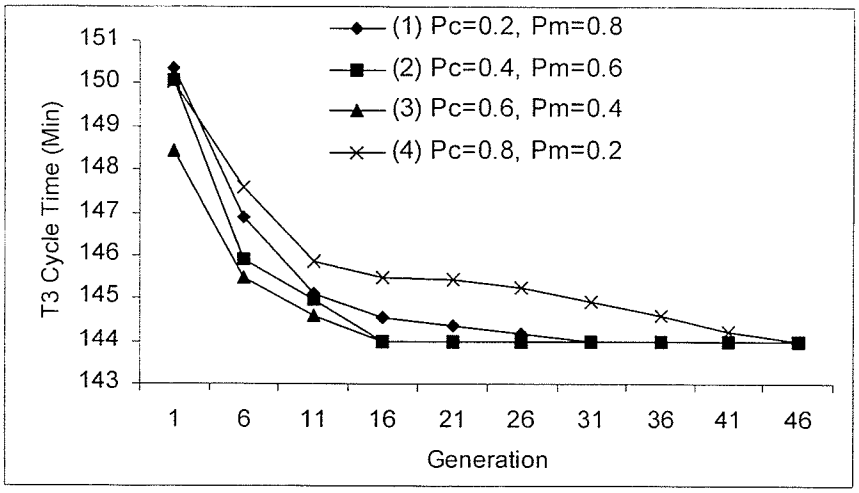


Figure 5.13 The converging process of alternative 3 for a robotic cell with three machines

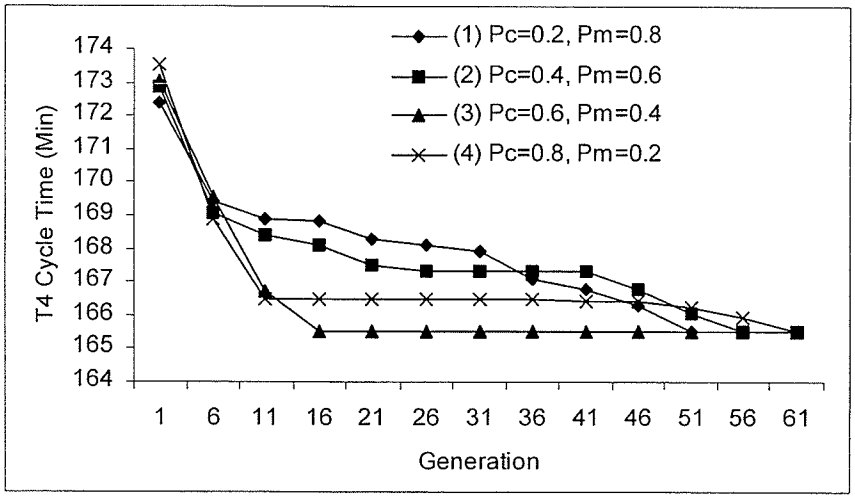


Figure 5.14 The converging process of alternative 4 for a robotic cell with three machines

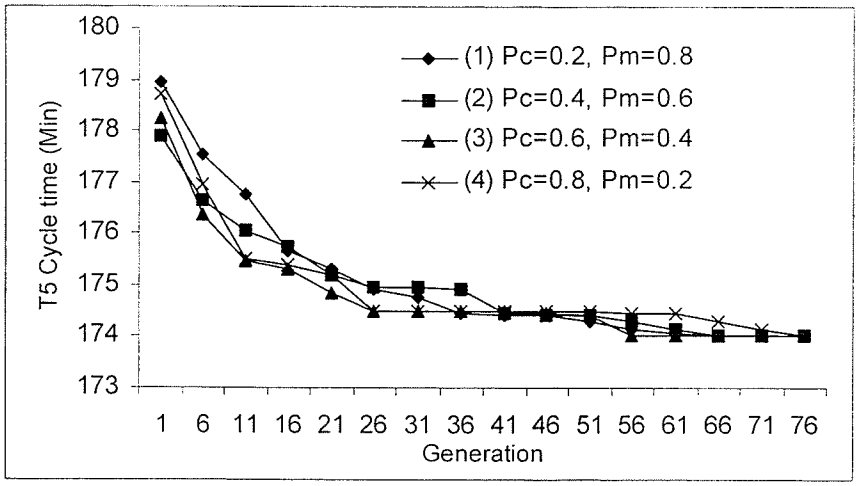


Figure 5.15 The converging process of alternative 5 for a robotic cell with three machines

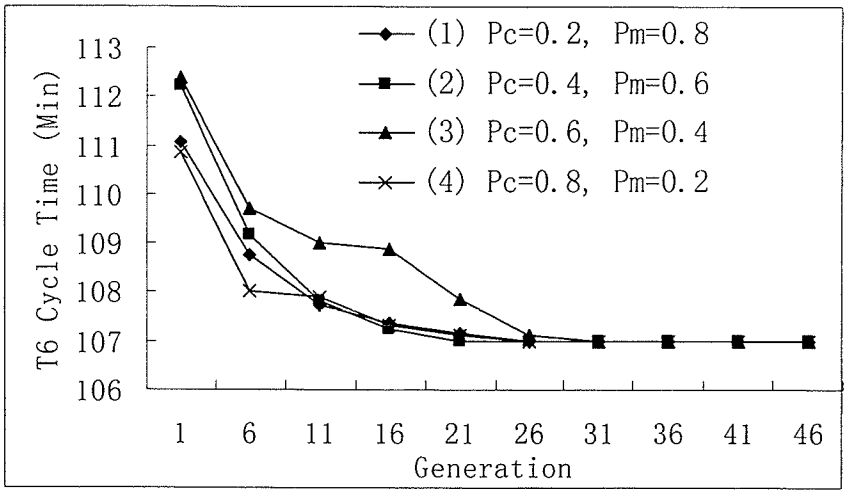
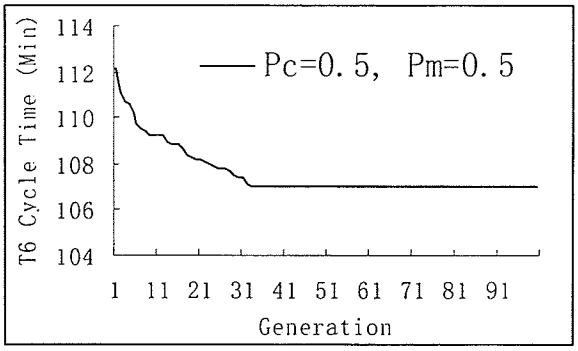


Figure 5.16 The converging process of alternative 6 for a robotic cell with three machines

Figures 5.12 through 5.16 display the results of each alternative with a random start of part sequence. The cycle time of four groups of operation rates converges on the same value, but on respective values for alternatives 2 through 6. Therefore, the cycle time of alternatives 2 through 6 can be obtained easily in these graphs. The cycle time of alternatives 1 through 6 is 241, 125.6, 144, 165.5, 174, and 107 minutes, respectively. Alternative 6 with the shortest cycle time is therefore, the optimal sequence.

Alternative 6 is tested with crossover 0.5 and mutation 0.5. The converging process is shown in Figures 5.17. The part schedule and the optimal cycle time are obtained. The optimal cycle time is 107 minutes. There are several solutions for part schedule, and each sequence satisfies the optimal cycle time.



3	7	5	6	1	2	4	8	10	9	107.0
3	7	5	6	1	2	4	8	10	9	107.0
3	7	5	9	6	10	1	2	4	8	107.0
3	7	5	6	1	2	4	8	10	9	107.0
3	7	5	6	1	2	4	8	10	9	107.0
3	7	5	6	1	2	4	8	10	9	107.0
3	7	5	9	6	10	1	4	2	8	107.0
3	7	5	6	1	2	4	8	9	10	107.0
3	7	5	1	6	2	4	8	10	9	107.0
3	7	5	6	1	2	4	8	10	9	107.0

Figure 5.17 The optimal part sequences and cycle time for a robotic cell with three machines

It is clear that the optimal cycle time and robot moving sequence have been obtained by using GA for different part types scheduling in robotic cells with two and three machines. Computational results indicate that the GA approach developed is feasible for the scheduling problems of different parts in robotic cells with two or three machines.

Chapter 6 Conclusions and Future work

In this work, the problems of identical and different part type in robotic cells with two and three machines have been investigated. The objective was to find the optimal cycle time for the robotic cells. For identical parts, where in all parts are of the same type, the sequence of part input vanishes. Therefore, cycle time depends only on robot move sequences in the robotic cells. For different part types, different sequences of part input may result in different cycle times, since all the parts require separate processing time. Consequently, both robot moving sequences and part sequences must be considered for optimal cycle time.

The production processes of parts in robotic cells with two and three machines are described in Chapter 3. A number of MPS repeatedly produced in the robotic cells is considered. The production of parts begins from the state that the cell is empty and the robot is at input buffer ready to pick up a part. A mathematical model is presented to describe the characteristics of a robotic cell with two or three machines. These formulations are used for both identical and different part types.

For identical parts in a robotic cell with two or three machines, a simulation tool has been developed. All results of tests are obtained through the programmed graphical windows, where input data and the optimal cycle time are displayed. For the same input data, the robot moving sequence decides the optimal cycle time. The Web-based simulation tool we developed is helpful in understanding the basic concept of product processing in a robotic cell. It lets users easily comprehend the process of part manufacture in a robotic cell. It also allows users to do different tests and to analyze the part process in a robotic

cell. By using the simulation tool, the optimal cycle time has been obtained. This simulation tool had been used in Dr. Peng's course (25.485-Computer Integrated Manufacturing and Automation) for two years. Students have used this tool in FMS study and assessment. It is efficient and practical.

For different parts in a robotic cell with two or three machines, the repetitive manufacturing process of an MPS was considered. The different parts belong to a part family. The objective is to minimize cycle time, by determining the sequence of parts and robot moves. An approach of genetic algorithm is proposed with a detailed description. Illustrative examples of the robotic cells with two and three machines are discussed.

The examples indicate that the genetic algorithm is effective and practical for optimizing the sequence of robot moves in robotic cells with two and three machines for different parts. The genetic algorithms can rapidly find the optimal solution and part schedule. However, convergence speed should be controlled to prevent local optimization. Different schemes of mutation and crossover result in different converging speeds of the algorithm. A set of the operation rates of crossover and mutation can be used to find the global optimal solution. Since genetic algorithms are stochastic approaches, the performance of the approaches is affected by operation rates. Based on the four groups of the operation rates of mutation and crossover, genetic algorithm is run several times until the same solution is obtained. The examples demonstrate that the genetic algorithm could easily solve the problems. So far, the part sequence problems associated with all the robot moving cycles in a robotic cell with two or three machines have been solved.

Further work for part processing in a robotic cell may include the following.

1. Introduce different loading and unloading time to different machines, and different travel time between machines. In the mean time, consider the corresponding constraints.
2. Investigate parts with different features, and different set-up time.
3. Include a fixed set of parts in the design of robot cells.
4. Consider the limited buffer storage in a robotic cell, and the trade-off between throughput rate and inventory cost.

References

[1] R. Macchiaroli and S. Riemma, "Clustering methods for production planning and scheduling in a flexible manufacturing system", Proceedings - IEEE International Conference on Robotics and Automation, pp.3155-3160, 1994.

[2] J. S. Basran, E. M. Petriu, and D. C. Petriu, "Flexible agent-based robotic assembly cell", Proceedings - IEEE International Conference on Robotics and Automation, Vol.4, pp.3461-3466, 1997.

[3] H. N. Geismar, C. Sriskandarajah, and N. Ramanan, "Increasing throughput for robotic cells with parallel machines and multiple robots", IEEE Transactions on Automation Science and Engineering, Vol.1, pp.84-89, 2004.

[4] P. Freedman and R. Alami, "Repetitive sequencing: From workcell tasks to workcell cycles", Proceedings - IEEE International Conference on Robotics and Automation, pp.1962-1967, 1990.

[5] N. Singh, "Systems Approach to Computer-Integrated Design and Manufacturing", John Wiley&Sons, Inc., 1996

[6] R. Logendran and C. Sriskandarajah, "Sequencing of robot activities and parts in two-machine robotic cells", International Journal of Production Research, Vol.34, pp.3447-3463, 1996.

[7] R. K. Miller and T. C. Walker, "FMS/CIM Systems Integration Handbook", The Fairmont Press, Lilburn, GA., 1990.

[8] S. P. Sethi, R. Chandrasekaran, I. Drobouchevitch, and C. Sriskandarajah, "Scheduling Multiple Parts in a Robotic Cell Served by a Dual-Gripper Robot", operations research, Vol. 52, pp.65-82, 2004.

[9] N. Brauner and G. Finke, "Optimal moves of the material handling system in a robotic cell", International Journal of Production Economics, Vol.74, pp.269-277, 2001.

[10] S. P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz, W. Kubiak, "Sequencing of parts and robot moves in a robotic cell", International Journal of Flexible Manufacturing Systems, Vol.4, pp.331-358, 1992.

[11] N. G. Hall, H. Kamoun, and C. Sriskandarajah, "Scheduling in robotic cells: Complexity and steady state analysis", European Journal of Operational Research, Vol.109, pp.43-65, 1998.

[12] H. Chen, C. Chu, and J. Proth, "Sequencing of parts in robotic cells", International Journal of Flexible Manufacturing Systems, Vol.9, pp.81-104, 1997.

[13] M. Gen and R. Cheng, "Genetic Alogorithms and Engineering Design" John Wiley & Sons, Inc, 1996.

[14] Y. Crama and J. V. D. Klundert, "Cyclic scheduling of identical parts in a robotic cell", Operations Research, Vol.45, pp.952-965, 1997.

[15] A. Agnetis, "Scheduling no-wait robotic cells with two and three machines", European Journal of Operational Research, Vol.123, pp.303-314, 2000.

[16] F. T. S. Chan, K. L. Mak, L. H. S. Luong, and X. G. Ming, "Machine-component grouping using genetic algorithms", Robotics and Computer-Integrated Manufacturing, Vol.14, pp.339-346, 1998.

[17] Y. P. Aneja and H. Kamoun, "Scheduling of parts and robot activities in a two machine robotic cell", Computers & Operations Research, Vol.26, No.4, pp.297-312, 1999.

[18] N. G. Hall, H. Kamoun, and C. Sriskandarajah, "Scheduling in robotic cells: classification, two and three machine cells", Operations Research, Vol.45, pp.421-439, 1997.

[19] A. Agnetis and D. Pacciarelli, "Part sequencing in three-machine no-wait robotic cells", Operations Research Letters, Vol.27, pp.185-192, 2000.

[20] D. P. Garg and C. D. Poppe, "Coordinated robots in a flexible manufacturing work cell", IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Vol. 1, pp.648-653, 2001.

[21] J. Bautista, A. Lusa, R. Suarez, M. Mateo, R. Pastor, and A. Corominas, "Application of genetic algorithms to assembly sequence planning with limited resources", Proceedings of the IEEE International Symposium on Assembly and Task Planning, pp.411-416, 1999.

[22] N. Sannomiya and H. Iima, "Application of genetic algorithm to scheduling problems in manufacturing processes", Proceedings of the IEEE Conference on Evolutionary Computation, pp.523-528, 1996.

[23] N. Jawahar, P. Aravindan, S. G. Ponnambalam, and A. A. Karthikeyan, "Genetic algorithm-based scheduler for setup-constrained FMC", Computers in Industry, Vol.35, pp.291-310, 1998.

[24] F. T. S. Chan, K. L. Mak, L. H. S. Luong, and X. G. Ming, "Machine-component grouping using genetic algorithms", Robotics and Computer-Integrated Manufacturing, Vol.14, pp.339-346, 1998.

[25] V. Caraffa, S. Ianes, T. P. Bagchi, and C. Sriskandarajah, "Minimizing makespan in a blocking flowshop using genetic algorithms", International Journal of Production Economics, Vol.70, pp.101-115, 2001.

[26] B. Gonzalez, M. Torres, and J. A. Moreno, "Hybrid genetic algorithm approach for the `no-wait' flowshop scheduling problem", IEE Conference Publication, pp.59-64, 1995.

[27] B. Gonzalez, M. Torres, and J. A. Moreno, "Hybrid genetic algorithm approach for the `no-wait' flowshop scheduling problem", IEE Conference Publication, pp.59-64, 1995.

[28] N. Jawahar, P. Aravindan, S. G. Ponnambalam, and A. A. Karthikeyan, "Genetic algorithm-based scheduler for setup-constrained FMC", Computers in Industry, Vol.35, pp.291-310, 1998.

[29] C. R. Standridge, "Teaching manufacturing systems simulation in a computer aided teaching studio", Winter Simulation Conference Proceedings, Vol.2, pp.1613-1618, 2001.

[30] J. P. Kenne and A. Gharbi, "A simulation optimization based control policy for failure prone one-machine, two-product manufacturing systems", Computers and Industrial Engineering, Vol.46, pp.285-292, 2004.

[31] S. Chen, L. Chen, and L. Lin, "Knowledge-based support for simulation analysis of manufacturing cells", Computers in Industry, Vol.44, pp.33-49, 2001.

[32] G. Zulch, S. Rottinger, and T. Vollstedt, "A simulation approach for planning and re-assigning of personnel in manufacturing", International Journal of Production Economics, Vol.90, pp.265-277, 2004.

[33] H. M. E. Korhonen, J. Heikkila, and J. M. Tornwall, "A simulation case study of production planning and control in printed wiring board manufacturing", Winter Simulation Conference Proceedings, Vol.2, pp.844-847, 2001.

[34] S. Kopacsi and G. L. Kovacs, "Knowledge based simulation system for flexible manufacturing", Proceedings - IEEE International Conference on Robotics and Automation, Vol.1, pp.858-863, 1993.

[35] C. McLean and S. Leong, "The expanding role of simulation in future manufacturing", Winter Simulation Conference Proceedings, Vol. 2, pp.1478-1486, 2001.

[36] T. Murata and H. Ishibuchi, "Performance evaluation of genetic algorithms for flowshop scheduling problems", Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, Vol.2, pp.812-817, 1994.

[37] N. G. Hall, H. Kamoun, and C. Sriskandarejah, "Scheduling in robotic cells: complexity and steady state analysis", European Journal of Operational Research, Vol.109, pp.43-65, 1998.

INTERNET-BASED TEACHING AND LEARNING FOR THE ROBOT CELL DESIGN

Qingjin Peng, Tao Luan
Department of Mechanical and Industrial Engineering
The University of Manitoba, Winnipeg, Manitoba, R3T 5V6, Canada
Email: Pengq@cc.umanitoba.ca

ABSTRACT

Industrial robots play an important role in advanced manufacturing systems. A major application of industrial robots is loading and unloading of machining parts in manufacturing cells. For a single robotic cell with n machines, the number of possible alternative sequences of robot moving is n factorial (n!). The process to decision the optimal design of a robot cell is often both daunting and time-consuming. In order to decide the shortest circle time in a robotic cell, a cost-effective tool, which can simulate the sequence of robot moving, is developed to assist the design and analysis of robot operations by the authors. The tool can simulate a single robot cell with two and three machines using a distributed sharing model. The Java programming language is used to create the program code. The Web browsers are used to provide a user-friendly interface to manipulate the simulate process. Very positive feedback has been received from users.

KEY WORDS

Internet-based teaching and learning, Robot cell, Computer aided tools, Simulation

1. INTRODUCTION

Computer Integrated Manufacturing and Automation is a core course of our Industrial Engineering Program. Production planning and control is a core course of our new Manufacturing Engineering Program. Mechanical engineering students take these courses as a technical elective. Flexible Manufacturing Systems (FMS) is an important section of these courses. It takes six hours to teach the sequence of robot moving in the FMS section, which includes the introduction of alternative robot moving, the analysis of the moving and the cycle time, and decision of the optimal alternatives.

The process of a product loading and unloading in a manufacturing cell or FMS system is carried out by robots. The robot performs a procedure of pickup, moving, load, unload, and drop parts in the cell. For a single robotic manufacturing cell with n machines, the number of possible alternative sequences of robot moving is n factorial (n!). The teaching and learning of this section is often both daunting and time-consuming. Many literatures provide different techniques to obtain the optimal cycle time and consequently the best sequences of robot moving, such as Ngentis discussed a no-wait robotic cell for an automated m-machine flow shop in which one robot is used to move the parts from a machine to the next [1]; Liu et al provided a dynamic sequencing of robot moves in a manufacturing cell [2]; Hall et al proposed a scheduling method in robotic cells for classification of two and three machine cells [3]; Aneja et al discussed the scheduling of parts and robot activities in a two machine robotic cell [4].

The method used to teach this material is based on the textbook [5], developed by Sethis et al. It characterizes single robot cell with two and three machines. There is not a suitable simulation tool which can be used to examine the theoretic methods and diverse solutions. The delivery of this section is only based on the lecture without related tutorial and lab to support the teaching and learning process. We proposed to develop a simulation tool to accommodate this demand.

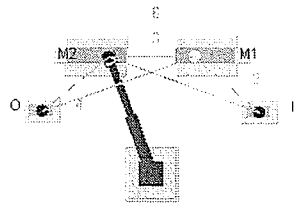
Simulation is a modeling and analysis tool widely used for the purpose of designing, planning and control of manufacturing systems [5]. One of the greatest advantages of a simulation model is that can compress or expand time. Manufacturing systems can be understood better with the help of simulation model. Once a model is developed, it provides an efficient tool for analyzing effects of various parameters and studying the implications of various alternatives.

Simulation modeling techniques with visual animation capabilities also provide an efficient means in learning, experimenting, and analyzing of a real-life complex system. Using simulation modeling, it is possible to create and test countless scenarios. The simulation has

been used extensively in engineering education. Such as, Standridge used simulation in his teaching with a mix of passive and active learning in his class [6]; A software tool developed for teaching/learning in a 8051 microcontroller shows the benefits of it as an alternative to traditional instruction tools [7]; A modular approach to the design and simulation of digital communication systems improves students' interest in the subject by addressing practical design issues from the material discussed in the class [8]. Therefore, simulation-based education methods provide valuable resources to boost interest in engineering education and enhance the ability to effectively teach and conceptually learn engineering subjects [9].

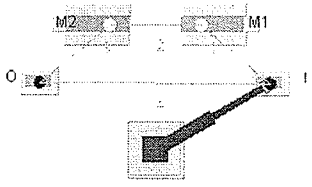
2. ROBOT MOVING SEQUENCE IN A MANUFACTURING CELL

2.1 Sequencing of Robot Moving in a Two-Machine Robotic Cell



The robot cycle time $T_1=6h+6v+a+b$

Figure 1. (a) Alternative 1



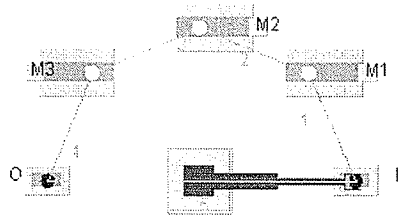
The robot cycle time:
$T_2 = 4h + 4v + max\{2h + 4v, a, b\}$

Figure 1. (b) Alternative 2

If we assume that a is the processing time of machine $M1$ and b is the processing time of machine $M2$; h is the time used by robot for each pick up, load, unload, and drop operation; v is the time of the robot moving between two adjacent locations; T is the cycle time of the robot operation. There are two alternatives of robot moving sequence in a two-machine robotic cell. The illustrations and cycle time are shown in Figures 1 (a) and (b).

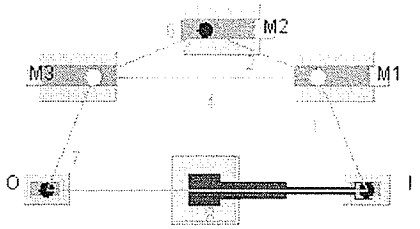
2.2 Sequencing of Robot Moving in a Three Machine Robotic Cell

Based on the analysis provided by Sethi et al, there are six alternatives in a three-machine robotic cell [10]. If we use the same parameters as in Section 2.1, and introduce c as the processing time of machine $M3$, the sequences and the cycle time of the robot moving can be indicated in Figures 2 (a) to (f).



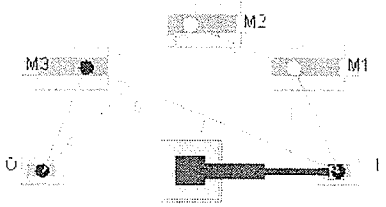
The robot cycle time: $T_1 = 8h + 8v + a + b + c$

Figure 2(a). Alternative 1



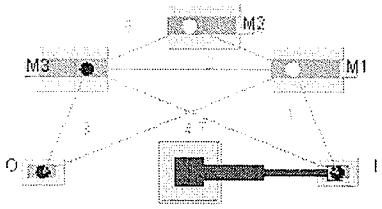
The robot cycle time:
$T_2 = j + max\{k, b, (k/2+a), (k/2+c), (a+b+c)/2\}$
Where: $j = 4h + 4v$ and $k = 8v + 4h$

Figure 2 (b). Alternative 2



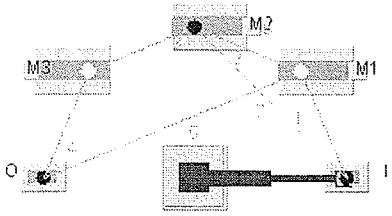
$T3 = j + max\{(k-2h+a), c, (a+b+k/2-2v)\}$

Figure 2 (c). Alternative 3



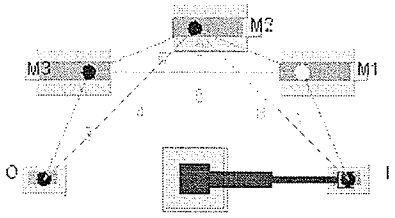
$T_4 = j + max\{(k+b), (k/2 +a +b -2v), (k/2 +b +c -2v)\}$

Figure 2 (d). Alternative 4



$T5 = j + max\{a, (k + c -2v), (k/2 +b +c -2v)\}$

Figure 2 (e). Alternative 5



$T_6 = j + max\{k, a, b, c)\}$

Figure 2 (f). Alternative 6

It can be found from Figures 1 and 2 that optimal sequence of a robot moving, which has shortest cycle time, depends on the value of a, b, c, h and v. The different products may require different machining time, which will result in the different optimal sequence. The process to search an optimal sequence only based on the formulae provided in Figures 1 and 2 is not easy to understand. It is also hard to evaluate the search result without a simulation tool. A simulation tool was developed based on these considerations. The developed tool provides a method for students to configure and test an FMS effectively.

3. THE SYSTEM DESIGN AND IMPLEMENTATION

Visual J++ language is used to write the simulation program in order that the simulation can be run on Web browsers. An 'Applet on HTML' is created and embedded into the Web browsers using the Applet class of Visual J++. The flow chart of the program running is shown in Figure 3.

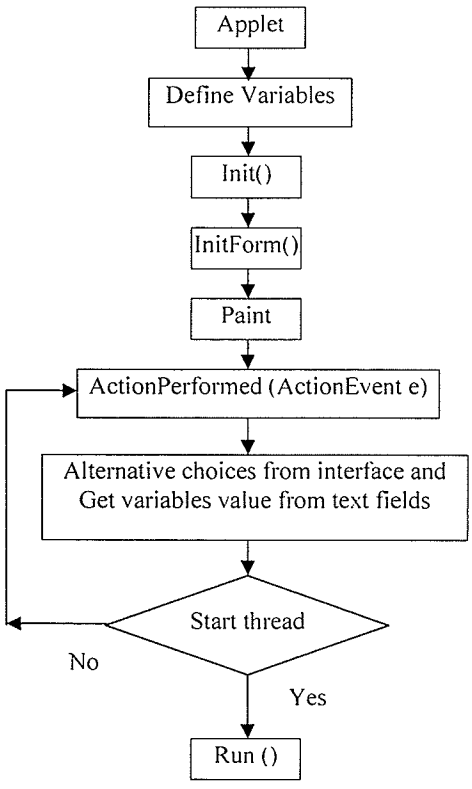


Figure 3. The flow chart of the program

The thread classes are used in the program, which involves using objects to implement a Runnable interface. The Runnable interface is activated using the run() method in order that the simulation can be paused or stopped at anytime during the program running.

The following procedure was used in the implementation of the simulation tool in order to meet the design objectives:

1). The demand requirement analysis: This was to collect related ideas and experience from faculties and students who had used software-based tool in the teaching and learning to find demand for the simulation tool.

2). The system design and function definition: This was to design the system structure, input and output, and interface used. The detailed development process was also considered.

3). Detailed system design: At this stage, the detailed programming flow chart was designed. The demand requirement was checked. Any initial improvement was made.

4). Software code development: This was to develop the software code.

5). Installation and test: This stage was for primary test of the software running, and evaluate the interactive process prior to use in the class.

6). Evaluation by students: This was to use the software tool developed in the course tutorial. The student report required contained not only technical issues, but the feedback of using this tool as well.

7). The system improvement: The improvement was made based on the student's feedback and problems in the application.

4. THE EVALUATION OF THE SIMULATION TOOL

The developed software tool of the sequence simulation of robot moving has been used in the teaching and learning. The tool can be accessed from any location through the Internet. The address of the simulation tool is:http://www.umanitoba.ca/faculties/engineering/mech_and_ind/prof/peng/RobotmoveSimulation/page1.htm. Its operation window is shown in Figure 4. In the teaching and learning applications in the last two years, the simulation tool had saved about three teaching hours in each course for new content development. It also increased the interest and understanding of students in the learning process. Students had an assignment associated with FMS study. A report was required from the students. The report included feedback and an evaluation of using this tool. The issue related the tool improvement was documented and used to further system improvement.

The feedback was collected from the classes of last year and this year. All students gave the positive comments. Some selected sentences from the feedback include:

- Very useful/ helpful/greatly increased the understanding of the various difference robot moving sequences;
- The textbook doesn't explain it as well as the simulation tool does;
- The software makes the sequence alternatives quite clear;
- Very easy to use.

The suggestion for the further improvement was also collected. Some of suggestions are listed as following.

- Add robot wait time display into the simulation;
- Allow users to choose the robot moving speed, and to create the new sequences;
- Allow time progress to change;
- Use 3D model.

5. DISCUSSION AND CONCLUSION

Many literatures provide different techniques to obtain the optimal cycle time and consequently the best sequences of robot moving. But none of them provide an easy procedure in a robot cell design. This paper discusses the development of an Internet-based tool in teaching and leaning to simulate the sequence of robot moving. It provides a cost-effective tool for students to set up a robot cell in a flexible manufacturing system.

The developed software has been used in the teaching and learning of courses: Computer Integrated Manufacturing and Automation, and production planning and control. It has increased interest and understanding of students in the learning process. The tool allows students to pursue a broad range of hypotheses and variable manipulation, and gain valuable practice with using the concepts, such as:

- Check the theoretical solution discussed in the class by the computer simulation process;
- Provide different hypotheses and find the solution by using the proposed software;
- Help to complete the assignment of robot moves; and
- Use it as an assistant tool to analyse the work process of a flexible manufacturing system.

Therefore, the developed web-based simulation tool has promoted deep learning process in manufacturing cell design. It allows users to test different hypotheses. It gives users opportunities to practise applications of their knowledge, and to enrich the theoretical approaches for a wider range of styles.

Further work will be the system improvement based on suggestions from the feedback. Some work has been done recently. Especially, Virtual Reality Modeling Language (VRML)-based 3D modeling simulation has been developed to enhance the capability of the simulation tool in visualization and interaction. Figure 5 shows its operation window.

6. ACKNOWLEDGMENT

This project is supported by the Teaching and Learning with Technology Grants Program at University of Manitoba.

REFERENCES

[1]. Ngentis, A., Scheduling no-wait robotic cells with two and three machines, European Journal of Operational Research 123 2 2000 Elsevier Science B.V. p 303-314 0377-2217.

[2]. Liu, S., and Lin, L., Dynamic sequencing of robot moves in a manufacturing cell, European Journal of Operational Research v 69 n 3 Sep 24 1993 Publ by Elsevier Science Publishers B.V. p 482-497 0377-2217.

[3]. Hall, G., Kamoun, H., and Sriskandarajah, C., Scheduling in robotic cells: classification, two and three machine cells , Operations Research v 45 n 3 May-Jun 1997 Informs p 421-439 0030-364X.

[4]. Aneja, Y.P., and Kamoun, H., Scheduling of parts and robot activities in a two machine robotic cell, Computers & Operations Research v 26 n 4 Apr 1999 Elsevier Sci Ltd p 297-312 0305-0548.

[5]. Singh, N., Systems Approach to Computer-integrated Design and Manufacturing, John Wiley & Sons Inc, 1996.

[6]. Standridge, C., Teaching simulation using case studies, 2000 Winter Simulation Conference Proceedings. (Cat. No.00CH37165). IEEE, Piscataway, NJ, USA; 2000; 2 vol.(xl+xxiv+2114) pp. p.1630-4, vol.2.

[7]. Del-Rio, A., and Andina, J., UV151: A simulation tool for teaching/learning the 8051 microcontroller, The 30th Annual Fronties in Education Conference, Building on A Century of Progress in Engineering Education. Conference Proceedings (IEEE Cat. No.00CH37135). Stripes Publishing, Champaign, IL, USA; 2000; 2 vol.623+596 pp. p.F4E/11-16 vol.2.

[8]. Soyjaudah, K., and Jahmeerbacus, M., A modular approach to the design and simulation of digital communication systems, International Journal of Electrical Engineering Education, vol.37, no.3; July 2000; p.226-32.

[9]. Iskander, MF., Advances in technology-based engineering education, IEEE Antennas and Propagation Soclaty International Symposium, Transmitting Waves of Progress to the Next Millennium. 2000 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (Cat. No.00CH37118). IEEE, Piscataway, NJ, USA; 2000; 4 vol. xi+2359 pp. p.1237 vol.3.

[10]. Sethi, S. P., Sriskandsrsjah, C., Sorger, G., Blazewicz, J., and Kubiak, W., Sequencing of parts and robot moves in a robotic cell, The International Journal of FMS, 4 (1992):331-358.

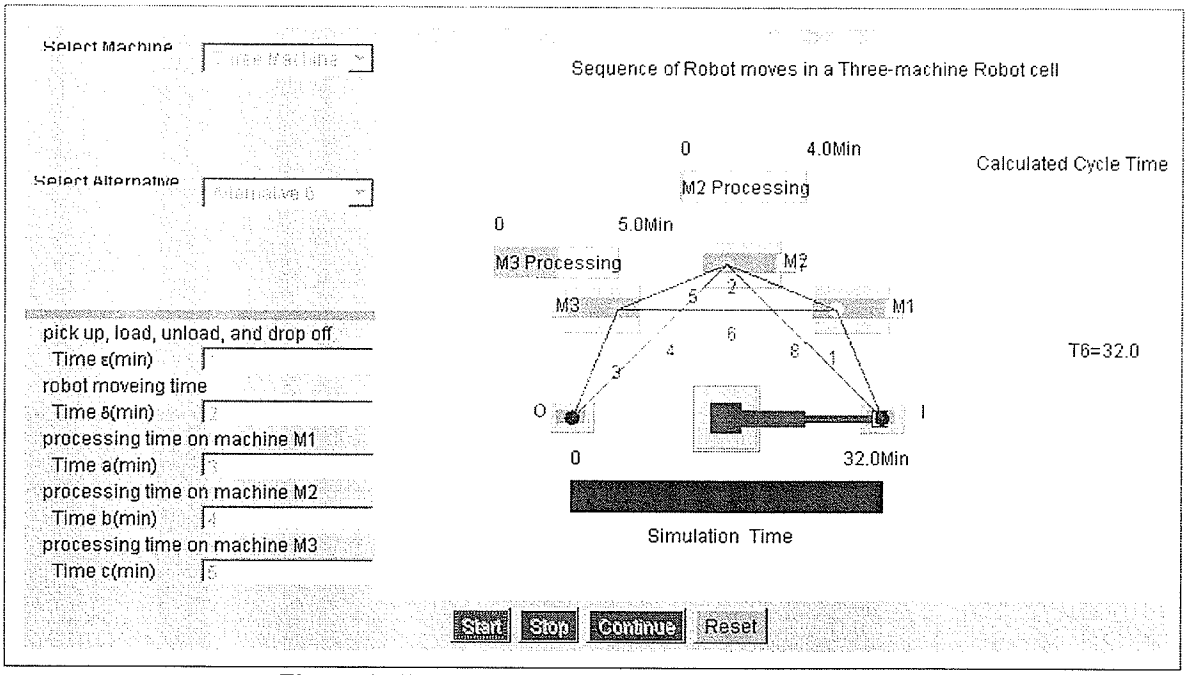


Figure 4. The operation window of the simulation tool

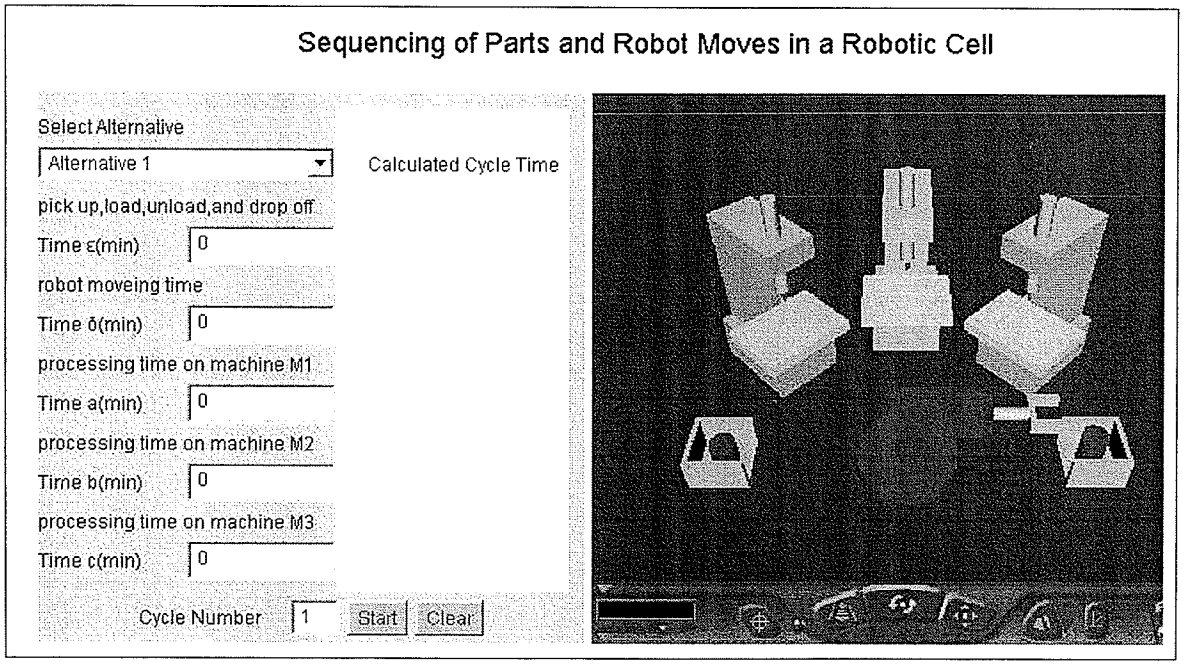


Figure 5. The 3D simulation window based on VRML

Genetic Algorithm-based Parts Scheduling in a Two-machine Robotic Cell

Tao Luan* and Qingjin Peng

Department of Mechanical and Industrial Engineering
The University of Manitoba
Winnipeg, Manitoba, R3T 5V6, Canada

ABSTRACT

This paper discusses different part types problems in a two-machine robotic cell. The different sequence of input parts will generate different cycle time. The research is to find the maximum parts throughout rate or the minimum cycle time of the robot operation. Some approaches related to the problem have been proposed. However, these methods are very complex on large-sized part problems or even medium-sized problems. A genetic algorithm is proposed to find solution of the problem that is characterized by a fixed path. The proposed algorithm employs the natural chromosome representation of genetic algorithm. A point cut crossover operator is developed for chromosome crossover operation. A mutation operator is designed to select two genes randomly in a chromosome and to exchange the genes positions. The optimal part sequence and cycle time are obtained through processing genetic evaluation and improving the solution performance. An example is presented.

1. INTRODUCTION

Industrial robots play an important role in computer integrated manufacturing systems. A major application of industrial robots is loading and unloading parts in robotic cells. For example, the robot is used to pick up a part from an input buffer, load and unload it on a lathe machine, then on a drill machine, finally discharge it in an output buffer.

Most of the existing research on robotic cells is for a single robotic cell which consists of an input device, several machines that perform different processing on each part, an output device, and a robot that transports the parts from one machine to another, or between machines. If the layout of machines, an input, and output device is arranged along a semi-circle, the robot is located in the center of the semi-circle and transports parts between machines and input/output devices, the robotic cell is called a robot-centred cell [3,5]. If the robot is on a rail and can move along the rail in the cell to load and unload parts on the machines, the robotic cell is called mobile-robot cell [3,5,6].

Many researchers focused on the problems of identical part types and different part types in the robot-centred cell, but the results can be used in mobile-robot cell [3]. Sethi et al [1] solved the problem of the robot move sequence in two machines and three machines for identical part types, parts schedule is limited on the fixed paths in two machines for different part types. Crama et al [2] developed the special case of the problem where the number of machines is arbitrary based on the concept of pyramidal permutation. A dynamic programming approach is used to solve the robot move sequence in a robotic cell with m machines for identical part types. Agentis [3,4] studied no-wait processing in the robotic cell with two and three machines. Logendran et al [5] solved the different parts problem in two machines robotic cell. Aneja et al [7] considered the different parts problem in two machines robotic cell, but each part can be produced by one of two different possible robot move cycles. Gilmore and Gomory's algorithm is adopted to solve the different parts problems [1,5,7]. It is for a special solvable class of the traveling salesman problem. This algorithm is complex to solve large-size parts schedule problems or even medium-size parts schedule problems.

In this paper, we focus on a single robotic cell with two machines and there are no intermediate buffers between machines as shown in Figure 1. The operation procedure of the two-machine robot-centered cell is assumed as follows:

* Corresponding author: Tel.: (204) 474-8572; Fax: (204) 275-7507; E-mail: umluant@cc.umanitoba.ca

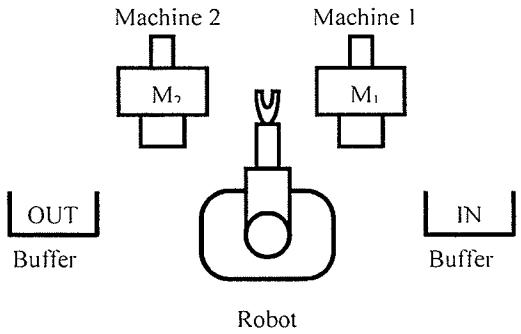


Figure 1: Two-machine robot-centered cell

(1) Each part will go through all elements of the cell in the order IN, M1, M2, OUT.
(2) Each machine processes only one part at a time.
(3) The set-up time is included in the processing time.
(4) The time of each picking-up, loading, unloading, and drop off is same.
(5) The robot travel time between any pair of adjacent location is same.

The objective of the research is to find the optimal solution of the sequence of robot moves in the two-machine robotic cell for different parts based on the minimum part set (MPS) [1,5].

Genetic algorithms (GA) are used to solve the problem because the problem belongs to combinatorial optimization problems. Genetic algorithms have the feature of flexible, global and stochastic searching mechanism. They are expected to improve the efficiency for schedule of different parts. This paper is organized as follows: the schedule of different part types in a robotic cell with two machines is introduced in Section 2. Section 3 designs the implementation of genetic algorithms. An example is illustrated in Section 4 followed by conclusions in Section 5.

2. SCHEDULE OF DIFFERENT PART TYPES IN A ROBOTIC CELL WITH TWO -MACHINE

There are two alternative sequences of robot moves in a two-machine robot cell. They are shown in Figure 2 and Figure 3. For alternative 1, the sequence of robot moves is in the order of IN-M_1-M_2-OUT-IN. For alternative 2, it is in the order of IN-M_1-M_2-OUT-M_1-M_2-IN.

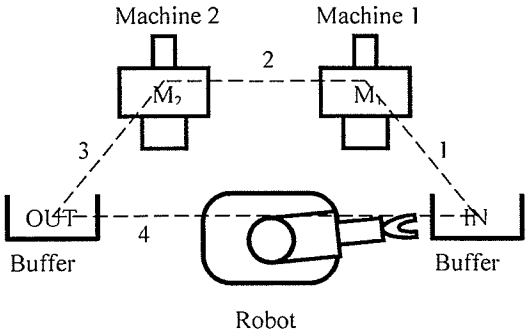


Figure 2: Alternative 1

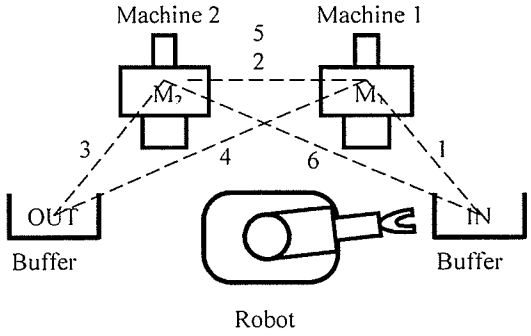


Figure 3: Alternative 2

2.1 ALTERNATIVE 1

For multi-part cycles with alternative 1 of robot moves, the cycle time can be obtained as follows [1]:

$$T_{1\sigma(i)\sigma(i+1)} = \varepsilon+\delta+\varepsilon+a_{\sigma(i)}+\varepsilon+\delta+\varepsilon+b_{\sigma(i)}+\varepsilon+\delta+\varepsilon+3\delta$$

$$= 6\varepsilon + 6\delta + a_{\sigma(i)} + b_{\sigma(i)}$$

The cycle time:

$$T_1 = \sum_{i=1}^{n} T_{1\sigma(i)\sigma(i+1)} \tag{1}$$

where σ is a given permutation of n parts. ε is the time of picking-up, loading, unloading, or drop off. δ is the robot travel time between any pair of adjacent location. $a_{\sigma(i)}$ and $b_{\sigma(i)}$ are part i processing time at M_1 and M_2, respectively. $T_{1\sigma(i)\sigma(i+1)}$ is the time to produce one part i. n is the number of parts.

2.2 ALTERNATIVE 2

For this case, the cycle time can be obtained as follows [1]:

$$\begin{aligned} T_{2\sigma(i)\sigma(i+1)} &= \varepsilon + \delta + \varepsilon + 2\delta + w_{1\sigma(i+1)} + \varepsilon + \delta + \varepsilon + 2\delta + \varepsilon + \delta + \varepsilon + \delta + w_{2\sigma(i+1)} \\ &= 8\delta + 6\varepsilon + w_{2\sigma(i+1)} + w_{1\sigma(i+1)} \end{aligned}$$

where $w_{1\sigma(i+1)}$ and $w_{2\sigma(i+1)}$ are the robot waiting times at M_1 and M_2, respectively.

$$w_{2\sigma(i+1)} = \max\{0, b_{\sigma(i+1)} - 4\sigma - 2\varepsilon\}$$

$$w_{1\sigma(i+1)} = \max\{0, \alpha_{\sigma(i+1)} - w_{2\sigma(i)} - 4\delta - 2\varepsilon\}$$

where $w_{2\sigma(i)} = \max\{0, b_{\sigma(i)} - 4\delta - 2\varepsilon\}$

Therefore, $T_{2\sigma(i)\sigma(i+1)} = 8\delta + 6\varepsilon + \max\{0, b_{\sigma(i+1)} - 4\delta - 2\varepsilon\} + \max\{0, a_{\sigma(i+1)} - 4\delta - 2\varepsilon - \max\{0, b_{\sigma(i)} - 4\delta - 2\varepsilon\}\}$

The cycle time:

$$T_2 = \sum_{i=1}^{n} T_{2\sigma(i)\sigma(i+1)} \tag{2}$$

3. THE IMPLEMENTATION OF GENETIC ALGORITHMS

Based on the formulations (1) and (2), a genetic algorithm is used to find optimal cycle time and parts schedule for a given fixed sequence of robot moves. The genetic algorithm consists of the encoding scheme, evaluation function, roulette wheel selection, genetic operators, and evolution strategy. They are illustrated as follows:

3.1 ENCODING SCHEME

Encoding scheme is used to encode a solution of the problem into a chromosome. It is a key for genetic algorithm to solve real world problems. Because the schedule of parts in a robotic cell belongs to a permutation schedule problem, the permutation of parts is designed to represent chromosomes for the problem. For example, if there are eight parts in a chromosome and the representation of the chromosome is:

1	2	3	4	5	6	7	8

The schedule of parts will be part1, part5, part3, part4, part2, part6, part7, part8.

3.2 INITIAL POPULATION

Initial population is a group of individuals or chromosomes. After initial population size and representation of chromosome are defined, the initial population is generated randomly. Each individual of the initial population presents a schedule of parts.

3.3 Evaluation Function

Evaluation function is used to measure the fitness value of a given individual. For the problem of parts schedule in a two machines robotic cell, the inverse of cycle time of alternative 1 and alternative 2 is used to determine the fitness for each individual, respectively. The evaluation function can be designed as follows:

For alternative 1: $eval(individual) = 1/T_1$

For alternative 2: $eval(individual) = 1/T_2$

3.4 Roulette Wheel Selection

A roulette wheel approach is used to select individuals from population to perform genetic operators. The approach is stochastic. Each individual occupies a 'slice of a pie'. A random number is generated. If the number is between the cumulative probabilities of q_k and q_{k+1}, the individual k is selected. By this way, the best individual that has the most probability will be selected to perform genetic operator. The roulette wheel approach includes following steps [9]:

(1) Calculate each fitness value for each individual

$$eval(individual_k),\quad k = 1,2,..., pop_size$$

(2) Calculate the total fitness for the population

$$F = \sum_{k=1}^{pop_size} eval(individual_k)$$

(3) Calculate selection probability p_k for each individual

$$p_k = \frac{eval(individual_k)}{F},\quad k = 1,2,..., pop_size$$

(4) Calculate cumulative probability q_k for each individual

$$q_k = \sum_{j=1}^{k} p_j,\quad k = 1.2,..., pop_size$$

where k is the number of individuals in the population.

The strongest individual with the most probability is selected by spinning the roulette wheel.

3.5 Genetic Operators

The genetic operators are used to create children at each generation. In this research, two genetic operators are introduced to deal with sequence of robot moves for different part types in a two-machine robotic cell. They are one point cut crossover and arbitrary two parts change mutation [8]. They are shown in Figures 4 and 5.

3.5.1 One Point Cut Crossover

Two parents are selected to generate one child. One cut point is randomly generated. Parent 1 is divided into two parts by the cut point. The child inherits the left part of the parent 1. The other part of the child comes from parent 2, takes over the parts that are not included the first part of the child and keeps the order of parent 2.

3.5.2 Mutation-Arbitrary Two Parts Change

Mutation is an operator of genetic algorithms. Arbitrary two parts change mutation is designed to perform the operation of mutation. Two parts in the parent of Figure 5 are selected randomly and exchanged their position to generate one child.

3.6 Evolution Strategy

After genetic operators, total population consists of old population and new created children. Based on evolution strategy, the new population for the next generation is selected from the old population and children. As the old

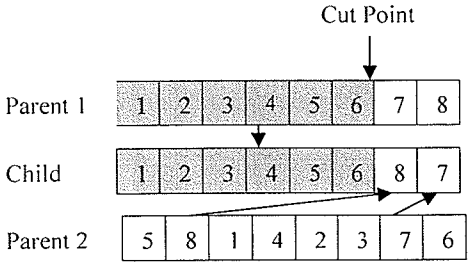


Figure 4: One point cut crossover

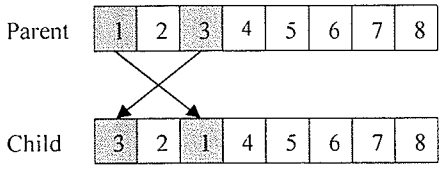


Figure 5: Arbitrary two parts change mutation

population and children compete for survival and the best or better individuals are picked out, the new population is better than the old population. The best or better population can be obtained from one the generation to another generation. Optimal solution is obtained after several generations.

3.7 OVERALL ALGORITHM

Genetic algorithm is implemented by combining one point cut crossover and arbitrary parts change mutation. It is described as follows:

Input: parts number, the robot travel time, the pick up, load, unload, and drop off time, each part processing time on the machine 1and machine 2, population size, the probability of crossover and mutation.

Output: cycle time and schedule of parts.

Begin:

Step 1: Randomly generate the initial population.

Step 2: If the termination criterion is not satisfied, go to Step 3. Otherwise output the result.

Step 3: Generate a random number.

Step 4: If the number satisfies the probability of crossover, two parents are selected from the initial population by the roulette wheel approach in order to generate one child. Otherwise one parent is selected from initial population by the roulette wheel approach in order to generate one child.

Step 5: If the number of children is not satisfied, go back to Step 3. Otherwise go to Step 6.

Step 6: Using fitness function to select the best population size individuals from the initial population and the children.

Step 7: Go back to Step 2.

4. AN ILLUSTRATIVE EXAMPLE

The genetic algorithm has been completed by Visual J++ language. An example is designed to illustrate the parts schedule problem with ten parts in the two-machine robotic cell. The robot travel time δ is 0.3 minutes. The pick up, load, unload, and drop off time ε is 0.2 minutes. The minimal production ratio (MPR) [5] is $r = (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8) = (1,2,1,1,1,2,1,1)$. The processing times on machine 1 and machine 2 are given in Table 1.

Table 1: The part number, type and processing time

Part number	1	2	3	4	5	6	7	8	9	10
Part Type	P1	P2	P2	P3	P4	P5	P6	P6	P7	P8
M_1 (min)	5.0	3.0	3.0	4.0	4.5	4.5	6.0	6.0	2.0	4.0
M_2 (min)	3.0	4.5	4.5	3.0	5.0	3.0	4.0	4.0	3.5	6.0

For the example, it is assumed that the set-up time is short for each part. The switch between different parts type is not affected by the set-up time. The question is focused on parts sequence. The objective is to find the optimal parts schedule solution for the alternative 1 and alternative 2 by using the genetic algorithm. By comparing

alternative 1 and 2, the optimal cycle time can be obtained. If the set-up time is longer, the research can be used to find different parts type sequence. The optimal parts type sequence can be obtained.

For alternative 1, four sets of operation rates are used. They are [crossover, mutation]=[0.2, 0.8], [0.4, 0.6], [0.6, 0.4], and [0.8, 0.2], respectively. Based on the four groups of operation rates and ten population size, the graphic of T_1 cycle time is shown in Figure 6.

For alternative 2, four sets of operation rates are used. They are [crossover, mutation]=[0.2, 0.8], [0.4, 0.6], [0.6, 0.4], and [0.8, 0.2], respectively. Based on the four groups of operation rates and ten population size, the graphic of T_2 cycle time is shown in Figure 7.

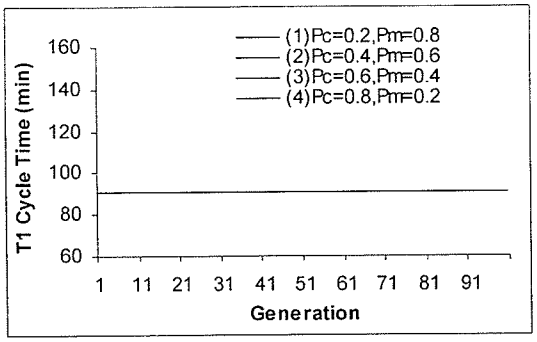


Figure 6: The converging process of alternative 1

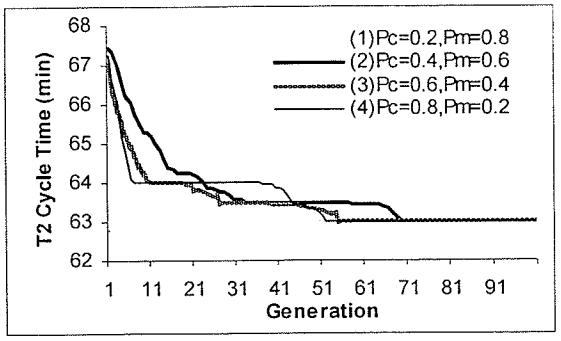


Figure 7: The converging process of alternative 2

From the Figure 6, the cycle time is not changed for the four sets of operation rates. Actually, the cycle time is the same for any parts sequence. The result also can be conformed from the formulation of (1).

From the Figure 7, the cycle time of each line is changed. This means that the cycle time is affected by the each parts schedule. For different sets of operation rates, the cycle time is converged on the same value after several generations. The four sets of operation rates are used to prevent local optimum situation. The solution is global optimum because the cycle time of different sets of operation rates is converged on the same value.

Comparing the optimal cycle time of alternative 1(90.75 minutes) with the optimal cycle time of alternative 2 (63.0 minutes), as the cycle time of alternative 1 is more than the cycle time of alternative 2, the optimal cycle time is the cycle time of alternative 2. The crossover is 0.5 and mutation is 0.5 for testing of alternative 2. The graphic is shown in Figure 8. From the Figure 8, the cycle time is converged on 63.0 minutes. The parts schedule is obtained as shown in Figure 9.

In the left side of the line in Figure 9, each row represents parts schedule. In the right side of the line, each row represents the cycle time of each parts schedule. The optimal cycle time is 63.0 minutes. However, the parts schedule is not only one sequence. There are several solutions for parts schedule. Each sequence satisfies the optimal cycle time.

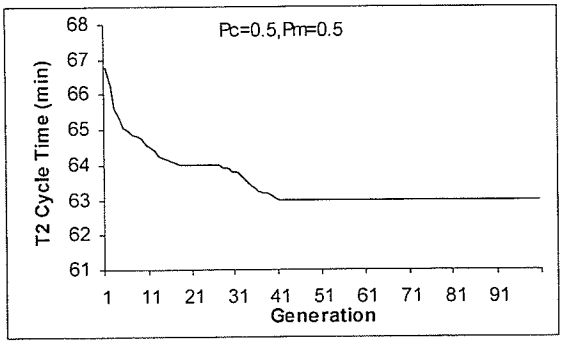


Figure 8: T2 cycle time with Pc=0.5 and Pm=0.5

1	3	5	8	6	9	10	7	4	2	63.0
1	9	10	7	6	3	5	8	4	2	63.0
1	9	10	7	6	3	5	8	4	2	63.0
1	3	5	8	6	9	10	7	4	2	63.0
1	9	10	7	6	3	5	8	4	2	63.0
1	9	10	7	6	3	5	8	4	2	63.0
1	3	5	8	6	9	10	7	4	2	63.0
1	9	10	7	6	3	5	8	4	2	63.0
1	3	5	8	6	9	10	7	4	2	63.0
1	9	10	7	6	3	5	8	4	2	63.0

Figure 9: Optimal parts sequence and cycle time of alternative 2

5. CONCLUSIONS

Genetic algorithms have shown an effective and practical solution for optimization problems of sequence of robot moves for different part types in a two-machine robotic cell. The example illustrates that genetic algorithms can rapidly find optimal solution and parts schedule. However, convergence speed should be controlled to prevent local optimum. Different scheme of mutation and crossover will affect the converging speed of the algorithm. A set of operation rates of crossover and mutation can be used to find the global optimal solution. Since genetic algorithms are stochastic approaches, the performance of the approach is affected by operation rates. Based on the four groups of mutation and crossover operation rates, genetic algorithm is run several times until the same solution is obtained in the example. Further research is to find optimal cycle time and parts schedule in a three-machine robotic cell.

REFERENCES

[1] *S. P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz and W. Kubiak, "Sequencing of parts and robot moves in a robotic cell", The International Journal of Flexible Manufacturing System, Vol.4, pp.331-358, 1992.*

[2] *Y. Crama and J. Van De Klundert, "Cyclic scheduling of identical parts in a robotic cell", Operations Research, Vol.45, No.6, pp.952-965, 1997.*

[3] *A. Agnetis, "Scheduling no-wait robotic cells with two and three machines", European Journal of Operational Research, Vol.123, No.2, pp.303-314, 2000.*

[4] *A. Agnetis and D. Pacciarelli, "Part sequencing in three-machine no-wait robotic cells", Operations Research Letters, Vol.27, No.4, pp.185-192, 2000.*

[5] *R. Logendran and C. Sriskandarajah, "Sequencing of robot activities and parts in two-machine robotic cells", International Journal of Production Research, Vol.34, No.12, pp.3447-3463, 1996.*

[6] *Nicholas G. Hall, H. Kamoun and C. Sriskandarajah, "Scheduling in robotic cells: classification, two and three machine cell", Operations Research, Vol.45, No.3, pp.421-439, 1997.*

[7] *Y.P. Aneja and H. Kamoun, "Scheduling of parts and robot activities in a two machine robotic cell", Computers & Operations Research, Vol.26, No.4, pp.297-312, 1999.*

[8] *T. Murata and H. Ishibuchi, "Performance evaluation of genetic algorithms for flowshop scheduling problems", Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, Vol.2, pp.812-817,1994.*

[9] *Mitsuo Gen and Runwei Cheng, "Genetic Alogorithms and Engineering Design", John Wiley & Sons, Inc, 1996.*