FUZZY-BASED METHODOLOGY FOR MULTI-OBJECTIVE SCHEDULING IN A ROBOT-CENTERED FLEXIBLE MANUFACTURING CELL

By

INDIRA MOLINA RESTREPO

A Thesis Submitted to the Faculty of Graduate studies
in Partial Fulfilment of the Requirements for the Degree of

MASTER OF SCIENCE

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

0-612-76818-X

FUZZY-BASED METHODOLOGY FOR MULTI-OBJECTIVE SCHEDULING IN A ROBOT-CENTERED FLEXIBLE MANUFACTURING CELL

BY

INDIRA MOLINA RESTREPO

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

Acknowledgments

I truly value the guidance of my advisor Dr. Balakrishnan. His wisdom, expertise and opportune feedback led to a successful completion of this thesis. I also acknowledge his financial support, and the fellowship awarded by the Faculty of Graduate Studies. Extended thanks to the members of the committee for their feedback. I also express my gratitude to Mr. Ken Tarte for his willingness to provide technical support, and to my fellow graduate students for sharing their experiences, best luck to all of them.

This thesis would have never been possible without my parents and family support. Their prayers and phone calls have always accompanied me. Thanks to those friends that did not care about making expensive long-distance phone calls to support me and remind me a little about home. Finally, but no least, I want to thank Mithun for his unconditional love and support. He made my transition to Canada easier and helped me to get over my homesickness.

Abstract

Scheduling problems in Flexible Manufacturing Cells present varying degrees of difficulty. The objective of the present work is to develop a fuzzy logic based methodology for generating the sequence of flow of parts through the cell by selecting appropriate movements of the robot within the manufacturing cell. The goal is to meet certain production objectives, given a multi-product batch. The production objectives taken into account are maximization of the throughput and machine utilization, as well as, minimization of penalty for tardy jobs and robot travel time.

The fuzzy-based approach uses membership functions to find the contribution of each part type to the objectives according to the specifications of the batch. The evaluation of these contributions generates the sequence of the parts within the cell. Two fuzzy based strategies have been developed: fuzzy-job and fuzzy-machine. These strategies are compared to well known dispatching rules. The effects of using priorities such as loading versus unloading, and sequencing parts in a sequential/non-sequential mode are investigated as well. Custom designed software is created to control the robot moves within the cell. The software has the ability of processing information in simulation (off-line) and in real time (on-line), and is successfully implemented and tested in the experimental cell. In general, the proposed fuzzy-based methodologies especially fuzzy-job show a superior performance compared to traditional dispatching rules. The model can be further expanded or modified to include different or more objectives.

Table of Contents

List of Figures

List of Tables

1.Introduction

1.1 Background

Evolving society and economy have an impact on manufacturing trends. Manufacturing design and processes are undergoing constant changes as a result of individualistic demands from customers, as well as, international competition. The survival of companies may depend on how quickly and how well they can satisfy consumer needs. This has led to industrial systems with high levels of flexibility. One method of achieving this flexibility is through automation and computerization of processes. In addition, it is necessary to improve productivity by reducing production costs and times. Flexible Manufacturing Systems (FMSs) have become popular due to their ability to produce medium volume and medium variety of parts (M-M). In an M-M system, flexibility and production are keys elements [1]. Flexible Manufacturing Cell (FMC) is a subset of FMS.

FMC is an automated system composed of a group of machines served by a material-handling device, where families of parts are processed. Parts are usually classified into families by using the concept of Group Technology (GT). Because of the flexibility, robots are usually employed for the movement of parts between machines as well as loading and unloading. However, that flexibility leaves numerous ways of routing the parts within the cell. Therefore, one of the most important problems encountered in the FMC environment deals with the assignment of given resources to different processes, in order to achieve the best efficiency. FMC scheduling deals with the efficient allocation of the resources for manufacturing products. The objective of scheduling is to find a way to assign and sequence the use of resources; thereby production objectives can be achieved.

Due to the complexity of scheduling in the FMS environment, various researchers have suggested dividing the problem into different levels. One such proposal is the framework provided by Suri and Whitney [2]. It provides three levels of decision problems:

1. upper level management, long term decisions regarding production and economical goals as well as setting policies for part-mix changes and system modification as well as expansion;
2. medium term decisions involving grouping of parts and balancing of workload; and
3. short term decisions including scheduling of work, selection of parts, tool management and reaction to system failures.

The problem investigated in the present work focuses on short-term decisions, specifically on scheduling jobs. The scheduling problem can be classified according to Graves [3] as follows.

1. Requirements generation.
    - In an open shop, production orders are generated by customer request.
    - In a closed shop, production orders are based on stock.
2. Nature of the arrival of parts and their processing times.
    - Input parameters are known and certain for a deterministic model.

- Input parameters are random variables with probabilistic distributions for a stochastic model.

3. Scheduling environment.
   - For a static environment, all jobs to be scheduled are available at the beginning of the scheduling process.
   - For a dynamic situation, the set of jobs to be processed is continuously changing over time.
4. Type of processing environment.
   - For a one stage-one processor problem, all jobs require only one processing step, using one single processing machine.
   - For a one-stage-parallel processor problem, one single processing step is required, but it can be performed on any of the processing machines.
   - In a multistage-flow shop problem, all the jobs require the same processing steps in a strict precedence.
   - In a multistage-job shop problem, each job has its own processing requirements in different sequences.
5. Scheduling criteria.
   - Cost related measures such as set-up costs, production costs, inventory costs, shortage costs and expediting costs are considered.
   - Performance related measures such as machine utilization, flow time, lateness and tardiness are key issues.

The scheduling problem considered in this thesis is an open, deterministic, static and multistage-job shop problem. The scheduling criteria include both cost related measures and performance related measures.

1.2 Objective

The main objective of this project is to develop a computer based intelligent technique to generate a sequence of parts, and control the movement of the robot within a manufacturing cell in order to fulfill certain production objectives, given a multi-product batch. Although standard industrial robots are heavily computerized, they do not possess the intelligence to dynamically alter preprogrammed motions. This thesis relies on an external PC to impart that additional control capability. The production objectives taken into account are:

- Maximize throughput, or, in other words, producing a batch in the shortest possible time. Throughput time refers to the time between the beginning of the batch production and its completion.
- Minimize penalty for tardy jobs. This is probably one of the most important objectives in a production environment. Tardy costs not only relate to tardy penalties, but also significantly to the loss of clients and future sales, as well as rush shipping costs [4].
- Maximize machine utilization. Machines usually vary in cost and energy efficiency. Therefore, it is important to use machine resources efficiently. In a static setup like the one in consideration, machine utilization is proportional to the throughput time,

thus the insertion of this objective is almost irrelevant. However, in a dynamic scheduling problem it would be an important parameter.

- Minimize robot travel time. It is difficult to measure the performance of a certain rule or heuristic in terms of robot travel time. At first sight, one may think that the shorter is the robot idle time, the better. However, this parameter needs to be looked at closely. The robot idle time will increase when the throughput time increases. Therefore, it is logical to assume that, if a dispatching rule gives a shorter throughput time than another, the robot idle time would be shorter as well. However, there are cases when more than one rule performs the best in terms of the throughput time. In such tied cases, if the robot idle times vary, the rule with the best performance is the one that gives the highest robot idle time. This means, decreasing energy costs and increasing robot life span, thus reducing production costs.

In the present scenario, each batch consists of orders placed by diverse customers. Thus, each batch involves multiple products, each with different specifications, but similar processing operations. These specifications depend on the customer's requirements and production costs involved. Customer requirements refer to processing times, due dates and penalties, whereas production costs refer to robot and machine idle costs. Depending upon the complexity of the machining requirements, the machining times vary from part to part. Certain parts may need to be processed on one or more than one machine, and the sequence of these operations may be relevant or not. The FMC considered in this work has the ability to process the parts both sequentially and non-sequentially. In a sequential process, the parts are routed through the cell to visit machines in a pre-specified order.

On the other hand, a non-sequential processing will allow the flexibility to route the parts bypassing the sequencing requirement. Due dates are set by the customer. There are many different types of customers; thus, it is necessary to assign penalty values to each part type. These penalties depend on the characteristics of the customer and terms of contracts. It may be more significant to have certain orders on time for an important client than for one for whom deadlines are not that critical. Penalties also involve fines, loss of future sales and rush shipping costs. Robots are an expensive investment. They have a life span, and hence it is necessary to use them effectively. The importance of efficient utilization of the robot to reduce costs is critical. Likewise, machines have an associated utilization cost. It may not be efficient to let the machine remain idle. It is important to balance the idle times of the robot versus that of the machines. Machine idle cost requires a production study, and these values have to be set by the user. In our setting, all data is randomly assigned.

The machines in the cell perform different operations such as milling, grinding, drilling, etc. It is assumed that there is only one operation at a time on a machine. The length of the operation depends on the machining requirements of the part type. Processing times are part specific and are not machine dependent. Preventive maintenance of the machines is assumed and, therefore, no breakdowns are expected. Parts are delivered to the machines when needed, and hence there are no buffers present in each machine. The task of the robot is to pick up parts from an input buffer, move them through the cell, and drop them off at an output buffer. The travel time of the robot between machines is known and remains fixed.

A PC based, custom designed software accomplishes the control of the FMC. The software manages the sequence of parts and robot moves within the cell. The software controls the cell both online and off-line. Simulations are important for experimental purposes and actual implementations, and hence there is a need for simulation capability. An experimental cell provides a better visualization of performance of various scheduling models developed. Any hardware specific aspects that cannot be simulated can be readily seen in real-time implementation.

The multi-objective nature of the problem under consideration is a difficult problem. Many approaches can be followed. For this project, a fuzzy-based approach is considered. Fuzzy logic is an artificial intelligence tool that is very useful when dealing with uncertain data found in multi-objective decision-making problems. It is easy to model and the computational times are found to be very minimal, an aspect that is very critical for real-time implementations.

The nature of the implemented technique explores the following issues.

- To verify the functionality of the project in a real setting versus computer simulation.
- To determine whether there is a difference between two commonly employed part loading strategies, namely 'loading' and 'unloading' priority.
- To examine the differences between sequential and non-sequential part processing.
- To compare the performance of the proposed methodology to some of the standard rules and heuristics.

1.3 Organization

The rest of this work is structured as follows. A literature review pertaining to scheduling of Flexible Manufacturing Systems is presented. Then, an overview of the fuzzy logic theory and model implemented is given. This is followed by a description of the experimental setup and the developed software. Next, experiments and results are examined. Finally, conclusions and recommendations are provided.

2. Review of literature

Extensive research has been done on aspects related to scheduling of FMS systems. Due to the complexity and nature of scheduling, several approaches have been considered, ranging from traditional solutions involving complex mathematical analysis to the recent approach, namely reasoning algorithms. In addition, a number of different dispatching rules have been proposed. The following sections examine dispatching rules and the progress made to develop new improved rules and heuristics, as well as, review traditional and artificial intelligence based approaches.

2.1 Dispatching rules

The number of dispatching rules reported in the literature is overwhelming. Ramasesh [5] proposed a classification of rules based on performance measure criteria: time, work-in-process, due date and cost. Time based rules such as SPT (Shortest Processing Time) have been found to be good to reduce flow time and machine idle time. However, jobs with long processing times tend to be tardy [4]. Due-date based rules have been found to be more effective for tardiness related criteria [6], but their performance decline when applied to congested shops [7]. Furthermore, SPT was found to perform well in congested shops and for very tight due dates [5,8].

Cost-based priorities are one of the most important criteria to evaluate the performance of a scheduling rule. The cost related to tardy jobs, machine utilization, late orders, etc usually vary from customer to customer. Vepsalainen and Morton [7] tested six priority rules with a weighted cost against two new proposed rules: Weighted COVERT and 'Apparent Tardiness Cost' (ATC). They assigned priorities based on the expected tardiness cost. The results indicated that the new rules are superior in minimizing weighted tardiness penalties. Throughput criteria were not considered in this study.

Chen and Lin [4] presented a multi-factor priority rule to improve the Weighted COVERT rule [7]. Processing time criteria were included in this study. This rule gives higher priority to those jobs that have a longer expected waiting time, shorter slack time, and higher ratio of tardiness cost over processing time. The objective was to reduce the total tardiness cost. Its performance was compared to five other priority rules. This rule outperformed for tardiness cost criteria under certain conditions; however, simple rules such as SPT and EDD (Earliest Due Date) performed better on throughput criteria.

Kutanoglu and Sabuncuoglu [9] examined recently proposed dispatching rules in terms of tardiness criteria. Their experiments investigated the bottleneck dynamics, resource pricing and the effects of inserted idleness. The experiments showed that different pricing schemes should be used in different environments. A Pricing scheme refers to the different methods employed to calculate extra costs if resource capacity (i.e. machines) is decreased. Moreover, the inserted idleness improved the performance of ordinary dispatching rules.

From the brief review, it can be concluded that no generalization can be made regarding various dispatching rules. No single rule has shown to be superior in all type of scenarios, neither has it shown to perform uniformly well on more than one criterion. This is compounded by the fact that, at times, the literature presents conflicting evidence on the performance of the same set of rules [5]. As a result, some authors have developed new heuristics in order to improve the performance of single rules.

2.2 Heuristics

A common belief among researchers is that a combination of simple dispatching rules or a combination of heuristics with simple dispatching rules performs better than single rules in many cases [6].

Gere [6] proposed the use of "good" priority rules and tailored them to a particular problem at hand. He showed the effectiveness of certain heuristics: 'alternate operation' and 'look ahead' in combination with priority rules. For further improvement, the schedule was re-run with an enhancement of priorities for late jobs; however, the improvement was not significant.

Wu and Wysk [10] employed a simulation-based method to evaluate the performance of a set of dispatching rules for a short planning horizon. An evaluation process was carried out to select the best rule at a given period of time. The selection of the time span depended on the characteristics of the system and measures of performance. The process was repeated based on a short time frame. By alternating rules in such a manner, they tend to compensate the undesirable effects that each produces, to make a scheduling strategy more sensitive to the dynamic changes of the system.

Holthaus and Ziegler [11] implemented a coordination rule, called 'look ahead job demanding' (LAJD). This rule is based on 'look ahead' information about machine idle times, and a mechanism is incorporated for demanding, offering and selection of jobs. The simulation demonstrated the effectiveness of the rule in improving flow-time and due-date based objectives compared to scheduling rules without coordination.

Another approach with multi-objective criteria was proposed by Pierreval and Mebarki [12]. They developed a simple heuristic dispatching strategy, called 'shift from standard rules' (SFSR). This rule is based on a dynamic selection of pre-determined dispatching rules. A new selection is carried out each time that a machine becomes available, depending on the objectives, operating conditions and actual system state. The rules employed to choose the dispatching rules contained thresholds, which are tuned off-line with a simulation technique. The thresholds need to be adapted if important changes occur in the configuration of the system. The rule performed well to meet the primary objective, but it did not perform as good for the secondary objective. These objectives are usually based on a measure of the level of work in progress and on the capability to meet the due dates.

A number of researchers have used heuristics to approach the FMS scheduling problem, without combining or using simple rules. To name a few, Moreno and Ding [13], as opposed to the traditional hierarchical approach, developed a concurrent solution to the loading and scheduling problem in a FMS. The solution proved to be quite effective.

Hathout [14] proposed a heuristic to maximize the throughput and optimize the sequence of robot moves. The ability of parts being routed sequentially versus non-sequentially, and their effect on throughput were investigated. Several rules pertaining to loading of machines were implemented, and the study clearly demonstrated that certain rules performed better in certain cases.

2.3 Traditional approaches

Many of the prior FMS scheduling research works have utilized linear programming techniques and modeling approaches.

Linear programming techniques are usually difficult to formulate and to solve. Proposed solutions cannot handle large-problem size. One such case was presented by King, Hodgson and Chafee [15]. They applied a branch and bound technique to optimize the moves of a robot within a two-machine cell. The trade-off between computational time and its influence on obtaining close-to-optimal solutions has been reported. Their solution became ineffective for problems when the number of parts increased past ten. Likewise, Chen, Chu and Proth [16] concluded that this technique is somewhat successful when adapting it to three machines. Nonetheless, they concluded that further investigation is required to solve large-size problems. Sethi and others [17] utilized a state space approach to address the problem of sequencing parts and robot moves in a robotic cell. The objective was to maximize the throughput of the system. Their solution was limited to two machines and one single part type. In addition, many issues such as unequal travel times between different machines were not addressed.

Petri-nets are common and useful tools for the modeling of FMCs. Cheng, Sun and Fu[18] used a time place Petri-net (TPPN) to model a FMS. They obtained an optimal schedule by using a heuristic search algorithm. Yalcin and Boucher [19] presented a solution based on colored Petri Nets (CPNs) to control the alternative machining and sequencing in a FMC. Both papers showed that Petri-nets could be used as an effective modeling tool.

Lin, Wakabayashi and Adiga [20] presented another modeling technique. They developed an object-oriented model of the entities involved in a cell and their interactions. Their emphasis was on analysis and modeling rather than the development of heuristics or algorithms.

2.4 Artificial intelligence (AI) approaches

The scheduling of a FMC has been found to be a difficult problem to solve due to its dynamic nature. The use of Artificial Intelligence tools has proved to be effective to approach such problems. The most known AI techniques utilized in the scheduling of FMC are knowledge based, neural networks, genetic algorithms and fuzzy logic.

Knowledge based systems use human expertise and knowledge of the environment to solve the problem. Lee [21] developed a knowledge-based scheduling system, where knowledge could be easily updated or extended. The proposed solution was flexible and versatile, and did not require long computational times. The system consisted of automated guided vehicles (AGVs) for material handling. The author suggested that this might lead to bottlenecks when applied to a system containing one single material handling device. Further investigation is required to adapt it to a FMC environment.

Chen and Guerrero [22] devised a rule-based system to assist the controller in a FMC in making good decisions by using the current system state. The results showed that this approach gives better results in comparison to either applying heuristics or Petri-net models separately.

One of the most critical issues when developing knowledge-based systems is finding the required knowledge. Simulation techniques are usually used to acquire information, but they do not usually provide the required knowledge for making decisions. Pierreval and Ralambondrainy [23] overcame this problem by utilizing learning algorithms in a flow shop. These algorithms were proposed to generate a set of rules from simulation experiments. The results were encouraging; however, further studies are required to apply the technique to more complex and dynamic cases.

Expert systems (ES) have been frequently adopted to tackle scheduling problems. Kusiak and Chen [24] suggested that using ES, combined with operations research approaches, seem to be more suitable than each one separately.

Neural networks attempt to reflect the learning and prediction abilities of the human mind [25]. Jain and Meeran [26] presented a work, based on training a back-error propagation network, to solve a job-shop-scheduling problem. Their problem was claimed not to be feasible for large-scale applications.

Genetic algorithms (GA) provide a methodology that has been found to perform better than heuristics methods. Moreover, when integrated with other search procedures, it has been shown to give even better results [25]. Its limitation could be the large amount of computational time required.

Fuzzy set theory can be useful in modeling and solving scheduling problems with uncertain data. In the same way, fuzzy logic is an excellent tool when it comes to multi-objective criteria. Kazerooni, Chan and Abhary [27], presented a multi-objective fuzzy approach that uses membership functions to find the share of each objective in final decision rules. The outcome of this final decision was applied to the selection of machines, after a job had been previously selected by the use of traditional scheduling rules. Thus, different rules were combined for the selection of jobs and for the selection of machines. A combination of FUZZY/STPT (Fuzzy/Shortest Total Processing Time) showed improvement in net profit and average lead-time. The proposed methodology proved to be easy to implement and could be improved to yield better results.

Vidyarthi and Tiwari [28] developed a fuzzy-based methodology to address the machine-loading problem. Even though the minimization of system unbalance and maximization of throughput were their objectives, they did not take into account the inclusion of due-date related objectives to reduce costs. The job ordering and sequencing, as well as the operation-machine allocation decisions are made based on the evaluation of membership functions.

2.5 Summary

FMS scheduling is a very complex problem. The many different aspects involved and their complexity makes it a challenging research area. Although dispatching rules have not been found to perform efficiently in all cases, they have been shown to be an efficient tool for experimental tests. Dispatching rules are usually easy and fast to program. New and sophisticated rules in combination with heuristics have been proposed, but they usually lack intelligence to perform well when different system configurations or when multi-objective criteria are introduced. In mathematical terms, the FMS scheduling problem is difficult to formulate and solve. Likewise, the problem has proved to be NP-Hard, which usually restricts the solution to no more than two machines and/or a large number of parts.

Artificial intelligence tools have proved to be far more effective than the techniques explained above. For the present study, fuzzy logic has been chosen over all the other AI techniques because of its ability to deal with uncertain data. As would be shown in this thesis, fuzzy logic is easy to model and is an excellent tool for decision making in multi-objective systems. Moreover, in comparison to ES or knowledge-based systems, fuzzy logic needs fewer rules, and the knowledge is easier to model. It is easier to train than neural networks. It also leads to faster solutions than neural networks or genetic algorithms. Very few studies on FMC scheduling using fuzzy logic have been undertaken, and most of the previous research has been done on FMS systems with more than one material handling device or none.

The present work has adopted some elements from the work done by Hathout[14] in terms of the layout and configuration of the system, as well as, management of some internal data. The modeling of the internal data is accomplished by using object-oriented programming.

3. Methodology

In a typical computerized machine cell serviced by a single robot, many objectives can be taken into account. One usual production objective is to maximize the throughput, or, in other words, to reduce as much as possible the total processing time of the batch; thus increasing productivity. However, the batch may be composed of multiple products ordered by different clients, and hence may have different due dates. Not only due dates but also the importance of the clients and the terms of contracts are important. It is needless to say that penalties due to late jobs must also be taken into consideration. Machines and robots are a high investment, and hence it is important for the production floor to use the resources effectively. These objectives constitute the multi-objective problem under consideration. More details have been provided in Section 1.2.

The methodology proposed is fuzzy-logic based, and can analyze a multi-objective problem. It is an extension of SPT (Shortest Processing Time) and WEDD (Weighted Earliest Due Date), and, has the ability to take into account factors such as machine and robot idle costs. SPT and WEDD are well known for their effectiveness in optimizing the throughput and tardiness cost, respectively. However, they are not effective in dealing with multi-objective problems. The performance of the fuzzy-based approach is compared to SPT and WEDD, used individually and the results will be reported in Chapter 5. Before describing the fuzzy-based approach, it will be useful to review the following:

SPT (Shortest Processing Time) - jobs are sequenced in increasing order of their processing times. The sequence is accomplished by selecting the part with the shortest machining processing time among many combinations of job/machine. As is well known, SPT provides good performance for throughput criteria. However, jobs with long processing times and early due dates tend to be tardy. SPT is defined by:

$$SPT = MinPT(j,m), \qquad 3.1$$

where $Min\,PT(j,m)$ = minimum processing time of the job/machine combinations.

WEDD (Weighted Earliest Due Date) - jobs are sequenced in increasing order of the ratio of the due date of the job to the penalty of the job assigned when the job is late. Although WEDD performs well in terms of due date and penalty, it does not take into account processing time information, and thus it leads to a poor use of resources. WEDD can be expressed as:

$$WEDD = Min\left(\frac{DD}{P}\right), \qquad 3.2$$

where

$Min(DD/P)$ = minimum ratio of the due date (DD) over the penalty (P) of the jobs.

3.1 Fuzzy logic

The foundation of fuzzy logic is fuzzy theory. Zadeh [29] introduced fuzzy set theory in 1965. The aid of fuzzy logic is to provide a framework to deal, in a natural way, with problems in which the source of imprecision is the absence of sharply defined criteria. A fuzzy set, A, in X is characterized by a *membership function* $\mu_A(x)$ which associates each element in X a real number in the interval [0,1]. The value of $\mu_A(x)$ at x represents the *grade of membership* of x in A. The closer this value is to unity, the higher is the level of membership of x in A. On the contrary, in crisp logic the membership values are either *0* or *1*. Fuzzy logic introduces the possibility of intermediate values, and a more precise way of defining grades of membership of an element in a domain.

Fuzzy set operators can be defined in terms of operations between membership functions, the same way it can be done for crisp sets. These operations are important because they can describe interactions between variables. The basic operations in fuzzy logic are intersection, union and complement. They are defined by Zadeh [29] as follows:

$$\text{Intersection:} \quad \forall x \in X : \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)), \qquad 3.3$$

$$\text{Union:} \quad \forall x \in X : \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)), \qquad 3.4$$

$$\text{Complement:} \quad \forall x \in X : \mu_{A'}(x) = 1 - \mu_A(x). \qquad 3.5$$

This is a very simple extension of the classical operations. Other extensions using simple algebraic transformations are given in Table 3.1 [30]. For reference, Zadeh's definition is included as the first type.

	Intersection	**Union**
Zadeh	$\min(\mu_A(x), \mu_B(x))$	$\max(\mu_A(x), \mu_B(x))$
Mean	$\frac{\mu_A(x) \times \mu_B(x)}{2}$	$\frac{2 \times \min(\mu_A(x), \mu_B(x)) + 4 \times \min(\mu_A(x), \mu_B(x))}{6}$
Product	$\mu_A(x) \times \mu_B(x)$	$(\mu_A(x) + \mu_B(x)) - (\mu_A(x) \times \mu_B(x))$
Bounded Sum	$\max(0, \mu_A(x) + \mu_B(x) - 1)$	$\min(1, \mu_A(x) + \mu_B(x))$

Table 3.1 Algebraic intersection and union operations

Fuzzy set theory is characterized by its capability of handling linguistic variables. This ability makes fuzzy logic an attractive tool to mimic the behavior of human experts. In addition, fuzzy sets have been shown to be a very effective tool when it comes to multi-objective decision-making [32]. Consequently, fuzzy logic is a useful tool to approach the FMC scheduling problem under consideration.

3.2 Multi-objective decision making

Bellman and Zadeh introduced the concept of fuzzy decision making in 1970 [31]. They defined it as "a decision process in which the goals and/or constraints, but not necessarily the system under control, are fuzzy in nature". They pointed out that fuzzy goals and constraints can be defined accurately as fuzzy sets. A fuzzy decision may be viewed as the intersection of the given goals and constraints. The major reasons for utilizing fuzzy sets when handling multi-objectives are [32]:

1. its ability to represent objectives,
2. its convenient forms for combining objectives, and
3. its realistic means of including different degrees of importance to the objectives.

Based on the work proposed by Bellman and Zadeh [31] as well as Yager [32], the decision-making technique used in the present work will incorporate material presented below.

Objectives (goals and constraints) can be easily represented by fuzzy sets. Assuming we have a set of alternatives in a decision $X=[X_1,X_2,....,X_n]$ and a particular objective A, we can associate with each element in X_i a number $\mu_A(X_i)$ in the interval [0,1] indicative of how well X_i satisfies objective A. The advantage of fuzzy sets derives from the fact that very fuzzy objectives as well as very precise objectives can be represented. Moreover, fuzzy sets allow the manipulation of subjective phenomenon.

In order to extend the above definition to combining multi-objectives in decision making, let us assume we would like to select among the set of alternatives X the one that best satisfies a set of objectives $A_1, A_2, \ldots, A_p$. Each alternative X_i, is assigned a number indicative of how well it satisfies the objectives as a group $\mu_A(X_i)$ and, of course, the X with the highest value is the best. The dilemma would be on how to combine the contribution of each element to each objective, in order to get an overall general contribution of each element to all the objectives as a whole. One approach is the linguistic connection of the objectives, stated as "we want an X_i from X such that X_i satisfies A_1 *and* A_2 *and...and* A_p". Therefore, by doing intersection we would be able to combine the objectives. Another approach includes bargaining procedures followed in game theory where solutions are negotiated. The second approach shows a remarkable similarity to the first one. For further details on the second procedure, please refer to [32].

The values of the alternatives are obtained by using membership functions. As mentioned in the previous section, there are different ways of representing the intersection of objectives. For instance, Zadeh's method chooses the minimal value among the objectives. However, if an alternative does not contribute to an objective at all (i.e. $\mu_A(X_i) = 0$), then the result of the intersection would be zero (0), thus excluding the contribution of the alternative to the other objectives even if those are close to unity. Similarly, product and bounded sum methods eliminate the contribution of the objectives if one among them has membership value of zero (0). On the contrary, mean method produces an average value of the contributions of the alternative to the different objectives. Hence, if the alternative does not benefit an objective at all, the solution does not go to zero (0). Furthermore, in our specific case we are not only interested in the best alternative, but also in the second or third best one. For example, in the case of an alternative (part type), which cannot be chosen because of no availability of resources (i.e. machine that needs to process being unavailable, blockage, etc), then second or even third best alternative has to be looked at.

Table 3.2 presents a batch composed of three different part types: *A*, *B* and *C*; and three objectives represented by membership functions $\mu_A(p)$, $\mu_B(p)$ and $\mu_C(p)$. In this table, columns 1 and 2 refer to the part type and the number of parts to be produced respectively. Column 3 refers to the job number. Columns 4,5 and 6 represent the processing times for the respective machines, namely M1, M2 and M3. This table shows the numerical results obtained by using the four methods shown in Table 3.1. As can be observed, the *mean* method is the only one that can assure us that we would have more than one alternative to choose from. For example, if part type B cannot be selected for any reason, by using any method but *mean*, we would not have a second choice. In fact, *mean* would leave us to choose a second alternative (part type C) and a third alternative (part type A).

Part type	# parts	Jobs #	M1	M2	M3	$\mu_A(p)$	$\mu_B(p)$	$\mu_C(p)$	Z	M	P	B
A	2	1,2	4	2	0	0	0.40	0.50	0	0.32	0	0
B	3	3,4,5	0	3	0	0.50	1	1	0.50	0.83	0.50	0.50
C	1	6	0	5	1	1	0	0.5	0	0.50	0	0

Z –Zadeh; **M** –Mean; **P** – Product; **B** - Bounded Sum.
Membership functions are only given for example purposes. Results do not show relevance to the data set.

Table 3.2 Example of intersection methods

3.3 Implemented fuzzy logic based solution

In the present study, the evaluation of the overall contribution of the fuzzy membership function of each part type determines the sequence of the jobs in a given batch. Two methods are proposed: fuzzy-job and fuzzy-machine. These strategies are mainly used to choose the jobs and machines in the sequence.

3.3.1 Fuzzy-job

As stated in the previous section, job sequencing is determined by evaluating the overall contribution of the fuzzy membership function of the part type to the optimal performance of the system. The fuzzy membership function is composed of membership functions that correspond to the objectives of the problem under consideration. The various membership functions are defined below.

- $\mu_{Th}(p)$: The throughput of part type p is defined by a membership function that is defined as the ratio of the difference between the maximum total processing time PT

of the part types, and the total processing time of the part type p to the difference between the maximum and minimum total processing time of the part types. This membership evaluates the contribution of the part type to maximize the throughput of the batch. This can be expressed as:

$$\mu_{Th}(p) = \frac{MaxPT - PT(p)}{MaxPT - MinPT}, \qquad 0 \le \mu_{Th}(p) \le 1, \tag{3.6}$$

where

$MaxPT$ = maximum processing time of part types,

$MinPT$ = minimum processing time of part types, and

$PT(p)$ = total processing time of part type p.

$PT(p)$ is defined by $PT(p) = N(p) \times \sum_{m=1}^{M} PT(p,m)$, 3.7

where

$N(p)$	=	number of jobs for part type p,
m	=	machine number, $m = 1,2,3,\ldots,M$,
M	=	number of machines, and
$PT(p,m)$	=	processing time of part type p on machine m.

- $\mu_P(p)$: The membership function for the penalty of part type p is defined as the ratio of the difference between the maximum total penalty TP of the part types, and the total penalty of the part type p to the difference between the maximum and minimum total penalty of the part types. This membership evaluates the contribution of the part type to minimize the total penalty due to late jobs. This can be expressed as:

$$\mu_P(p) = \frac{MaxTP - TP(p)}{MaxTP - MinTP}, \qquad 0 \le \mu_P(p) \le 1, \tag{3.8}$$

where

$MaxTP$ = maximum total penalty of part types,

$MinTP$ = minimum total penalty of part types, and

$TP(p)$ = total penalty of part type p.

$TP(p)$ is defined by $TP(p) = \dfrac{DD(p)}{P(p)}$, 3.9

where

$DD(p)$ = due date of part type p, and

$P(p)$ = penalty of part type p.

- $\mu_{IC}(p)$: The membership function for the machine idle cost of part type p is defined as the ratio of the difference between the total machine idle cost of part p, and the minimum total machine idle cost of the parts to the difference between the maximum and minimum total machine idle cost of the part types. This membership evaluates the contribution of the part type to minimize the total machine idle cost when producing the batch. This can be expressed as:

$$\mu_{IC}(p) = \frac{C(p) - MinC}{MaxC - MinC}, \qquad 0 \le \mu_{IC}(p) \le 1, \tag{3.10}$$

where

$MaxC$ = maximum total machine idle cost of part types,

$MinC$ = minimum total machine idle cost of part types, and

$C(p)$ = total machine idle cost for part type p.

$C(p)$ is defined by $C(\text{p}) = \sum_{m=1}^{M} U \times IC(m)$, 3.11

where

$$U = \begin{cases} 1, & \text{if machine is needed for processing the part type } p\text{, and} \\ 0, & \text{if machine is not needed for processing the part type } p\text{, and} \end{cases}$$

$IC(m)$ = machine idle cost of machine m.

- $\mu_T(p)$:The membership function for the robot travel time of part type p is defined as:

$$\mu_T(p) = \begin{cases} 1, & \text{if jobs visits 1 machine, and} \\ 1/M, & \text{if job visits more than 1 machine,} \end{cases}$$

where,

M = number of machines part type p has to visit.

This membership evaluates the contribution of the part type to minimize the total robot travel time when producing the batch.

- μ_O^p: The overall membership function of part type p is the average *"mean"* of the individual membership function of the penalty, throughput, machine idle cost and robot travel times for part type p. By using the mean method defined in Section 3.1 this can be expressed as:

$$\mu_O(p) = \frac{\mu_P(p) + \mu_{Th}(p) + \mu_{IC}(p) + \mu_T(p)}{4} \qquad 0 \le \mu_O(p) \le 1. \qquad 3.12$$

3.3.2 Fuzzy-machine

Fuzzy-machine is similar to fuzzy-job. However, instead of evaluating the contribution of the part type to meeting the objectives of the system, the evaluation is given by obtaining a membership function for each combination of part type and machine. Thus, for fuzzy-job, there would be as many membership functions as number of part types, while for fuzzy-machine, there would be as many membership functions as number of part types and machines each part has to visit. For example, for the batch given in Table 3.3, the number of membership functions for fuzzy-job would be three $(\mu_o(A), \mu_o(B), \mu_o(C))$, and for fuzzy-machine would be six $(\mu_o(A,M1)$, $\mu_o(A,M2)$, $\mu_o(B,M2)$, $\mu_o(B,M3)$, $\mu_o(C,M3)$, $\mu_o(C,M4))$. The variables employed in Table 3.3 are the same as those used in Table 3.2.

Part type	# parts	Jobs #	M1	M2	M3	M4
A	2	1,2	4	2	0	0
B	2	3,4	0	13	7	0
C	2	5,6	0	0	5	8

Table 3.3 Batch example 3.1

Penalty and robot travel time values remain constant for each part type. These values do not vary according to the machine. Therefore, these membership functions are defined as in fuzzy-job. Throughput and machine idle cost memberships are defined somewhat differently as described below.

- $\mu_{Th}(p,m)$: The membership function for the throughput of part type p on machine m is defined in the same terms as in fuzzy. The difference is that the processing time is given by *PT(p,m)* instead of *PT(p)*. $\mu_{Th}(p,m)$ can be defined as:

$$\mu_{Th}(p,m) = \frac{MaxPT - PT(p,m)}{MaxPT - MinPT}, \qquad 0 \le \mu_{Th}(p,m) \le 1, \qquad 3.13$$

where

$MaxPT$ = maximum processing time of combinations part-type/machine, and

$MinPT$ = minimum processing time of combinations part-type/machine.

- $\mu_{IC}(p,m)$: The membership function for the machine idle cost of part type p on machine m is defined as before, but with the difference that instead of the total machine idle cost C of part p, the membership function is defined in terms of machine idle cost of part p on machine m. $\mu_{IC}(p,m)$ can be defined as:

$$\mu_{IC}(p,m) = \frac{IC(m) - MinIC}{MaxIC - MinIC}, \qquad 0 \le \mu_{IC}(p,m) \le 1, \qquad 3.14$$

where

$MaxIC$ = maximum machine idle cost of machines, and

$MinIC$ = minimum machine idle cost of machines.

3.3.3 Numerical example

For the batch data given in Table 3.4, the numerical results are shown below. For fuzzy-job, there are three membership functions that are evaluated: $\mu_o(A), \mu_o(B), \mu_o(C)$. For part type A, the membership functions are determined as follows.

Membership function for *throughput*: $\mu_{Th}(A) = \frac{40-12}{40-12} = 1$.

Membership function for *penalty*: $\mu_P(A) = \frac{17.5-13.3}{17.5-13.3} = 1$.

Membership function for *machine idle cost*: $\mu_{IC}(A) = \frac{3-3}{5-3} = 0$.

Membership function for *robot travel time*: $\mu_T(A) = \frac{1}{2} = 0.5$.

Overall membership function: $\mu_O(A) = \frac{1+1+0+0.5}{4} = 0.625$.

Part Type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	2	1,2	4	2	0	0	40	3	12	13.3	3
B	2	3,4	0	13	7	0	35	2	40	17.5	4
C	2	5,6	0	0	5	8	55	4	26	13.75	5

Machine idle cost rate: M1=2, M2=1, M3=3, M4=2.

Table 3.4 Batch example 3.2

The numerical results for the rest of the parts are shown in Table 3.5. As can be noticed, part type C (machine M3) would be the first in the sequence followed by part type A (machine M1), and finally part type B (machine M2).

Part type	$\mu_P(p)$	$\mu_{Th}(p)$	$\mu_{IC}(p)$	$\mu_T(p)$	$\mu_O(p)$
A	1	1	0	0.50	**0.62**
B	0	0	0.50	0.50	**0.25**
C	0.89	0.50	1	0.50	**0.72**

Table 3.5 Numerical results for fuzzy-job

For the fuzzy-machine method, membership functions are expressed in terms of part type and machine. For the combination of part type A and machine M1, the various membership functions are given as follows. Notice that penalty and robot travel time membership functions have the same numerical value as part type A in fuzzy-job.

Membership function for *throughput*:

$$\mu_{Th}(A,M1)=\frac{13-4}{13-2}=0.81.$$

Membership function for *machine idle cost*:

$$\mu_{IC}(A,M1)=\frac{2-1}{3-1}=0.5.$$

Overall membership function

$$\mu_O(A,M1)=\frac{1+0.81+0.5+0.5}{4}=0.70.$$

The rest of the calculations are given in Table 3.6. From the results presented in the table, the sequence would be Part type C-Machine M3, Part type A-M1, Part type A-M2, Part type C-M4, Part type B-M3, and Part type B-M2.

Part type / machine	$\mu_P(p,m)$	$\mu_{Th}(p,m)$	$\mu_{IC}(p,m)$	$\mu_T(p,m)$	$\mu_O(p,m)$
Part A, M1	1	0.81	0.50	0.50	**0.70**
Part A, M2	1	1	0	0.50	**0.62**
Part B, M2	0	0	0	0.50	**0.12**
Part B, M3	0	0.54	1	0.50	**0.51**
Part C, M3	0.89	0.72	1	0.50	**0.77**
Part C, M4	0.89	0.45	0.50	0.50	**0.58**

Table 3.6 Numerical results for fuzzy-machine

Having defined the methodology used for fuzzy-job and fuzzy-machine, the next chapter provides an overview of the structure of the implemented software, the experimental setup and the results obtained when comparing the different strategies.

4. Experimental setup and software

4.1 Layout of the flexible manufacturing cell

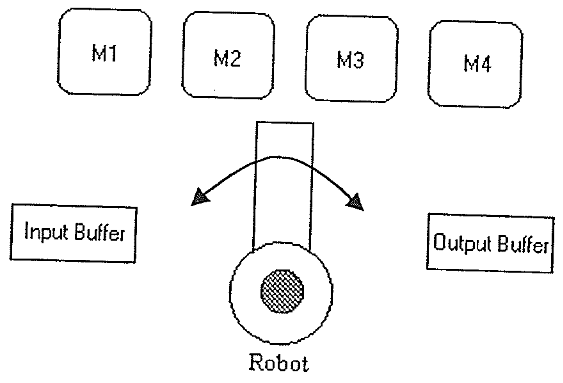


Figure 4.1 FMC Organization

The proposed methodology was tested on an experimental flexible manufacturing cell located at the University of Manitoba. It consists of a central robot arm serving four machining stations, an input buffer, and an output buffer (Figure 4.1). The machines are labeled M1, M2, M3 and M4. The robot has five degrees of freedom, and it was used for part transfer between different stations. The input buffer contains the raw parts that need to be processed. The parts are deposited on the output buffer when processing is completed. Each machine has a sensor that detects when a part is placed on the machine. The cell is controlled by graphic software written in Borland C++[33,34] using OWL (Object Windows Library) programming [35], that runs on a central PC. The PC communicates with the robot controller and sensors through an I/O board (Figure 4.2). There are fourteen robot subprograms controlled by a master program. The structure of the programs is shown in Appendix A. Each subprogram corresponds to a unique robot move within the cell. The master program continuously monitors the status of inputs of the robot controller and runs a subprogram according to the action required to be taken.

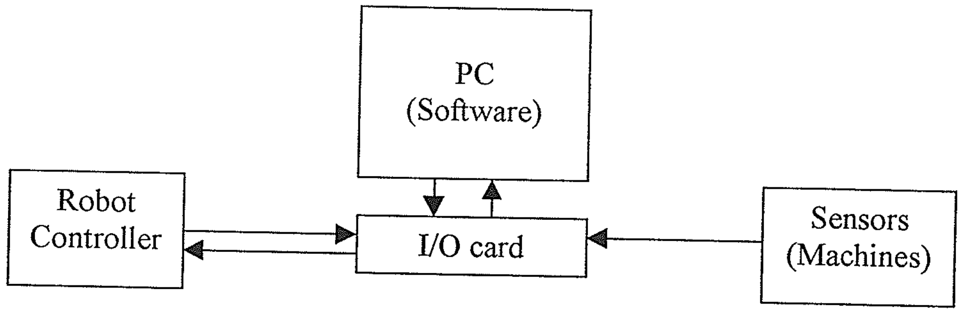


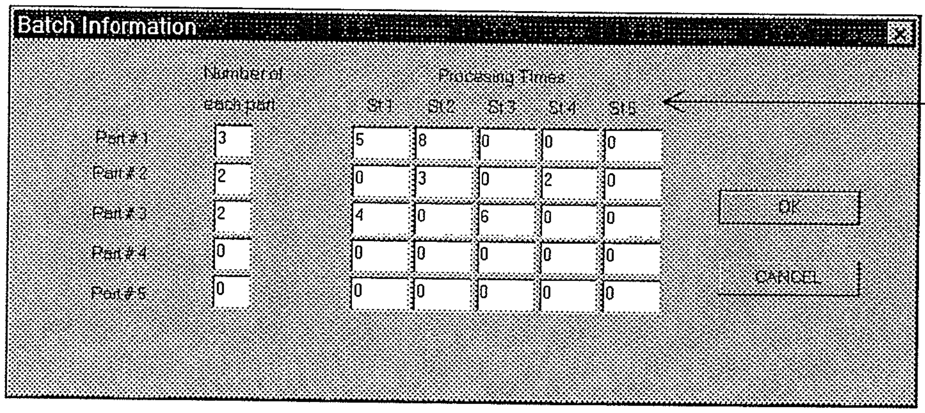
Figure 4.2 FMC I/O configuration

The PC software has a graphic interface where the user enters a batch data and its characteristics. An internal algorithm processes the data and sends orders to the robot through the I/O card. The master program in the robot controller responds to the PC controller and runs an appropriate subprogram; thus, the corresponding operation will be performed. At the end of each operation, the robot sends a signal to the I/O card, acknowledging to the PC that the requested operation has been finished and it is available for new operations. When loading a part onto a machine, the part triggers a sensor that sends a signal to the PC. This signal starts a timer in the software according to the part processing requirements. This signal indicates whether a station is occupied as well. The software continuously monitors the status of the timers to know when a part is finished and is ready to be picked up. Besides the ability of operating in real time, the software is capable of operating off-line, simulating results on the screen. Simulations are a quick way of assessing results prior to hardware implementations. They provide an overall picture of the system so that flaws or malfunctions can be visualized ahead of time.

4.2 Software operation

The software has the capability of online, real time implementation as well as off-line simulation, in both text and graphic mode. A windows-based graphic environment permits the insertion of data about the batch conditions in a user-friendly manner. The following figures show how data should be entered on each dialogue window in the software. For illustrative purposes, batch data corresponding to batch #1 shown in Appendix G.1 will be used.

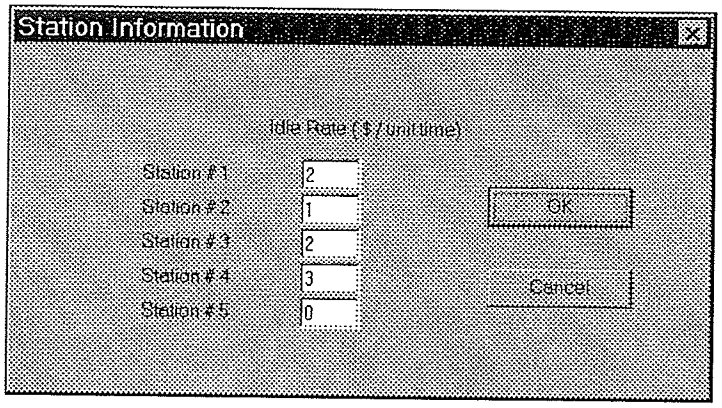
The first screen (Figure 4.3) is the *"Batch Information"* window. The user is allowed to insert information about the number of parts to be produced, and the processing times. If a part is not required to visit a machine, the processing time should be entered as zero. The second screen is the *"Stations"* window (Figure 4.4). Machine idle cost rate data must be provided at this stage. If a machine is not needed for the current batch, the data entered must be zero.


Batch Information

	Number of each part	Processing Times St 1	St 2	St 3	St 4	St 5
Part # 1	3	5	8	0	0	0
Part # 2	2	0	3	0	2	0
Part # 3	2	4	0	6	0	0
Part # 4	0	0	0	0	0	0
Part # 5	0	0	0	0	0	0

St # identifies the station (machine) number

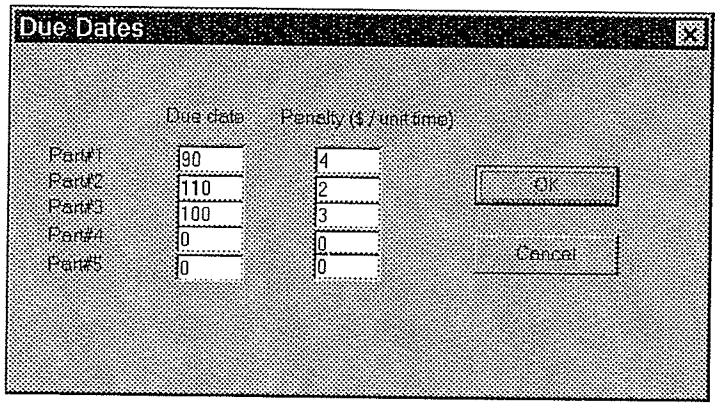
Figure 4.3 Batch information window



Station Information

	Idle Rate ($ / unit time)
Station #1	2
Station #2	1
Station #3	2
Station #4	3
Station #5	0

Figure 4.4 Station idle rate window

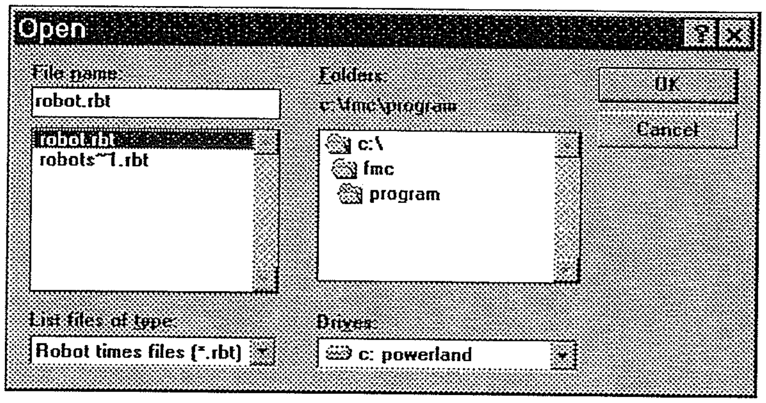


Due Dates

	Due date	Penalty ($ / unit time)
Part#1	90	4
Part#2	110	2
Part#3	100	3
Part#4	0	0
Part#5	0	0

Figure 4.5 Due date/Penalty window

The third screen is the *"Due Dates"* window (Figure 4.5). For each station, data about the due date and penalty rate for tardy jobs must be inserted. If a machine is not used, the values must be entered as zero. In addition, when the option *"Robot"* is selected on the main menu, a window pops up requesting the name of a file that contains the robot travel times (Figure 4.6). The file has an extension .rbt. The advantage of using files instead of dialogues is to avoid typing long sets of data (in this case 41). The .rbt data files contain robot travel times between various operations. Appendix D shows a sample of "Robot.rbt" file, which is the default file. Travel times can be changed using any word processor. However, the structure of the file cannot be altered, only the robot travel time data can be adjusted, if necessary. More details about robot travel times and their significance are given in Section 4.4.



Open

File name: robot.rbt

robot.rbt
robots~1.rbt

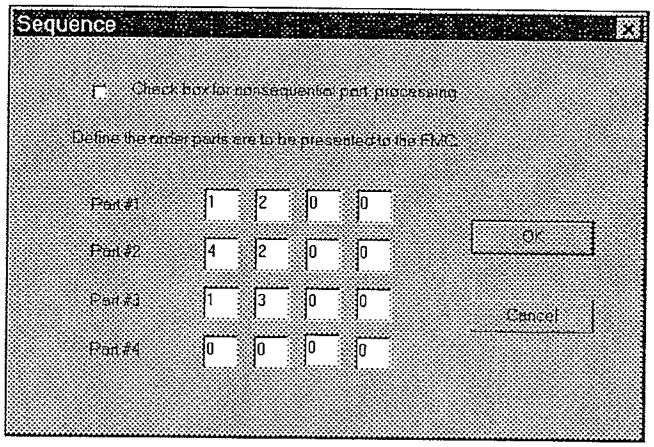
Folders: c:\fmc\program

c:\
fmc
program

List files of type: Robot times files (*.rbt)

Drives: c: powerland

Figure 4.6 Robot idle time file window



Sequence

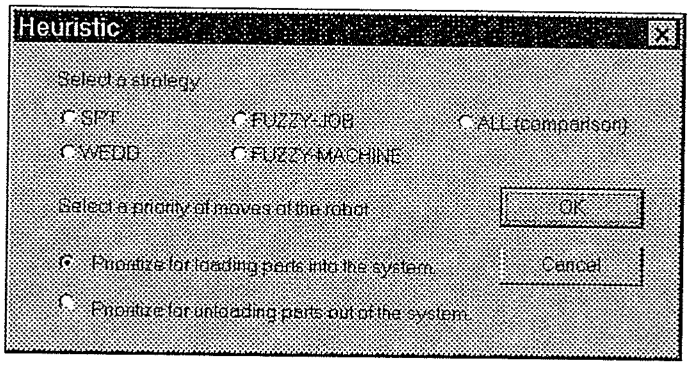
Check box for nonsequential part processing

Define the order parts are to be presented to the FMC

Part #1	1	2	0	0
Part #2	4	2	0	0
Part #3	1	3	0	0
Part #4	0	0	0	0

Figure 4.7 Part sequence window

Having inserted the numerical data, additional considerations must be entered. The window titled *"Sequence"* (Figure 4.7) allows the end-user to choose between a sequential versus a non-sequential loading of machines. If "sequential" is chosen, the order of the machines has to be inserted. If it is not chosen, the order of the parts has no effect on the sequence. The *"Heuristics"* window (Figure 4.8) contains the different strategies that can be run: SPT, WEDD, Fuzzy-Job, Fuzzy-Machine and ALL. If "ALL" is chosen, the software runs the four strategies for a loading and unloading priority, as well as sequential and non-sequential options. Therefore, a total of sixteen simulation results (combination of strategy-priority-mode) are shown. At the end of the simulation, a list of the combinations is displayed in descending order of performance for each objective. When "ALL" is selected, choosing "text mode" on the *"Mode"* window, as well as "simulation" on the *"Run-Setup"* window is recommended. In addition, in the *"Sequence"* window, "non-sequential" mode must be checked and the order of the sequence of the parts must be entered. The *"Heuristics"* window also provides two robot movement priorities: "loading" or "unloading". The rest of the windows allow the user to choose from two options. The *"Run-Setup"* window gives the option of choosing FMC (on-line implementation) or simulation (off-line). The *"Mode"* window provides the two modes in which the simulation can be carried out: text or graphic. Text mode is especially useful for test and analysis purposes.


Heuristic

Select a strategy

SPT FUZZY-JOB ALL (comparison)

WEDD FUZZY-MACHINE

Select a priority of moves of the robot

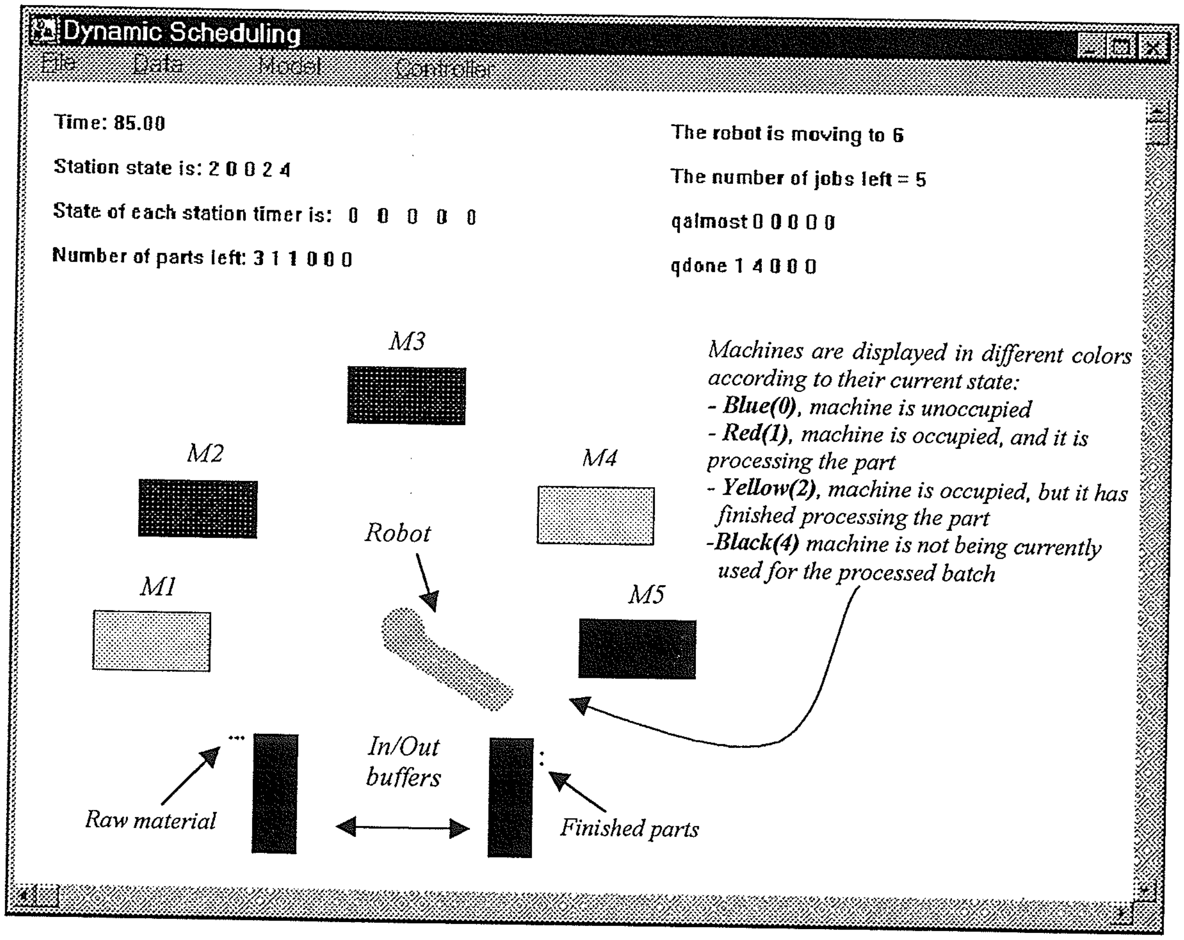
[✓ Prioritize for loading parts into the system.]

Prioritize for unloading parts out of the system.

Figure 4.8 Heuristics window

Finally, the user has to click on the menu *"Begin"* to be able to start the simulation. If the user has chosen "text" on the *"Mode"* window, a window pops up prompting for the name of the file where the user wants to store all the data. The file must have extension

.scn. A sample file is given in Appendix E. If the user chooses "graphic", a screen simulation appears on the main window (Figure 4.9). In this figure, "time" refers to the value of the general timer, or the time that has elapsed since the beginning of the simulation. The "station state" shows the state of the five stations respectively. Each number corresponds to a color as shown on the figure (blue=0,red=1,yellow=2,black=4). The "state of each station timer" refers to the processing time left for each machine. The "number of parts left" provides the number of parts left for each part type. The queues "qdone" and "qalmost" indicate which machine has finished its processing and which machine is almost finished, respectively.


Time: 85.00

Station state is: 2 0 0 2 4

State of each station timer is: 0 0 0 0 0

Number of parts left: 3 1 1 0 0 0

The robot is moving to 6

The number of jobs left = 5

qalmost 0 0 0 0 0

qdone 1 4 0 0 0

Machines are displayed in different colors according to their current state:
*- **Blue(0)**, machine is unoccupied*
*- **Red(1)**, machine is occupied, and it is processing the part*
*- **Yellow(2)**, machine is occupied, but it has finished processing the part*
*-**Black(4)** machine is not being currently used for the processed batch*

Figure 4.9 Main screen

Once the simulation is finished, general results about the performance of the strategy are shown. These results refer to throughput time, robot idle time, tardiness cost and machine idle cost. Other features of the software pertain to the I/O communication protocol. These features allow the user to test inputs and outputs prior to actual implementations.

4.3 Logic and data structure management on software

This section will provide software details on the implementation of the proposed fuzzy logic based methodology. Details of the fuzzy logic model were provided earlier in Section 3.3. After entering all the data as outlined in the previous section, the user can initiate the simulation. At this point, internal calculations are made to determine the contribution of each part type to the objectives of the system using the fuzzy theory presented earlier. Then, the program enters a loop, which continues until all the parts have been processed. This loop can be part of the sequential or non-sequential option. This depends on the specifications selected by the user. Inside each loop, an internal variable responds to the choice of the user to follow either a loading or unloading priority. This internal variable will lead the sequence of operations within the loop: loading, shifting, unloading, and moving and waiting. Loading means the robot will pick up a part from the input buffer and load a machine. Unloading means the robot will pick a finished part from a machine and drop it off at the output buffer. Shifting corresponds to unloading a part from a machine and transferring to another machine for further processing. Moving and waiting, refers to the movement of the robot to a machine and waiting until the processing ends prior to picking up the part. This last action is a look-ahead feature, which seeks to reduce the throughput time of a batch by saving extra travel time. Instead of waiting for the next part to be finished, the robot keeps track of which parts are almost done and goes next to the corresponding station, thus saving movement time. If a loading priority is chosen, the program first looks at the possibility of loading. If it is not possible, it considers the possibility of shifting a part. If that is also not possible, then it considers unloading. Finally, if nothing else is possible, the robot moves to the machine with the earliest finishing time. In the case of an unloading priority, the program first looks at the possibility of unloading, then shifting, then loading, and finally moving and waiting. Appendix F provides an overview of the loading priority logic.

Several structures were implemented to manage and use various data effectively. These include data pertaining to processing time, due date, penalty, cost, time and so forth. For instance, each job has a number of related parameters such as machines to visit and processing times, location, part type, etc. In terms of programming, these data would be unmanageable if it is not organized as a structure. Therefore, these parameters are collected in a job matrix (data structure) as shown in Figure 4.10. The first five variables correspond to the processing time requirements of the part. If the part needs to be processed on a machine, the variable corresponding to that machine would have as its value the processing time, otherwise that value would be zero. When the part is being processed, that value will be continuously updated towards zero. When the value reaches zero, it signifies that the part does not need any more processing on that machine. The sixth variable states how many machining processes are required by the part. This value is decreased every time the part is processed on a machine. The seventh variable indicates the part type. This is done for identification purposes, especially when dealing with software-based timers. The eighth variable shows the current location of the part. This value could be 0 for input buffer, 6 for output buffer or the machine number where the part is located. The ninth variable points to the status of the part. These values could be 1 if the part is being worked on, 2 if part is waiting to be unloaded, 3 if part is being shifted and 4 if the part is done.

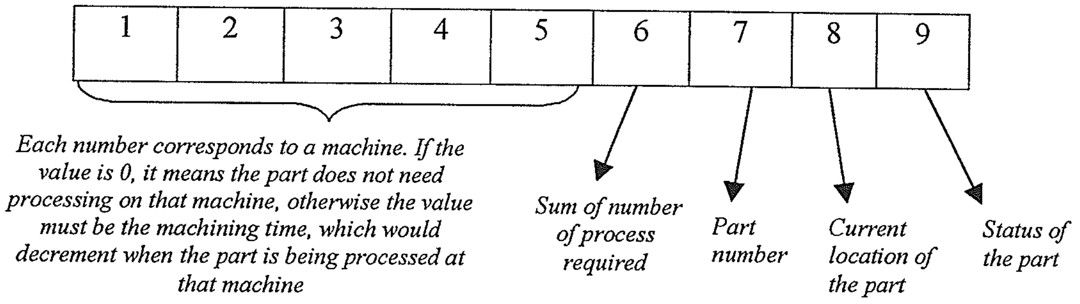

1
2
3
4
5
6
7
8
9
Each number corresponds to a machine. If the value is 0, it means the part does not need processing on that machine, otherwise the value must be the machining time, which would decrement when the part is being processed at that machine
Sum of number of process required
Part number
Current location of the part
Status of the part

Figure 4.10 Job matrix

Timers are managed in two different ways depending on the type of mode: off-line or online. In off-line mode, when a part is loaded onto a machine, a timer is set to a value corresponding to the machine processing time given by the job matrix. This value decreases as the general timer increases. When the timer reaches a value of zero, the corresponding machine number is inserted into a "job done" queue indicating that the processing of the part is completed. In online simulation, the program also manages the machine timers, but there are differences. Once the PC sends a command to the robot to load a part, it begins scanning the sensor located at the respective machine. Once the sensor is triggered indicating that the part is in place, the program starts the timer that corresponds to the machine where the part is being processed. The program keeps track of these timers. Once the timer has reached the time that is stored in the job matrix, a "job done" queue gets the information, and when possible, the program sends a command to the robot to pick up the part. Then, the timer is reset.

Besides the "job done" queue, there is a "job almost done" queue. This queue is useful for moving and waiting actions. This queue has a list of machines that are about to complete their processing. This queue is fed when a part is loaded onto a station for processing. However, since that queue is updated constantly, machines are increasingly ordered according to their remaining processing times.

The primary difference between sequential and non-sequential mode is that, in sequential mode there is a matrix that stores the sequence of the parts. In non-sequential mode, a "check" function instead of a matrix format is used to prevent conflicts that may rise as a result of the movement of the parts through certain routes.

4.4 Robot travel time issues

As stated in Section 4.1, there are fourteen robot subprograms. Each subprogram corresponds to a specific robot move as shown in Appendix A. The number of programs is presently limited since the robot has only fourteen inputs, and each program has to be called by an input triggered by an output from the PC. This limitation causes difficulty if all the robot travel times have to be made equal. Each robot subprogram corresponds to a part of a total robot move. This means, that a robot move or subprogram is part of a bigger move. For instance, if the robot is needed to go from the input buffer to pick up a new part, and then load it onto machine M1 (IN-M1), the PC would have to send two signals to call two different programs: program 1 and program 3 (Appendix C). Thus, if the speed of movement of the robot in program 1 or 3 were changed, then the travel distance of other moves that involve program 1 or 3 would change. Therefore, it is quite difficult to set all robot travel times equal. In fact, this would never be the case in a real situation. These travel times were measured and any one value was found to vary from 4 to 6 seconds. Since the objective of simulation is to obtain results that correspond as closely as possible to those in real time implementations, it is necessary to measure the robot travel times as accurately as possible. The robot utilized for the experiments is routinely used in many assembly tasks that require high precision and repeatability in positioning. It has been found to maintain both of these parameters over many trials, and hence it was deemed unnecessary to evaluate either of these parameters. However, the travel time as it shifts from one program to another is of greater importance in the experiments to be conducted. For instance, the robot could be waiting at a central location before the PC sends a request. Since the robot controller scans the inputs sequentially, the time of call from the PC to the actual time of response may vary. The measured travel time will also include a measure of repeatability of time measurements.

In order to measure the robot travel times accurately, small modifications were done in the robot programs. These modifications were inserted instructions. In Appendix A, these instructions are highlighted in bold lettering. For example, in order to measure the robot travel time between output buffer and input buffer (OUT-IN); we would need to measure the time just after the robot has dropped off the part at the output buffer, and immediately after the gripper has been triggered to pick up a new part. As seen in Appendix C, the related robot subprograms are numbered 2 and 1, in that sequence. The problem now is that; program 2 goes from a central position to the output buffer and returns to the central position. The time needed for the robot to go from the central position to the output buffer should not be accounted, and neither should be the time after the robot picks up the part in program 1. To solve this problem, an instruction was inserted in the robot sub-programs at these moments. The robot sets or resets an output to the PC, hence the software is instructed to start or end the timer, respectively. In this case, for program 2, the inserted instruction sets output 7. Then, the PC registers that signal and the software registers the exact time. Later, program 1 resets output 7, and the software records that moment and calculates the time between these events. These modifications were done in all the robot programs. The first program in the sequence sets output 7, and the last resets output 7. The software records the time between these events. The accuracy of this time is ± one hundredth of a second. All the travel time data collected for the experimental setup is stored in file "Robot.rbt" given in Appendix D.

This section has provided an outline of the experimental and software structure developed. In addition, aspects related to the user-operation and internal logic of the software, as well as, management of the robot travel time has been explained. The next chapter provides details of the experiments and results. In addition, aspects such as online versus off-line results, loading versus unloading priority, and sequential versus non-sequential mode are also investigated.

5. Experiments and results

The objective of the experiments is to compare the performance of the two proposed methodologies: fuzzy-job and fuzzy-machine in real-time as well as through off-line simulation. In order to study their performance in a multi-objective setting, they are compared to two dispatching rules: SPT and WEDD, as well as to the heuristic proposed by Hathout [14]. In Hathout's work, that preceded this work, all the parts in the batch have equal processing times. Comparisons with her heuristic are possible only when such conditions are met. Comparisons to many other heuristics reported in the literature are difficult due to specific characteristics of different FMCs. For example, handling devices are usually omitted or Automated Guided Vehicles (AGVs) are used for material handling. Part movement using AGVs, in comparison to using a robot, follows a different path generation procedure. Furthermore, many authors do not usually provide details about their software logic. For these reasons, comparisons with a wide variety of models are not attempted.

5.1 Real time implementation versus simulation

One of the most interesting and challenging aspects of this work is the real time implementation, or, in other words, the online control of the robot moves within the manufacturing cell, an aspect that is never mentioned by many investigators. Most authors have limited their studies to simulations only. Real time implementations allow readily visualizing the relationship between various parameters that contribute to the throughput in a cell. In addition, they allow seeing the effect of uncertainties such as slight differences in the time synchronization between the robot and PC controllers that may give rise to altogether a different type of part routing than the one determined.

Several challenges were encountered during the real time implementation. For a meaningful comparison between the real time implementation and the simulation, the robot travel times must be precisely measured and matched to those in the simulation. A C++ program was designed for this purpose, as well as, some modifications were performed to the robot sub-programs as outlined in the previous chapter. The robot travel times were measured to an accuracy of one hundredth of a second using a PC-based timer. These times were then entered into the simulation. Table 5.1 presents the throughput times, in seconds, for four sets of batches. Relevant data for these batches is given in Appendix G-1. A small difference of about 1% to 3% between the real time and the simulation results can be seen. Figure 5.1 shows the same results graphically. The plot is essentially linear with only a small variation. The results from real time implementation are always slightly greater than those obtained from simulation. The small variation could be attributed to a number of factors. The first one may be due to the imprecision in the measurement of the robot movement times. Initial experiments showed that measuring the robot move times repetitively yields slightly different results. The second factor might be the slight communication time delay between the PC and the robot controller. It should be pointed out that, the robot controller scans the main program sequentially, and this will cause a small delay in response. The sequential scanning built into the hardware of the robot controller cannot be modified.

	BATCH # 1 Throughput time (s)			BATCH # 2 Throughput time (s)		
	Real	Simulation	%Difference	Real	Simulation	% Difference
SPT	218.49	216.24	1.0 %	412.16	402.22	2.5 %
WEDD	216.66	211.85	2.2 %	428.05	415.39	3.0 %
FUZZY-JOB	219.65	216.85	1.3 %	428.00	415.39	3.0 %
FUZZY-MACH	216.59	211.85	2.2 %	412.16	402.22	2.5 %

	BATCH # 3 Throughput time (s)			BATCH # 4 Throughput time (s)		
	Real	Simulation	%Difference	Real	Simulation	% Difference
SPT	459.12	445.75	3.0 %	701.27	681.60	2.9 %
WEDD	458.85	445.75	2.9 %	691.35	671.77	2.9 %
FUZZY-JOB	366.64	357.56	2.5 %	691.14	671.77	2.9 %
FUZZY-MACH	458.94	445.75	2.9 %	691.03	671.77	2.8 %

Table 5.1 Comparison of throughput time between real time versus simulation

All robot moves correspond to those programmed on the software. Feedback loops function accordingly. The purpose of the feedback loops is to ensure that the robot is responding to the request initiated, and to prevent the software from sending additional requests to the robot until the robot has finished the programmed move.

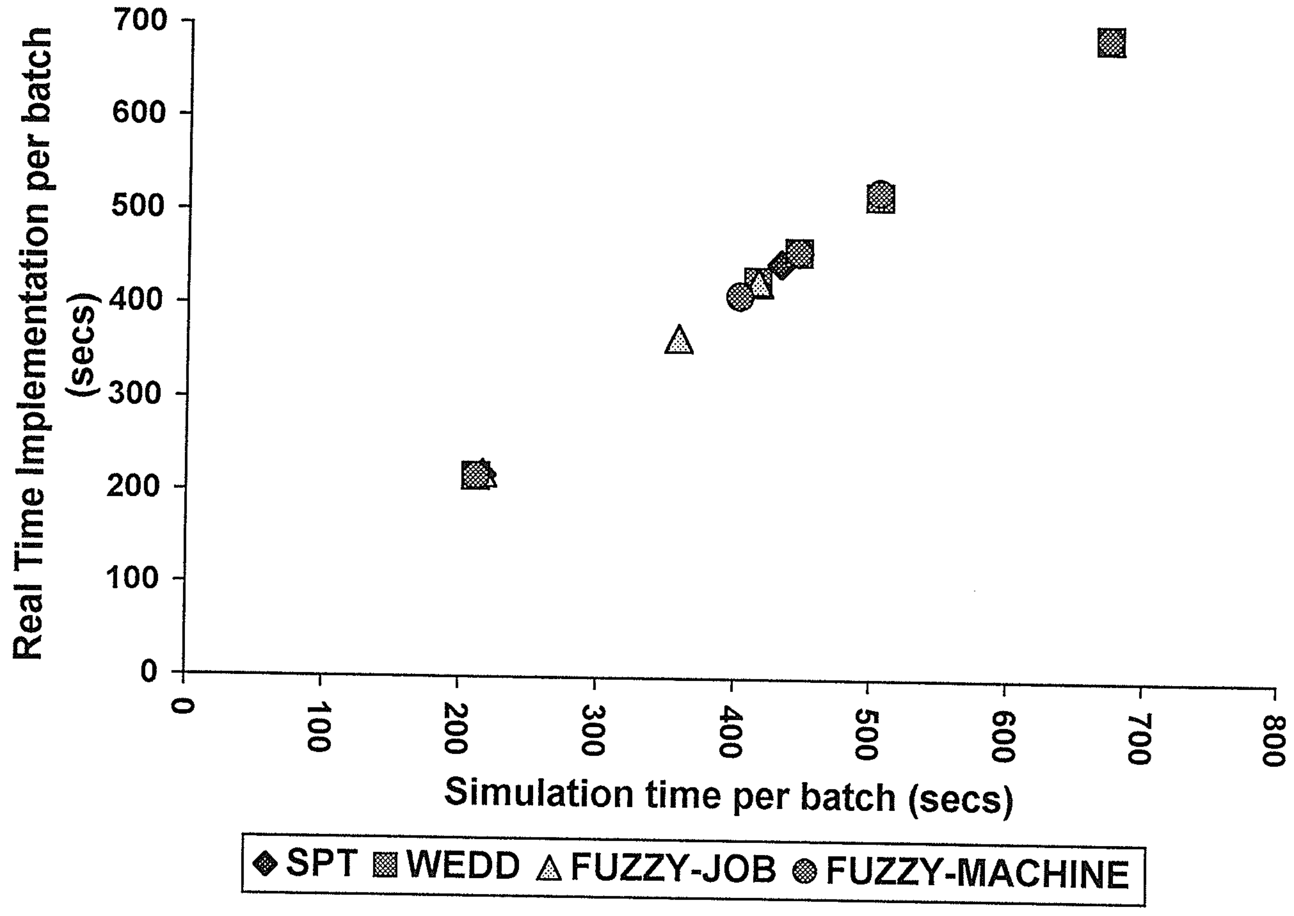


Figure 5.1 Plot of throughput for real time versus simulation for different strategies

5.2 Validation of the proposed methodologies

Some parts or products may require their manufacturing operations to follow a fixed sequence. There are also examples of part processing where the sequence is irrelevant. In the latter cases, it is wise to take advantage of the flexibility provided by the robot to move parts through the system more efficiently. Moreover, in some cases, it has been found that non-sequential operations maximize the throughput, especially when processing times are long [14]. For the present work, the experiments consider both sequential and non-sequential options. Comparisons are made in terms of machine idle cost, tardiness cost, and throughput resulting from employing different strategies. Criteria related to robot idle time can be only analyzed when more than one rule performs best and their robot idle times are not equal. Further details related to this issue are given in Section 1.2. Therefore, this criterion is analyzed only when such a comparison is possible.

In order to compare the performance of various scheduling options, a set of data that produces a wide variety of part processing requirements and due dates was generated. For the tardiness costs to be more meaningful, data from real-life situations should be used. No such data could be found. The effect of due date penalties is reflected by variations in due date indicated. Again, it was chosen arbitrarily. The data for different batches were randomly generated to study the effect of variations in processing times, processing requirements and due dates.

5.2.1 Sequential case

For the sequential case, it is assumed that the sequence follows the order in which the machines are numbered. However, it is important to clarify that the software designed has the capability of processing any sequence. The sequence followed is only for illustrative purposes. For instance, for batch 5.1 (See Table 5.2), the sequence for part type A is M1-M2, for part type B, it is M2-M4 and M1-M3 for part type C. The part processing is simulated using a loading priority. Some attributes can be noticed for this batch. The part type with the longest processing time (part type A) is also the part with the longest due date, and it has the highest penalty for tardiness. Besides, the part with the shortest processing time and shortest machining time (part type B - machine M2) has the lowest penalty and the earliest due date. Choosing the right sequence to reach multiple production goals is challenging in view of the conflicting objectives. It is known that choosing parts with the shortest processing time usually gives the best throughput. For this batch, the part with the shortest processing time (part type B) has the lowest tardiness penalty. However, it may be beneficial to start the process by choosing the part with the highest penalty in order to avoid high production costs. The proposed fuzzy methodologies are designed to look at these aspects and find a middle ground so that all the objectives can be fulfilled as far as possible.

Part type	# parts	Job #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	3	1,2,3	45	48	0	0	280	4	279	70	3
B	2	4,5	0	43	0	42	220	2	170	110	4
C	2	6,7	44	0	46	0	220	3	180	73.3	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

DD(p) – Due date of part type p
P(p) – Penalty of part type p
PT(p) – Total processing time of part type p
TP(p) – Total penalty of part type p, defined by: DD(p) / P(p)
C(p) –Total machine idle cost for part type p.

Table 5.2 Batch example 5.1

Simulation results for batch 5.1 are shown in Figure 5.2. This figure shows the resulting improvement when using fuzzy-job and fuzzy-machine over SPT and WEDD. The bar graph compares the performance of the different strategies. The percentage difference of improvement is defined as given below. For this particular case, the strategies selected are fuzzy-job and fuzzy-machine, and the strategies to which comparison is made are SPT and WEDD.

$$\text{\% difference of improvement} = 100 - \frac{\text{measure of criterion obtained from the strategies selected}}{\text{measure of criterion obtained from the strategies to which comparison is made}} \%$$

For this batch, fuzzy-job and fuzzy-machine produce identical performance, and the figure presents the results using them as base line. In this example, SPT chooses part type B first. Part type B has the smallest processing time, and the highest due date over penalty ratio. At the same time, SPT tends to leave part type A for processing to the end. Part type A has the highest penalty; therefore, tardiness cost would tend to be higher

(Figure 5.2). On the other hand, WEDD chooses part type A, which has the lowest due date over penalty ratio. However, that part type has a very long processing time, which at the end gives rise to a long robot idle time, and thus a longer throughput time. This produces a very high tardiness cost. Fuzzy-job and fuzzy-machine perform 71% better than WEDD in this case. Although, it is expected that WEDD would tend to give the best results when it comes to tardiness criterion, it did not turn out to be so since WEDD does not take into account processing times.

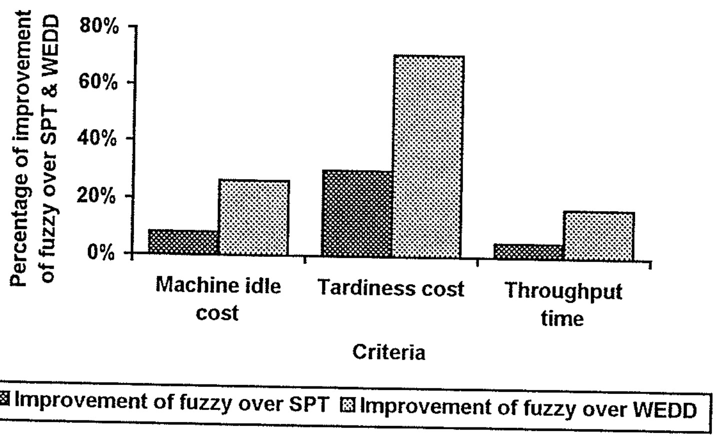


Figure 5.2 Percentage difference of improvement of fuzzy-job and fuzzy-machine with respect to SPT and WEDD for batch 5.1

Table 5.3 shows the part sequence for batch 5.1 given by fuzzy-job, fuzzy-machine, SPT, and WEDD. The sequence is displayed in a tabular form and the actions taken at various stages use the following abbreviations. A letter represents an action, with L referring to loading, U referring to unloading, and S referring to shifting. For example, L06-M1 means loading job number 6 onto machine 1; S06-M3, means shifting job number 6 from the current position to machine 3; and, U04 means unloading job number 4. The 'Time' represents the time during which the action is being performed.

Time (s)	Fuzzy-Job	Fuzzy-Machine	SPT	WEDD
5	L06-M1	L06-M1	L04-M2	L01-M1
15	L04-M2	L04-M2	L06-M1	L04-M2
48			S04-M4	
49	S06-M3	S06-M3		
58				S04-M4
63			L05-M2	
64	L07-M1	L07-M1		
68			S06-M3	
69	S04-M4	S04-M4		
73				L05-M2
83			L07-M1	
84	L05-M2	L05-M2		
95			U04	
100	U06	U06		
105				U04
106			S05-M4	
110	S07-M3	S07-M3		
116				S05-M4
119			U06	
125	L01-M1	L01-M1		
126				S01-M2
129			S07-M3	
130	U04	U04		
140	S05-M4	S05-M4		
141				L02-M1
144			L01-M1	
153			U05	
161	U07	U07		
163				U05
171	S01-M2	S01-M2		
179				U01
180			U07	
186	L02-M1	L02-M1		
189				S02-M2
190			S01-M2	
191	U05	U05		
204				L03-M1
205			L02-M1	
224	U01	U01		
234	S02-M2	S02-M2		
242				U02
243			U01	
249	L03-M1	L03-M1		
252				S03-M2
253			S02-M2	
267				L06-M1
268			L03-M1	
287	U02	U02		
297	S03-M2	S03-M2		
305				U03
306			U02	
315				S06-M3
316			S03-M2	
330				L07-M1
350	U03	U03		
355	**END**	**END**		
366				U06
369			U03	
374			**END**	
376				S07-M3
427				U07
432				**END**

Table 5.3 Part sequence of fuzzy-job, fuzzy-machine, SPT and WEDD for batch 5.1

As can be noticed from Table 5.3, fuzzy-job and fuzzy-machine result in a throughput time of 355 seconds and SPT and WEDD has a throughput time of 369 and 432 seconds, respectively. Another example is given in Table 5.4. In this case, part type A has the longest due date and the highest tardiness penalty. Part type B has the smallest processing time, while part type C has values in between, and the shortest processing time (machine M3).

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	5	1,2,3,4,5	28	32	0	0	480	3	300	60	3
B	5	6,7,8,9,10	0	25	30	0	460	2	275	230	4
C	5	11,12,13, 14,15	0	0	23	35	440	2	290	220	5

Machine idle cost rate: M1=2; M2=1; M3=3; M4=2

Table 5.4 Batch example 5.2

Simulation results are shown in Figure 5.3. Fuzzy-job and fuzzy-machine find a middle ground by initially choosing part type C. SPT chooses part type C as well, since it has the lowest processing time. Results show the same performance for fuzzy-job, fuzzy-machine and SPT, even in terms of robot idle time. There is a 10.7% improvement over WEDD in terms of tardiness cost.

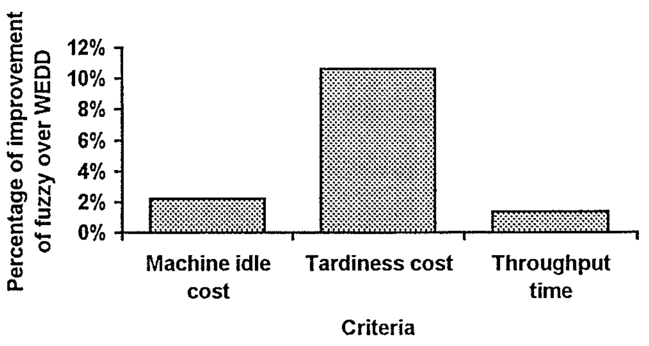
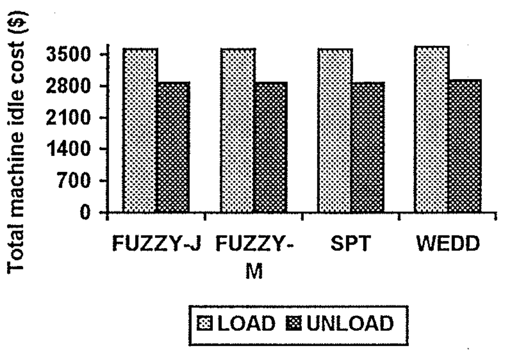
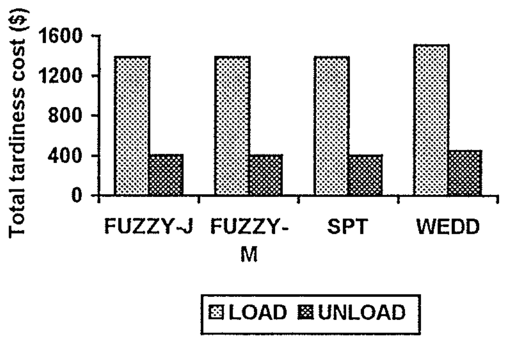


Figure 5.3 Percentage difference of improvement of fuzzy-job and fuzzy-machine with respect to WEDD for batch 5.2

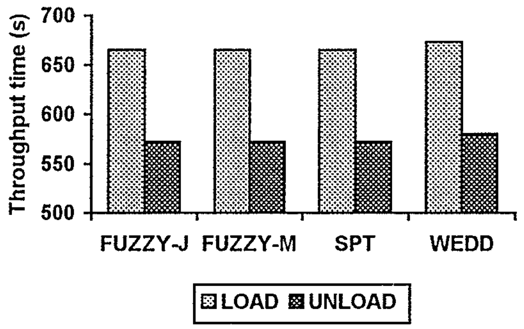
In order to study whether further enhancement can be obtained by any of the different strategies, more experiments were conducted using an unloading priority. In some cases, when using an unloading priority, results tend to be better; however, the results are not generalizable. For illustration purposes, simulation results of batch 5.2, using an unloading priority are shown in Figure 5.4. As seen in this figure, unloading priority yields better results than loading priority, however, this is not always the case. More results and analysis of loading versus unloading priority are given in subsequent sections.



(a) Machine idle cost



(b) Tardiness cost



(c) Throughput time

Figure 5.4 Loading versus unloading priority for batch 5.2

For illustrative purposes, three more simulation results are shown below. Once again, fuzzy-job and fuzzy-machine are compared against SPT and WEDD, but taking into account loading and unloading priority. There are eight combinations (methodology-priority) in total. In addition, when parts have the same processing time, results are also compared to those of Hathout [14]. Results are shown in figures and tables that indicate the improvement in percentage of the best performance compared to the worst, and the improvement of the best methodology compared to SPT. The data set for the first set of experiments is given in Table 5.5.

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	3	1,2,3	15	18	0	0	120	4	99	30	3
B	2	4,5	0	18	0	12	140	2	60	70	4
C	2	6,7	15	0	16	0	130	3	62	43.3	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

Table 5.5 Batch example 5.3

	Machine idle cost	Tardiness cost	Throughput time
Best strategy	Fuzzy-job-load Fuzzy-job-unload	Fuzzy-job-unload	Fuzzy-job-load Fuzzy-job-unload
Best	1392	826	221
Worst	1592	927	246
IBSP(worst)[1]	12.5%	10.9%	10.1%
IBSP (SPT)[2]	9.4% load	6.7% load	7.5% load

[1] Improvement of the best strategy performance over the worst
[2] Improvement of the best strategy performance over SPT

Table 5.6 Tabulated results - batch example 5.3

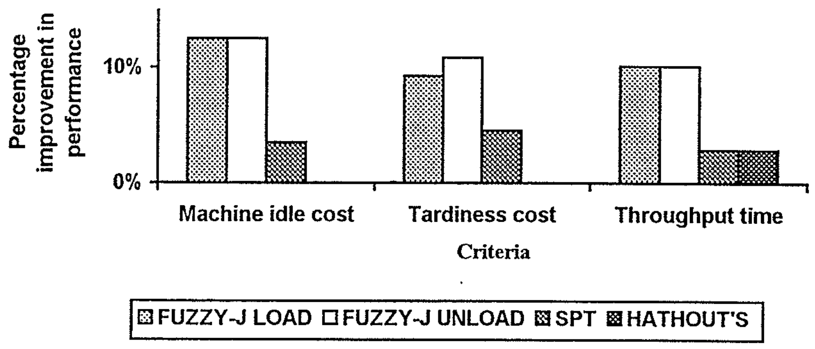


Figure 5.5 Performance plots - batch example 5.3

From the results shown in Table 5.6 and Figure 5.5, it is clear that fuzzy-job outperforms the other methodologies. For a fuzzy-job methodology, there is a remarkable similarity between the results of loading and unloading priority, except in terms of tardiness cost. Although the unloading priority is 1.8% better than loading priority in terms of the tardiness cost criterion, the percentage difference is too small to conclude that fuzzy-job with unloading priority yields the best results. In terms of robot idle time, there is not much difference in performance between the priorities. For this specific case, comparisons against Hathout's heuristic are possible since the processing times are equal. However, these comparisons are only possible in terms of throughput since Hathout's problem does not take into account other criterion. The throughput time for an identical batch of parts with loading priority produces a 7.5% improvement for fuzzy-job strategy in comparison to Hathout's.

In Table 5.6, the strategy that performed "best" has been identified. No consistent pattern could be observed in regards to the strategy that performed "worst". This statement applies to all the tables in which the "best" and "worst" performance has been compared.

Part Type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	4	1,2,3,4	85	85	0	0	840	1	680	840	3
B	5	5,6,7,8,9	0	85	0	85	870	2	850	435	4
C	3	10,11,12	0	0	85	0	760	3	255	253.3	1

Machine idle cost rate: M1=1; M2=3; M3=1; M4=2

Table 5.7 Batch example 5.4

Another batch example is given in Table 5.7. In this case, fuzzy-machine has the best performance (Table 5.8, Figure 5.6). As in the previous example, loading and unloading priority give similar results. Loading performance is better by 35.3% for tardiness cost. In terms of throughput, unloading and loading have the same performance, however the robot idle time for the unloading priority is longer, which means less robot-related production cost. The difference is 0.8%. This difference is minimal; hence, it can be concluded that fuzzy-machine with loading priority has the best performance. For this specific case, comparisons against Hathout's heuristic are also possible since all processing times are equal to 85 seconds. In terms of throughput, Hathout's heuristic gives identical results to those obtained using fuzzy-machine.

	Robot idle time	**Machine idle cost**	**Tardiness cost**	**Throughput time**
Best strategy	Fuzzy-mach-unload	Fuzzy-mach-load Fuzzy-mach-unload	Fuzzy-mach-load	Fuzzy-mach-load Fuzzy-mach-unload
Best	N/A	2865	55	895
Worst	N/A	3705	380	1015
Improvement	N/A	22.7%	85.5%	11.8%
IBSP (SPT)[1]	N/A	2.4%	15.3%	1.1%
ILP (unload)[2]	-0.8%	-	35.3%	-

[1] Improvement of the best strategy performance over SPT
[2] Improvement of loading over unloading priority

Table 5.8 Tabulated results - batch example 5.4

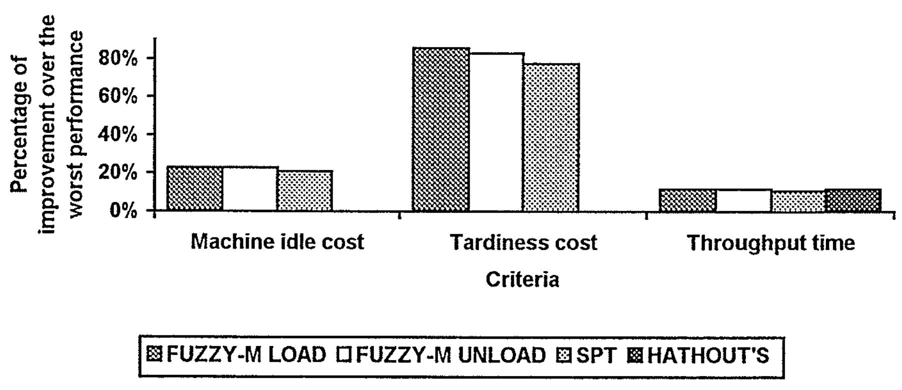


Figure 5.6 Performance plots - batch example 5.4

The next case is given in Table 5.9. SPT and fuzzy-machine have the best performance (Table 5.10, Figure 5.7) Unloading priority for both methodologies seems to perform better than loading priority except for the tardiness cost criterion. An important factor to be noticed in Table 5.10 is that, unloading is better (>6%) than loading priority in all cases. Nevertheless, when loading is better in tardiness criterion, the difference is 23%. Thus, SPT and fuzzy-machine with loading priority have the best performance.

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	4	1,2,3,4	48	43	0	0	440	1	364	440	3
B	5	5,6,7,8,9	0	41	0	49	470	2	450	235	4
C	3	10,11,12	0	0	45	0	310	3	135	103.3	1

Machine idle cost rate: M1=1; M2=2; M3=1; M4=3

Table 5.9 Batch example 5.5

	Machine idle cost	Tardiness cost	Throughput time
Best strategy	SPT-unload Fuzzy-mach-unload	SPT-load Fuzzy-mach-load	SPT-unload Fuzzy-mach-unload
Best	1957	167	539
Worst	2503	463	617
Improvement	21.8%	63.9%	12.6%
ISFU (load) [1]	5.4%	-23%	2.8%

[1] Improvement of SPT & fuzzy-machine unloading priority over SPT & fuzzy-machine loading priority

Table 5.10 Tabulated results - batch example 5.5

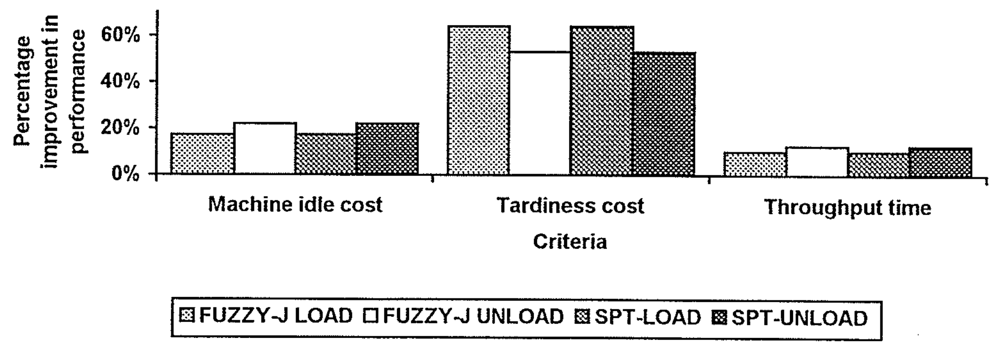


Figure 5.7 Performance plots - batch example 5.5

5.2.1.1 Summary of results for sequential mode

In order to compare the results, thirty batches were analyzed. Some of the results were shown in previous sections. In this section, a compilation of the results from all the trials is given. Figures show the percentage of how many times the methodology-priority combination was the best in the thirty trials. For compactness, letters represent the combination methodology-priority. For example, FJ means fuzzy-job, FM means fuzzy-machine, W and S mean WEDD and SPT respectively. Likewise, L and U represent loading and unloading priority, respectively.

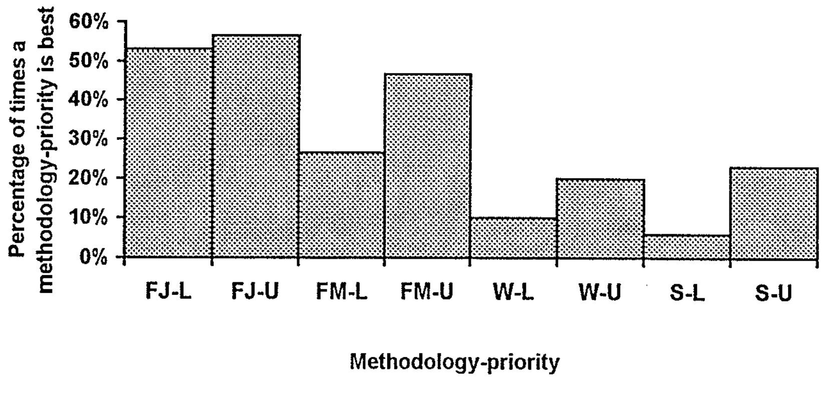


Figure 5.8 Performance for machine idle cost criterion

The first aspect to analyze is the performance of the methodologies with respect to the machine idle cost criterion (Figure 5.8). Fuzzy-job has the best performance. It is best 56.6% of the time with unloading priority, and 53.3% of the time with loading priority. SPT and WEDD with loading priority have the worst performance. Unloading priority seems to yield somewhat better results than the loading priority.

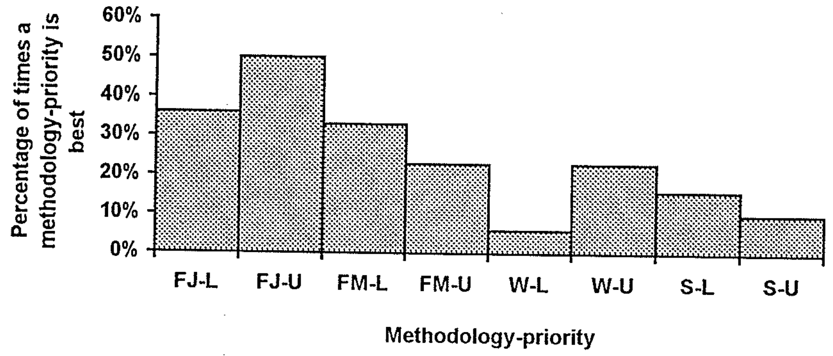


Figure 5.9 Performance for tardiness cost criterion

For the tardiness cost criterion (Figure 5.9), fuzzy-job with unloading priority has the best performance followed by fuzzy-job with loading priority. Indeed, it was expected that unloading priority would perform best in terms of the tardiness criterion since jobs would be unloaded faster, thus reducing tardiness costs. However, this is not always the case. For instance, the loading priority performs better for fuzzy-machine and SPT. WEDD with the loading priority showing the worst performance, followed by SPT.

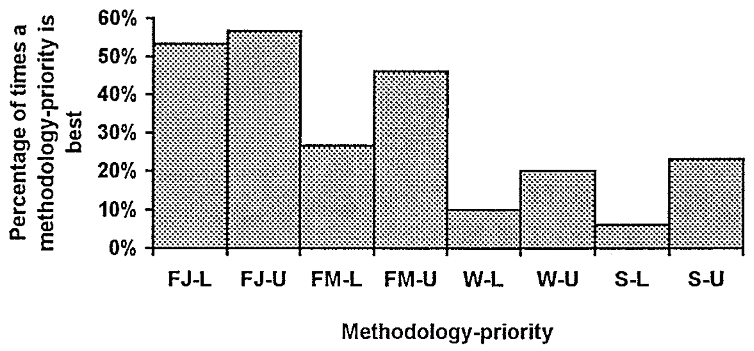


Figure 5.10 Performance for throughput criterion

In terms of throughput criterion (Figure 5.10), results are very similar to before, fuzzy-job with unloading priority showing the best performance. SPT and WEDD with loading priority have the worst performance.

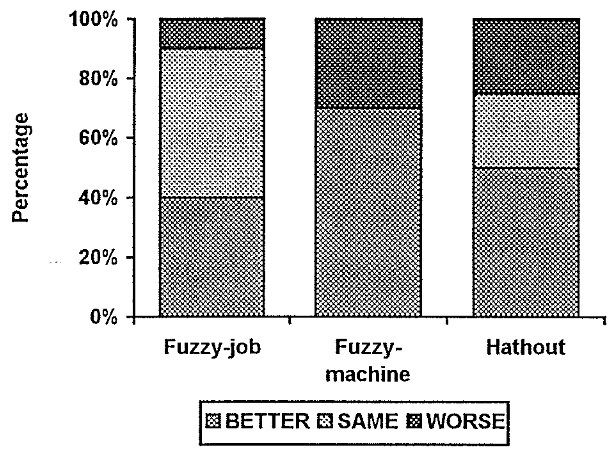
In general, fuzzy-job with the unloading priority gives the best results; SPT and WEDD show poor performance. Furthermore, the unloading priority usually performs better than the loading priority, except for the tardiness cost criterion, where results are difficult to generalize.

5.2.2. Non-sequential case

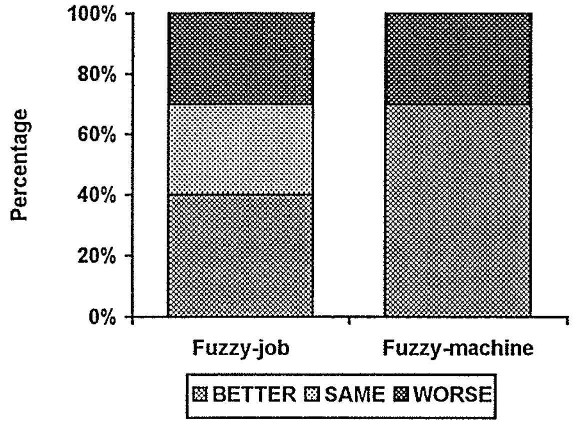
Under certain circumstances, the sequence a part or product goes through a set of machines may not be important. This suggests that flexibility in the route a part can go through in processing may offer an improvement over a specific route. In the previous section, fuzzy-job has been shown to perform the best, followed by fuzzy-machine. However, one of the hypotheses is that, fuzzy-machine might have better performance in a non-sequential mode. In addition, the non-sequential mode may yield better results. To verify this hypothesis, sets of experiments were conducted.

The first experiment compares sequential and non-sequential mode of fuzzy-job, fuzzy-machine and Hathout's heuristic [14]. Ten experiments were done, using a loading priority. Batch data for these experiments are given in Appendix G-2. Figure 5.11 shows the results obtained for the two criteria: tardiness cost and throughput time.

For brevity, robot idle time and machine idle cost data are omitted. Machine idle cost results are proportional to the results obtained for the throughput time criterion. Robot idle time criterion can be analyzed only in specific cases, as stated before. Comparisons using the throughput are made with results from Hathout's [14] whenever all the parts in a batch have identical processing times.



(a) Throughput time



(b) Tardiness cost

Figure 5.11 Performance of non-sequential respect to sequential mode

In terms of throughput (Figure 5.11.a), the non-sequential priority performs better 40% of the time for fuzzy-job, and remains the same 50% of the time. For fuzzy-machine, non-sequential priority performs better 70% of the time, but its performance is worse 30% of the time. For Hathout, results are better 50% of the time, and 25% of the time are worse or remain the same. For tardiness cost (Figure 5.11.b), results are the same for fuzzy-machine. However, for fuzzy-job, it is improved 40% of the time, and worse 30% of the time. Non-sequential mode yields better results or at least same results for both throughput and tardiness cost criteria; however, there are cases when the performance is very low as measured by the percentage shown. One interesting aspect is that fuzzy-machine has the best improvement when in non-sequential mode. However, as shown later, this improvement is not good enough to outperform fuzzy-job. Table 5.11 shows the numerical results obtained for the ten trials. Results indicate the throughput time and the tardiness cost for each batch. Tardiness costs are shown in parentheses. The best results are highlighted in bold lettering.

Batch #	**Fuzzy-job Sequential**	**Fuzzy-job Non-seq**	**Fuzzy-Mach Sequential**	**Fuzzy-Mach Non-seq**	**Hathout Sequential**	**Hathout Non-seq**
1	208 (**1227**)[1]	208 (1287)	**204 (957)**	208 (1057)	N/A[2]	N/A
2	415 (308)	415 (308)	502 (1079)	**470 (808)**	N/A	N/A
3	655 (300)	655 (300)	782 (1479)	**750 (1148)**	N/A	N/A
4	323 (**1684**)	**319** (1908)	323 (**1684**)	**319** (1908)	N/A	N/A
5	945 (107)	**921 (102)**	915 (75)	**891 (42)**	N/A	N/A
6	452 (4286)	**450 (3935)**	452 (4286)	**445 (3705)**	N/A	N/A
7	221 (841)	221 (841)	239 (885)	**224 (768)**	239	**224**
8	**592 (267)**	602 (438)	**562 (181)**	572 (273)	562	562
9	445 (4255)	445 (**3645**)	**445** (4255)	452 (**3679**)	**445**	452
10	940 (3791)	**787 (989)**	940 (3791)	**743 (696)**	940	**928**

[1] Results represent throughput time (tardiness cost)
[2] Not applicable since processing times are not equal

Table 5.11 Throughput time and tardiness cost for sequential versus non-sequential

Three batches and their results are further analyzed next. Results are shown in the same format as before. In this section, sequential and non-sequential modes are taken into consideration; thus, there are a total of sixteen combinations of methodology-priority-mode. The data for the first batch is given in Table 5.12. For this batch, fuzzy-job has the best performance. From the results presented in Figure 5.12, it can be observed that there is no difference between the unloading and loading priority, and, the sequential and non-sequential method of processing. The improvement over SPT is up to 18.8% in tardiness cost criterion (Table 5.13, Figure 5.12). This difference is a result of SPT leaving the part with the highest processing time (part type C) to be processed in the end. The part also has the earliest due date. Therefore, jobs (part type C) may tend to be tardy.

Part type	**# parts**	**Jobs #**	**M1**	**M2**	**M3**	**M4**	**DD(p)**	**P(p)**	**PT(p)**	**TP(p)**	**C(p)**
A	3	1,2,3	35	38	0	0	180	4	219	45	3
B	2	4,5	0	33	0	32	200	2	130	100	4
C	2	6,7	34	0	36	0	190	3	140	63.3	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

Table 5.12 Batch example 5.6

	Machine idle cost	**Tardiness cost**	**Throughput time**
Best strategy	Fuzzy-job	Fuzzy-job	Fuzzy-job
Best	1498	744	295
Worst	2034	1307	362
Improvement	26.3%	43.1%	18.5%
IBSP (SPT)[1]	9.2%	18.8%	6.0%

[1] Improvement of the best strategy performance over SPT

Table 5.13 Tabulated results - batch example 5.6

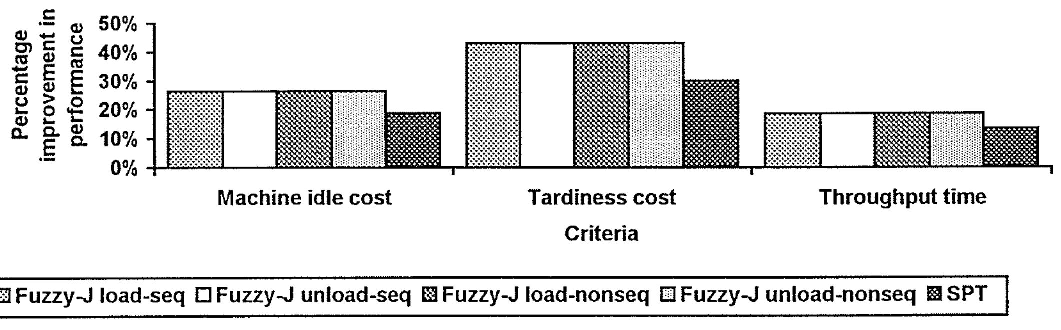


Figure 5.12 Performance plots - batch example 5.6

The data for another batch is given in Table 5.14. For this case, fuzzy-machine has the best performance (Table 5.15, Figure 5.13). Sequential mode performs better than the non-sequential. There is a similar behavior between the loading and unloading priorities.

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	4	1,2,3,4	68	68	0	0	640	1	544	640	3
B	5	5,6,7,8,9	0	68	0	68	670	2	680	335	4
C	3	10,11,12	0	0	68	0	510	3	204	170	1

Machine idle cost rate: M1=1; M2=3; M3=1; M4=2

Table 5.14 Batch example 5.7

	Robot idle time	**Machine idle cost**	**Tardiness cost**	**Throughput time**
Best strategy	Fuzzy-m-unload-seq	Fuzzy-m-load-seq Fuzzy-m-unload-seq	Fuzzy-m-load-seq	Fuzzy-m-load-seq Fuzzy-m-unload-seq
Best	422	2202	121	742
Worst	540	2923	534	845
Improvement	21.8%	24.6%	77.3%	12.2%
IBSP (SPT)[1]	2.3%	3.1%	14.2%	1.3%
ILP (unload)[2]	-1.2	-	46.5%	-

[1] Improvement of the best strategy performance over SPT
[2] Improvement of loading over unloading priority

Table 5.15 Tabulated results - batch example 5.7

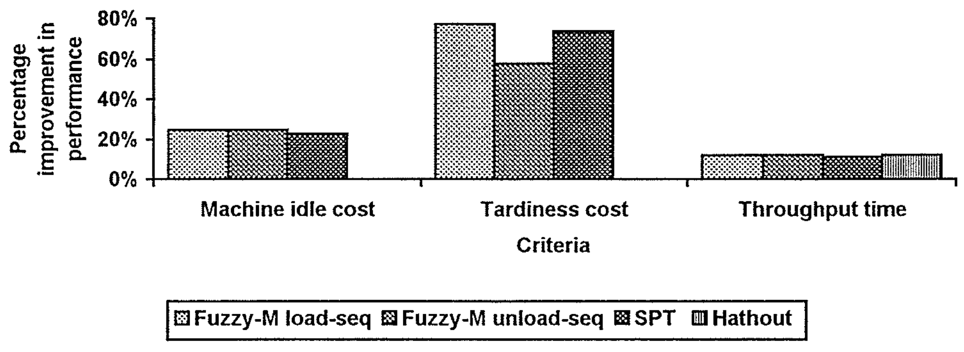


Figure 5.13 Performance plots - batch example 5.7

As regards to the robot idle time criterion, unloading outperforms loading priority by a slight margin of 1.2%. However, when it comes to the tardiness criterion, the loading priority outperforms the unloading priority by a margin of 46.5%. In this case, Hathout's heuristic yields similar results to those given by the fuzzy-machine in terms of throughput.

The data for the last illustrative example is given in Table 5.16. In this instance, SPT and fuzzy-machine with non-sequential mode outperform (Table 5.17, Figure 5.14). There is not much difference in performance between the loading or unloading priorities. SPT has good performance since the part type with the smallest processing time, has also the smallest due date over penalty ratio. In this case, Hathout's heuristic with unloading priority yields a result 0.2% better than SPT or fuzzy-machine.

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	5	1,2,3,4,5	50	45	0	0	580	3	475	193.3	3
B	5	6,7,8,9,10	0	45	52	0	560	2	485	280	4
C	5	11,12,13, 14,15	0	0	52	47	540	2	495	270	5

Machine idle cost rate: M1=2; M2=1; M3=3; M4=2

Table 5.16 Batch example 5.8

Best strategy	Machine idle cost	Tardiness cost	Throughput time
	SPT-nonseq Fuzzy-mach-nonseq	SPT-nonseq Fuzzy-mach-nonseq	SPT-nonseq Fuzzy-mach-nonseq
Best	2964	696	743
Worst	4620	3981	950
Improvement	35.8%	82.5%	21.8%

Table 5.17 Tabulated results - batch example 5.8

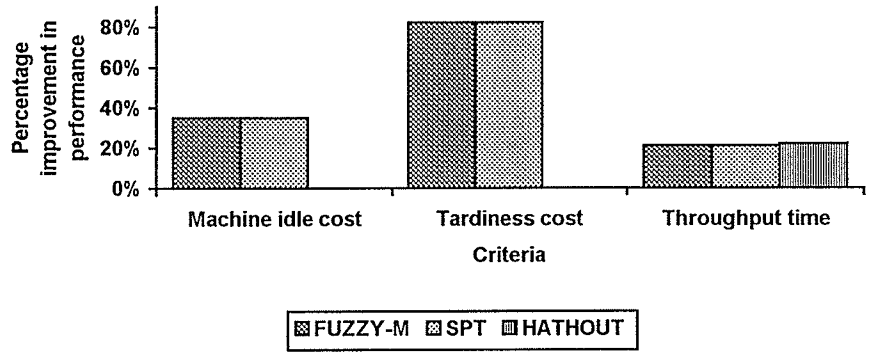


Figure 5.14 Performance plots - batch example 5.8

5.2.2.1 Summary of results for non-sequential mode

As in the previous section, experiments have been performed for thirty batches. Some of the results were shown in previous sections. In this section, a recompilation of data from the thirty trials is given. As before, figures show the percentage of how many times a particular combination of methodology-priority-mode was the best in the thirty trials. The acronyms used to abbreviate the combinations are the same as those used in section 5.2.1.1. Two additional abbreviations, S and N meaning sequential or non-sequential, respectively, have been added.

The first observation relates to the machine idle cost criterion (Figure 5.15). The best performance is achieved by fuzzy-job, for the loading priority and non-sequential mode.

The second best is fuzzy-job, with loading priority and sequential mode. Fuzzy-job strategy performs the best followed by fuzzy-machine, SPT and WEDD in descending order. SPT with the unloading priority and sequential mode performs the worst. In general, the loading priority, in combination with the non-sequential mode, produces the best performance for all strategies.

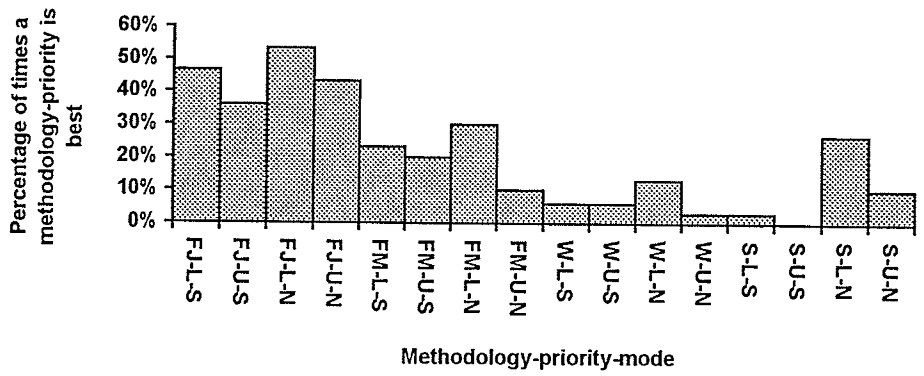


Figure 5.15 Performance for machine idle cost criterion

For the tardiness cost criterion, results are a bit different, as seen in Figure 5.16. The best performance is given by fuzzy-job in combination with the unloading priority and non-sequential mode, followed by fuzzy-job with loading priority and non-sequential mode.

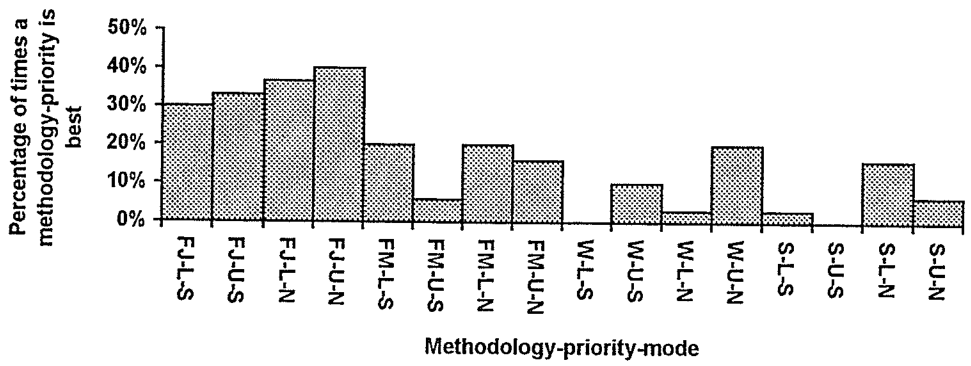


Figure 5.16 Performance for tardiness cost criterion

As in section 5.2.1.1, the unloading priority is the best in terms of tardiness cost for fuzzy-job and WEDD, and the loading priority is the best for fuzzy-machine and SPT. The worst performance was given by SPT.

In terms of the throughput criterion (Figure 5.17), results are very similar to the machine idle cost criterion, fuzzy-job with loading priority and non-sequential producing the best performance.

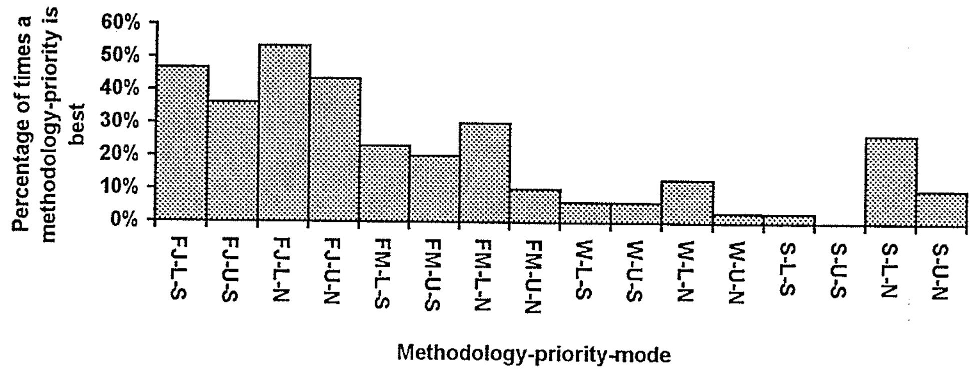


Figure 5.17 Performance for throughput criterion

To conclude, a fuzzy-job methodology has been shown to be the best, followed by fuzzy-machine. SPT in a sequential mode produces the worst performance. In terms of tardiness cost, results are similar to those obtained for the sequential mode. The unloading priority has proved to be the best priority for fuzzy-job and WEDD, with the loading priority being the best for fuzzy-machine and SPT. Moreover, in terms of throughput and machine idle cost criteria, results are opposite to those obtained for the sequential mode. Loading priority turns out to be the best priority for non-sequential mode, whereas, unloading is the best priority for sequential mode. Furthermore, the non-sequential mode produces better performance than the sequential mode.

6. Conclusions and recommendations

6.1 Conclusions

The performance of the proposed fuzzy logic based methodologies is very promising. They have shown much better performance than traditional dispatching rules such as SPT and WEDD in a multi-objective scheduling environment. SPT and WEDD may still be good when considering single objectives such as maximizing the throughput time or minimizing the tardiness cost, respectively. This thesis has shown that fuzzy methodologies are able to combine several objectives for effective scheduling of jobs. The results presented also show that fuzzy-job is more effective than fuzzy-machine. As indicated before, the difference in performance can be attributed to the way each strategy analyzes the contribution of the jobs to reach the objectives. Fuzzy-job considers the attributes of the job only, while fuzzy-machine evaluates the contribution of the job-machine combination. The enhancement in performance shown by the fuzzy-job comes from analyzing each job, keeping in perspective all the machines and their ability to process a set of jobs that constitutes a batch. It does not restrict the analysis to just a job-machine combination. The results also indicate a slight difference in performance between SPT and WEDD. SPT has a tendency to perform better in machine idle cost and throughput time criteria, while WEDD performs better in tardiness cost criterion.

Studies conducted to evaluate loading and unloading priority strategies did not result in any generalizable results. For the tardiness cost criterion, the unloading priority has proved to be the best priority for fuzzy-job and WEDD, while the loading priority has been the best choice for fuzzy-machine and SPT. In regards to improving the machine idle cost and throughput time, the results are different. Loading priority turns out to be the best priority for the non-sequential mode, whereas, unloading is the best priority for the sequential mode. In addition, non-sequential loading of parts proves to be more effective than sequential loading of parts.

The methodologies were successfully implemented in an automated machine cell. The two strategies performed quite well and the results obtained from simulation (off-line) show only a marginal difference with those from actual implementation (online). The slight difference is unavoidable due to the preset control architecture of the robot and communication aspects.

The capability of the custom designed software used to evaluate the performance of the two strategies can be effectively used for simulations (off-line) and actual implementations (online). The software has the ability of producing simulation in text or graphic mode for valuable data collection for further studies. An example is the generation of different sequences for a wide variety of batches for SPT, WEDD, fuzzy-job and fuzzy-machine methodologies, or all of them. When the sequences of all the methodologies are displayed, comparisons of the performance of the strategies for each objective can be easily seen. Further insight into results that show inconsistencies can be made using this tool.

6.2 Recommendations

In spite of the effectiveness of fuzzy-job in improving the throughput in a multi-objective environment, assigning random weights to the objectives could further enhance the results. This assignment could be complex since there are four membership functions, which could be assigned ten weight values ranging from 0 to 1. The number of possible combinations would be enormous (10^4). There are two possible ways of assigning and evaluating these weights. The first one could be by using rules which would require substantial user input and further experiments. A second approach, which may be more efficient, is by using techniques such as genetic algorithms. These approaches may require substantial computation times, and the improvement in performance will have to be evaluated against the computational time that would be needed. This would require further investigation.

Although the non-sequential methodology proved to be more efficient than the sequential mode, further research can be done to enhance the performance of the non-sequential mode. In the present work, no special techniques were utilized to check the movements of parts in the non-sequential method. An object-oriented function that checks for the presence of no conflicts in part allocation was utilized. However, by using intelligent techniques with look-ahead features, these conflict-checks can be further enhanced and results for the non-sequential mode may show further improvement.

The model can be easily expanded or modified to change or add more objectives. Furthermore, it would be very interesting to adapt the model to different types of scheduling problems such as the stochastic arrival of parts, dynamic environment, and different kinds of processing environment such as those explained in section 1.1.

8. References

[1] Singh, N. *Systems approach to computer design and manufacturing*, New York: John Wiley & Sons, Inc., 1996.

[2] Suri, T. and Whitney, C. “Decision support requirement in flexible manufacturing”, *Journal of Manufacturing Systems* 3.1 (1985): 61-69.

[3] Graves, S. “ A review of production scheduling”, *Operations Research* 29.4 (1981): 646-675.

[4] Chen, S and Lin, L. “Reducing total tardiness cost in manufacturing cell scheduling by a multi-factor priority rule”, *International Journal of Production Research* 37.13 (1999): 2939-2956.

[5] Ramasesh, R. “Dynamic job shop scheduling: a survey of simulation research”, *Journal of Management Science* 18.1 (1990): 43-57.

[6] Gere, W. "Heuristics in job shop scheduling", *Management Science* 13.3 (1966): 167-190.

[7] Vepsalainen, A. and Morton, T. “Priority rules for job shops with weighted tardiness costs”, *Management Science* 33.8 (1987): 1035-1047.

[8] Baker, K. “Sequencing rules and due-date assignments in a job shop”, *Management Science* 30. 9 (1984): 1093-1104.

[9] Kutanoglu, E. and Sabuncuoglu, I. “An analysis of heuristics in a dynamic job shop with weighted tardiness objectives”, *International Journal of Production Research* 37.1 (1999): 165-187.

[10] Wu, S. and Wysk, R. “An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing”, *International Journal of Production Research* 27.9 (1989): 1603-1623.

[11] Holthaus, O and Ziegler, H. “Improving job shop performance by coordinating dispatching rules”, *International Journal of Production Research* 35.2 (1997): 539-549.

[12] Pierreval, H. and Mebarki, N. “Dynamic selection of dispatching rules for manufacturing system scheduling”, *International Journal of Production Research* 35.6 (1997): 1575-1591.

[13] Moreno, A. and Ding, F ”A constructive algorithm for concurrently selecting and sequencing jobs in an FMS environment”, *International Journal of Production Research* 31.5 (1993): 1157-1169.

[14] Hathout, L. "Dynamic robot control and part loading in a flexible manufacturing cell", Master's thesis, *Department of Mechanical and Industrial Engineering, University of Manitoba*. 2000.

[15] King, R., Hodgson, T. and Chafee, F. "Robot task scheduling in a flexible manufacturing system", *IIE Transactions* 25.2 (1993): 80-87.

[16] Chen, H., Chu, C. and Proth, J. "Sequencing of parts in robotic cells", *The International Journal of Flexible Manufacturing Systems* 9 (1997): 81-103.

[17] Sethi, S.P., at al. "Sequencing of parts and robot moves in a robotic cell", *The International Journal of Flexible Manufacturing Systems* 4 (1992): 331-358.

[18] Cheng, C, Sun, T. and Fu, L. " Petri-Net based modeling and scheduling of a flexible manufacturing system", *Proceedings of the International Conference on Robotics and Automation* 1 (1994): 513-517.

[19] Yalcin, A. and Boucher, T. "An architecture for flexible manufacturing cells with alternate machining and alternate sequencing", *IEEE transactions on Robotics and Automation* 15.6 (1999): 1126-1130.

[20] Lin, L., Wakabayashi, M. and Adiga, S. "Object-oriented modelling and implementation of control software for a robotic flexible manufacturing cell", *Robotics & Computer Integrated Manufacturing* 11.1 (1994): 1-12.

[21] Lee, A. *Knowledge-based flexible manufacturing systems (FMS) scheduling*, New York: Garland Publishing Inc., 1994.

[22] Chen, H. and Guerrero, H. "Robot scheduling system for flexible manufacturing cells", *IEEE International Engineering Management Conference* (1990): 113-118.

[23] Pierreval, H. and Ralambondrainyi, H. "A simulation and learning technique for generating knowledge about manufacturing systems behaviour", *International Journal of Production Research* 41.6 (1990): 461-474.

[24] Kusiak, A. and Chen, M. "Expert systems for planning and scheduling manufacturing systems", *European Journal of Operational Research* 34 (1988): 113-130.

[25] Jones, A., Yuehwern, Y. and Wallace, E. "Monitoring and controlling operations". *The handbook of Industrial Engineering*, March 2000.
< http://www.mel.nist.gov/msidlibrary/doc/iehandbook.pdf>.

[26] Jain, A.S. and Meeran, S. "Scheduling a job-shop using a modified back-error propagation neural network", *Proceedings of the First Turkish Symposium on Intelligent Manufacturing Systems* 30-31 (1996): 462-474.

[27] Kazerooni, A., Chan, F.T.S. and Abhary, K. "Real-time operation selection in an FMS using simulation - a fuzzy approach", *Production Planning & Control* 8 (1997): 771-779.

[28] Vidyarthi, N.K. and Tiwari, M.K. "Machine loading problem of FMS: a fuzzy-based heuristic approach", *International Journal of Production Research* 39.5 (2001): 953-979.

[29] Zadeh, L.A. "Fuzzy sets", *Information and Control* .8 (1965): 338-353.

[30] Cox, E,. *The fuzzy systems handbook.* 2nd ed, New York: AP professional, 1998.

[31] Bellman, R.E. and Zadeh, L.A. "Decision-making in a fuzzy environment", *Management Science* 17 (1970): 141-164.

[32] Yager, R. "Fuzzy decision making including unequal objectives", *Fuzzy Sets and Systems* (1978): 87-95.

[33] Hubbard, J. *Schaum's outline of theory and problems of programming with C++*, New York: McGraw-Hill, 1996.

[34] Swan T., *Tom Swan's mastering Borland C++ 5*, Indiana: Sams Publishing, 1996.

[35] Spencer, I. *Teach yourself OWL programming in 21 days*, Indiana: Sams Publishing, 1995.

[36] Park, Y. "Optimizing robot's service movement in a robot-center FMC", *Computers and Industrial Engineering* 27.1-4 (1994): 47-50.

Appendix A - Robot program

Main robot program

```
10    Velocity 1000-2000 mm/sec              // minimum and maximum velocity
20    Robot coordinates                      // sets move's frame of reference
30    TCP 0 (Tool Center Point)              // indicates robot's point of reference
40    Frame 0                                // sets work envelope
50    Move to position X,Y at Velocity=50%   // robot moves to a recorded position
60    Move to position X,Y at Velocity=75%
70    Set output 6                           // sets output 6
80    Jump to 240 if input 1=1               // program goes to a line instruction
100   Jump to 260 if input 2=1               // when input is high
110   Jump to 280 if input 3=1
120   Jump to 300 if input 4=1
130   Jump to 320 if input 5=1
140   Jump to 340 if input 6=1
150   Jump to 360 if input 7=1
160   Jump to 380 if input 8=1
170   Jump to 400 if input 9=1
180   Jump to 420 if input 10=1
190   Jump to 440 if input 11=1
200   Jump to 460 if input 12=1
210   Jump to 480 if input 13=1
220   Jump to 500 if input 14=1
230   Jump to 70                             // program goes to instruction line # 70
240   Call Program 1                         // calls program #1
250   Jump to 70
260   Call Program 2
270   Jump to 70
280   Call Program 3
290   Jump to 70
300   Call Program 4
310   Jump to 70
320   Call Program 5
330   Jump to 70
340   Call Program 6
350   Jump to 70
360   Call Program 7
370   Jump to 70
380   Call Program 8
390   Jump to 70
400   Call Program 9
410   Jump to 70
420   Call Program 10
430   Jump to 70
440   Call Program 11
450   Jump to 70
460   Call Program 12
470   Jump to 70
480   Call Program 13
490   Jump to 70
500   Call Program 14
510   Jump to 70
520   Return
```

Subprograms

Program 1:	Robot picks up a part from input buffer, and goes to central position.
Program 2:	Robot goes from central position to output buffer, and then it goes back to central position.
Program 3:	Robot goes from central position holding a part and drops it at machine M1.
Program 4:	Robot goes from central position holding a part and drops it at machine M2.
Program 5:	Robot goes from central position holding a part and drops it at machine M3.
Program 6:	Robot goes from central position holding a part and drops it at machine M4.
Program 7:	Robot goes from central position to a position above machine M1.
Program 8:	Robot goes from above machine M1 to pick up a part from M1, and goes to central position.
Program 9:	Robot goes from central position to a position above machine M2.
Program 10:	Robot goes from above machine M2 to pick up a part from M2, and goes to central position.
Program 11:	Robot goes from central position to a position above machine M3.
Program 12:	Robot goes from above machine M3 to pick up a part from M3, and goes to central position.
Program 13:	Robot goes from central position to a position above machine M4.
Program 14:	Robot goes from above machine M4 to pick up a part from M4, and goes to central position.

Subprogram layout

The following is a sample of program 2.

```
10      Reset output 6                                   // resets output 6
20      Move to position X,Y at velocity of 100%         // robot moves to recorded position
30      Move to position X,Y at velocity of 50%
40      Rectangular coordinates
50      Move to position X,Y at velocity of 50%
60      Move to position X,Y at velocity of 50%, fine
70      Gripper wait 1 second                            // gripper is activated
80      Set output 7                                     // sets output 7 (travel time purposes)
90      Move to position X,Y at velocity of 50%
100     Move to position X,Y at velocity of 100%
110     Return
```

Appendix B – PC I/O allocation

Inputs

Input #	Function
0	Input buffer
1	Sensor located on machine M1
2	Sensor located on machine M2
3	Sensor located on machine M3
4	Sensor located on machine M4
5	Signal from the robot
6	Signal from the robot

Outputs

Output #	Function
0	Robot picks up a part from input buffer, and goes to central position
1	Robot goes from central position to output buffer, then it goes back to central position
2	Robot goes from central position holding a part and drops it at machine M1
3	Robot goes from central position holding a part and drops it at machine M2
4	Robot goes from central position holding a part and drops it at machine M3
5	Robot goes from central position holding a part and drops it at machine M4
6	Robot goes from central position to a position above machine M1
7	Robot goes from above machine M1 to pick up a part from M1, and goes to central position
8	Robot goes from central position to a position above machine M2
9	Robot goes from above machine M2 to pick up a part from M2, and goes to central position.
10	Robot goes from central position to a position above machine M3
11	Robot goes from above machine M3 to pick up a part from M3, and goes to central position
12	Robot goes from central position to a position above machine M4
13	Robot goes from above machine M4 to pick up a part from M4, and goes to central position

Appendix C - Notation for robot moves and corresponding programs

Notation	Robot Move description	Sequence/ Programs
IN M1	Robot picks up a part at input buffer and loads it onto machine M1	1,3
IN M2	Robot picks up a part at input buffer and loads it onto machine M2	1,4
IN M3	Robot picks up a part at input buffer and loads it onto machine M3	1,5
IN M4	Robot picks up a part at input buffer and loads it onto machine M4	1,6
M1 IN	Robots loads a part onto M1 and goes to input buffer to pick up a new part.	3,1
M2 IN	Robots loads a part onto M2 and goes to input buffer to pick up a new part.	4,1
M3 IN	Robots loads a part onto M3 and goes to input buffer to pick up a new part.	5,1
M4 IN	Robots loads a part onto M4 and goes to input buffer to pick up a new part.	6,1
M1 OUT	Robots unloads a part from M1 and goes to output buffer to drop it off	7,8,2
M2 OUT	Robots unloads a part from M2 and goes to output buffer to drop it off	9,10,2
M3 OUT	Robots unloads a part from M3 and goes to output buffer to drop it off	11,12,2
M4 OUT	Robots unloads a part from M4 and goes to output buffer to drop it off	13,14,2
OUT IN	Robot drops off a part at the output buffer and goes to input buffer to pick a new part	2,1
OUT M1	Robot drops off a part at the output buffer and goes to M1 to unload a finished part	2,7,8
OUT M2	Robot drops off a part at the output buffer and goes to M2 to unload a finished part	2,9,10
OUT M3	Robot drops off a part at the output buffer and goes to M3 to unload a finished part	2,11,12
OUT M4	Robot drops off a part at the output buffer and goes to M4 to unload a finished part	2,13,14
M1load M2unload	Robot loads a part onto M1 and goes to M2 to unload a part	3,9,10
M1load M3unload	Robot loads a part onto M1 and goes to M3 to unload a part	3,11,12
M1load M4unload	Robot loads a part onto M1 and goes to M4 to unload a part	3,13,14
M2load M1unload	Robot loads a part onto M2 and goes to M1 to unload a part	4,7,8
M2load M3unload	Robot loads a part onto M2 and goes to M3 to unload a part	4,11,12
M2load M4unload	Robot loads a part onto M2 and goes to M4 to unload a part	4,13,14
M3load M1unload	Robot loads a part onto M3 and goes to M1 to unload a part	5,7,8
M3load M2unload	Robot loads a part onto M3 and goes to M2 to unload a part	5,9,10
M3load M4unload	Robot loads a part onto M3 and goes to M4 to unload a part	5,13,14
M4load M1unload	Robot loads a part onto M4 and goes to M1 to unload a part	6,7,8
M4load M2unload	Robot loads a part onto M4 and goes to M2 to unload a part	6,9,10
M4load M3unload	Robot loads a part onto M4 and goes to M3 to unload a part	6,11,12
M1unload M2load	Robot unloads a part from M1 and goes to M2 to load that part	7,8,4
M1unload M3load	Robot unloads a part from M1 and goes to M3 to load that part	7,8,5
M1unload M4load	Robot unloads a part from M1 and goes to M4 to load that part	7,8,6
M2unload M1load	Robot unloads a part from M2 and goes to M1 to load that part	9,10,3
M2unload M3load	Robot unloads a part from M2 and goes to M3 to load that part	9,10,5
M2unload M4load	Robot unloads a part from M2 and goes to M4 to load that part	9,10,6
M3unload M1load	Robot unloads a part from M3 and goes to M1 to load that part	11,12,3
M3unload M2load	Robot unloads a part from M3 and goes to M2 to load that part	11,12,4
M3unload M4load	Robot unloads a part from M3 and goes to M4 to load that part	11,12,6
M4unload M1load	Robot unloads a part from M4 and goes to M1 to load that part	13,14,3
M4unload M2load	Robot unloads a part from M4 and goes to M2 to load that part	13,14,4
M4unload M3load	Robot unloads a part from M4 and goes to M3 to load that part	13,14,5

Appendix D - Structure of file "Robot.rbt"

(The robot travel times correspond to the times measured in seconds for the project)

```
IN M1 5.13
IN M2 4.78
IN M3 5.27
IN M4 5.66
M1 IN 5.05
M2 IN 5.05
M3 IN 5.77
M4 IN 5.82
M5 IN 0.0
M1 OUT 4.83
M2 OUT 4.85
M3 OUT 4.5
M4 OUT 4.73
M5 OUT 0.0
OUT IN 5.4
OUT M1 5.66
OUT M2 5.77
OUT M3 5.38
OUT M4 5.5
OUT M5 0.0
M1LOAD M1UNLOAD 0.0
M1LOAD M2UNLOAD 5.5
M1LOAD M3UNLOAD 5.44
M1LOAD M4UNLOAD 5.11
M1LOAD M5UNLOAD 0.0
M2LOAD M1UNLOAD 5.11
M2LOAD M2UNLOAD 0.0
M2LOAD M3UNLOAD 5.44
M2LOAD M4UNLOAD 5.11
M2LOAD M5UNLOAD 0.0
M3LOAD M1UNLOAD 4.73
M3LOAD M2UNLOAD 4.89
M3LOAD M3UNLOAD 0.0
M3LOAD M4UNLOAD 4.62
M3LOAD M5UNLOAD 0.0
M4LOAD M1UNLOAD 4.56
M4LOAD M2UNLOAD 4.34
M4LOAD M3UNLOAD 4.12
M4LOAD M4UNLOAD 0.0
M4LOAD M5UNLOAD 0.0
M5LOAD M1UNLOAD 0.0
M5LOAD M2UNLOAD 0.0
M5LOAD M3UNLOAD 0.0
M5LOAD M4UNLOAD 0.0
M5LOAD M5UNLOAD 0.0
M1UNLOAD M1LOAD 0.0
M1UNLOAD M2LOAD 4.78
M1UNLOAD M3LOAD 5.45
M1UNLOAD M4LOAD 6.05
M1UNLOAD M5LOAD 0.0
M2UNLOAD M1LOAD 5.22
```

```
M2UNLOAD M2LOAD 0.0
M2UNLOAD M3LOAD 5.54
M2UNLOAD M4LOAD 6.1
M2UNLOAD M5LOAD 0.0
M3UNLOAD M1LOAD 4.83
M3UNLOAD M2LOAD 4.45
M3UNLOAD M3LOAD 0.0
M3UNLOAD M4LOAD 5.71
M3UNLOAD M5LOAD 0.0
M4UNLOAD M1LOAD 5.11
M4UNLOAD M2LOAD 4.72
M4UNLOAD M3LOAD 5.45
M4UNLOAD M4LOAD 0.0
M4UNLOAD M5LOAD 0.0
M5UNLOAD M1LOAD 0.0
M5UNLOAD M2LOAD 0.0
M5UNLOAD M3LOAD 0.0
M5UNLOAD M4LOAD 0.0
M5UNLOAD M5LOAD 0.0
```

Note: the notation is explained in Appendix C

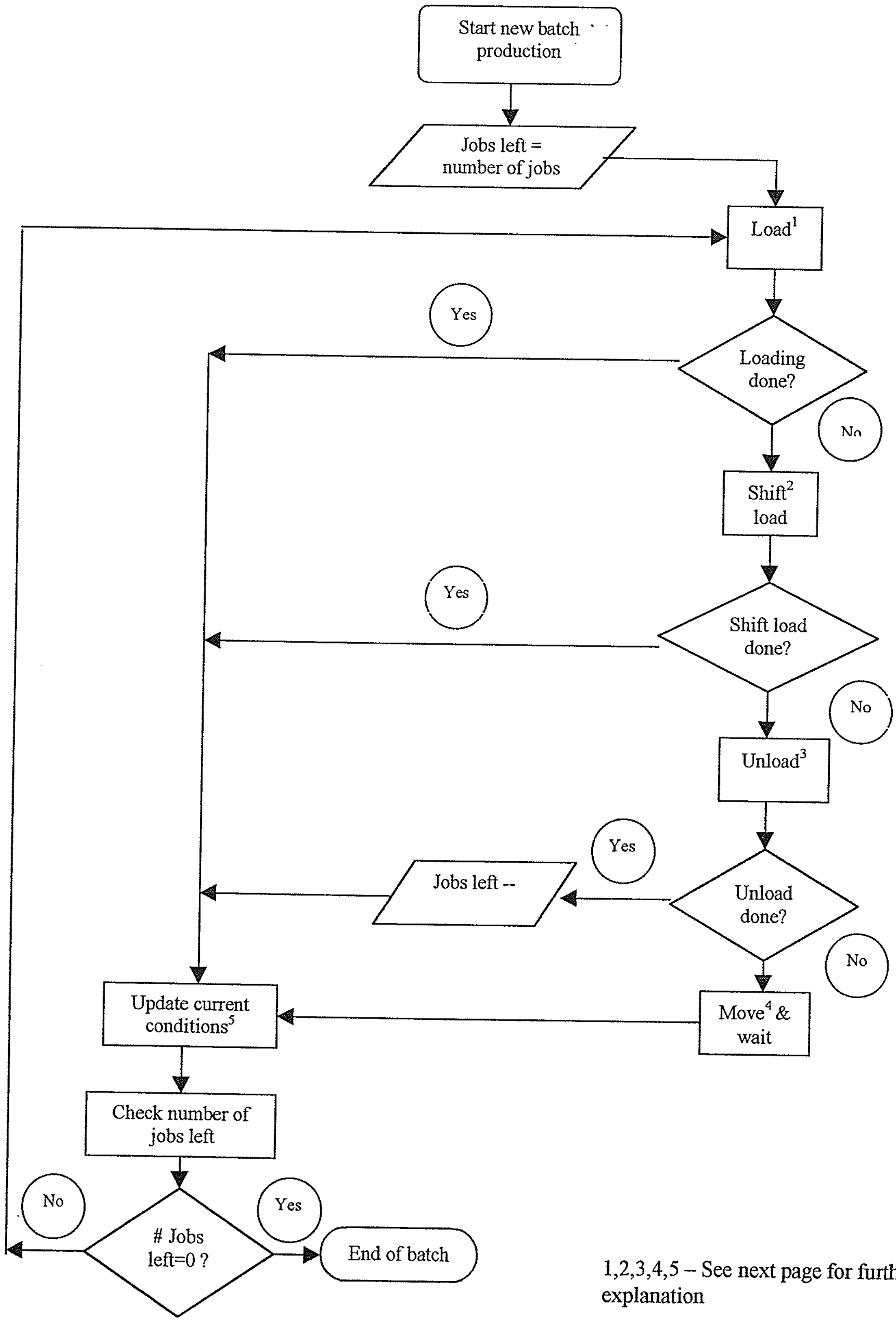
Appendix E - Structure of file "type *.scn"

```
Robot is moving to 0 to load job #1
Robot is moving to 1 to load job #1
Robot is moving to 0 to load job #6
Robot is moving to 3 to load job #6
Robot is moving to 0 to load job #4
Robot is moving to 4 to load job #4
Robot is moving to1 to unload job #1
Robot is moving to 2 to load job #1
Robot is moving to3 to unload job #6
Robot is moving to 1 to load job #6
Robot is moving to 0 to load job #7
Robot is moving to 3 to load job #7
Robot is moving to2 to unload job #1
Robot is moving to 6 to load job #1
Robot is moving to4 to unload job #4
Robot is moving to 2 to load job #4
Robot is moving to 0 to load job #5
Robot is moving to 4 to load job #5
Robot is moving to1 to unload job #6
Robot is moving to 6 to load job #6
Robot is moving to3 to unload job #7
Robot is moving to 1 to load job #7
Robot is moving to2 to unload job #4
Robot is moving to 6 to load job #4
Robot is moving to4 to unload job #5
Robot is moving to 2 to load job #5
Robot is moving to1 to unload job #7
Robot is moving to 6 to load job #7
Robot is moving to 0 to load job #2
Robot is moving to 1 to load job #2
Robot is moving to2 to unload job #5
Robot is moving to 6 to load job #5
Robot is moving to1 to unload job #2
Robot is moving to 2 to load job #2
Robot is moving to 0 to load job #3
Robot is moving to 1 to load job #3
Robot is moving to2 to unload job #2
Robot is moving to 6 to load job #2
Robot is moving to1 to unload job #3
Robot is moving to 2 to load job #3
Robot is moving to2 to unload job #3
Robot is moving to 6 to load job #3
Idle robot time is: 8
The total machine idle cost is: 1552
The tardiness cost is: 1057
The throughput time is: 208
```

Note:
0, refers to input buffer
1,2,3,4,5, refer to machine station number
6, refers to output buffer

```
Time:0     qdone00000
Time:5     qdone00000
Time:10    qdone10000
Time:15    qdone10000
Time:20    qdone10000
Time:25    qdone13000
Time:30    qdone13400
Time:35    qdone34000
Time:40    qdone34000
Time:45    qdone42000
Time:50    qdone42100
Time:55    qdone42100
Time:60    qdone42100
Time:65    qdone42130
Time:70    qdone41300
Time:75    qdone13000
Time:80    qdone13200
Time:85    qdone13200
Time:90    qdone13240
Time:95    qdone13240
Time:100   qdone32400
Time:105   qdone24000
Time:110   qdone24100
Time:115   qdone24100
Time:120   qdone41000
Time:125   qdone10000
Time:130   qdone12000
Time:135   qdone12000
Time:140   qdone20000
Time:145   qdone20000
Time:150   qdone21000
Time:155   qdone21000
Time:160   qdone10000
Time:165   qdone00000
Time:170   qdone00000
Time:175   qdone20000
Time:180   qdone21000
Time:185   qdone21000
Time:190   qdone10000
Time:195   qdone00000
Time:203   qdone20000
Time:208   qdone20000
```

Appendix F - Loading priority logic



1,2,3,4,5 – See next page for further explanation

1

LOAD

Robot picks up a selected part from input buffer and loads it on the respective machine.

Priority:
1. Fuzzy-job or Fuzzy-machine
2. According to part order

Conditions:
- Machine available
- No conflict with other parts
- If process is sequential, part must be the first one in the sequence.

2

SHIFT LOAD

Robot unloads a finished part from a machine, and loads it onto a new machine.

Priority:
First part done

Conditions:
- Machine available
- No conflict with other parts
- If process is sequential, part must be the next one required in the sequence.

3

UNLOAD

Robot picks up a finished part from a machine and drops it off at the output buffer.

Priority:
First part done

4

MOVE AND WAIT

Robot moves to a machine and waits until current processing is finished.

Priority:
Part almost done with shortest remaining processing time.

Conditions:
- If part needs to visit other machine, machine has to be available
- No conflict with other parts
- If process is sequential, part must be the next one required in the sequence.

5

UPDATE CURRENT CONDITIONS

Controller updates:

- Timers
- Counters
- Queues
- Job matrixes

Appendix G - Batch data

G.1 Batches used for real time versus simulation experiments

Batch #1

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	3	1,2,3	5	8	0	0	90	4	39	22.5	3
B	2	4,5	0	3	0	2	110	2	10	55	4
C	2	6,7	4	0	6	0	100	3	20	33.3	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

DD(p) – Due date of part type p
P(p) – Penalty of part type p
PT(p) – Total processing time of part type p
TP(p) – Total penalty of part type p, defined by: DD(p) / P(p)
C(p) – Total machine idle cost for part type p.[1]

Batch #2

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	4	1,2,3,4	28	23	0	0	240	1	204	240	3
B	5	5,6,7,8,9	0	21	0	29	270	2	250	135	4
C	3	10,11,12	0	0	25	0	210	3	75	70	1

Machine idle cost rate: M1=1; M2=2; M3=1; M4=3

Batch #3

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	3	1,2,3	45	48	0	0	280	4	279	70	3
B	2	4,5	0	48	0	42	220	2	180	110	4
C	2	6,7	45	0	46	0	220	3	182	73.3	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

[1] For further details refer to Section 3.3

Batch #4

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	5	1,2,3,4,5	30	25	0	0	480	3	275	160	3
B	5	6,7,8,9,10	0	25	32	0	460	2	285	230	4
C	5	11,12,13,1415	0	0	32	27	440	2	295	220	5

Machine idle cost rate: M1=2; M2=1; M3=3; M4=2

G.2 Batches used for sequential versus non-sequential experiments

Batch #1

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	3	1,2,3	5	8	0	0	90	4	39	22.5	3
B	2	4,5	0	3	0	2	110	2	10	55	4
C	2	6,7	4	0	6	0	100	3	20	33.3	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

Batch #2

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	3	1,2,3	55	58	0	0	340	4	339	85	3
B	2	4,5	0	53	0	52	280	2	210	140	4
C	2	6.7	54	0	56	0	300	3	220	100	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

Batch #3

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	3	1,2,3	95	98	0	0	580	4	579	145	3
B	2	4,5	0	93	0	92	420	2	370	210	4
C	2	6.7	94	0	96	0	480	3	380	160	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

Batch #4

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	4	1,2,3,4	8	3	0	0	80	1	44	80	3
B	5	5,6,7,8,9	0	1	0	9	90	2	50	45	4
C	3	10,11,12	0	0	5	0	70	3	15	23.3	1

Machine idle cost rate: M1=1; M2=2; M3=1; M4=3

Batch #5

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	4	1,2,3,4	88	83	0	0	840	1	684	840	3
B	5	5,6,7,8,9	0	81	0	89	870	2	850	435	4
C	3	10,11,12	0	0	85	0	760	3	255	253.3	1

Machine idle cost rate: M1=1; M2=2; M3=1; M4=3

Batch #6

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	5	1,2,3,4,5	8	12	0	0	180	3	100	60	3
B	5	6,7,8,9,10	0	5	10	0	160	2	75	80	4
C	5	11,12,13,14,15	0	0	3	5	140	2	90	70	5

Machine idle cost rate: M1=2; M2=1; M3=3; M4=2

Batch #7

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	3	1,2,3	15	18	0	0	120	4	99	30	3
B	2	4,5	0	18	0	12	140	2	60	70	4
C	2	6,7	15	0	6	0	130	3	62	43.3	4

Machine idle cost rate: M1=2; M2=1; M3=2; M4=3

Batch #8

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	4	1,2,3,4	48	48	0	0	440	1	384	440	3
B	5	5,6,7,8,9	0	48	0	48	470	2	480	235	4
C	3	10,11,12	0	0	48	0	310	3	144	103.3	1

Machine idle cost rate: M1=1; M2=3; M3=1; M4=2

Batch #9

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	5	1,2,3,4,5	10	5	0	0	180	3	75	60	3
B	5	6,7,8,9,1 0	0	5	12	0	160	2	85	80	4
C	5	11,12,13 , 14,15	0	0	12	7	140	2	95	70	5

Machine idle cost rate: M1=2; M2=1; M3=3; M4=2

Batch #10

Part type	# parts	Jobs #	M1	M2	M3	M4	DD(p)	P(p)	PT(p)	TP(p)	C(p)
A	5	1,2,3,4,5	50	45	0	0	580	3	475	193.3	3
B	5	6,7,8,9,1 0	0	45	52	0	560	2	485	280	4
C	5	11,12,13 , 14,15	0	0	52	47	540	2	495	270	5

Machine idle cost rate: M1=2; M2=1; M3=3; M4=2