DYNAMIC ROBOT CONTROL AND PART LOADING IN A FLEXIBLE MANUFACTURING CELL

BY

LINDA LEILA HATHOUT

A Thesis Submitted to the Faculty of Graduate Studies
In Partial Fulfilment of the Requirements for a Degree of

Master of Science

Department of Mechanical and Industrial Engineering
University of Manitoba
Winnipeg Manitoba

0-612-56128-3

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

Dynamic Robot Control and Part Loading in a Flexible Manufacturing Cell

BY

Linda Leila Hathout

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

Abstract

This project focuses on the dynamic scheduling and control aspects of a flexible manufacturing cell (FMC) having a single robot to move parts to be processed. One issue rarely addressed in the literature on FMC optimization is the relationship between the optimal intra-cell, or robot move sequence and the type, quantity and the order of parts in the batch scheduled for production. If all parts have the same due date and the order of servicing the parts can be changed, then the sequence of robot moves should not be fixed but, rather, the moves should be determined by the properties of the batch and the properties of the FMC. The dynamic scheduling of robot moves and part sequencing for this scenario has not been investigated previously.

The objective of this project is to develop a computer program to control and optimize the throughput of parts by considering the overall system's properties in a working FMC. Software is created to control the robot and to dynamically make decisions. In particular, real time robot control decisions are made during production by continuously polling the FMC's status to obtain information from sensors. The software then bases decisions to move the robot based on the system's current status. The software's computational time is minimal, so that processing is in real time. The software also allows for simulation and experimentation with the control variables of the FMC. This latter aspect allows the user the ability to adjust the decision support systems for faster batch throughput times. The program facilitates greater flexibility in the utilization of a FMC while still taking full advantage of its capacity.

Acknowledgments

I would like to thank Dr. S. Balakrishnan and Dr. N. Popplewell for their excellent input and time. I also greatly appreciate the time and patience of Terri Willard, Patrick Oliver and Jon Crocker. Thank you for being my sounding board and for your technical expertise.

Dedication

This work is dedicated to my mom, Angele Hathout. She believed in me, encouraged me and supported me throughout my engineering studies. Her confidence in me gave me no choice but to be confident in myself.

Table of Contents

Tables

Note for the following tables the letter in front of the table number indicates the appendix where the table is located.

Figures

Note for the following tables the letter in front of the table number indicates the appendix where the table is located.

Appendices

1. Introduction to the Flexible Manufacturing Environment

The importance of flexibility in manufacturing is becoming more and more critical in the current highly competitive marketplace. Customer demands are becoming increasingly individualistic, leading to a greater variation in the products, which in turn could mean a variation in the process. A competitive edge requires manufacturers to address individual customer demands and reduce the manufacturing cycle time while cutting production costs. As computing speeds increase and the cost of robots decreases, optimization is becoming not only possible, but essential. A flexible manufacturing system (FMS) is an ideal place to take full advantage of flexible automation for catering to small-lot, multi-product batch production.

In the proceeding section a brief overview of flexible manufacturing will be given to better ground developments of this problem. Following which, the thesis's objective will be stated.

1.1. The Flexible Manufacturing Environment

A FMS is defined as consisting "of machines where production operations are performed, linked by a material handling system, all under central computer control" (Buzacott & Shanthikumar, 1980). The material handling system provides the flexibility to allow the parts to follow a variety of routings. In a FMS, production and assembly processes may exist at varied states of automation; what makes the manufacturing process flexible is the ability to rapidly change the system to accommodate multiple product objectives.

There are four stages of decision making for FMS, as identified by Stecke (1985): namely, the design, planning, scheduling and control stages. At the planning phase, considerations include part type selection, process planning, machine grouping, determining production ratios, batching of parts, scheduling of batches, allocation of pallets and fixtures, and allocation of operations and tools among machines. A flexible manufacturing cell (FMC) is one aspect of the manufacturing process found in some FMSs. FMCs are affected by all the previously mentioned factors. They are also affected by the scheduling and control of the parts in the system. Ideally, FMCs are the most automated and dynamic processes of the system.

A manufacturing cell is the most flexible of computer integrated manufacturing systems and it is best suited for low-volume, high-variety production. A manufacturing cell is a cluster of machines or processes that are located in close proximity and dedicated to the manufacturing of a family of parts. The parts are similar in their processing requirements, such as operations, tolerances, and machine tool capacities. In many cells, robots are the primary material handling systems used for part loading and unloading. Robots are seen as the archetypes of flexible automation. The scheduling problem at the cell level is characterised by a short lead time, dynamically changing environments, the versatility of the machines, and the need for real time decision making.

Cell level scheduling needs to be:

1. responsive to changing environments;
2. flexible in accommodating different scheduling needs; and
3. intelligent for scheduling decision support.

To further break down the problem at the FMC level, there are five major subgroups of factors regulating the functionality of a cell. They are; FMS production factors, material input factors, material handling factors, workstation equipment factors and layout factors. Please note the factors bulleted below with an arrow are considered in this work.

FMS production factors

- ➢ diversity of parts manufactured by the cell;
- ➢ the sequence of processing a job/part (engineering design dependent);
- ➢ flexibility of the cell verses efficiency of the cell; and
- ♦ manufacturing priorities; throughput, job deadlines, or other objective such as optimizing the use of machinery with a given hourly cost.

Material input factors

- ➢ different types of material input, how many different core parts the cell can handle;
- ➢ the number of different part types arriving in batches;
- ♦ part queue priority;
- ♦ infinite queue of material input versus batches of parts;
- ♦ different part types that arrive randomly and require job identification at the cell; and
- ♦ material input requirements at subassembly machines.

Material handling factors

- robot, automated guided vehicles(AGV), and humans;
- type of gripper used, and the number of different grippers necessary;
- single gripper/holder versus dual or multiple grippers /holders;
- speed of loading/unloading and movement;
- smart material handling systems, e.g. force sensors; and
- safe operating speed that minimizes the pick-up and placement errors.

Work station equipment

- the number of stations/machines in the cell;
- number of part holders, fixtures or buffers at each station;
- fixturing issues, including the number of different part shapes that can be handled simultaneously by a machine;
- position of buffer, in front or behind the machine;
- whether concurrent loading/unloading and machining is possible;
- machine maintenance times and frequency and machine breakdown frequency;
- degree of interruption for online or offline maintenance;
- whether an operation can be done on more than one machine, machine flexibility; and
- the number of operations or programs which can be performed at a machine; can different programs be called on a part to part basis or a batch to batch basis?

Layout

- travel time versus operation time;
- location of material supply and exit with respect to machine stations; and
- safe access to machines for online maintenance while the cell is working.

The greatest challenge in implementing a FMC is the timing and interdependency of all the components of the cell as well as the integration of a wide variety of often conflicting objectives. To run an individual cell, there is usually a control computer that can be used to simultaneously control many processes or pieces of equipment. This computer communicates directly with the process or equipment controllers for online control. The computer takes, as input, pertinent information about the process parameters and uses this input according to the programmed control logic in order to make decisions about how to set various outputs. In the most sophisticated FMCs, the online computer has complete control of the process, forming a closed control loop, which requires no human intervention except for maybe dealing with unforeseen developments.

Because of the complexity of FMSs, simulations are often used to ensure that a FMC is operating optimally to meet the objectives of the FMS. Simulations are beneficial in optimizing a FMC's utilization for the following reasons. They are:

- simulations do not need explicit mathematical functions to relate variables;
- they can represent complex systems in order for management to get a "feel" for the real system;
- simulations can compress or expand time for test purposes;
- they are useful in observing phenomenon and developing management predictions.

With this environment in mind, this thesis focuses on the sequencing of robot moves and sequencing of parts in a FMS. The study examines an approach to dynamic part and robot move sequencing for a FMC by using a dynamic hierarchical decision making strategy. Not all the variables mentioned previously are addressed fully. However, this work will create a generic FMC environment which can look at questions of buffer size, number of machines, machining times, robot move times, part processing requirements, part loading orders, and the impact of non-sequential processing. Future studies can incorporate variables not considered. The scenario examined is expanded upon next.

1.2. Objective

The objective of this study is to develop a dynamic hierarchical decision making structure for part sequencing and intelligent intra-cell robot movement in order to increase throughput, without human supervision, for varying, multi-product batches. Because there are so many system variables, it is important to refine the problem's scope yet still provide a system that is sufficiently generic to apply to various FMC situations.

Here, the flexible manufacturing system's objective is to reduce lead time, that is the time between the customer placing an order and the completion of that order. In the scenario considered, individual orders consist of requests for multiple products, which are processed during a period of sales. All products should be delivered within the same time span. The processing required at the FMC level involves several machining stations that can perform their operations irrespective of whether a previous operation has been performed. Hence, the machine processes that each part needs to go through can be done non-sequentially. Needless to say, machines must be structured to meet this type of flexibility. For example, different part shapes must be accommodated using flexible fixturing.

Parts enter the cell as a batch. The batch can consist of parts which:

1. are identical even though they may require different processing in order to become different products; or

2. are different due to different processes at other cells, yet such parts share a need for at least one of the machining processes available at the particular cell considered.

In either case, the sequencing of parts for processing, as well as the necessary robot moves within the cell, have to be addressed. The only difference is that, in the first case, the parts do not need to be distinguished upon entry into the system. In the second case, the parts need to be identified and sorted - perhaps by having more than one input buffer.

The machines in the cell all perform unique operations (e.g. milling, drilling, deburring etc.). It is possible that they perform part-specific operations in which each part type needing a specific machine can call up one of several programs that have been pre-programmed into the machine controller. However, for simplicity, each machine performs one unique operation on all parts that require this operation. This means that the machining process at each machine is the same throughout each batch, but it is distinct from the other machining processes. Each machine has room for one part to be machined at a given time. There is a buffer behind each machine that automatically reloads the machine once the part currently at the machine has been removed. A sensor is located at each of these machines and provides feedback to a control computer to indicate that the machining process is underway or has been completed. Breakdowns of machines are assumed to be infrequent. They are not considered to be an issue in this work.

An experimental, computer integrated cell was developed to test different control strategies. In this cell, an articulated robot with five degrees of freedom performs the task of material handling. The robot has one gripper. Its "pick and place" operations are performed at relatively high speeds, each around 0.5 seconds. Parts considered are not exceptionally heavy and the buffers and fixtures are designed to correct minor inaccuracies in a part's position. Parts are placed initially in the input buffer. When a part has undergone all the required operations, it is moved from the FMC to an output buffer. Compared to the time taken to pick up and place the parts, the movement of the robot is sufficiently fast that most of its move times between stations are roughly equal (i.e. about 5.0 seconds).

Given this scenario, the problem is one of dynamically sequencing both parts and robot moves without operator intervention. The purpose is to develop a better understanding of how the batch manufacturing time can be reduced by exploring generally occurring batch production situations. This is achieved bearing in mind the following objectives;

- to develop a functional, computer controlled FMC;
- to examine whether concurrently processing various parts is beneficial;
- to develop a robot movement decision hierarchy;
- to examine the effects of load versus unload part prioritizing;
- to examine the effects of various buffer sizes;
- to test the effect of sequential and non-sequential part processing; and
- to determine the influence of part sequencing on the batch throughput.

The objectives of the last four items is really to identify general trends and not to define hard set rules.

Having stated the objectives and provided a background of the environment from which this work arises, the next section summarizes the body of research that deals with aspects of the optimization of this type of FMC. The focus is on the tools and methodologies proposed previously. Subsequent to the literature review, a description is given of the FMC used and an overview of the software developed. The last section provides details of the experiments and analyses conducted.

2. Literature Review

The literature search examines research dealing with aspects of optimization in FMCs that reduce the parts' lead time. The specific questions involve sequencing a robot's movement and part loading in a FMC. The problem of sequencing parts and robot moves are addressed in many different fields including computer science, applied mathematics, management and manufacturing/industrial engineering. The applied mathematical approach, in general, is to develop analytically the relationship between different variables that can be solved using branch and bound algorithms or "travelling salesman problem" solving tools, with the objective of finding the optimal cyclic robot movement in a FMC. Most management work in this area has focussed on decision hierarchy questions at the FMS level and cost reduction issues related to reducing product lateness. The focus of management research is on developing general heuristics for quick implementation in flow shop environments. The computer science interest is generally in the modelling of such problems. It has lead to the development of FMC models using petri net modelling frameworks and applying object oriented approaches to FMC modelling. Engineering efforts at improving FMCs has resulted in continuous improvements in automation, quality, and maintainability as well as increasing the efficiency of these cells. Production engineering work in this area is limited to documentation of case specific, process improvements achieved through alterations in current production flow sequences or different part prioritization hierarchies. The next four sections examines the progress made by using these different approaches.

2.1. Cyclic Robot Movement in a FMC

Robot movement and part scheduling questions are known to be NP-hard making them computationally intensive and challenging to solve. Sethi et al (1992) examine a FMC problem and determine the optimal cyclic robot move sequence for manufacturing a batch of parts in the shortest time. However, their solution is limited to a flow line manufacturing system where parts have to visit all the three stations considered and the solutions are based on cyclic scheduling. Sethi et al expand the cyclic scheduling to look at multi-part type problems involving two machines. They develope empirical formulas for the cycle times in order to solve the problem of sequencing different part types for a given sequence of robot moves for two machines. Sethi et al state that their problem-solving technique becomes cumbersome when the number of machines is more than two, and when multiple part types are processed. Kamoun et al (1993) examine an optimal cyclic pattern for one part type and three machines. Chen et al (1997) study a multi-part type production problem on four machines by using a branch and bound technique. Their technique is to first find an optimal robot cycle, and then choose a part sequence that produces the lowest cycle time. Some success is achieved when adapting their heuristics and algorithms to three machines. However, they conclude that further investigation is required to adapt the same solution to larger size problems.

The previous work concerns the analysis of optimal scheduling for deterministic cycles of one or a few parts through a FMC. The solutions are limited, in terms of both the number of machines or the complexity of the parts handled, and the order of the parts that the resulting NP-hard equations can be solved. For example, in Chen et al (1997), the assumption is that all the parts need to visit all the stations. This assumption is not realistic in a truly flexible systems. Another disadvantage of using a fixed robot program arises when there is a minor system change or a machine breakdown. The robot controller will be unable, then, to negotiate the new environment and a system shutdown will result. From this review, the need becomes clear for developing a real-time scheduling system for a dynamic environment.

2.2. Flow Shop Approach to FMC Scheduling

A more dynamic part sequencing solution than that developed so far, requires that robot movements not be predetermined completely, rather, they should be structured to be responsive to changing system needs. Niemi and Davies (1989) develop a simulation of such a dynamic FMC optimization control system for two machines. They also test different movement control logic. However, no generalizable results are found. Such a case is also presented by Yang-Byung (1990). In this study, the optimal robot's service movement is examined in a robot-centred FMC. Here, five unique part types are processed in a FMC where each part has a unique set of process requirements in a specified order and requires a specific time on each machine. Several different robot service movement decision rules are tested. The best results are produced when the robot first services the part with the shortest remaining process time. In this case, the parts arrive randomly at the cell and the make-span (the time to produce one part) can be optimized.

King (1993) looks at the case of a two-machine cell, with buffers at each machine, for processing a batch of parts needing different processing times on each machine. The objective is to determine the optimal sequence of robot moves to minimize the make-span of the batch. King found that a branch and bound technique becomes ineffective as the number of parts increases past ten. The approach taken was to treat the problem as an open flow shop. This type of open flow shop problems deal with non-pre-emptive shop scheduling that, addresses part move routing. It usually has the objective of minimizing the makespan of a part. Moreover, a part's route is not given in advance although a predetermined processing time is know at each machine. Routes are determined by queues in front of machines and by the remaining process requirements. Rebaine and Strusevich (1999) consider the question of two-machine, open shop scheduling with parts having different transportation times. They develop a linear time algorithm that can find the optimal schedule in such a flow shop scenario.

The research reviewed so far does not handle well the scenario under consideration (i.e. batches with parts having multiple, and varying machining requirements). Most documented objectives are to achieve a reduced makespan for a single part or to reduce the cost of late parts. The heuristic rules examined in these papers can be generally applied when developing a robot move decision hierarchy within an FMC. However, their applicability to a dynamic cell environment is not easy and it has not been researched fully.

2.3. Modelling FMC Scheduling

There has been much work in modelling FMCs by using advanced simulation and modelling tools. The application of a Timed Place Petri-Net (TPPN) or a Coloured Petri Net (CPN) has proven most useful. A Petri net model consists of places (graphically represented as circles) and transitions (graphically represented as bars) connected via a set of directed arcs. Places may contain tokens (represented by dots inside a circle) which move through the network (i.e. from place to place) according to certain rules. Cheng et al (1994) used a TPPN for solving resource allocation questions in a FMS. They look at a job shop scenario using automated guided vehicles (AGV) and apply a heuristic search method to determine the near optimal schedule of part processing. More related to the question at hand, Yalcin and Boucher (1999) use CPN to solve a FMC problem with alternative machining and alternative part sequencing. Both these papers prove that Petri nets can be used as an effective modelling tool.

Another approach presented to FMC simulation is using object oriented (OO) modelling to create open-ended simulation software for flexible manufacturing. Like Petri net modelling, the object oriented approach focuses on object definition and actions occurring during state transitions. Lin et al (1994) demonstrate how OO can be used in such a manner.

In these modelling studies, the focus is on the tools themselves with very little concern for a tool's experimental implementation or experiments using the tools to gain insight into better robot move rules. The assumption in these papers of an infinite incoming buffer also leads to the objective of optimizing the makespan of a part rather than a batch of parts.

The framework developed by Petri net modelling, adapted to address FMC questions, serves as a useful example of a type of information architecture for any modelling software. The reviewed papers dealing with Petri nets serve as an example of how a net can be structured to handle time and place data related to objects moving through a model. These ideas prove to be valuable in developing simulation software. This is equally true for observations made about how to implement object oriented coding for flexible, manufacturing related simulations.

2.4. Sequencing and Loading

Thus far, the focus has been on sequencing part movements using a robot-controlled, decision-making strategy, given that the parts are in the FMC. Yet, if parts come in batches, it is possible to select the order that the parts are produced. While many management articles deal with this question in relation to due dates and minimizing late costs, they do not address the situation where all parts have the same due date so that the optimization objective changes. In cases where the part loading sequence can be changed, most flexible production plans solve the loading problem first and then separately select a fixed cyclical robot move schedule based on the loading results (Moreno & Ding, 1993). As demonstrated by Moreno and Ding, concurrently selecting and sequencing jobs in an FMS proves far more effective. While their method is not transferable to the FMC, the idea should be.

In most of the literature surveyed, sequential processing is generally assumed. However, Kumar and Li (1994) look at the optimal assembly time for a printed circuit board (PCB) assembly where the sequence of operations is not critical. Their paper attempts to solve the problem of the robots movement as a "travelling salesman" problem. This scenario relates closely to the one presented here. However, the authors present an optimal solution for the manufacturing of just one type of PCB.

It was confirmed from the literature that concurrent loading and robot move decision-making is preferable in a dynamic FMC environment. In a specific PCB case, the implementation of non-sequential processing takes fuller advantage of the available machine capacity and results in a significant reduced throughput time.

2.5. Summary of Literature Review

The techniques employed to develop models for the design and operation of a FMS at various decision levels are: computer simulations, deterministic combinatorial scheduling techniques, queuing networks, petri nets and hierarchical control. These techniques can be classified further into FMS optimization through combinatorial methods or optimization through heuristic rules (Chen 1990). Chen proposes a compromise, which produces optimal cycles, based on an evaluation of profiles of the parts. This solution is acceptable provided there is a limited degree of variation of batches, part profiles, and the FMC's composition. There is also a trade-off, within such an optimization solution, between finding an optimal schedule that will reduce the processing time, and the cost of waiting while the control computer searches for the optimal sequence.

The approach used here in the development of the control program and simulation incorporates ideas from the structure proposed by prior research using Petri net modelling and object oriented programming. The methodology for implementing the robot movement decision hierarchy also partially stems from these approaches. The hierarchy of decisions attempts to concurrently select parts for loading into the system and to sequence robot moves based on the current status of the overall system. Based on the review, the effect of variables like buffers, as well as the number of parts contained in them, loading part orders, and non-sequential processing merit further investigation.

The software and FMC layout design developed in this thesis incorporate several ideas generated as a result of the literature review. The software and hardware is designed to answer issues related to optimizing the batch size, number of parts and to identify sequencing aspects that have not been thoroughly explored previously. What follows is a brief description of the FMC used to implement online testing of the control program and an overview is given of the program's structure.

3. Organization and Layout of FMC System

The flexible manufacturing system designed for this project consists of a flexible manufacturing cell and a control computer that runs specially designed software. A detailed description of the cell is given next, followed by an explanation of how the software functions. To emphasise how this software works to produce a batch of parts, a detailed step-by-step description is given of state changes in the cell. These details highlight the robot move decision structure, implemented within the software, and give comparisons between different heuristics.

3.1. Flexible Manufacturing Cell

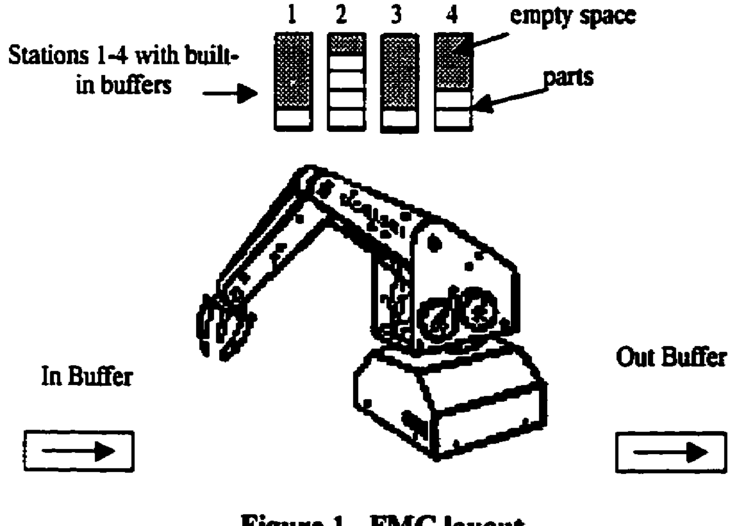


Figure 1. FMC layout

A general description of the FMC set-up was given in the Introduction. (See also the schematic shown in Appendix A.) This section provides more details of the FMC located in the Computer Integrated Manufacturing and Automation Laboratory at the

University of Manitoba. The cell consists of an "Asea" articulated robot with five degrees of freedom that is used for material handling. The gripper is a suction cup located at the end of the robot's arm. There are four processing or machining stations, each with a gravity feed buffer which can hold up to five parts. (See Figure 1.) A machine, and the buffer behind it, is occasionally called a station when the location of both is being referenced. The processing or machining times in the cell are simulated through timers in a programmable logic controller (PLC) which are triggered through limit switches located at the bottom of each of the four buffers. The logic developed for the programmable logic control can be found in Appendix B. There is one input buffer for parts entering the cell and a drop-off location for the finished parts. The robot controller has 14 unique, pre-set programs that are controlled by a master program. This master program responds to commands from the input/output board of the personal computer. An outline of this program can be found in Appendix C.

The IBM compatible computer has an I/O board that reads the status of the timers in the PLC and the status of the switches at the processing stations. The latter indicate whether each station is occupied. The I/O board also monitors an input bit that is set high when the robot is performing a task and low when the robot is available. Based on this information, the software determines what the robot's next move should be and sets the corresponding output bit on the I/O board to high. The master program for the robot controller reads the status of the input bit, which controls the use of one of fourteen subprograms. The subprograms correspond to different tasks that the robot is required to perform.

3.2. FMC Controller and Simulation Software

A single program written in C++ controls the FMC both in an on-line, real time environment as well as in an off-line simulation. The program is written to test the effect of different variables on the total throughput. Once the appropriate variables have been entered into the program, the simulation is started. The program makes decisions as to which task to perform next based on input criteria, the decision hierarchy, FMC variables and the existing feedback from the FMC. Further details are given in the following sections.

Before providing details on how the program logic works, it is important to understand how the program works from an operational perspective. The screenshot shown in Figure 2 presents the program's main window when a simulation is running. The next set of screenshots (Figures 3 and 4) show the program windows that allow the user to set FMC variables and select the type of test to run. The second screen (Figure 3) allows the user to input the part processing requirements and the quantity of each part to be made in the batch. The third screen (Figure 4) allows the user to define the movement times between each station (labelled 1-5). In addition, the times from the input to each station, as well as those from the individual stations to the output can be defined. In the experimental flexible manufacturing cell used, the movement times between each location are invariably 5 +/- 1 seconds.

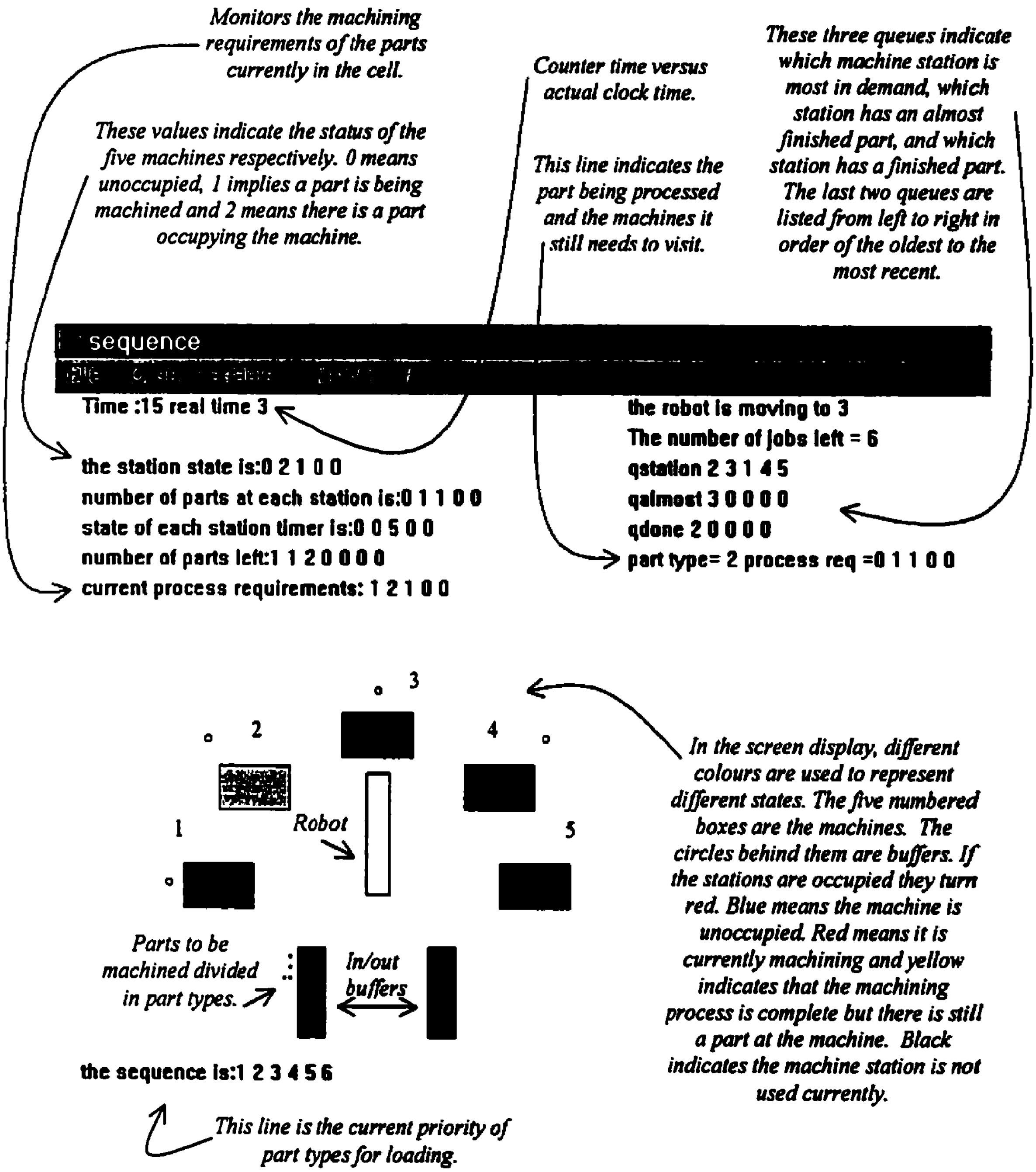


Figure 2. Main screen

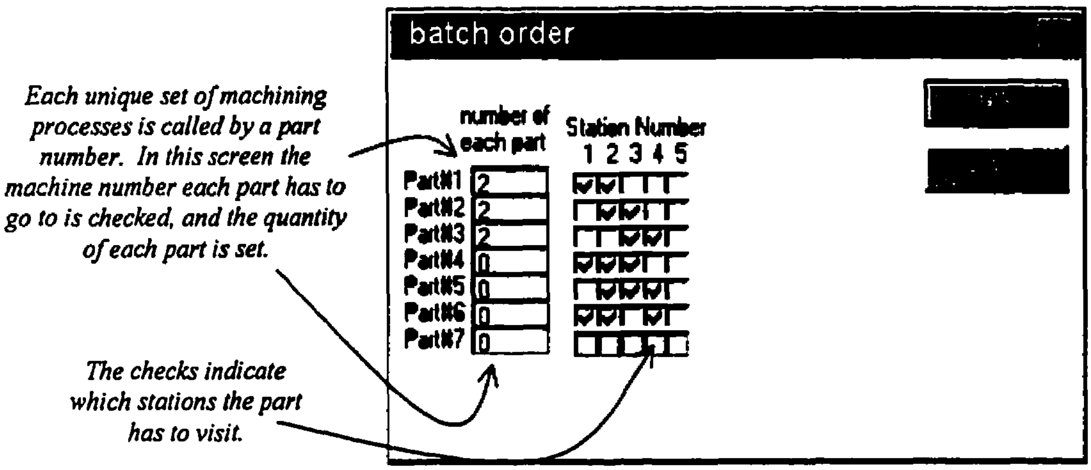


Figure 3. Batch order screen

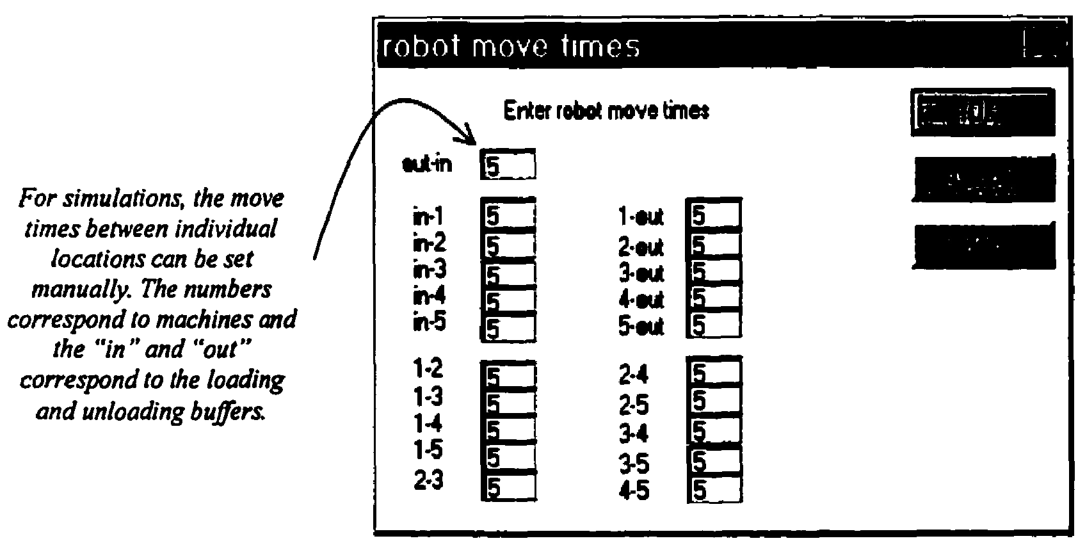


Figure 4. Input of robot move times screen

The next input screen (Figure 5) allows the user to set the number of buffers at each station and the processing time required at each station. In the experimental FMC, the station processing times are simulated through programmable logic controller timers which allow each processing time to be varied. In a field FMS, all these operational variables could be set automatically via a local area network that oversees all operations.

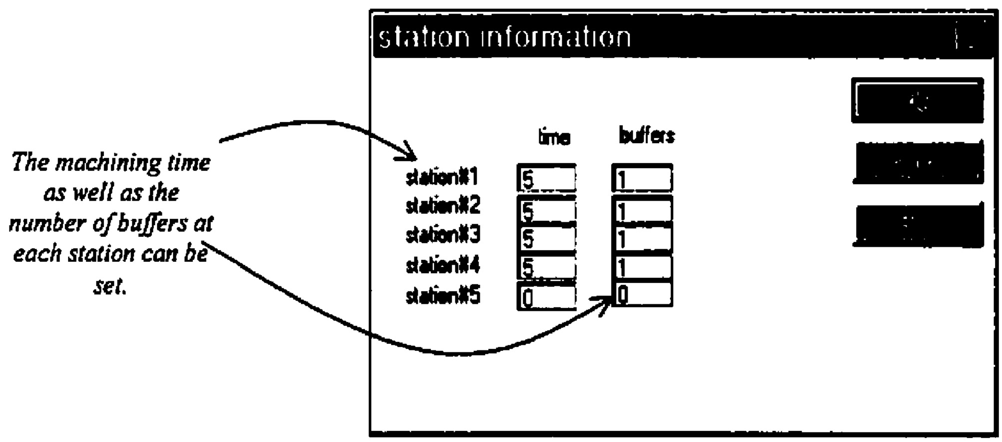


Figure 5. Station information screen

The next two screens (Figures 6 and 7) provide the operator with control over the simulation variables. The sequence screen (Figure 6) allows the operator to test a specific order of parts to be loaded into the cell. Here, it is also possible to choose between processing each part in a sequential or a non-sequential manner. The heuristic screen (Figure 7) allows for the selection between two robot movement priorities: loading versus unloading.

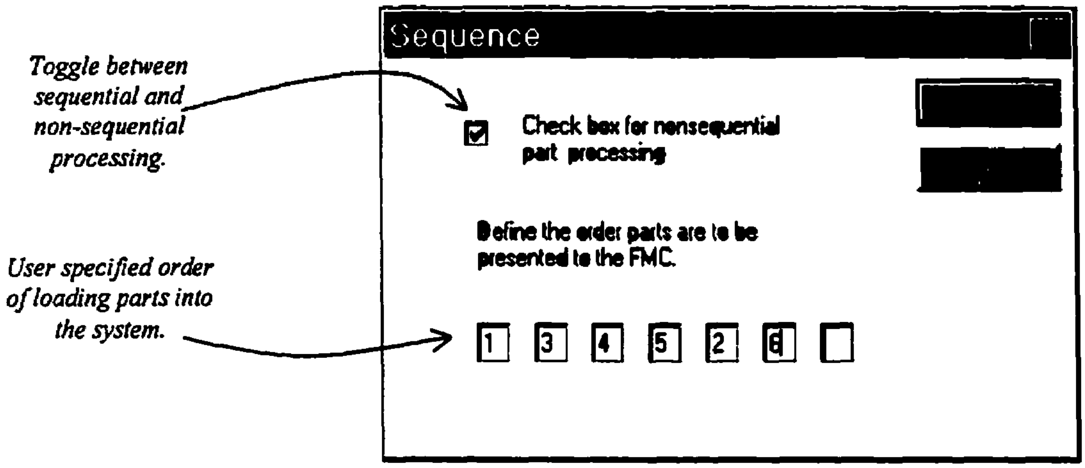


Figure 6. Part sequencing information screen

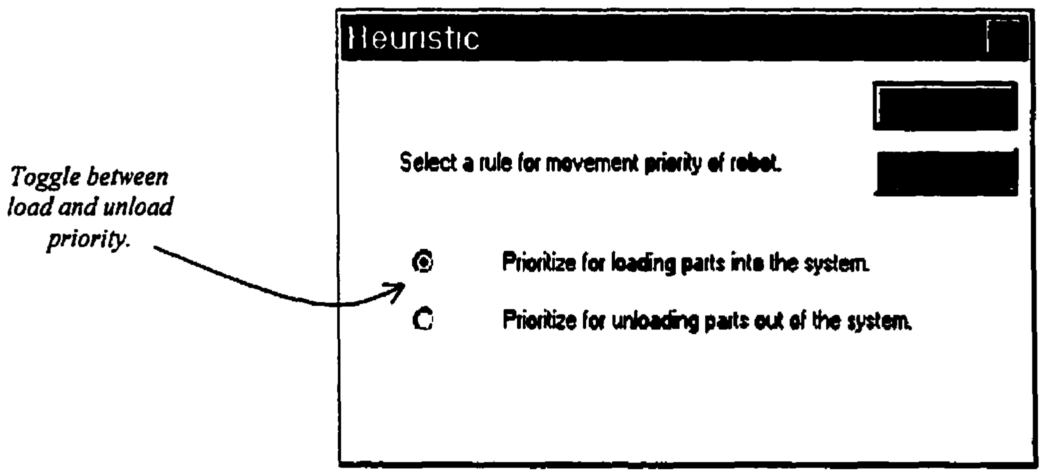


Figure 7. Heuristic selection screen

The run screen (Figure 8) allows the operator to choose the type of test to be run. The choices are to run a batch of parts through the FMC, to run a simulation quickly on the computer, to run a simulation on the computer in real time, or to run a simulation by stepping through each individual robot move by clicking the mouse button. There are also two other options needed to perform multiple tests. The first is an option to test all the possible sequences that a batch containing parts with different machining requirements could have. The second option allows the operator to run a series of tests in which consecutive tests have machining times all incremented by five seconds.

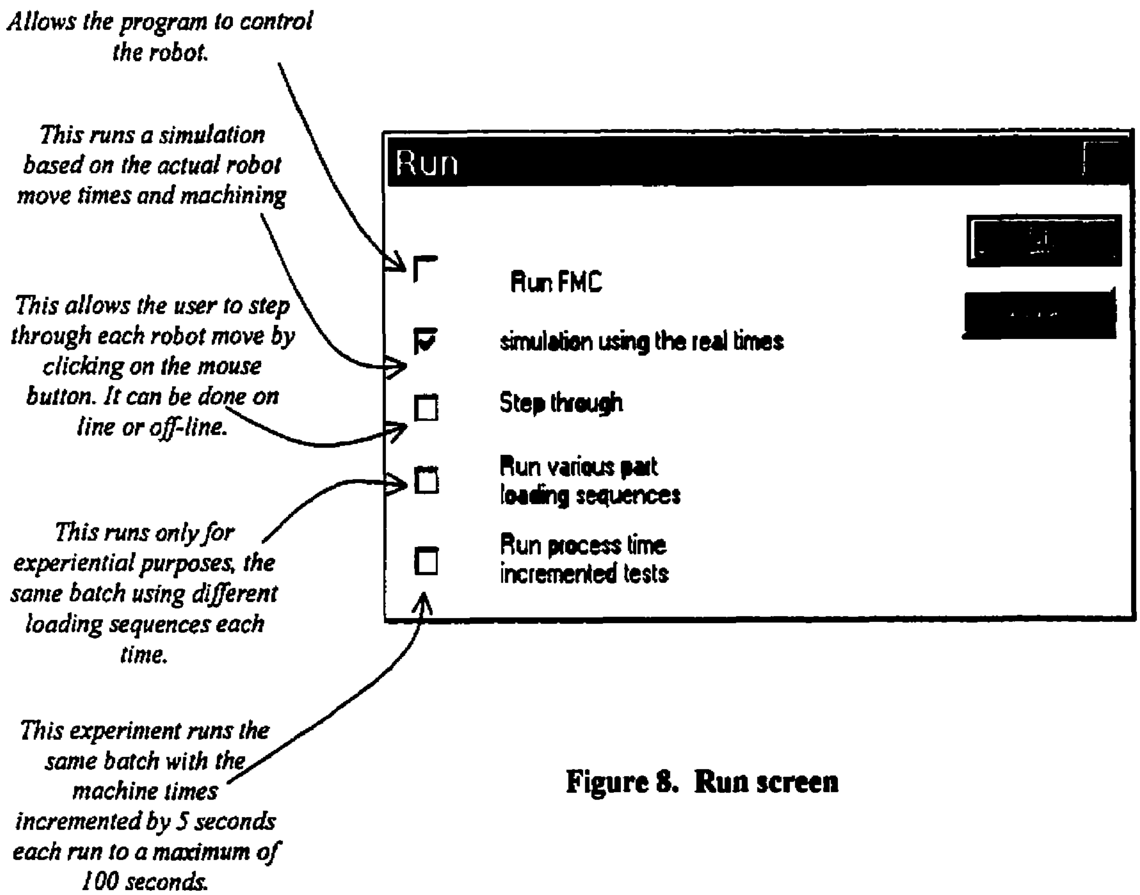


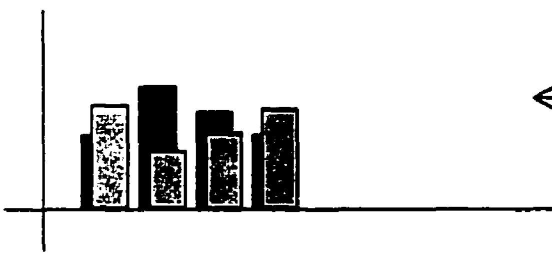
Figure 8. Run screen

At the end of each program, statistics are shown (Figure 9) that indicate each station's utilization, its occupied and free times. There are also statistics on the (arithmetic) average makespan of each part. The user is also provided with the option, at the end of each test, to save data in a user-defined file. An example of the data's output format is given in Appendix G. When multiple tests are conducted, data is filed automatically in a pre-specified file. All variables related to the given test are also included in this file.

The batch took 1010 minutes to complete.

the average residence time of part type 1, is 730/5 =146.0 seconds
the average residence time of part type 2, is 795/5 =159.0 seconds
the average residence time of part type 3, is 975/5 =195.0 seconds
the average residence time of part type 4, is 1000/5 =200.0 seconds ← *Statistics on the time each part spends in the cell.*
the average residence time of part type 5, is 945/5 =189.0 seconds
the average residence time of part type 6, is 1330/5 =266.0 seconds

The percent of total process time to total time and idle time to total time at each station was.
station#1 time used = 37.12%, time idle = 52.97%, time free = 9.90%
station#2 time used = 61.88%, time idle = 29.20%, time free = 8.91% ← *Statistics on machine usage.*
station#3 time used = 49.50%, time idle = 38.61%, time free = 11.88%
station#4 time used = 37.12%, time idle = 50.99%, time free = 11.88%
station#5 time used = 0.0%, time idle = 0.0%, time free = 100.0%
station utilization graph.



← *Histogram of machine use. The dark bars indicates machine utilization and light bars indicates the idle time.*

Figure 9. Statistics screen

Now a brief description will be given on how this program works. Once the simulation has started, the program enters a condition based program loop, which continues until all specified parts have been made. There are two types of cycles that the user may choose - the first type prioritises loading parts into the system, the second prioritises unloading. Appendix D gives a schematic of the simulation cycle.

One of four actions is taken during each cycle: load, shift part from one station to the next, unload, or move the robot and wait at the next station needing unloading or shift a part to its next operation. The order in which these choices are presented depends on whether the unload or load priority is selected. The unload priority first looks at the option of unloading a part. If this is not possible, then it considers shifting a part from one station to the next. If shifting is not an option, then it may load a part. If all else is impossible the final choice is to move the robot to a station that has almost finished machining and waiting. The load priority first looks at the option of loading a part, then shifting a part from one station to the next, then unloading, and lastly the moving and waiting command is considered.

During each cycle, the program's clock is incremented by the total time it takes for the robot to move. When the move and wait action is called, the clock is incremented by the time it takes the robot to move plus the time remaining for processing the part at the station to which the robot moves. As the main clock is incremented, the timers (representing the appropriate machining times) corresponding to the stations processing parts are decremented. There are also two queues, the 'done station' queue and the 'almost done station' or 'working station' queue. These queues keep track of which part is done first and which parts are to be done next. In the case of a station finishing processing when the clock is incremented, the station number gets added to the 'done station queue', and that station number is removed from the working station queue. For example, if the 'working station queue' indicates that machine 3 is done machining first, followed by stations 1 and 4, the working queue is 314. If machine 2 is holding a finished part, the 'done' queue is 2. If the timer is incremented by an amount which is equal or more than the time required to finish machining the part at machine 3, then the done queue is 23 and the working queue becomes 14.

Each part moving through the system is entered in a matrix. (See Figure 10.) An individual part has an identification number and nine variables of information defined when the part enters the system. The first five variables are Boolean variables that identify the stations that a particular part needs to visit. (Note that these variables could also be turned into integer variables and used to store unique machining times for each station.) As the part visits each station for processing, the integer value representing the status changes, binary numbers 0 and 1 are used to represent the state. The sixth variable monitors how many stations that the robot still has to visit by reading the binary number created by the first five Boolean variables. For example, if the first five variables describing a part are [0,1,1,0,1], then the sixth variable is 2+4+16=22. If the third operation is complete, the sixth variable is 18. The seventh variable indicates the part number, which would be used if the part needs to be unloaded to a part-specific buffer. The eighth variable gives the current station at which the part is located. The ninth variable indicates whether the part is in transition (i.e. held by the robot), in a buffer, being worked on, waiting for unloading or is out of the system. Each of these states has a number affiliated to it. There is also a buffer matrix that keeps track of the part queues at each station. Figure 11 provides a graphical representation of these matrices.

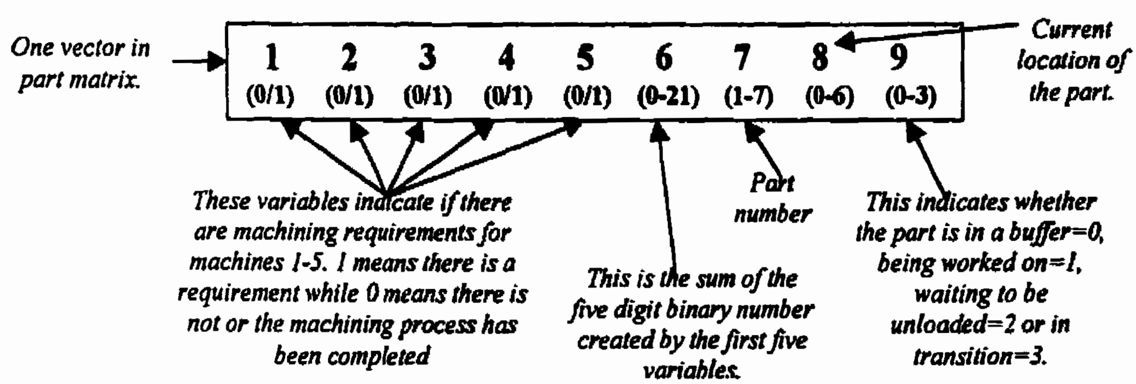


Figure 10. Job matrix

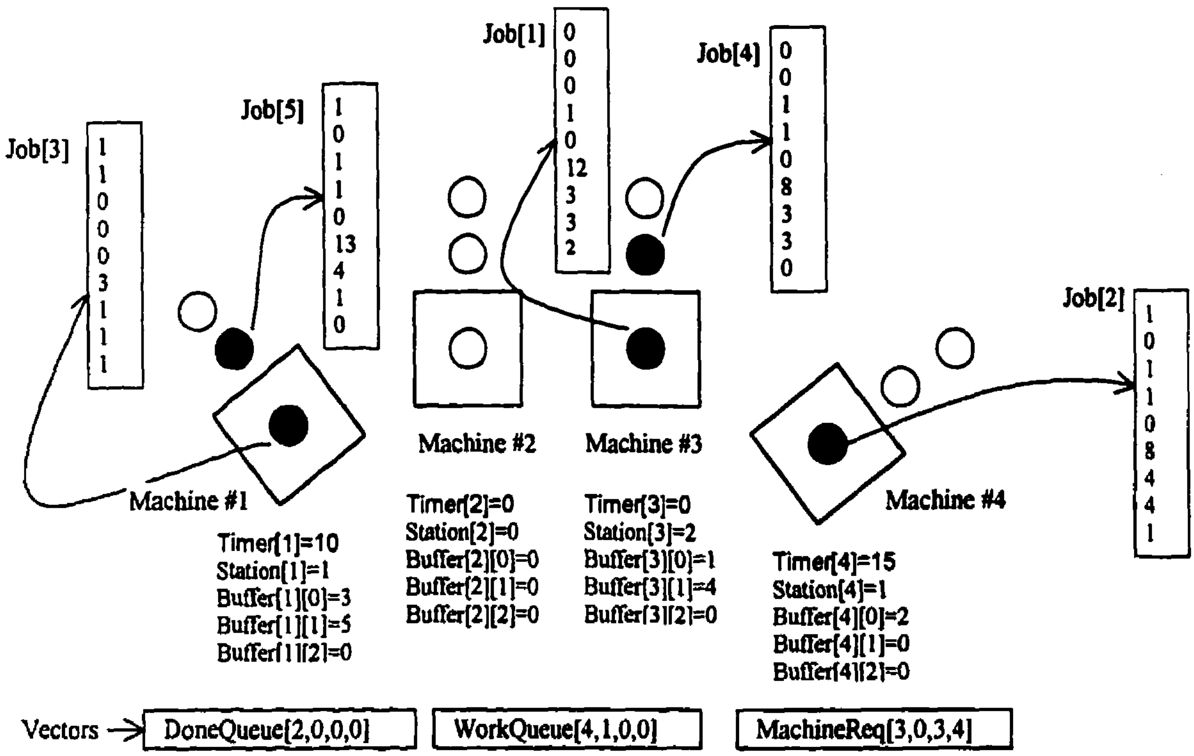


Figure 11. Matrix example

Of the four operations that can be performed in each cycle, the load operation is the most complicated. The load operation triggers a subroutine that determines whether there is any part that can enter the system. The primary requirements for loading a part into the cell is the availability of space to load the part onto a required machine, or into a limited capacity buffer. The program has been developed to accommodate two types of part processing, sequential and non-sequential processing. The loading rules for both these processes are the same. However, when parts are loaded sequentially, the option of which part to choose is more restricted. The following paragraph explains the hierarchy of the part selection priorities.

In the sequential case, a part can be loaded only if the first machine in the series of machines that the part needs to go to is free. Given that there may be more than one part that is a suitable candidate for loading, there is an order of priority for selecting a part. The first priority is given to the first part in the order of feasible parts which needs the currently least occupied station (e.g. a station having the fewest waiting parts). If there is more than one station with the fewest parts (e.g. two stations that are occupied by one part while the other stations have two or more), priority goes to the station that is most in demand by the parts in the current batch. If no clear priority is found, then the same search is performed for stations that have one more occupancy.

The process of part selection for the non-sequential case goes through the same order of priorities as in the previously defined sequential process. However, in this case, a part is a candidate providing one of its machining requirements is met. There are two additional criteria in the case of non-sequential processing. The part to be loaded, after having met all the other requirements, must go through a conflict check. The conflict check is a subroutine that ensures that, if a part is loaded into the system, it will not cause a blockage to the further processing of parts. For example, consider a situation when two parts of the same type need to visit two machines, and both machines have no buffers. If the parts are loaded consecutively on machines 1 and 2, there is no way for either part to proceed to the other station. This scenario is defined as a conflict situation, which needs to be avoided or resolved.

The other unique feature of non-sequential loading is that the cell can be loaded to capacity. This feature means that all the free buffer space in the cell can be loaded in such a way that no process is in conflict. This is not desirable because it tends to reduce the number of options available for the movement of parts and it usually leaves only one free buffer space for manoeuvring. Consequently, the total batch time is increased greatly. This situation does not arise in sequential processing. However, capping current machining requirements of parts in the system by making it a part loading condition, solves the problem of insufficient manoeuvring room. The limit is set based on the number of buffers at each station plus two. Hence, the number of parts, requiring a particular machine, permitted to enter into the cell, is fixed at the number of buffers at that machining station plus two. For example: if there are two buffers at each of three stations, then the current requirements of the parts in the cell cannot be more than four requests for processing on machine 1, four requests on machine 2, and four requests on machine 3.

The choice of setting the cap at the number of buffers plus two was determined based on numerical experimentation performed by using an exhaustive search algorithm. Values greater or less than two proved to be unsatisfactory. The first arrangement, where no cap was imposed, allowed the cell to be loaded to its maximum capacity. This arrangement made it possible to load the cell to the point where any additional part would completely block the system. It made shifting parts difficult and led to the conclusion that loading the system to maximum capacity resulted in buffers being used more as storage locations than temporary transfer points. The chosen cap of the number of buffers plus two provided the best compromise between overloading the system and switching to an unload always rule, regardless of the number of buffers. In conclusion, to benefit from the buffers, the time spent by parts in the buffers should be minimal.

The shift load, and unload routines are the other places in the program in which a part selection hierarchy is used. In both cases a decision needs to be made as to which of 'x' number of finished parts is to be unloaded or shifted first. Here, again, priority is given to the parts at a station that is in high demand. If demand among stations with finished parts is equal, then the 'done parts queue' is referred to, in order to see which part will be finished first. Based on these criteria, a decision is made as to which part to unload next. In the case of shifting a part from station to station, the same two criteria are examined. First, the part being shifted can be shifted only to a station that has free buffer space. Second, a conflict check is performed in the case of non-sequential processing.

3.3. Hierarchy of Decisions to Robot Move

Having stated the basic criteria for making decisions regarding robot moves and part selection rules, this section provides a clear understanding of how these decisions affect the batch processing time. This section also demonstrates the differences between the loading and unloading rules, as well as the non-sequential and sequential loading rules. The differences are demonstrated in four step-by-step analyses of state changes in one particular robot loading and sequencing example. There are three kinds of parts to be processed, parts A, B and C as shown in Table 1. The robot's move times are each 5 seconds and there is one buffer at each station.

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine time 15 s	Machine time 15 s	Machine time 15 s	Machine time 15 s
A	2	X	X		
B	2		X	X	
C	2			X	X

Table 1. Batch order 1.1

The next two tables provide a step by step account of the processing of this batch of parts through the FMC. The objective is to highlight the difference between the robot's movements chosen when an unload priority is used versus those when the load priority is used to process parts sequentially. For example, part B needs to visit station 2 before proceeding to station 3.

The notation used to describe the station changes is as follows;

- L indicates that the robot has gone to the input buffer, taken a new part and deposited it at a station.
- U indicates that the robot has picked up a part and dropped it off at the system's out buffer.
- S indicates that the robot has picked-up a part and dropped it off at a station.
- Parts are identified by a part letter followed by a number to indicate which specific one of a part type is being referenced (e.g. A1 or A2),
- A station, S, and its affiliated buffers, are designated by the notation Sxy. The y indicates what buffer position the part has taken at station x. Moreover y=0 implies

that the part at a station is currently being machined or that machining at station x has finished on this part.

- If there is no letter in front of a part name but a station identified after the part, then the part has shifted up one place in the buffer space. For example if a part B number 3 (B3) was loaded on station 3 buffer 2 (S32) and the part at station 3 buffer 0 (S30) was unloaded, then part B3 would move to S31.
- 'Done' means the part has completed processing at a station.

Tables 2.a and 2.b present the experimental results of both a numerically simulated sequential loading and unloading run. The 'time' column gives the time that the action in the 'robot action' column was finished. The 'other actions' column indicates if any other action occurred at that time, such as a part has been finished machining or has moved from one buffer space to another. The 'machine req' (machining requirements) column has four numbers. A given number represents the number of processing requirements at each station that is needed by all the parts currently in the system. This information is used to decide which parts enter the cell next. Values in 'machine req' only change when a part no longer needs to be machined at a station and it is no longer occupying that station. The last column shows the done queue, which indicates the order in which the stations have finished their machining jobs.

Time (s)	Robot action	Other actions	Machine Req	Done queue
5	L B1-S20		0110	0000
15	L C1-S30		0121	0000
20		B1 done	0121	2000
25	L A1-S10		1221	2000
30		C1 done	1221	2300
35	LB2-S21		1331	2300
40		A1 done	1331	2310
45	L C2-S31		1342	2310
55	L A2-S11		2442	2310
60		C2-S30	2432	2310
65	S C1-C40		2432	2100
70		B2-S20	2332	2100
75	S B2-S31	C2 done	2332	1300
75		B1-S30	2322	1300
80	S C2-S41	C1 done	2322	1400
85		B2 done	2322	1420
85		A2 S10	1322	4200
90	S A1-S21	B1 done	1322	4230
90		A1-S20	1222	4230
95	S B2-S31		1222	4300
100		C2 S40	1221	3000
105	U C1	A2 done	1221	3100
105		A1 done	1221	3120
115	S A2-S21	C2 done	0221	3240
120		B2-S30	0211	3240
125	U B1		0211	2400
130		A2-S20	0111	2400
135	U A1	B2 done	0111	4300
145	U C2	A2 done	0110	3200
155	U B2		0100	2000
165	C A2		0000	0000

Table 2.a Time history of sequential loading

Time (s)	Robot action	Other actions	Machine req	Done queue
5	L B1-S20		0110	0000
15	L C1-S30		0121	0000
20		B1 done	0121	2000
25	L A1-S10		1221	0000
30		C1 done	1221	2300
35	S B1-S31		1121	3000
40		A1 done	1121	3100
40		B1-S30	1121	3100
45	S C1-S40		1111	1000
55	S A1-S20	B1 done	0111	3000
60		C1 done	0111	3400
65	U B1		0101	4000
75		A1 done	0101	4200
75	UC1		0100	2000
80	UA1		0000	0000
90	L B1-S20		0110	0000
95	L C1-S30		0121	0000
100		B1 done	0121	2000
105	L A1-S10		1221	0000
110		C1 done	1221	2300
115	S B1-S31		1121	3000
120		A1 done	1121	3100
120		B1-S30	1121	3100
125	S C1-S40		1111	1000
130	S A1-S20	B1 done	0111	3000
135		C1 done	0111	3400
145	U B1		0101	4000
155		A1 done	0101	4200
160	UC1		0100	2000
165	UA1		0000	0000

Table 2.b Time history of sequential unloading

The two examples differ only in the priority rule used for the robot's moves. Sequential loading always loads parts if there is place to load a part. If not, the computer considers part shifts and, if this is also not possible, then the computer considers unloading parts. In sequential unloading, the hierarchy of robot actions is the reverse - first the possibility of unloading a part is considered, then shifting a part, and then loading parts. These two cases produce sequences of events that are different even though the total batch time is the same. The robot's move time forms the bottleneck in these examples. Examining the 'done queue' list shows this. If it tends to have one or more parts waiting, it is because the robot cannot keep up with the demand for its action.

Once the first level of decision making is determined (whether to load, shift unload or wait), the second level of decision making comes into play. Here the program decides which part is to be loaded, shifted or unloaded. The flow chart in Appendix D lists the order of priority at this point. The order in which parts are loaded is determined by the batch demand for a machine and whether space is available in the FMC for a new part. To explain the term batch demand for a machine, consider 50 unprocessed parts of which 30 require machine 3, 20 require machine 4 and 10 require machine 2. Given this batch, the control program first checks if there is a part which can be loaded on machine 3; if not, it checks if there is a part which needs machine 4 and then machine 2. In this example, the machines most in demand are 3 and 4. Hence, they are loaded first by parts that require them, in this case part B and part C.

One important decision that has to be made in this model is whether or not to continue to adhere to a set of rules or to make special conditions for unique cases. For example, if the robot has just loaded the buffer of one station with a part and the part being machined at that station is finished, it may be beneficial to take advantage of the robot's position to shift that part (such as at time 35 in the load example of Table 2.a, given in bold lettering) rather than addressing the next part in the queue according to the hierarchy set out. Through experimentation, it is determined that the benefit of implementing such a policy is limited to cases where the robot's move time is the bottleneck. Otherwise imposing such a policy is no longer beneficial; rather, it increases the batch throughput time.

The non-sequential unload example given in Table 3.b shows a case in which no unload, shift or load operations are possible. Hence, the robot moves to the station that would first be done machining a part and then the robot waits for the next required action. At time 150 seconds (see bold lettering), the robot is at station 3 and, at this station, no more parts are to be machined nor are there any more parts to be loaded. By checking another queue in the program, called the 'working queue', the program knows ahead of time that station 4 is the next station with a completed part and, hence, the robot moves to station 4 and waits to perform the next task. This kind of "look ahead" feature reduces the robot's movement time by having it take advantage of "free time" to move to another location. The robot arrives at station 4 at time 155 seconds and waits 5 seconds for the part to be finished, as opposed to waiting until the part is finished before moving to station 4. By taking advantage of the "look ahead" feature the 5 seconds required to move from station to station is saved.

In the sequential load examples given in Table 2.a, the order in which parts are loaded is B C A B C A and all the parts are loaded immediately, meaning the first six robot actions load parts into the cell. In the non-sequential load example shown in Table 3.a, the load order is A B A C C and only after two shifts occur is the final part B loaded. The reason that the order is different is because two parts can be loaded immediately on station 2, namely parts A and B. However, if there is a restriction requiring the parts to be processed in order, then part A cannot start on station 2. It is also important to observe that the non-sequential loading cannot immediately load all six parts because, in this case, if the last part B is added then further movement is blocked. This situation does not occur in sequential loading.

Time (s)	Robot action	Other Actions	Machine req	Done queue
5	L A1-S20		1100	0000
15	L B1-S30		1210	0000
20		A1 done	1210	2000
25	L A2-S10		2310	2000
30		B1 done	2310	2300
35	L C1-S40		2321	2300
40		A2 done	2321	2310
45	L C2-S31		2332	2310
50		C1 done	2332	2314
55	S A1-S11		2232	3140
60		C2-S30	2222	1400
65	S B1-S20		2222	1400
75	L B2-S31	C2 done	2332	1430
75		B2-S30	2332	1400
80	S C2-S41	B1 done	2322	1420
85		A1 S10	1322	4200
90	S A2-S21	B2 done	1322	4230
95		C2-S40	1322	2300
100	S C1-S31	A1 done	1321	2310
105		A2-S20	1321	3100
110	U B1	C2 done	1221	3140
115		C1-S30	1221	1400
120	S B2-S21	A2 done	1211	1420
125		B2-S20	1211	4000
130	U A2	C1 done	1111	4300
135	U A1	B1 done	0111	4320
145	U C2		0110	3200
155	U C1		0100	2000
165	C B2		0000	0000

Table 3.a Time history of non-sequential loading

Time (s)	Robot action	Other Actions	Machine req	Done queue
5	L A1-S20		1100	0000
15	L B1-S30		1210	0000
20		A1 done	1210	2000
25	L A2-S10		2310	2000
30		B1 done	2310	2300
35	S A1-S11		2210	3000
40		A2 done	2210	3100
45	S B1-S20		2200	1000
50		A1-S10	2200	1000
55	S A2-S11		1200	0000
60		B1 done	1200	2000
65	L B2-S20	A1 done	1310	2100
70		A2-S20	1310	2100
75	U B1		1210	1000
80		B2 done	1210	1300
85	UA1	A2 done	0210	1320
95	UA2		0110	3000
105	S B2-S20		0100	0000
115	L C1-S30		0111	0000
120		B2 done	0111	2000
125	L C2-S40		0122	2000
130		C1 done	0122	2300
135	U B2		0022	3000
140		C2 done	0022	3400
145	S C1-S41	C1-S40	0012	4000
150	**S C2-30**		**0011**	**0000**
160		C1 done	0011	4000
165	U C1	C2 done	0010	3000
175	U C2		0000	0000

Table 3.b Time history of non-sequential unloading

The next significant point about non-sequentially loading the parts is that there is a possibility to stifle movement in the FMC by overloading it with parts. If all the parts in a batch need to visit three machines and six of the parts are loaded, for example, the number of free buffer spaces for manoeuvring is limited. Suppose, a part on machine 3 needs to go to station 2. However, the part at machine 2 is waiting for a place at station 1 to become available. So, instead of having the flexibility to select an efficient solution, the system is left to choose whatever option is left. This problem does not exist with the sequential processing because the bottleneck machine naturally regulates the cell's capacity. In the non-sequential situation, therefore, an additional condition is added to the load rule that restricts the number of parts that need a specific machine to two. Two is more than the number of parts that can be accommodated at the station buffers. In the non-sequential load example (Table 3.a), part B2 is not loaded at 55 seconds. Part B2 is loaded only when both the machining requirements for machines 2 and 3 (both of which part B needs) by the parts currently in the cell are less than three (See column 4 of Table 3.a.). If there were two buffers, this number would be four.

This section has provided an overview of the software developed to handle dynamic FMC control problems on-line or off-line for simulation purposes. The key programming decisions have been outlined as well as the overall system's structure. The last part of this section provided several examples of batch production runs to highlight the nuances between the various rules. The next section describes various aspects related to the implementation of the proposed cell control strategy. This will allow a clearer understanding of the complexity of the robot's control as well as part loading issues.

4. Experiments and Analysis

The first experiment is to verify the functionality of the software and hardware for controlling the robot and responding to FMC conditions. Once verified, questions concerning loading and the robot's movement can be examined. The experiments, to be conducted by using the simulation software, are broad in scope and place minimal restrictions on any FMC condition. The intent behind the design of this software is that it can be used to perform similar experiments in both an industrial and a laboratory setting. Keeping in mind that variables in a specific industrial FMC are generally more restricted. For example, in industry, the variability amongst batches would be known as would be the processing times and number of machines in the cell. With this information, a tighter picture of the flexibility required by the cell would allow improvements in the sequencing of parts, in addition to the fine-tuning of the control program for specific situations. The present studies serve to provide an idea of the improvements that the software can afford, both through its implementation and its capacity as a simulation tool.

4.1. On-line Dynamic Control of a FMC

The first experiment is intended to verify the functionality of the on-line FMC control system. This proved to be successful. All robot movements corresponded appropriately to the programmed decision structures. Feedback loops from the system also functioned appropriately. They caused the robot to wait until the machining operations were completed before initiating the pick-up of parts.

To compare the throughput timing results from the simulations with those of the actual system, each individual robot sub-program is run and timed manually. The times are entered into the simulation, and the total simulated processing time are compared with the actual processing time. The results show a minor discrepancy between the two times. For a short test run, the discrepancy is around ±5% of the simulated time. This variation can be attributed to both the imprecision in the timing of the robot's sub-programs and the processor speed of the robot's controller which requires roughly half a second (it varied slightly) to poll the input information from the computer controller. This is a typical situation of stand-alone controllers whose hardware details are proprietary and cannot be controlled or monitored.

To conclude, the on-line robot controller software, with FMC sensor feedback, does accomplish the desired goal of creating a computer controlled robot that responds to the status of a FMC. To achieve a more accurate representation of the FMC during simulation would require recording the robot's movement times to an accuracy of one tenth of a second. However, no additional results of significance would be gained.

4.2. Concurrent Processing

The next test compares the results of individually processing part types (all part As then all part Bs) versus concurrently processing parts. The significance of concurrent processing over non-concurrent processing seems to be influenced greatly by the total order and the process that each part type has to undergo. Hence, the benefit of concurrent processing is composition dependent. The following three examples corroborate this assertion. The load priority is used in all cases, the processing machines invariably have no buffers, and the robot's movement between positions is always assumed to take 5 seconds.

If each part type in batch order 2.1 (given in Table 4) is processed individually, the total processing time is 545+1045+845+745=3180 seconds. In comparison, the total throughput for the same forty parts produced concurrently is 1085 seconds. Batch order 2.1 is an extreme case in which each part has only one process requirement. Producing all of one part type at a time resulted in three idle machines. The batch throughput time becomes the process time of each part on a specified station plus the time for the robot to load, and unload the part. Hence, the concurrent processing of the parts proves to be very beneficial and reduces the throughput time by almost a third.

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine Time 40 s	Machine Time 90 s	Machine Time 70 s	Machine Time 60 s
A	10	X			
B	10		X		
C	10			X	
D	10				X

Table 4. Batch order 2.1

If each part type in batch order 2.2 (Table 5) is processed individually, the total processing time is 1145*4=4580 seconds. The total throughput time for the same forty parts produced concurrently is 4395 seconds. This case is the opposite of case 1, that is all the parts, although they may still be unique, need to visit the same three stations.

Concurrent processing of parts has very little effect in this case. The minor improvement shown is due to the fact that concurrently processing the four part types can be done without interruption caused by the completion of all part A's and starting the next batch of part B's.

Part Type	Number Produced	Station 1 Machine Time 40 s	Station 2 Machine Time 90 s	Station 3 Machine Time 70 s	Station 4 Machine Time 60 s
A	10	X	X	X	
B	10	X	X	X	
C	10	X	X	X	
D	10	X	X	X	

Table 5. Batch order 2.2

If each part type is processed individually, as in batch order 2.3 (Table 6), the total processing time is 1090+1120+1145+1165=4520 seconds. The total throughput for the same forty parts produced concurrently is 4310 seconds. This example is a more realistic model of the types of parts that would be sent for processing in a FMS. In this case, parts need at least a couple of the machining processes in the FMC. Here, concurrently processing parts results in a 5% improvement.

Part Type	Number Produced	Station 1 Machine Time 40 s	Station 2 Machine Time 90 s	Station 3 Machine Time 70 s	Station 4 Machine Time 60 s
A	10	X	X		
B	10		X	X	
C	10	X	X	X	
D	10		X	X	X

Table 6. Batch order 2.3

Data from the next two tests are compared with those for batch order 2.3. The objective is to demonstrate the effect of changing the machining times on the throughput times for both concurrent and individual part type processing. For batch order 2.4 (Table 7), the processing times are halved. In this case, the total time for processing forty parts is 620+630+690+685=2630 seconds. The time it took to concurrently process the parts is 2530 seconds.

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine Time 20 s	Machine Time 45 s	Machine Time 35 s	Machine Time 30 s
A	10	X	X		
B	10		X	X	
C	10	X	X	X	
D	10		X	X	X

Table 7. Batch order 2.4

In the last batch order 2.5 (Table 8), the process times are halved again. Process by parts has a total process time of 380+387+452+450=1669 seconds. The total throughput for the same forty parts produced concurrently is 1638 seconds.

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine Time 10 s	Machine Time 22 s	Machine Time 17 s	Machine Time 15 s
A	10	X	X		
B	10		X	X	
C	10	X	X	X	
D	10		X	X	X

Table 8. Batch Order 2.5

The next table summarizes the results of batch order 2.3, 2.4 and 2.5.

Case	Total machining time required (s)	Total throughput time (s) by part	Total Throughput time(s) concurrent	% improvement	% decrease for part throughput	% decrease for concurrent throughput
2.3	7100	4520	4310	4.65	------	------
2.4	3550	2630	2530	3.80	41.8	41.3
2.5	1740	1669	1638	1.86	36.5	35.3

Table 9. Summary 2

This table demonstrates the effect of reducing the batch processing time requirements. There are two trends related to the 'total machining requirements' of a batch. The first shows that, as the total batch processing time is reduced, the closer the total throughput to the total machining requirements. It should be noted that, when processing one part type at a time with no buffers, the robot's total move time is 700 seconds. As the processing is completed more quickly, the robot is in greater demand compared to when the robot's moves occur while all the machines are busy machining parts. Reducing the machining time leaves the stations idle while the robot is moving. Essentially the robot becomes the bottleneck. Hence, even though the total machining requirement in batch order 2.4 is half that for batch order 2.3, the total reduction in throughput is less than 50%. (See the "percent decrease by parts" columns given in Table 9.) The change is even less from case 4 to 5. This observation is true for both the concurrent processing and by processing by part. The other observable trend is the influence of the processing time on the improvements afforded by concurrent processing. Column 4 of the table indicates a clear reduction of benefits brought through concurrent processing as the ratio of the processing to robot move times reduces.

What these experiments show is that the ability to concurrently process parts with different processing requirements in the same FMC reduces the throughput time of the batch in comparison to processing all one part type followed by another part type. Hence, the ability to dynamically respond to different part processing requests and FMC conditions takes fuller advantage of the capacity of a FMC over a fixed robot move cycle for processing one part type at a time.

It can also be argued that developing fixed cycles for a combination of part types would produce a superior batch throughput to that afforded by separately processing single part types. However, the flexibility that this dynamic response allows would still produce better results. Proof of this assertion is beyond the scope of this work.

4.3. Load versus Unload Prioritization

The previous section provided a detailed example of a complete batch production. The machine time for all stations was 15 seconds. This section will look at the effect of increasing the machining time while the robot's move time stays at 5 seconds for all inter-cell movements. The objective is to observe the effect of changing the ratio of the machining to robot move times on the total production time under the following, individually applied conditions.

1. The robot prioritizes unloading parts; parts are processed sequentially; or
2. the robot prioritizes loading parts; parts are processed sequentially; or
3. the robot prioritizes unloading parts; parts are processed non sequentially; or
4. the robot prioritizes loading parts; parts are processed sequentially.

In all four cases the 15 parts defined in Table 10 are to be produced. The simulation software ran a series of tests starting with all machining times set to 5 seconds. This time was incremented by 5 seconds for each subsequent test. The number of buffers at each station was always two.

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine Time 5 – 100 s	Machine Time 5 – 100 s	Machine Time 5 – 100 s	Machine Time 5 – 100 s
A	5	X	X		
B	5		X	X	
C	5			X	X

Table 10. Batch order 3.1

Additional data for these tests and relevant graphical representation of results is given more conveniently in Appendix E. A summary of the results are shown in the graph below, as well as in Table 11.

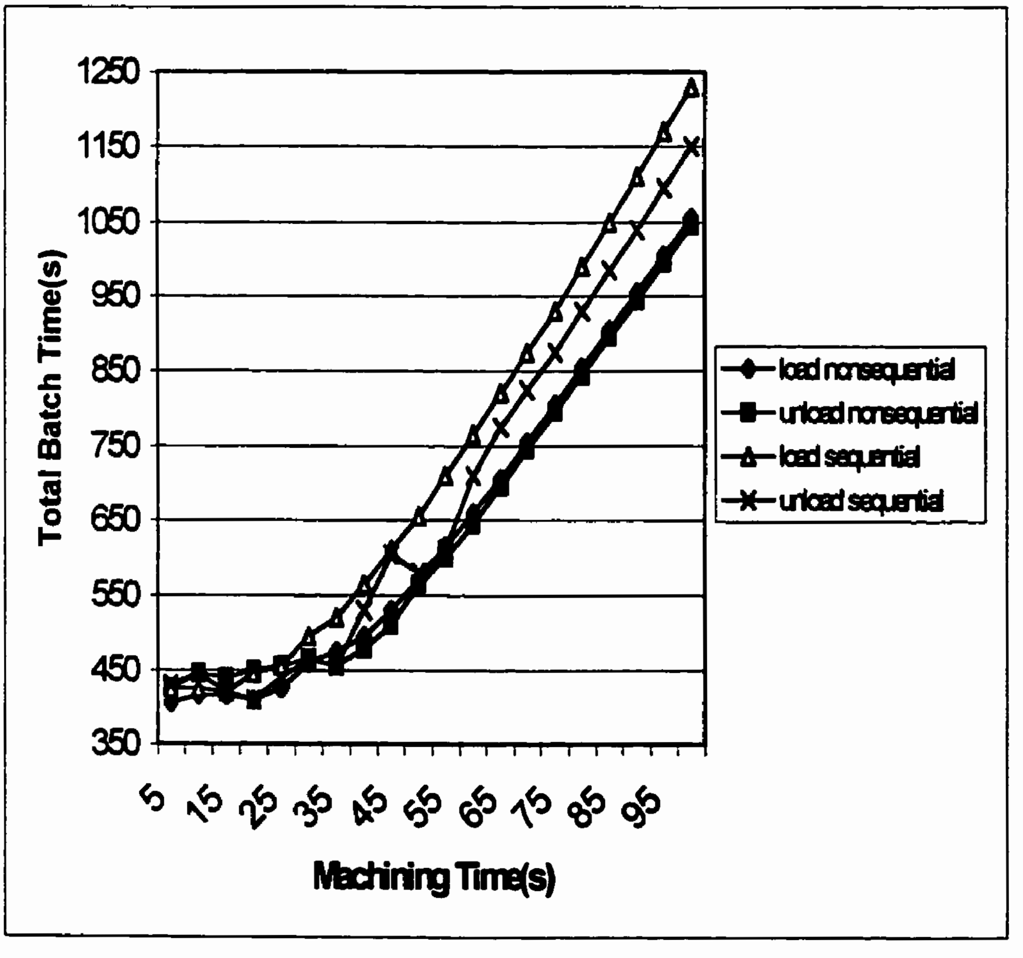


Figure 12. Effect of machining time on load and unload rules

Machining time(s)	non-sequential		sequential	
	load	Unload	load	unload
5	405	425	425	430
10	415	445	425	445
15	415	440	420	420
20	410	450	445	410
25	425	455	455	440
30	460	465	495	465
35	475	455	520	455
40	495	480	565	530
45	530	510	610	605
50	570	565	655	580
55	615	600	710	610
60	660	645	765	710
65	705	695	820	775
70	755	745	875	825
75	805	795	930	875
80	855	845	990	930
85	905	895	1050	985
90	955	945	1110	1040
95	1005	995	1170	1095
100	1055	1045	1230	1150

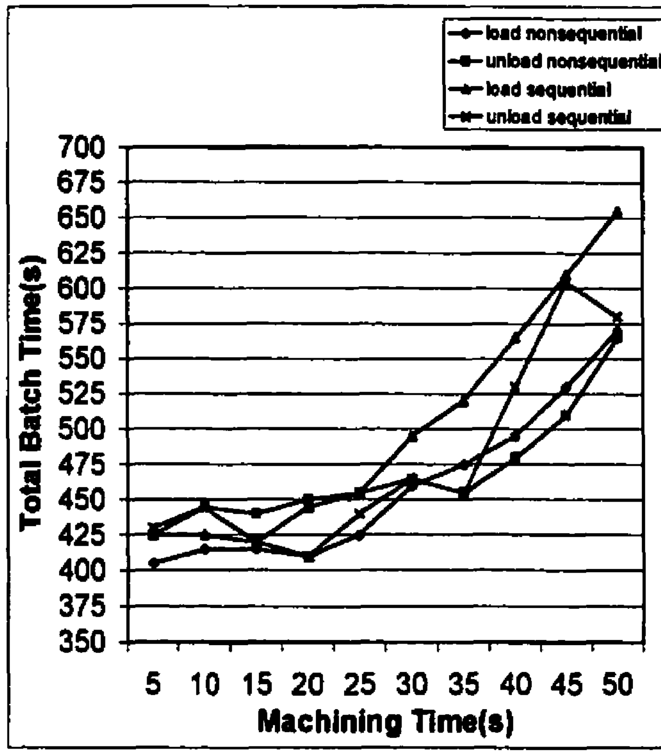


Table 11. Machining time data **Figure 13. Highlight machining time effect**

The results shown in Table 11 indicate that, in both the sequential and non-sequential cases, prioritizing the unload results in a lower total batch time than prioritizing the loading when the ratio of the machining and robot move times is about six or more. This test also indicates that non-sequential processing of parts is faster than sequential process. When the average machining time is less than about six times the robot's move time, there seems to be no observable pattern. This trend is shown in Figure 13, (Note the results given in Figure 12 for 0-30 seconds are plotted on a larger scale in Figure 13).

The observations made about this particular experiment are generally applicable. For all variations in batch sizes and part requirements, the unload priority gives a lower total time than when the load is prioritized. This observation holds true regardless of the number of buffers or parts. What cannot be generalized is at what ratio of the average machining time to robot move time does the system become stable (i.e. achieves a constant relationship). Thus far, it has been observed that, for the parameters used in these simulation studies, all batch simulations become stable under a machining to robot move time ratio of around 10:1.

4.4 Analysis of Buffer Size

This section will examine the effects of buffer size on the batch processing time. Here again four tests will be conducted in order to understand the impact of the buffer sizes for both non-sequential and sequential unloading. The same batch order employed in the previous experiment will be used and, as in the previous cases, 15 parts will be manufactured with the machining time incremented progressively by 5 seconds from 5 to 100 seconds.

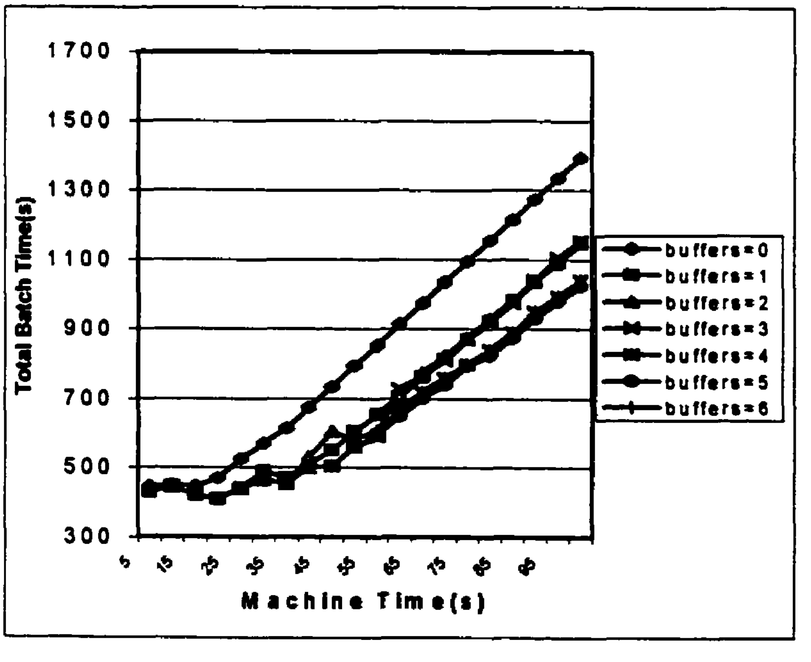


Figure 14.a Effect of buffers on sequential processing

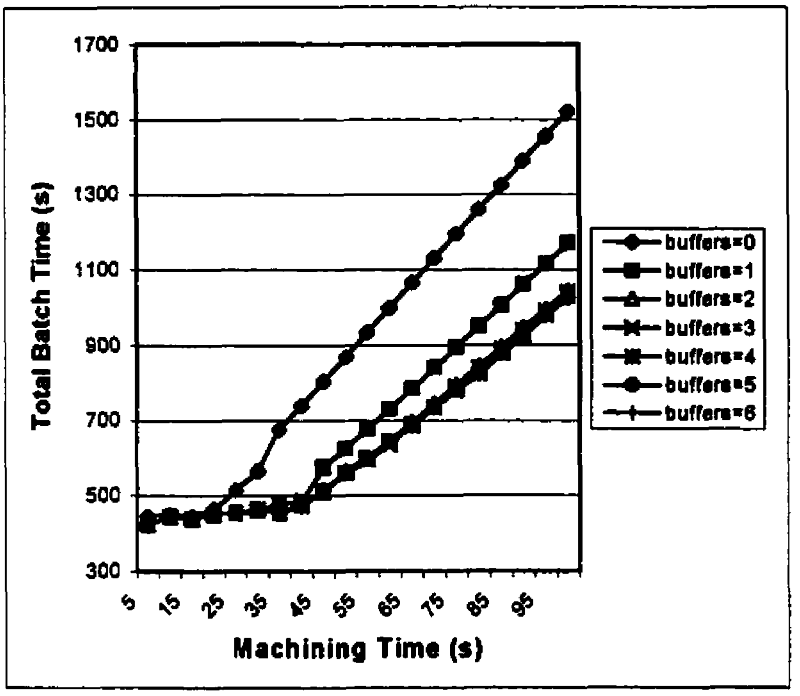


Figure 14.b Effect of buffers on non-sequential processing

Machining Time	Unload always non-sequential buffers=0	buffers=1	buffers=2	buffers=3	buffers=4	buffers=5	buffers=6
5	445	425	425	425	425	425	425
10	450	445	445	445	445	445	445
15	445	440	440	440	440	440	440
20	465	450	450	450	450	450	450
25	515	455	455	455	455	455	455
30	565	460	465	465	465	465	465
35	675	480	455	455	455	455	455
40	740	485	480	475	475	475	475
45	805	575	510	515	515	515	515
50	870	625	565	560	560	560	560
55	935	675	600	595	600	600	600
60	1000	730	645	635	645	645	645
65	1065	785	695	690	685	690	690
70	1130	840	745	735	735	735	735
75	1195	895	795	790	780	780	780
80	1260	950	845	840	825	825	825
85	1325	1005	895	890	880	880	880
90	1390	1060	945	940	925	925	925
95	1455	1115	995	990	980	980	980
100	1520	1170	1045	1040	1030	1025	1025

Table 12.a Effect of buffers on non-sequential processing

Machining Time	Unload always sequentially buffers=0	buffers=1	buffers=2	buffers=3	buffers=4	buffers=5	buffers=6
5	445	430	430	430	430	430	430
10	450	445	445	445	445	445	445
15	445	420	420	420	420	420	420
20	470	410	410	410	410	410	410
25	525	440	440	440	440	440	440
30	570	490	465	465	465	465	465
35	615	470	455	455	455	455	455
40	675	510	530	500	500	500	500
45	735	550	605	505	505	505	505
50	795	605	580	560	560	560	560
55	855	655	610	590	595	595	595
60	915	705	710	730	670	650	650
65	975	760	775	765	720	700	700
70	1035	815	825	810	760	740	740
75	1095	870	875	870	800	795	795
80	1155	925	930	920	840	825	825
85	1215	980	985	975	890	875	875
90	1275	1035	1040	1040	950	930	930
95	1335	1090	1095	1105	995	980	980
100	1395	1145	1150	1155	1040	1025	1025

Table 12.b Effect of buffers on sequential processing

From Figures 14.a and 14.b, as well as Tables 12.a and 12.b (taken from Appendix F), it would appear that no significant improvement is produced by increasing the buffer size to more than two in the non-sequential case and more than four in the sequential case. The sequential example shows that having more buffers in the FMC ensures that stations past the bottleneck have less chance of remaining idle. In both examples, since there are only 15 parts in the batch, this inevitably decreases the need for buffers.

The data given in Table 12 also shows that the point where more buffers become beneficial is related directly to the average machining time. For example, in the non-sequential case it is only after the average machining time is greater then 65 seconds that there is a benefit to having four rather than three buffers. In the analogous sequential case there is a benefit after 60 seconds. In both cases, as the ratio of robot move time to machining time increases, the more the robot sits idle. If extra buffers are available, the robot can take advantage of this idle time to fill the buffers with parts. This ensures that when a part is removed from a machine there is always a new part immediately available for machining. By decreasing machine idle time, the total batch throughput is also decreased.

4.5 Sequential versus Non-sequential Processing

The next experiment further examines the difference between non-sequential and sequential part processing. Four different batch orders are tested. The composition of each order is intended to examine the effect of having different bottleneck machines in the cell. All the orders demand equal machining times and each station has two buffers. In all tests, 40 parts are processed (10 of each part type A,B,C,D). Moreover, each part has the same total machining time requirements for all cases, although these requirements are not necessarily on the same machines for each case. In the four different cases considered, each specific machine requires 90 seconds to machine a part. However, the machines may have different total machining demands (meaning machine 1 processes 20 parts while machine 2 processes 30 parts). The details of the batch requirements are given in the tables below.

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine Time 90 s	Machine Time 90 s	Machine Time 90 s	Machine Time 90 s
A	10	X	X		
B	10	X		X	
C	10			X	X
D	10	X	X		X

Table 13. Batch order 3.1

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine Time 90 s	Machine Time 90 s	Machine Time 90 s	Machine Time 90 s
A	10	X	X		
B	10		X	X	
C	10			X	X
D	10	X	X		X

Table 14. Batch order 3.2

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine Time 90 s	Machine Time 90 s	Machine Time 90 s	Machine Time 90 s
A	10	X	X		
B	10		X	X	
C	10			X	X
D	10	X		X	X

Table 15. Batch order 3.3

Part Type	Number Produced	Station 1	Station 2	Station 3	Station 4
		Machine Time 90 s	Machine Time 90 s	Machine Time 90 s	Machine Time 90 s
A	10	X	X		
B	10		X		X
C	10			X	X
D	10	X		X	X

Table 16. Batch order 3.4

The results for the four batch orders are summarized in Table 17. What is clear from this table is that non-sequential processing is always faster than sequential processing. However, what needs to be explained is the noticeable variation in the throughput times given that all the batch cases have the same total machining time. Through experimentation, these variations have been attributed to the location of the bottleneck machine. Indeed, the batch requirements were chosen specifically to demonstrate this effect. In all cases, three of the machines are required to machine 20 parts while the fourth is required to machine 30 parts. Clearly the machine with the additional load corresponds to the bottleneck. (The batch orders from example 3.1 to 3.4 each have a different machine forming the bottleneck).

Batch Order	Non-sequential seconds	Sequential Seconds
3.1	2710	2900
3.2	2710	3030
3.3	2790	3035
3.4	2790	3010

Table 17. Summary 3

From the results shown in Table 17, it would seem that non-sequential processing is slightly faster if the bottleneck machines are used first in the process. However, given that the processing is non-sequential, the machine order should, intuitively, not influence the results. This first observation indeed turns out to be incorrect. The reason for the variation in the batch times will be explored further in the next experiment, which links the part loading order to this throughput variation.

In regard to the throughput times of sequential processing, this example indicates that the location of the bottleneck machine should hardly affect the batch's throughput time. Through further experimentation (in the next section), it has been found that the loading order of the parts (more to follow), as well as the location of the bottleneck machine, does affect the total throughput time. The next experiment will demonstrate that, in the sequential case, when all the different orders parts can be introduced into the system, the throughput times are greater, on average, when the bottleneck machine is at the front of the queue of machines. Moreover, the throughput time decreases as the bottleneck machine progresses to the end of this queue.

4.6 Load Sequence of Parts

The last experiment will address the issue of the loading sequence of parts. As it stands, the program is capable of searching the parts that remain to be processed in order to find a suitable fit with the current state of the FMC. For example, if three machines are busy and the fourth machine is idle, the program searches the parts remaining to be processed to see if any part requires machining at the free station. Consequently, the order of parts introduced to the system is superseded by the requirements of the cell. However, in most cases, several parts may need the machines that are currently free or have free buffer space. This situation is where the 'search order' that the program uses for checking the remaining part types for loading suitability becomes critical. Until now, all the parts were processed in the same order that they entered the system (i.e. A,B,C,D). Indeed, there is no obvious reason for selecting one part over another, or pairing parts in a specific manner.

This test will examine how alternative loading sequences of parts affects the total throughput of a batch. Using the same scenarios presented in the previous experiment in section 4.5, 24 individual but different experiments were conducted. Each experiment involved a different loading sequence. The 24 experiments were conducted by using both the sequential and non-sequential rules. The results are presented in the table below. Complete results can be found in Appendix G.

The first, most observable result from Table 18 is that there is a marked variation in the batch throughput time that clearly depends on the order of the loading sequence. In the case of non-sequential processing, this variation is around 1% while, in the sequential case, variations are between 1.6 and 7.8%. This lower variability supports the conclusions given in section 4.5 that non-sequential processing is more beneficial than sequential processing. Lower batch throughput times are also obtained consistently by using non-sequential processing. This difference is also indicated by the average processing times listed at the bottom of Table 18. Furthermore it is clear that the best non-sequential results do not correspond to the best sequential results.

Non-sequential				(results in seconds)			Sequential			
Sequence	case 1	case 2	case 3	case 4		Sequence	case 1	Case 2	case 3	case 4
1234000	[redacted]	[redacted]	2790	2790		1234000	2900	[redacted]	3035	2935
1243000	2725	2725	2805	2805		1243000	2900	[redacted]	3035	2935
1324000	[redacted]	[redacted]	2770	2770		1324000	2900	[redacted]	3035	2935
1342000	[redacted]	[redacted]	2770	2770		1342000	[redacted]	[redacted]	3035	2935
1423000	2725	2725	[redacted]	[redacted]		1423000	[redacted]	[redacted]	3035	2935
1432000	[redacted]	[redacted]	2795	2795		1432000	[redacted]	[redacted]	3035	2935
2134000	[redacted]	[redacted]	[redacted]	[redacted]		2134000	3470	[redacted]	3035	2935
2143000	2725	2725	[redacted]	[redacted]		2143000	3470	[redacted]	3035	2935
2314000	[redacted]	[redacted]	[redacted]	[redacted]		2314000	3470	[redacted]	3035	2935
2341000	2770	2770	2725	2725		2341000	3375	3130	[redacted]	2815
2413000	2865	2865	2725	2725		2413000	3375	3130	[redacted]	2815
2431000	2865	2865	2725	2725		2431000	3375	3130	[redacted]	2815
3124000	[redacted]	[redacted]	2770	2770		3124000	2900	[redacted]	3035	2935
3142000	[redacted]	[redacted]	2770	2770		3142000	[redacted]	[redacted]	3035	2935
3214000	2720	2720	2725	2725		3214000	3470	[redacted]	3035	2935
3241000	2780	2780	2750	2750		3241000	3375	3130	[redacted]	[redacted]
3412000	[redacted]	[redacted]	2785	2785		3412000	3305	3130	[redacted]	[redacted]
3421000	2715	2715	2750	2750		3421000	3225	3130	[redacted]	[redacted]
4123000	2785	2785	2760	2760		4123000	3305	3130	[redacted]	[redacted]
4132000	[redacted]	[redacted]	[redacted]	[redacted]		4132000	3305	3130	[redacted]	[redacted]
4213000	2795	2795	2725	2725		4213000	3225	3130	[redacted]	[redacted]
4231000	2795	2795	2725	2725		4231000	3225	3130	[redacted]	[redacted]
4312000	[redacted]	[redacted]	[redacted]	[redacted]		4312000	3305	3130	[redacted]	[redacted]
4321000	2720	2720	2725	2725		4321000	3225	3130	[redacted]	[redacted]
Highest	2865	2865	2805	2805		Highest	3470	3130	3035	2935
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]		[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Standard Deviation	48.4	48.4	31.4	31.4		Standard Deviation	249.8	51.1	97.0	61.3
Average	2741.5	2741.5	2743.8	2743.8		Average	3180.1	3080.3	2940.3	2875.0

Table 18. Load Sequence

Overall, several patterns are observable in Table 18, such as the interchangeable nature of part type 1 and 3 in the non-sequential case. However, what has not been determined is how to predict when a specific loading sequence results in a higher than average throughput time, or how to select a sequence which would produce the lowest throughput time. Various hypotheses were perused to correlate the loading patterns and throughput results but no conclusive pattern was found. This problem requires further investigation.

5. Conclusions

The objective of this thesis is to enhance the notion of flexibility in "flexible" manufacturing cells so that a cell's utility can be enhanced by using a computer implemented scheduler. This aim has been achieved by allowing multiple parts to be processed concurrently without using predetermined cycles of robot movement. The program developed to control the FMC performs reliably. It provided tools for dynamically selecting parts and controlling the robot's movements to complete complex batch demands with lower throughput times than is possible by processing one part type at a time. The other significant contribution was to develop a dynamic manufacturing cell which could process parts non-sequentially, an issue which has not been addressed adequately in the literature.

The simulation program, developed in conjunction with the control software, allows users to experiment with a multitude of variables that exist in the FMC environment prior to selecting a strategy that best fits a production run. Experiments using this software demonstrated its potential as a tool for examining different FMC control heuristics, and the effect of buffers on the throughput as well as part loading order questions. From these experiments a general understanding of the complexity of this enhanced flexible environment can be gained.

The first critical issue that the simulation software helped to verify was that, in all cases, the concurrent processing of parts is more desirable than processing parts in cycles when trying to minimize the batch throughput time. The software also helped to study the relationship between robot's move to machining times. When this ratio was low (between 1:1 and 1:10), the throughput times were less predictable due to the robot's path changing. However, it was observed that this relationship became more predictable once the ratio of the robot's move to machining times was higher than 1:10. At this point all robot movements occurred while all the stations were engaged in machining. This resulted in a steady increase in the throughput time.

The software also helped to confirm that, from the perspective of batch throughput time, using an 'unload always' robot movement rule is consistently superior (albeit marginally) to a 'load always' rule. Numerical simulations also suggested that there was a limit to the number of buffers that could be added to reduce the throughput time. It was shown that, after a point, buffers began to work more as a storage device than as dynamic transfer points. Increasing the number of buffers can enhance the performance of the cell but excess work in progress in arguably not desirable.

The question of non-sequential processing versus sequential processing was also examined using the simulation software. The results show that non-sequential processing, when possible, reduces batch throughput times, thereby increasing the utilization of a cell. Factors that increase or decrease the significance of this improvement include the number of machines in the FMC and the location of the "bottle neck machine" or the machine most in demand in the machining processing cycle.

The final set of experiments showed the effect on the batch throughput time of the sequence in which parts are introduced into the cell. Consideration of sequence in which parts are introduced proves to be significant depending on the process requirements of the parts in the batch. An experiment of 24 different part sequences was tested with four different batch orders of 40 parts. Each batch required a total of 135 minutes of machining. As a result of the different part orders there was a 1% variation in the batch throughput time using non-sequential processing, and a 7.8% variation in sequential processing. Overall, the loading sequence had a greater effect on sequential processing than that produced by non-sequential processing. However, no easily generalized patterns were obvious for selecting the sequence that generates the smallest batch throughput time. Thus far, the only way to determine the best sequence in which to load part is to simulate all combinations and allow the software to choose the best part sequence for a given scenario which can not be done in real time.

6. Further Work

The software has the flexibility to handle a broad range of FMC designs and part processing possibilities. It would be worth while to find an industrial FMC setting to test the software. Given a specific situation, the software's variables could be honed to specific criteria and the simulation tool could be used better to examine the effects of different production options such as the machine layout, buffer size, batch size, non-sequential processing (where possible) and loading sequences of parts.

The question of 'loading sequence' of parts has not been answered. Although search algorithms exist that can find optimal sequences for batch processing, most are only effective off-line. Furthermore, they require a long processing time or they can handle only limited conditions. In my opinion, a more robust dynamic solution is required. It is in this area that a neural network could, perhaps, be used gainfully to identify a weighting scheme for selecting loading sequences to produce the shortest batch throughput time.

7. Reference

Buzacott,J.A., Shanthikumar,J.G. (1980) "Models for Understanding Flexible Manufacturing Systems" *Management Science*, Vol. 12, No.4 pp 339-349.

Chen,H., Chu,C., Proth,J.M., (1997) "Sequencing of Parts in Robotic Cells" *The International Journal of Flexible Manufacturing Systems*, Vol. 9, pp 81-104.

Chen,H.G., Guerrero,H.H. (1990) "Robot Scheduling System for Flexible Manufacturing Cells" *IEEE International Engineering Management Conference*, pp 113-118.

Cheng,C.W., Sun,T.H., Fu,L.C. (1994) "Petri–Net Based Modeling and Scheduling of a Flexible Manufacturing System" 1994 IEEE, pp 513-517.

Kamoun,H., Hall,N.G., Sriskandarajah,C. (1993) "Scheduling in Robotic Cells: Heuristics and Cell Design", Working Paper 93-07, University of Toronto.

King,R.E., Hodgson,T.J., Chafee, F. W. (1993) "Robot Task Scheduling in a Flexible Manufacturing Cell" *IIE Transactions* Vol. 25, No.2 pp 80-87.

Kumar,R., Li,H., (1994). "Assembly Time Optimization of PCB Assembly" *Proceedings of the American Control Conference Baltimore Maryland* pp 306-310, June.

Lin,L., Wakabayashi, M., Adiga,S., (1994) "Object–Oriented Modeling and Implementation of Control Software for a Robotic Flexible Manufacturing Cell" *Robotics & Computer-Integrated Manufacturing* Vol. 11, No.1 pp 1-12.

Moreno,A.A., Ding,F.Y. (1993) "A Constructive Heuristic Algorithm for Concurrently Selecting and Sequencing Jobs in an FMS" *Environment International Journal of Production Research* Vol. 31, No.5 pp 1157-1169.

Niemi,E., Davies,B.J. (1989) "Simulation of An Optimizing FMS-Cell Control System" *Robotics & Computer-Integrated Manufacturing*, Vol. 5, No.2/3 pp 229-234.

Park,Y.B., (1994) "Optimizing Robot's Service Movement In A Robot-Centered FMC" *Computers & Industrial Engineering* Vol. 27, No.1-4 pp47-50.

Rebaine,D., Strusevich,V.A. (1999) "Two Machine Open Shop Scheduling with Special Transportation Times" *Journal of the Operational Research Society* Vol. 50, pp 756-764.

Stecke,K.E. (1985) "Design, Planning, Scheduling, and Control Problems of Flexible Manufacturing Systems" *Annals of Operations Research* 3, pp 3-12

Sethi,S.P., Sriskandarajah,C., Sorger,G., Blazewicz,J., Kubiak,W. (1992) "Sequencing of Parts and Robot Moves in a Robotic Cell" *The International Journal of Flexible Manufacturing Systems*, Vol. 4, pp 331-358.

Singh,N., (1996) *Systems Approach to Computer-Integrated Design and Manufacturing* John Wiley & Sons, Inc New York.

Yalcin,A., Boucher,T.O., (1999) "An Architecture for Flexible Manufacturing Cells With Alternative Machining and Alternative Sequencing" *IEEE Transactions on Robotics and Automations*, Vol. 15, No.6 pp 1126-1130.

Wemmerlov,U., Hyer,N.L., (1997) "Research issues in Cellular Manufacturing" *International Journal of Production Research*, Vol. 25, pp 13-431.

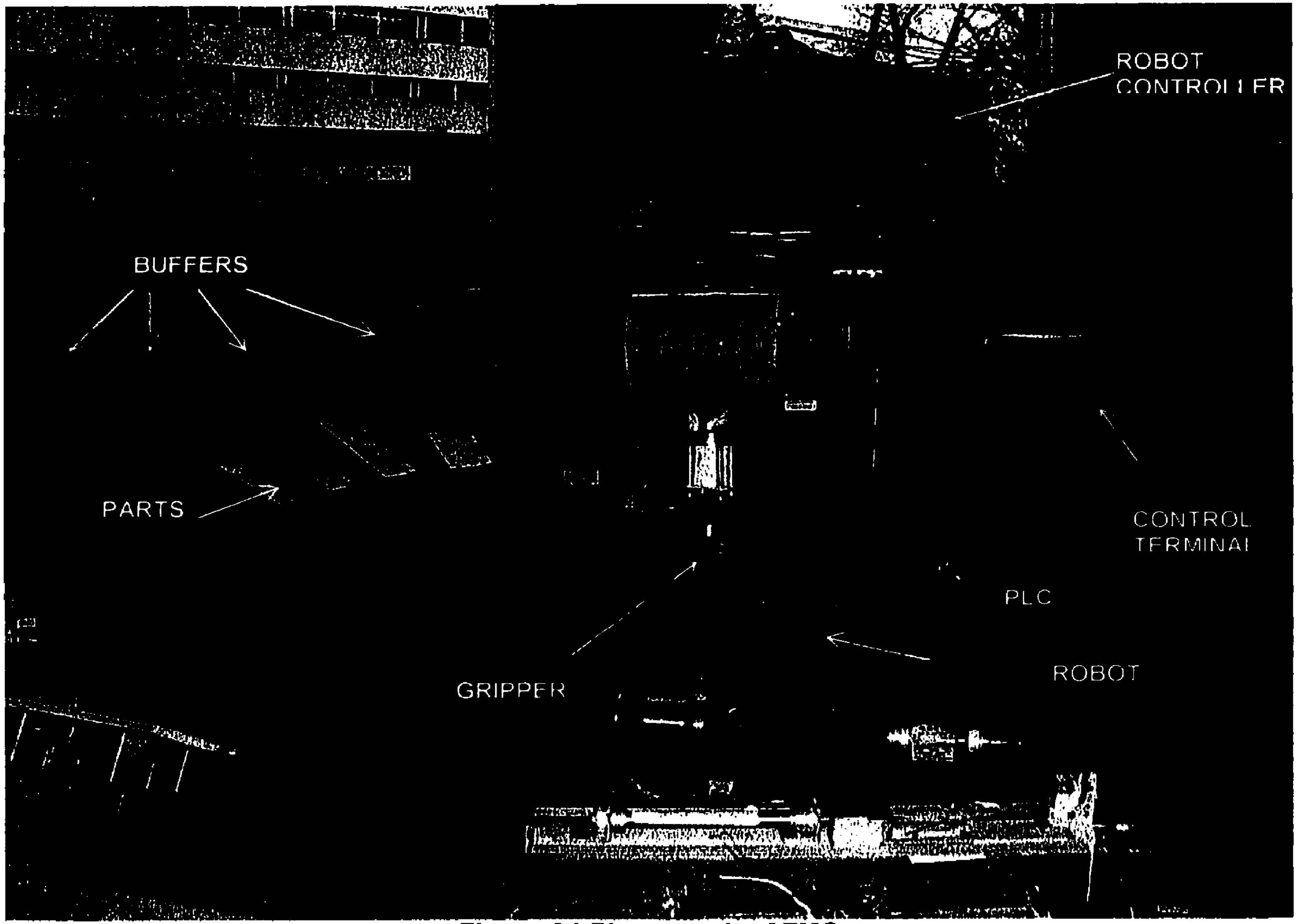


Figure A1 Photograph of FMC

Appendix B .... Robot program

This is the main robot control program. It loops continuously during operation.

```
10   Velocity 1000-2000 mm/sec   (This sets the minimum and maximum movement speed)
20   Robot coordinates   (This sets the movement's frame of reference)
30   Tool center point 0   (Indicates the robot's point of reference)
40   Frame 0   (This sets the work envelop)
50   Jump to line 200 if input 1 is high    (These jump-to statements are prompted by
60   Jump to line 220 if input 2 is high    the control program)
70   Jump to line 240 if input 3 is high
80   Jump to line 260 if input 4 is high
90   Jump to line 280 if input 5 is high
100  Jump to line 300 if input 6 is high
110  Jump to line 320 if input 7 is high
120  Jump to line 340 if input 8 is high
130  Jump to line 360 if input 9 is high
140  Jump to line 380 if input 10 is high
150  Jump to line 400 if input 11 is high
160  Jump to line 420 if input 12 is high
170  Jump to line 440 if input 13 is high
180  Jump to line 460 if input 14 is high
200  Call program 21                        (The call command allows this program to run
210  Jump to line 540                       another program from within the main program)
220  Call program 22
230  Jump to line 540
240  Call program 23
250  Jump to line 540
260  Call program 24
270  Jump to line 540
280  Call program 25
290  Jump to line 540
300  Call program 26
310  Jump to line 540
320  Call program 27
330  Jump to line 540
340  Call program 28
350  Jump to line 540
360  Call program 29
370  Jump to line 540
380  Call program 30
390  Jump to line 540
400  Call program 31
410  Jump to line 540
420  Call program 32
430  Jump to line 540
440  Call program 33
```

```
450  Jump to line 540
460  Call program 34
470  Jump to line 540
480  Reset output 1            (If no call is made, the internal variable is reset)
490  Return                    (The program returns to the beginning)
```

The following is an example of one of the 14 programs called from within the main program.

```
10   Velocity 1000-2000 mm/sec                (Same setup parameters as in the
20   Robot coordinates                         main program)
30   Tool center point 0
40   Frame 0
50   Set output 1 high      (This ensures no other message is set before this routine is completed)
60   Move to xxyyzz                           (Preset location)
70   Move to xxyyzz at 20% velocity fine      (Slows down for delicate placement)
80   Turn off gripper                         (Part is released)
90   Move to xxyyzz at 20% velocity fine      (Robot slowly retracts)
100  Move to xxyyzz
120  Return                                   (Program returns to main program)
```

The following is a description of the function of each called program.

Program 21 goes to the input buffer and picks a new part and returns it to a central position.
Program 22 goes from the central position with a part and releases it at the output buffer.
Program 23 goes from the central position with a part and releases it at the top of the buffer at station #1 and returns to the central location.
Program 24 goes from the central position with a part and releases it at the top of the buffer at station #2 and returns to the central location.
Program 25 goes from the central position with a part and releases it at the top of the buffer at station #3 and returns to the central location.
Program 26 goes from the central position with a part and releases it at the top of the buffer at station #4 and returns to the central location.
Program 27 moves from the central to just above station #1.
Program 28 moves from the central to just above station #2.
Program 29 moves from the central to just above station #3.
Program 30 moves from the central to just above station #4.
Program 31 precedes program 27, it picks up the part at station #1 and returns to the home position.
Program 32 precedes program 28, it picks up the part at station #2 and returns to the home position.
Program 33 precedes program 29, it picks up the part at station #3 and returns to the home position.
Program 34 precedes program 30, it picks up the part at station #4 and returns to the home position.

Appendix C .... PLC program

Logic Ladder

Inputs from limit switches at the processing stations located at the bottom of each buffer;
At station #1 the limit switch 00103 goes high when the station is occupied.
At station #2 the limit switch 00102 goes high when the station is occupied.
At station #3 the limit switch 00101 goes high when the station is occupied.
At station #4 the limit switch 00100 goes high when the station is occupied.

The outputs from the PLC to the I/O board connected to the PC are based on the status of 4 timers.
An output becomes high only when a timer (started by a switch at on of the stations) has been on for a pre-set number of seconds.
Timer T1, which is associated to switch 00100 sets output 10000 or 0 high
Timer T2, which is associated to switch 00101 sets output 10001 or 1 high
Timer T3, which is associated to switch 00102 sets output 10002 or 2 high
Timer T4, which is associated to switch 00103 sets output 10003 or 3 high

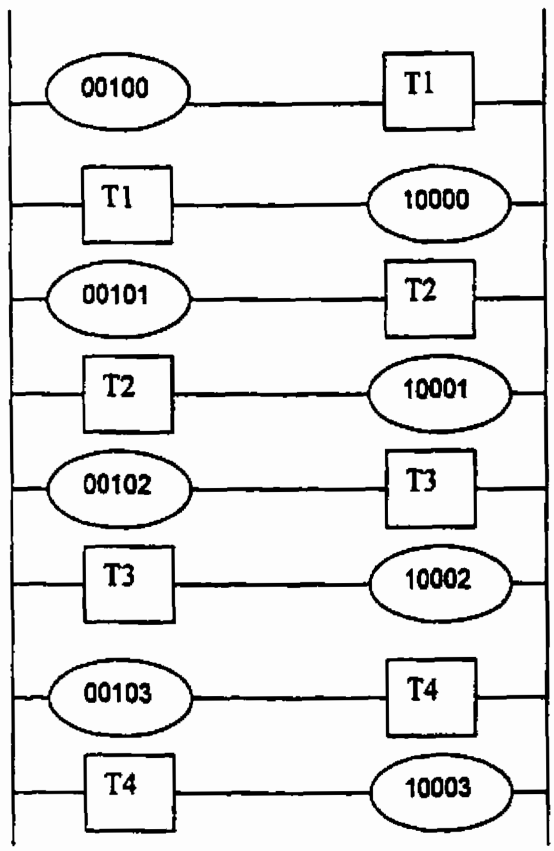


Figure C1 Logic ladder

Appendix D..........................Program Flow Chart (Example Prioritizing Loading)

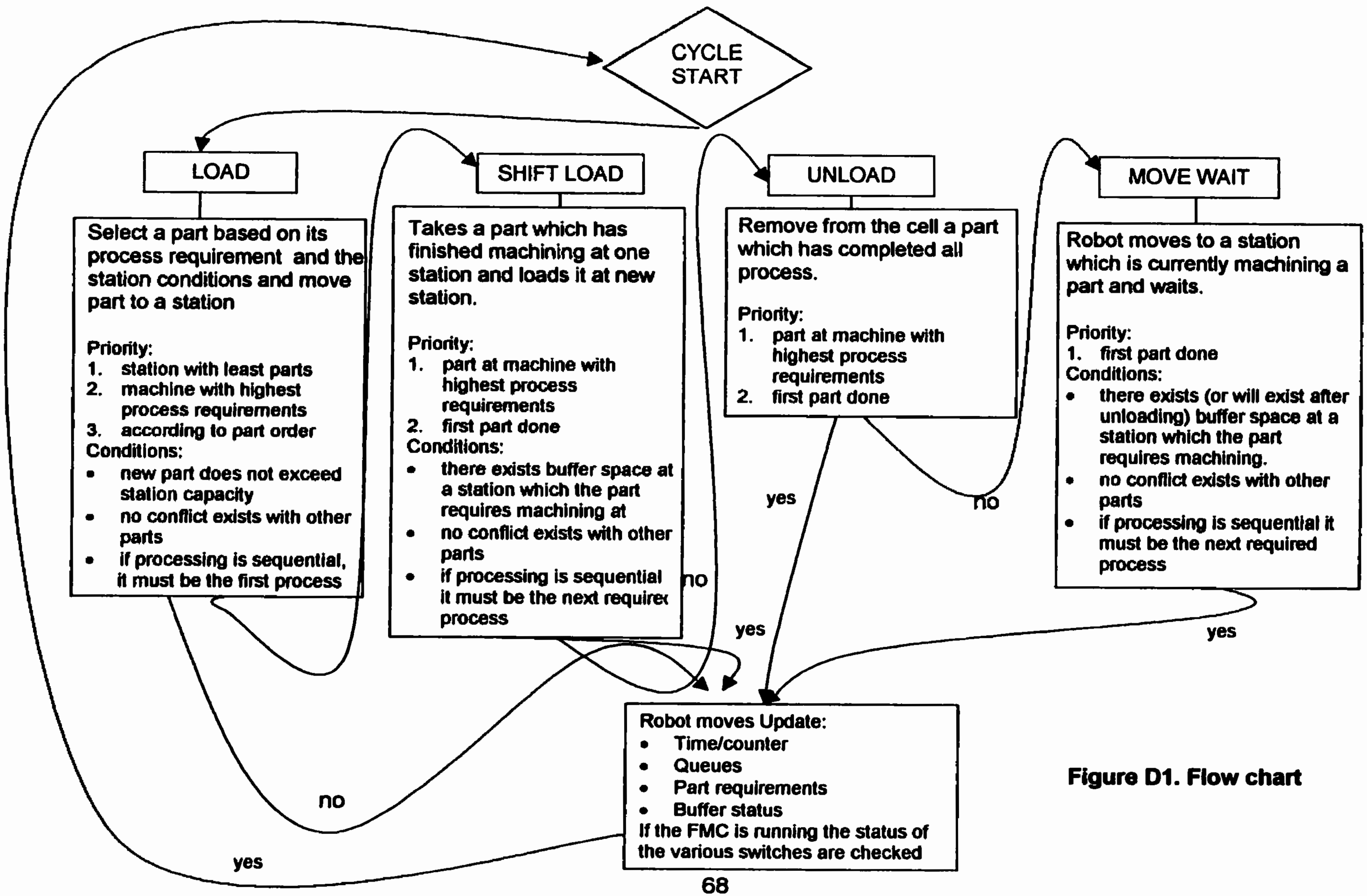


Figure D1. Flow chart

Appendix E .... Data for Unload Versus Load Comparison

Condition: Two Buffers, 15 Parts				
machining time	load nonsequential	unload nonsequential	load sequentially	unload sequentially
5	405	425	425	430
10	415	445	425	445
15	415	440	420	420
20	410	450	445	410
25	425	455	455	440
30	460	465	495	465
35	475	455	520	455
40	495	480	565	530
45	530	510	610	605
50	570	565	655	580
55	615	600	710	610
60	660	645	765	710
65	705	695	820	775
70	755	745	875	825
75	805	795	930	875
80	855	845	990	930
85	905	895	1050	985
90	955	945	1110	1040
95	1005	995	1170	1095
100	1055	1045	1230	1150

Table E1 Load Versus Unload

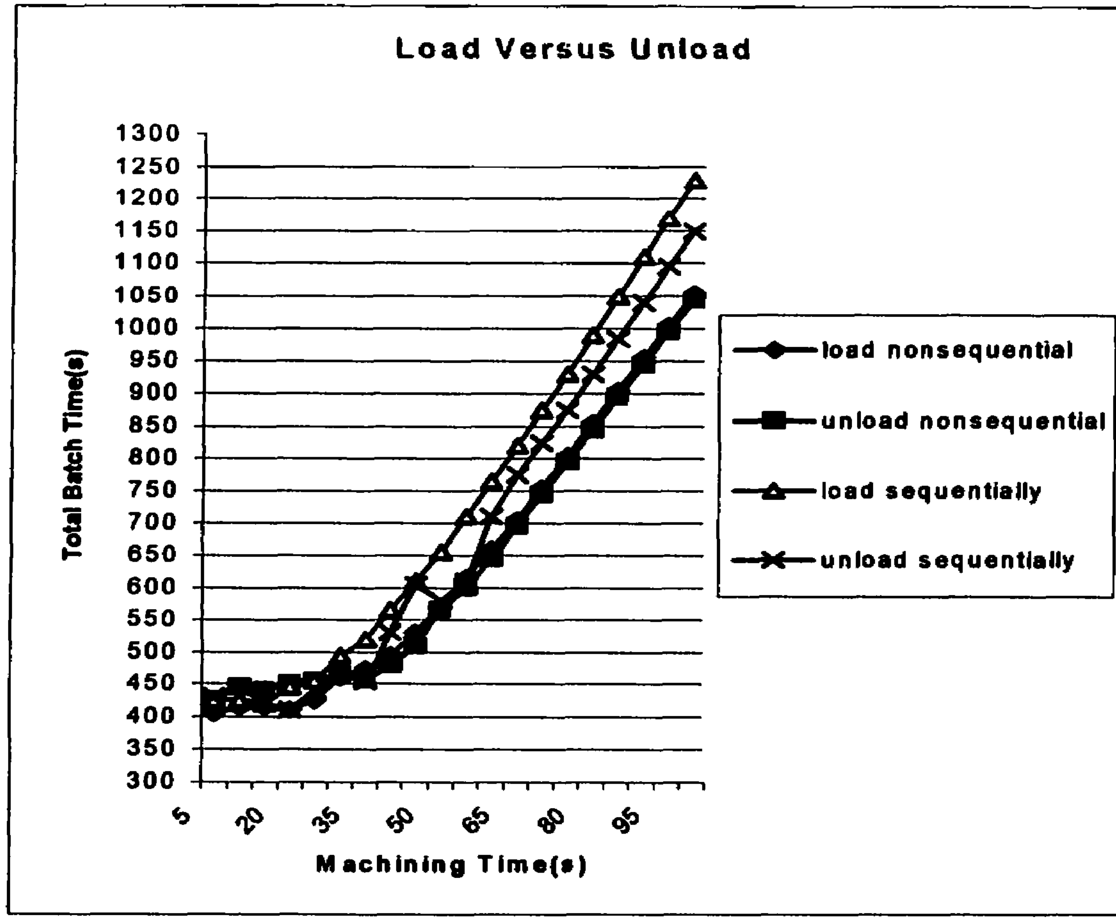


Figure E1 Load versus unload

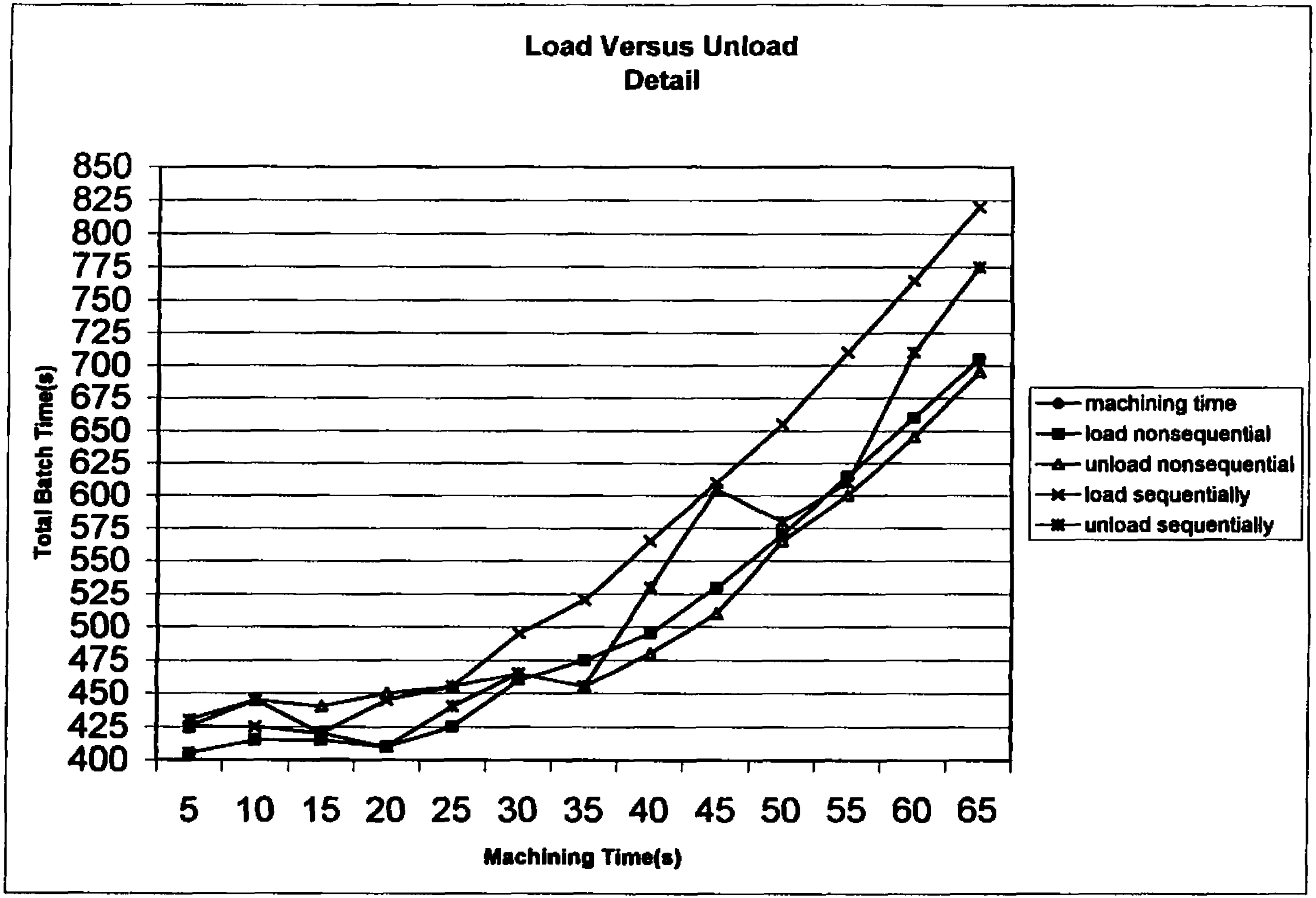


Figure E2 Load versus unload detail

Appendix F .... Data for Buffer Analysis

This file contains 4 worksheets

The objective of this study, as explained in section 5.3, is to examine the effects of buffers on the batch throughput time.

The four experiments conducted tested the batch throughput time, with incrementing machining time from 5-100 seconds with 0,1,2,3,4,5,6 buffers.

The four different scenarios tested where; unloading priority sequential processing, loading priority sequential processing, unloading priority nonsequential processing, and loading priority nonsequential processing.

Machining Time(s)			Unload Priority Sequential Processing				
	buffers=0	buffers=1	buffers=2	buffers=3	buffers=4	buffers=5	buffers=6
5	445	430	430	430	430	430	430
10	450	445	445	445	445	445	445
15	445	420	420	420	420	420	420
20	470	410	410	410	410	410	410
25	525	440	440	440	440	440	440
30	570	490	465	465	465	465	465
35	615	470	455	455	455	455	455
40	675	510	530	500	500	500	500
45	735	550	605	505	505	505	505
50	795	605	580	560	560	560	560
55	855	655	610	590	595	595	595
60	915	705	710	730	670	650	650
65	975	760	775	765	720	700	700
70	1035	815	825	810	760	740	740
75	1095	870	875	870	800	795	795
80	1155	925	930	920	840	825	825
85	1215	980	985	975	890	875	875
90	1275	1035	1040	1040	950	930	930
95	1335	1090	1095	1105	995	980	980
100	1395	1145	1150	1155	1040	1025	1025

Table F1 Unload Priority Sequential Processing Data

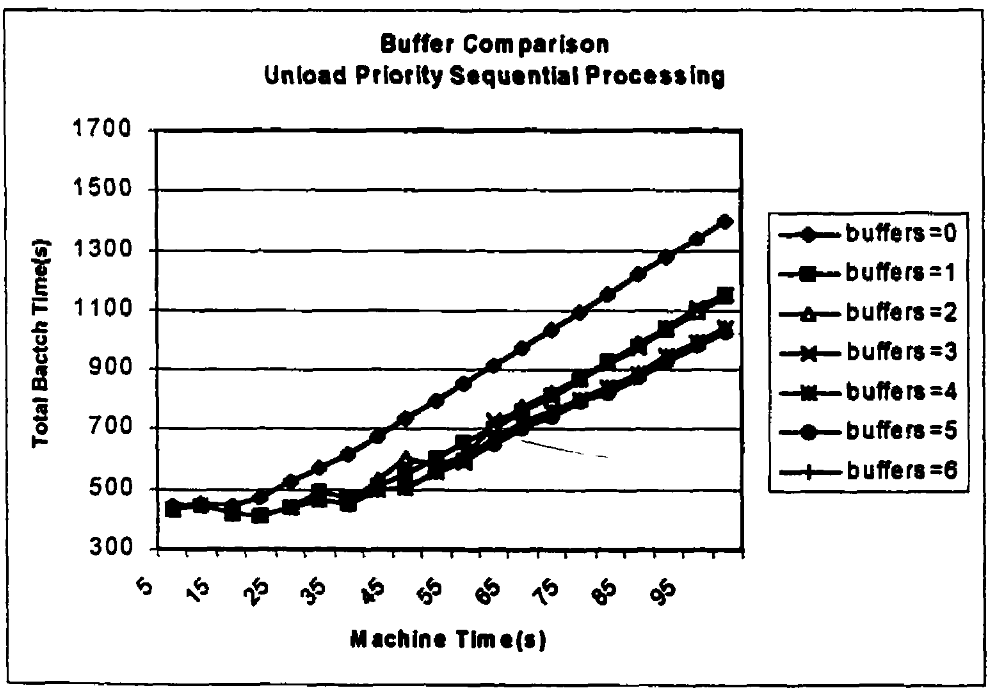


Figure F1 Unload Priority Sequential Processing

Machining Time (s)	Load Priority Sequential Processing						
	buffers=0	buffers=1	buffers=2	buffers=3	buffers=4	buffers=5	buffers=6
5	445	435	425	425	425	425	425
10	445	435	425	425	420	415	415
15	455	435	420	430	405	400	400
20	515	445	445	445	445	375	375
25	575	450	455	450	445	395	395
30	645	495	495	495	465	435	435
35	715	525	520	515	490	480	480
40	785	555	565	545	535	525	525
45	855	615	610	590	580	570	570
50	925	680	655	635	625	615	615
55	995	745	710	685	670	660	660
60	1065	810	765	735	715	705	705
65	1135	875	820	785	760	750	750
70	1205	940	875	835	805	795	795
75	1275	1005	930	885	850	840	840
80	1345	1070	990	935	895	885	885
85	1415	1135	1050	985	940	930	930
90	1485	1200	1110	1035	985	975	975
95	1555	1265	1170	1085	1030	1020	1020
100	1625	1330	1230	1135	1075	1065	1065

Table F2 Load Priority Sequential Processing Data

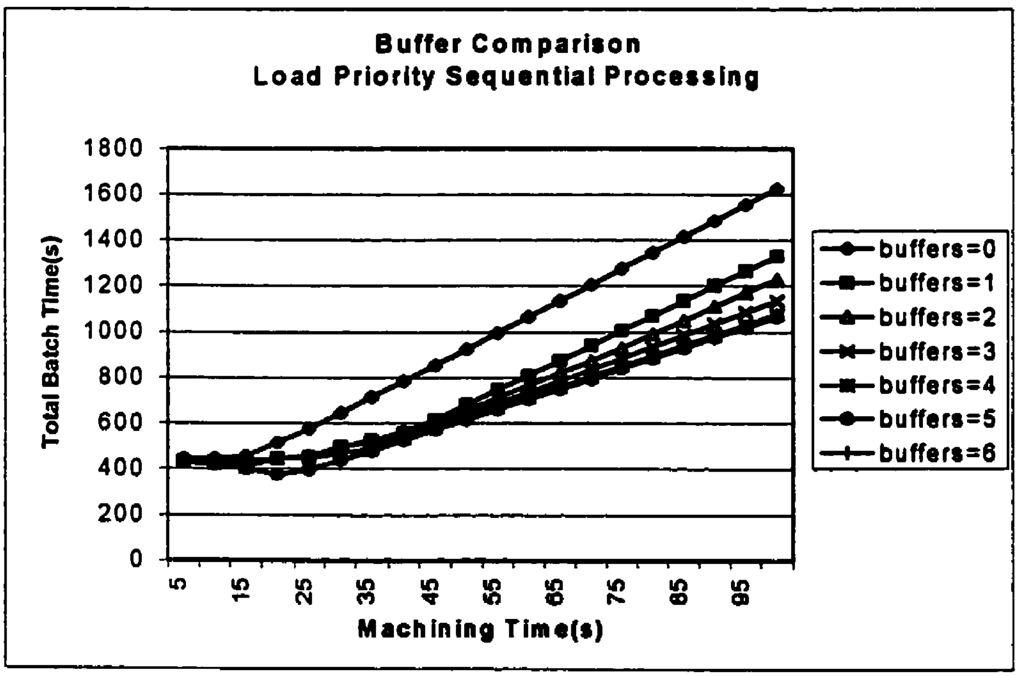


Table F2 Load Priority Sequential Processing

Machine	Unload Priority Nonsequential Processing						
Time (s)	buffers=0	buffers=1	buffers=2	buffers=3	buffers=4	buffers=5	buffers=6
5	445	425	425	425	425	425	425
10	450	445	445	445	445	445	445
15	445	440	440	440	440	440	440
20	465	450	450	450	450	450	450
25	515	455	455	455	455	455	455
30	565	460	465	465	465	465	465
35	675	480	455	455	455	455	455
40	740	485	480	475	475	475	475
45	805	575	510	515	515	515	515
50	870	625	565	560	560	560	560
55	935	675	600	595	600	600	600
60	1000	730	645	635	645	645	645
65	1065	785	695	690	685	690	690
70	1130	840	745	735	735	735	735
75	1195	895	795	790	780	780	780
80	1260	950	845	840	825	825	825
85	1325	1005	895	890	880	880	880
90	1390	1060	945	940	925	925	925
95	1455	1115	995	990	980	980	980
100	1520	1170	1045	1040	1030	1025	1025

Table F3 Unload Priority Non-sequential Processing Data

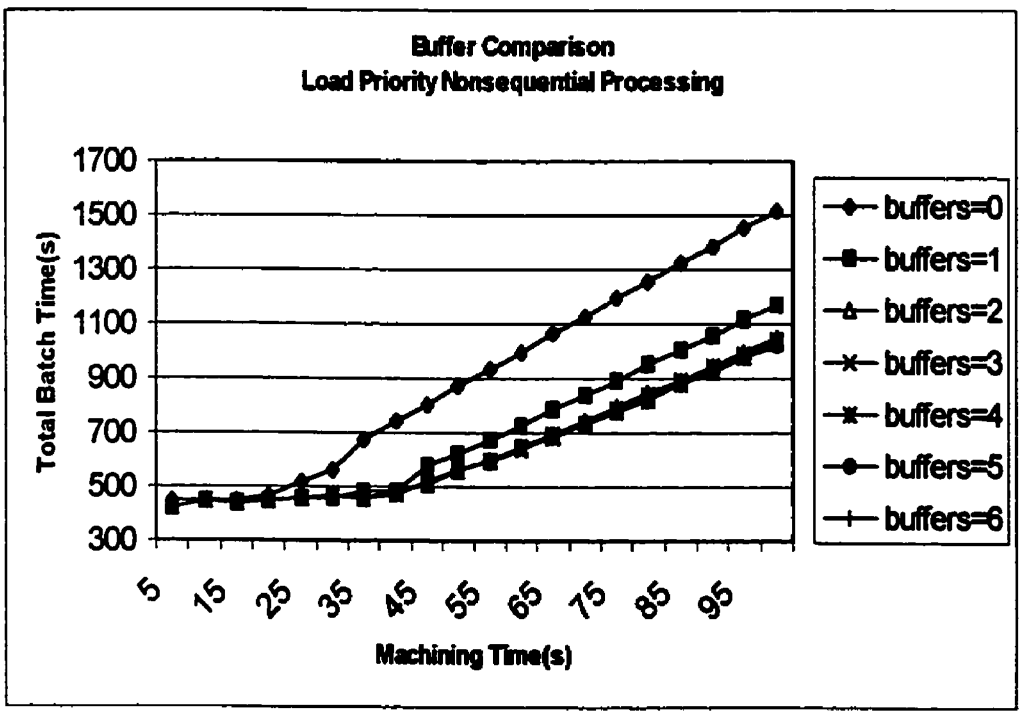


Figure F3 Unload Priority Non-sequential Processing

Machining Time	Load Priority Nonsequential Processing						
	buffers=0	buffers=1	buffers=2	buffers=3	buffers=4	buffers=5	buffers=6
5	445	405	405	410	420	420	420
10	445	405	415	410	440	425	425
15	455	410	415	430	435	415	415
20	510	450	410	430	435	435	435
25	565	465	425	430	435	435	435
30	635	495	460	445	440	450	450
35	705	550	475	450	470	470	470
40	775	610	495	510	495	525	525
45	845	670	530	530	535	555	555
50	915	735	570	575	575	585	585
55	985	800	615	615	620	630	630
60	1055	865	680	655	665	675	675
65	1125	930	705	700	710	720	720
70	1195	995	755	745	755	765	765
75	1265	1060	805	790	800	810	810
80	1335	1125	855	840	845	855	855
85	1405	1190	905	890	890	900	900
90	1475	1255	955	940	935	945	945
95	1545	1320	1005	990	980	990	990
100	1615	1385	1055	1040	1030	1035	1035

Table F4 Load Priority Non-sequential Processing Data

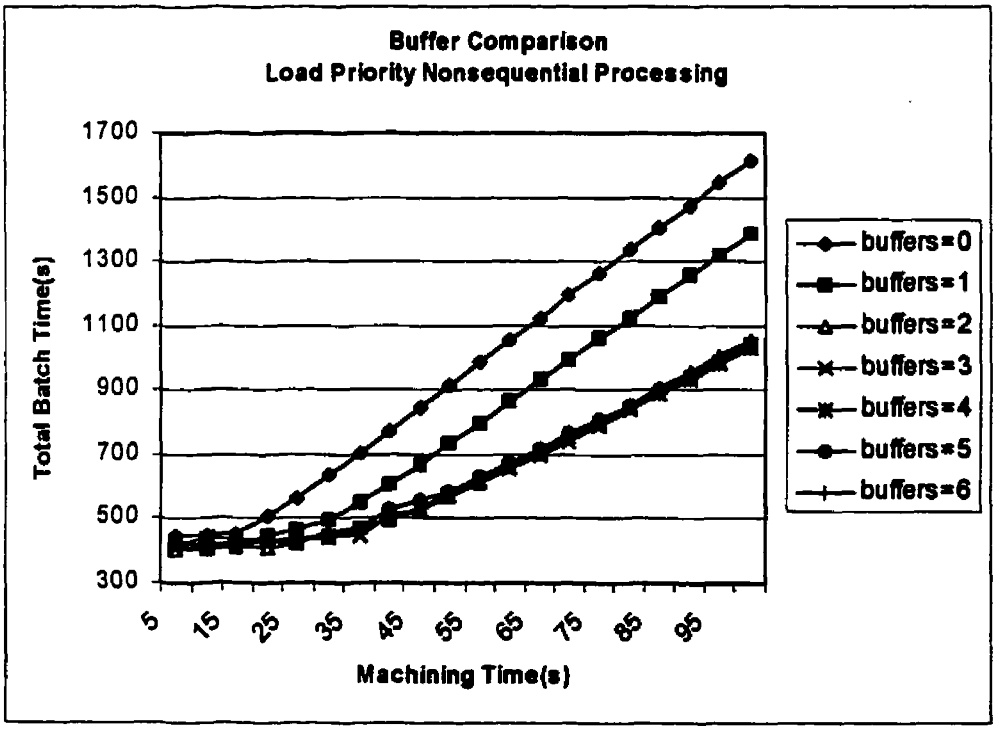


Figure F4 Load Priority Non-sequential Processing

Appendix G .... Data for sequential vs nonsequential and load sequence study

This file contains 9 worksheets
Batch order 3.1,3.2,3.3,3.4 refer to the batch order specified in section 5.4.

Four different cases were run using 24 different part loading sequences under the sequential(seq) and nonsequential (ns) part processing conditions using prioritizing unloading only.

The data is summarized in the worksheet-titled Summary of Data for Different Load Sequences.

The day of the experiment was: 29 9 2000 The time of the experiment was: 10:51:17.23 **Batch order 3.1 non-sequential data**

heuristic used for robot move prioritization 2 (unload)

parts were loaded sequentailly/nonsequentially 0/1 1

station 1	processing time	90	buffers	2
station 2	processing time	90	buffers	2
station 3	processing time	90	buffers	2
station 4	processing time	90	buffers	2

10	Part type 1	needed to be processed on the following stations	1	1	0	0	0
10	Part type 2	needed to be processed on the following stations	0	1	1	0	0
10	Part type 3	needed to be processed on the following stations	0	0	1	1	0
10	Part type 4	needed to be processed on the following stations	1	1	0	1	0

sequence number	part order	station utilization idle time free time st#1 utiliz	st#1 busy	st#1 free	st#2 utiliz	st#2 busy	st#2 free	st#3 utiliz	st#3 busy	st#3 free	st#4 utiliz	st#4 busy	st#4 free	part 1 total residence time/part	part 2	part 3	part 4	total run time
1	1234000	1800	145	765	2700	0	10	1800	125	785	1800	140	770	382	372	466	498.5	2710
2	1243000	1800	160	765	2700	0	25	1800	85	840	1800	80	845	416	362	455.5	468.5	2725
3	1324000	1800	55	855	2700	0	10	1800	70	840	1800	75	835	403.5	367	440	467.5	2710
4	1342000	1800	75	835	2700	0	10	1800	145	765	1800	75	835	403.5	372	440	472	2710
5	1423000	1800	155	770	2700	0	25	1800	80	845	1800	75	850	416	364	455.5	451.5	2725
6	1432000	1800	225	685	2700	0	10	1800	70	840	1800	55	855	443.5	369	434.5	473.5	2710
7	2134000	1800	205	705	2700	0	10	1800	120	790	1800	120	790	408	359	405	474.5	2710
8	2143000	1800	220	705	2700	0	25	1800	55	870	1800	65	860	431.5	355	407	466	2725
9	2314000	1800	205	705	2700	0	10	1800	120	790	1800	120	790	408	359	405	474.5	2710
10	2341000	1800	165	805	2700	60	10	1800	115	855	1800	115	855	415	380	400	524.5	2770
11	2413000	1800	260	805	2700	140	25	1800	35	1030	1800	75	990	423.5	357	379.5	537	2865
12	2431000	1800	260	805	2700	140	25	1800	35	1030	1800	75	990	423.5	357	379.5	537	2865
13	3124000	1800	55	855	2700	0	10	1800	70	840	1800	75	835	403.5	367	440	467.5	2710
14	3142000	1800	75	835	2700	0	10	1800	145	765	1800	75	835	403.5	372	440	472	2710
15	3214000	1800	315	605	2700	0	20	1800	60	860	1800	135	785	435	346	413.5	489	2720
16	3241000	1800	125	855	2700	70	10	1800	55	925	1800	105	875	385	350	391.5	522	2780
17	3412000	1800	145	765	2700	0	10	1800	200	710	1800	210	700	396.5	386	435	506	2710
18	3421000	1800	405	510	2700	5	10	1800	40	875	1800	55	860	495.5	381	343.5	535.5	2715
19	4123000	1800	55	930	2700	60	25	1800	85	900	1800	30	955	351	373	407.5	472	2785
20	4132000	1800	300	610	2700	0	10	1800	100	810	1800	40	870	463	370	424.5	460.5	2710
21	4213000	1800	55	940	2700	70	25	1800	90	905	1800	30	965	362	353	392.5	479	2795
22	4231000	1800	55	940	2700	70	25	1800	90	905	1800	30	965	362	353	392.5	479	2795
23	4312000	1800	300	610	2700	0	10	1800	100	810	1800	40	870	463	370	424.5	460.5	2710
24	4321000	1800	60	860	2700	10	10	1800	55	865	1800	90	830	364	366	430	469.5	2720

Table G1 Batch order 3.1 non-sequential data

The day of the experiment was: 29 9 2000 The time of the experiment was: 11:01:33.77 **Batch order 3.2 non-sequential data**

heuristic used for robot move prioritization 2 (unload)

parts were loaded sequentailly/nonsequentially 0/1 1

station 1	processing time	90 buffers	2
station 2	processing time	90 buffers	2
station 3	processing time	90 buffers	2
station 4	processing time	90 buffers	2

10 Part type 1	needed to be processed on the following stations	1	1	0	0	0
10 Part type 2	needed to be processed on the following stations	0	1	1	0	0
10 Part type 3	needed to be processed on the following stations	0	0	1	1	0
10 Part type 4	needed to be processed on the following stations	1	1	0	1	0

sequence number	part order	station utilization idle time free time st#1 utiliz	st#1 busy	st#1 free	st#2 utiliz	st#2 busy	st#2 free	st#3 utiliz	st#3 busy	st#3 free	st#4 utiliz	st#4 busy	st#4 free	part 1 total residence time/part	part 2	part 3	part 4	total run time
1	1234000	1800	145	765	2700	0	10	1800	125	785	1800	140	770	382	371.5	466	498.5	2710
2	1243000	1800	160	765	2700	0	25	1800	65	840	1800	80	845	416	361.5	455.5	468.5	2725
3	1324000	1800	55	855	2700	0	10	1800	70	840	1800	75	835	403.5	367	440	467.5	2710
4	1342000	1800	75	835	2700	0	10	1800	145	765	1800	75	835	403.5	372	440	472	2710
5	1423000	1800	155	770	2700	0	25	1800	80	845	1800	75	850	416	363.5	455.5	451.5	2725
6	1432000	1800	225	685	2700	0	10	1800	70	840	1800	55	855	443.5	368.5	434.5	473.5	2710
7	2134000	1800	205	705	2700	0	10	1800	120	790	1800	120	790	408	358.5	405	474.5	2710
8	2143000	1800	220	705	2700	0	25	1800	55	870	1800	65	860	431.5	355	407	466	2725
9	2314000	1800	205	705	2700	0	10	1800	120	790	1800	120	790	408	358.5	405	474.5	2710
10	2341000	1800	165	805	2700	60	10	1800	115	855	1800	115	855	415	379.5	400	524.5	2770
11	2413000	1800	260	805	2700	140	25	1800	35	1030	1800	75	990	423.5	357	379.5	537	2865
12	2431000	1800	260	805	2700	140	25	1800	35	1030	1800	75	990	423.5	357	379.5	537	2865
13	3124000	1800	55	855	2700	0	10	1800	70	840	1800	75	835	403.5	367	440	467.5	2710
14	3142000	1800	75	835	2700	0	10	1800	145	765	1800	75	835	403.5	372	440	472	2710
15	3214000	1800	315	605	2700	0	20	1800	60	860	1800	135	785	435	346	413.5	489	2720
16	3241000	1800	125	855	2700	70	10	1800	55	925	1800	105	875	385	349.5	391.5	522	2780
17	3412000	1800	145	765	2700	0	10	1800	200	710	1800	210	700	396.5	385.5	435	506	2710
18	3421000	1800	405	510	2700	5	10	1800	40	875	1800	55	860	495.5	380.5	343.5	535.5	2715
19	4123000	1800	55	930	2700	60	25	1800	85	900	1800	30	955	351	373	407.5	472	2785
20	4132000	1800	300	610	2700	0	10	1800	100	810	1800	40	870	463	369.5	424.5	460.5	2710
21	4213000	1800	55	940	2700	70	25	1800	90	905	1800	30	965	362	353	392.5	479	2795
22	4231000	1800	55	940	2700	70	25	1800	90	905	1800	30	965	362	353	392.5	479	2795
23	4312000	1800	300	610	2700	0	10	1800	100	810	1800	40	870	463	369.5	424.5	460.5	2710
24	4321000	1800	60	860	2700	10	10	1800	55	865	1800	90	830	364	365.5	430	469.5	2720

Table G3 Batch order 3.2 non-sequential data

The day of the experiment was: 29 9 2000 The time of the experiment was: 11:11:50.09 **Batch order 3.3 non-sequential data**

heuristic used for robot move prioritization 2 (unload)

parts were loaded sequentailly/nonsequentially 0/1 1

station 1	processing time	90	buffers	2
station 2	processing time	90	buffers	2
station 3	processing time	90	buffers	2
station 4	processing time	90	buffers	2

10	Part type 1	needed to be processed on the following stations	1	1	0	0	0
10	Part type 2	needed to be processed on the following stations	0	1	1	0	0
10	Part type 3	needed to be processed on the following stations	0	0	1	1	0
10	Part type 4	needed to be processed on the following stations	1	0	1	1	0

sequence number	part order	station utilization idle time free time st#1 utiliz	st#1 busy	st#1 free	st#2 utiliz	st#2 busy	st#2 free	st#3 utiliz	st#3 busy	st#3 free	st#4 utiliz	st#4 busy	st#4 free	part 1 total residence time/part	part 2	part 3	part 4	total run time
1	1234000	1800	90	900	1800	45	945	2700	70	20	1800	190	800	396	336	461	462.5	2790
2	1243000	1800	45	960	1800	85	920	2700	95	10	1800	50	955	377	357	367	508.5	2805
3	1324000	1800	75	895	1800	35	935	2700	60	10	1800	265	705	371.5	372	503.5	463.5	2770
4	1342000	1800	75	895	1800	35	935	2700	60	10	1800	265	705	371.5	372	503.5	463.5	2770
5	1423000	1800	90	820	1800	60	850	2700	0	10	1800	150	760	439.5	363	461	476	2710
6	1432000	1800	105	890	1800	100	895	2700	85	10	1800	230	765	387.5	393	420	512.5	2795
7	2134000	1800	145	765	1800	60	850	2700	0	10	1800	110	800	437.5	345	356.5	472.5	2710
8	2143000	1800	55	855	1800	110	800	2700	0	10	1800	225	685	436.5	345	430.5	480	2710
9	2314000	1800	145	765	1800	60	850	2700	0	10	1800	110	800	437.5	345	356.5	472.5	2710
10	2341000	1800	95	830	1800	50	875	2700	0	25	1800	225	700	415	337	443.5	467.5	2725
11	2413000	1800	110	815	1800	75	850	2700	0	25	1800	165	760	466.5	366	436.5	479.5	2725
12	2431000	1800	110	815	1800	75	850	2700	0	25	1800	165	760	466.5	366	436.5	479.5	2725
13	3124000	1800	75	895	1800	35	935	2700	60	10	1800	265	705	371.5	372	503.5	463.5	2770
14	3142000	1800	75	895	1800	35	935	2700	60	10	1800	265	705	371.5	372	503.5	463.5	2770
15	3214000	1800	95	830	1800	40	885	2700	15	10	1800	140	785	389.5	352	446	461.5	2725
16	3241000	1800	100	850	1800	65	885	2700	15	35	1800	150	800	459	365	430.5	476	2750
17	3412000	1800	75	910	1800	85	900	2700	65	20	1800	200	785	421.5	376	452	480.5	2785
18	3421000	1800	100	850	1800	65	885	2700	15	35	1800	150	800	459	365	430.5	476	2750
19	4123000	1800	120	840	1800	35	925	2700	50	10	1800	75	885	387.5	367	369.5	503.5	2760
20	4132000	1800	145	765	1800	90	820	2700	0	10	1800	150	760	420.5	362	437	501.5	2710
21	4213000	1800	135	790	1800	115	810	2700	0	25	1800	65	860	401.5	342	360	501.5	2725
22	4231000	1800	135	790	1800	115	810	2700	0	25	1800	65	860	401.5	342	360	501.5	2725
23	4312000	1800	145	765	1800	90	820	2700	0	10	1800	150	760	420.5	362	437	501.5	2710
24	4321000	1800	65	860	1800	115	810	2700	0	25	1800	330	595	413.5	377	448	529	2725

Table G4 Batch order 3.3 non-sequential data

The day of the experiment was: 29 9 2000 The time of the experiment was: 11:27:25.20 **Batch order 3.4 non-sequential data**

heuristic used for robot move prioritization 2 (unload)

parts were loaded sequentailly/nonsequentially 0/1 1

station 1	processing time	90	buffers	2
station 2	processing time	90	buffers	2
station 3	processing time	90	buffers	2
station 4	processing time	90	buffers	2

10	Part type 1	needed to be processed on the following stations	1	1	0	0	0
10	Part type 2	needed to be processed on the following stations	0	1	0	1	0
10	Part type 3	needed to be processed on the following stations	0	0	1	1	0
10	Part type 4	needed to be processed on the following stations	1	0	1	1	0

sequence number	part order	*station utilization idle time free time* st#1 utiliz	st#1 busy	st#1 free	st#2 utiliz	st#2 busy	st#2 free	st#3 utiliz	st#3 busy	st#3 free	st#4 utiliz	st#4 busy	st#4 free	part 1 total residence time/part	part 2	part 3	part 4	total *run* time
1	1234000	1800	90	900	1800	45	945	1800	190	800	2700	70	20	396	335.5	461	462.5	2790
2	1243000	1800	45	960	1800	85	920	1800	50	955	2700	95	10	377	356.5	367	508.5	2805
3	1324000	1800	75	895	1800	35	935	1800	265	705	2700	60	10	371.5	372	503.5	463.5	2770
4	1342000	1800	75	895	1800	35	935	1800	265	705	2700	60	10	371.5	372	503.5	463.5	2770
5	1423000	1800	90	820	1800	60	850	1800	150	760	2700	0	10	439.5	362.5	461	476	2710
6	1432000	1800	105	890	1800	100	895	1800	230	765	2700	85	10	387.5	393	420	512.5	2795
7	2134000	1800	145	765	1800	60	850	1800	110	800	2700	0	10	437.5	344.5	356.5	472.5	2710
8	2143000	1800	55	855	1800	110	800	1800	225	685	2700	0	10	436.5	345	430.5	480	2710
9	2314000	1800	145	765	1800	60	850	1800	110	800	2700	0	10	437.5	344.5	356.5	472.5	2710
10	2341000	1800	95	830	1800	50	875	1800	225	700	2700	0	25	415	337	443.5	467.5	2725
11	2413000	1800	110	815	1800	75	850	1800	165	760	2700	0	25	466.5	366	436.5	479.5	2725
12	2431000	1800	110	815	1800	75	850	1800	165	760	2700	0	25	466.5	366	436.5	479.5	2725
13	3124000	1800	75	895	1800	35	935	1800	265	705	2700	60	10	371.5	372	503.5	463.5	2770
14	3142000	1800	75	895	1800	35	935	1800	265	705	2700	60	10	371.5	372	503.5	463.5	2770
15	3214000	1800	95	830	1800	40	885	1800	140	785	2700	15	10	389.5	351.5	446	461.5	2725
16	3241000	1800	100	850	1800	65	885	1800	150	800	2700	15	35	459	364.5	430.5	476	2750
17	3412000	1800	75	910	1800	85	900	1800	200	785	2700	65	20	421.5	375.5	452	480.5	2785
18	3421000	1800	100	850	1800	65	885	1800	150	800	2700	15	35	459	364.5	430.5	476	2750
19	4123000	1800	120	840	1800	35	925	1800	75	885	2700	50	10	387.5	366.5	369.5	503.5	2760
20	4132000	1800	145	765	1800	90	820	1800	150	760	2700	0	10	420.5	362	437	501.5	2710
21	4213000	1800	135	790	1800	115	810	1800	65	860	2700	0	25	401.5	342	360	501.5	2725
22	4231000	1800	135	790	1800	115	810	1800	65	860	2700	0	25	401.5	342	360	501.5	2725
23	4312000	1800	145	765	1800	90	820	1800	150	760	2700	0	10	420.5	362	437	501.5	2710
24	4321000	1800	65	860	1800	115	810	1800	330	595	2700	0	25	413.5	377	448	529	2725

Table G4 Batch order 3.4 non-sequential data

The day of the experiment was: 29 9 2000 The time of the experiment was: 11:37:34.04 **Batch order 3.1 sequential data**

heuristic used for robot move prioritization 2 (unload)

parts were loaded sequentailly/nonsequentially 0/1 0

station 1	processing time	90 buffers	2
station 2	processing time	90 buffers	2
station 3	processing time	90 buffers	2
station 4	processing time	90 buffers	2

10 Part type 1	needed to be processed on the following stations	1	1	0	0	0
10 Part type 2	needed to be processed on the following stations	1	0	1	0	0
10 Part type 3	needed to be processed on the following stations	0	0	1	1	0
10 Part type 4	needed to be processed on the following stations	1	1	0	1	0

		station utilization idle time free time												part 1	part 2	part 3	part 4
sequence number	part order	st#1 utiliz	st#1 busy	st#1 free	st#2 utiliz	st#2 busy	st#2 free	st#3 utiliz	st#3 busy	st#3 free	st#4 utiliz	st#4 busy	st#4 free	total residence time/part			
1	1234000	2700	0	200	1800	5	1095	1800	20	1080	1800	5	1095	333.5	360	338.5	452
2	1243000	2700	0	200	1800	5	1095	1800	20	1080	1800	5	1095	333.5	360	338.5	452
3	1324000	2700	0	200	1800	5	1095	1800	20	1080	1800	5	1095	333.5	360	338.5	452
4	1342000	2700	0	105	1800	5	1000	1800	15	990	1800	20	985	333	357.5	338.5	454.5
5	1423000	2700	0	105	1800	5	1000	1800	15	990	1800	20	985	333	357.5	338.5	454.5
6	1432000	2700	0	105	1800	5	1000	1800	15	990	1800	20	985	333	357.5	338.5	454.5
7	2134000	2700	570	200	1800	0	1670	1800	0	1670	1800	35	1635	390	638.5	336.5	452.5
8	2143000	2700	570	200	1800	0	1670	1800	0	1670	1800	35	1635	390	638.5	336.5	452.5
9	2314000	2700	570	200	1800	0	1670	1800	0	1670	1800	35	1635	390	638.5	336.5	452.5
10	2341000	2700	570	105	1800	0	1575	1800	0	1575	1800	35	1540	360	638.5	346	627
11	2413000	2700	570	105	1800	0	1575	1800	0	1575	1800	35	1540	360	638.5	346	627
12	2431000	2700	570	105	1800	0	1575	1800	0	1575	1800	35	1540	360	638.5	346	627
13	3124000	2700	0	200	1800	5	1095	1800	20	1080	1800	5	1095	333.5	360	338.5	452
14	3142000	2700	0	105	1800	5	1000	1800	15	990	1800	20	985	333	357.5	338.5	454.5
15	3214000	2700	570	200	1800	0	1670	1800	0	1670	1800	35	1635	390	638.5	336.5	452.5
16	3241000	2700	570	105	1800	0	1575	1800	0	1575	1800	35	1540	360	638.5	346	627
17	3412000	2700	500	105	1800	665	840	1800	660	845	1800	5	1500	665	362	648.5	855
18	3421000	2700	420	105	1800	665	760	1800	670	755	1800	5	1420	362	644	648.5	855
19	4123000	2700	500	105	1800	665	840	1800	660	845	1800	5	1500	665	362	648.5	855
20	4132000	2700	500	105	1800	665	840	1800	660	845	1800	5	1500	665	362	648.5	855
21	4213000	2700	420	105	1800	665	760	1800	670	755	1800	5	1420	362	644	648.5	855
22	4231000	2700	420	105	1800	665	760	1800	670	755	1800	5	1420	362	644	648.5	855
23	4312000	2700	500	105	1800	665	840	1800	660	845	1800	5	1500	665	362	648.5	855
24	4321000	2700	420	105	1800	665	760	1800	670	755	1800	5	1420	362	644	648.5	855

Table G5 Batch order 3.1 sequential data

The day of the experiment was: 29 9 2000 The time of the experiment was: 11:49:52.68 **Batch order 3.2 sequential data**

heuristic used for robot move prioritization 2 (unload)

parts were loaded sequentailly/nonsequentially 0/1

station 1	processing time	90	Buffers	2
station 2	processing time	90	Buffers	2
station 3	processing time	90	Buffers	2
station 4	processing time	90	Buffers	2

10 Part type 1	needed to be processed on the following stations	1	1	0	0	0
10 Part type 2	needed to be processed on the following stations	0	1	1	0	0
10 Part type 3	needed to be processed on the following stations	0	0	1	1	0
10 Part type 4	needed to be processed on the following stations	1	1	0	1	0

sequence number	part order	station utilization idle time free time st#1 utiliz	st#1 busy	st#1 free	st#2 utiliz	st#2 busy	st#2 free	st#3 utiliz	st#3 busy	st#3 free	st#4 utiliz	st#4 busy	st#4 free	part 1 total residence time/part	part 2	part 3	part 4	total run time
3	1324000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
4	1342000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
5	1423000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
6	1432000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
7	2134000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
8	2143000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
9	2314000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
10	2341000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
11	2413000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
12	2431000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
13	3124000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
14	3142000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
15	3214000	1800	865	365	2700	225	105	1800	30	1200	1800	30	1200	678	426.5	332.5	736.5	3030
16	3241000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
17	3412000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
18	3421000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
19	4123000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
20	4132000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
21	4213000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
22	4231000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
23	4312000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130
24	4321000	1800	1060	270	2700	420	10	1800	155	1175	1800	10	1320	675	469.5	436.5	956.5	3130

Table G6 Batch order 3.2 sequential data

The day of the experiment was: 29 9 2000 | The time of the experiment was: 12:41:20.21 | **Batch order 3.3 sequential data**

heuristic used for robot move prioritization 2 (unload)

parts were loaded sequentailly/nonsequentailly 0/1 0

station 1	processing time	90 buffers	2
station 2	processing time	90 buffers	2
station 3	processing time	90 buffers	2
station 4	processing time	90 buffers	2

10	Part type 1	needed to be processed on the following stations	1	1	0	0	0
10	Part type 2	needed to be processed on the following stations	0	1	1	0	0
10	Part type 3	needed to be processed on the following stations	0	0	1	1	0
10	Part type 4	needed to be processed on the following stations	1	0	1	1	0

		station utilization idle time free time												part 1	part 2	part 3	part 4	
sequence number	part order	st#1 utiliz	st#1 busy	st#1 free	st#2 utiliz	st#2 busy	st#2 free	st#3 utiliz	st#3 busy	st#3 free	st#4 utiliz	st#4 busy	st#4 free	total residence time/part				total run time
3	1324000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
4	1342000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
5	1423000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
6	1432000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
7	2134000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
8	2143000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
9	2314000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
10	2341000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
11	2413000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
12	2431000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
13	3124000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
14	3142000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
15	3214000	1800	870	365	1800	490	745	2700	10	325	1800	35	1200	725.5	525	333	641.5	3035
16	3241000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
17	3412000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
18	3421000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
19	4123000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
20	4132000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
21	4213000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
22	4231000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
23	4312000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845
24	4321000	1800	930	115	1800	985	60	2700	0	145	1800	50	995	396	764.5	330	835.5	2845

Table G7 Batch order 3.3 sequential data

The day of the experiment was: 29 9 2000 The time of the experiment was: 13:55:49.44 **Batch order 3.4 sequential data**

heuristic used for robot move prioritization 2 (unload)

parts were loaded sequentailly/nonsequentailly 0/1 0

station 1	processing time	90	Buffers 2
station 2	processing time	90	Buffers 2
station 3	processing time	90	Buffers 2
station 4	processing time	90	Buffers 2

10	Part type 1	needed to be processed on the following stations	1	1	0	0	0
10	Part type 2	needed to be processed on the following stations	0	1	0	1	0
10	Part typə 3	needed to be processed on the following stations	0	0	1	1	0
10	Part type 4	needed to be processed on the following stations	1	0	1	1	0

		station utilization idle time free time												part 1	part 2	part 3	part 4	
sequence number	part order	st#1 utiliz	st#1 busy	st#1 free	st#2 utiliz	st#2 busy	st#2 free	st#3 utiliz	st#3 busy	st#3 free	st#4 utiliz	st#4 busy	st#4 free	total residence time/part				total run time
3	1324000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
4	1342000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
5	1423000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
6	1432000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
7	2134000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
8	2143000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
9	2314000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
10	2341000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
11	2413000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
12	2431000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
13	3124000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
14	3142000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
15	3214000	1800	605	530	1800	405	730	1800	445	690	2700	15	220	646	512	532.5	792	2935
16	3241000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
17	3412000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
18	3421000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
19	4123000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
20	4132000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
21	4213000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
22	4231000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
23	4312000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815
24	4321000	1800	765	250	1800	645	370	1800	725	290	2700	0	115	392	640.5	500	1163	2815

Table G8 Batch order 3.4 sequential data

Summary of Data for Different Load Sequence

	Non-sequential					sequential			
Sequence	case 1	case 2	case 3	case 4	Sequence	case 1	case 2	case 3	case 4
1234000	2710	2710	2790	2790	1234000	2900	3030	3035	2935
1243000	2725	2725	2805	2805	1243000	2900	3030	3035	2935
1324000	2710	2710	2770	2770	1324000	2900	3030	3035	2935
1342000	2710	2710	2770	2770	1342000	2805	3030	3035	2935
1423000	2725	2725	2710	2710	1423000	2805	3030	3035	2935
1432000	2710	2710	2795	2795	1432000	2805	3030	3035	2935
2134000	2710	2710	2710	2710	2134000	3470	3030	3035	2935
2143000	2725	2725	2710	2710	2143000	3470	3030	3035	2935
2314000	2710	2710	2710	2710	2314000	3470	3030	3035	2935
2341000	2770	2770	2725	2725	2341000	3375	3130	2845	2815
2413000	2865	2865	2725	2725	2413000	3375	3130	2845	2815
2431000	2865	2865	2725	2725	2431000	3375	3130	2845	2815
3124000	2710	2710	2770	2770	3124000	2900	3030	3035	2935
3142000	2710	2710	2770	2770	3142000	2805	3030	3035	2935
3214000	2720	2720	2725	2725	3214000	3470	3030	3035	2935
3241000	2780	2780	2750	2750	3241000	3375	3130	2845	2815
3412000	2710	2710	2785	2785	3412000	3305	3130	2845	2815
3421000	2715	2715	2750	2750	3421000	3225	3130	2845	2815
4123000	2785	2785	2760	2760	4123000	3305	3130	2845	2815
4132000	2710	2710	2710	2710	4132000	3305	3130	2845	2815
4213000	2795	2795	2725	2725	4213000	3225	3130	2845	2815
4231000	2795	2795	2725	2725	4231000	3225	3130	2845	2815
4312000	2710	2710	2710	2710	4312000	3305	3130	2845	2815
4321000	2720	2720	2725	2725	4321000	3225	3130	2845	2815
highest	2865	2865	2805	2805	highest	3470	3130	3035	2935
lowest	2710	2710	2710	2710	lowest	2805	3030	2845	2815
deviation	48.35	48.35	31.36	31.36	deviation	249.80	51.08	97.04	61.29
average	2741.46	2741.46	2743.75	2743.75	average	3180	3080	2940	2875

Table G9 Summary of Data for load sequence