Fuzzy Control of Flexible Manufacturing Systems

Paolo Dadone

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Hugh F. VanLandingham, Chair
John S. Bay
Hanif D. Sherali

December 5, 1997
Blacksburg, Virginia

Keywords: Fuzzy logic, Scheduling, Evolutionary programming

Fuzzy Control of Flexible Manufacturing Systems

Paolo Dadone

(ABSTRACT)

Flexible manufacturing systems (FMS) are production systems consisting of identical multipurpose numerically controlled machines (workstations), automated material handling system, tools, load and unload stations, inspection stations, storage areas and a hierarchical control system. The latter has the task of coordinating and integrating all the components of the whole system for automatic operations. A particular characteristic of FMSs is their complexity along with the difficulties in building analytical models that capture the system in all its important aspects. Thus optimal control strategies, or at least good ones, are hard to find and the full potential of manufacturing systems is not completely exploited.

The complexity of these systems induces a division of the control approaches based on the time frame they are referred to: long, medium and short term. This thesis addresses the short-term control of a FMS. The objective is to define control strategies, based on system state feedback, that fully exploit the flexibility built into those systems. Difficulties arise since the metrics that have to be minimized are often conflicting and some kind of trade-offs must be made using "common sense". The problem constraints are often expressed in a rigid and "crisp" way while their nature is more "fuzzy" and the search for an analytical optimum does not always reflect

production needs. Indeed, practical and production oriented approaches are more geared toward a good and robust solution.

This thesis addresses the above mentioned problems proposing a *fuzzy scheduler* and a *reinforcement-learning* approach to tune its parameters. The learning procedure is based on evolutionary programming techniques and uses a performance index that contains the degree of satisfaction of multiple and possibly conflicting objectives. This approach addresses the design of the controller by means of language directives coming from the management, thus not requiring any particular interface between management and designers.

The performances of the *fuzzy scheduler* are then compared to those of commonly used heuristic rules. The results show some improvement offered by fuzzy techniques in scheduling that, along with ease of design, make their applicability promising. Moreover, fuzzy techniques are effective in reducing system congestion as is also shown by slower performance degradation than heuristics for decreasing inter-arrival time of orders. Finally, the proposed paradigm could be extended for on-line adaptation of the scheduler, thus fully responding to the flexibility needs of FMSs.

Acknowledgements

The work of this thesis as well as my scientific growth and maturation was inspired, supported and encouraged by my advisor Dr. Hugh F. VanLandingham. Being always available to stimulating discussions with a very friendly attitude made my work much easier and pleasant. I will always be indebted to Dr. VanLandingham for being my mentor and having faith in my ideas and me.

Some special thanks are also due to the members of my committee, Dr. John S. Bay and Dr. Hanif D. Sherali. Dr. Bay's preciseness and attention in revising my draft was very appreciated. I am also very thankful to Dr. Sherali for giving me such a rigorous and insightful introduction to the world of optimization. Thank to Dr. Bruno Maione, my Italian advisor, for introducing me to the scheduling problem from a very valuable system theoretic perspective. I am also indebted to every professor I had during my college education for teaching me the things I know, but most of all for contributing in creating my person and forging me.

The biggest thanks are for the persons that made all this possible, those to whom I owe for being what I am now and to whom I dedicate this study: my parents. My father Andrea and my mother Antonella inspired my desire for thorough and in depth

education as well as the love for traveling and interaction with diverse realities, a big part of this experience. They not only inspired but also supported all my endeavors confirming the fact that I can always count on them.

The familiar places and friendly people around me made everything much easier while performing my research work in Blacksburg. My first and good friends in Blacksburg, Christos Kontogeorgakis and Timur Oral, let me have a very enjoyable transition from Italy to the States. The long and exhausting discussions with Christos have always been very insightful and valuable for me. Timur's extreme kindness along with his jovial spirit gave me relief when overloaded with work. I want to thank Craig Pendleton and Lee Williams for their companionship in partying as well as in studying, thank to them I could understand more about the American culture and about people. Heartfelt thanks go to Semih and Jane Olcmen for having been my supportive friends here in Blacksburg. The nice Blacksburg with its calm life and splendid surroundings, the snow, the frozen trees and the red-brown leafs of fall were a very nice contour to the research work. Final thanks go also to all my Italian friends regardless of time and distance they are always successful in making me feel at home when I am with them.

TABLE OF CONTENTS

TABLE OF FIGURES

INDEX OF TABLES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The increasing demand for low cost, low-to-medium volume production of modular goods with many different variations creates the need for production systems that are flexible and that allow for small product delivery times. This leads to production systems working on small batches, having low setup times and mainly characterized by many degrees of freedom in the decision making process. This type of system is known as flexible manufacturing systems (FMS). They are highly automated systems with built in redundancies that should provide the desired flexibility. Even though there is no universally accepted definition of what an FMS is, we can refer to the ones given by Tempelmeier and Kuhn [51] and Viswanadham and Narahari [53]. Thus, a FMS can be defined as a *production system consisting of identical multipurpose numerically controlled machines (workstations), automated material and tools handling system, load and unload stations, inspection stations, storage areas and a hierarchical control system*. For a more practical definition of an FMS (i.e., number of workstations, different parts, etc.) we can refer to a literature study by Young-On [61] stating that 60% of the FMSs in the world have less than eight workstations and work on less than ten different parts.

A FMS has some built in "hardware" redundancies that should give it the necessary flexibility. These redundancies, although useful, create difficult control problems, i.e. the search for the "best" action to take in a particular situation. The real problem in achieving the desired flexibility presently consists of finding a correct "solution" of the control problem. Such a problem is typically divided into three hierarchical problems characterized by the time frame they refer to: long, medium and short-term control. The long-term control of a manufacturing plant, or strategic production planning, is the level at which long-term strategies for the production of product groups over a large time horizon are studied, and approximate plans are formulated. The medium-term control, or tactical planning, has the objective of

determining the best tactics (e.g., demand forecasting, requirements planning, maintenance planning, distribution planning, etc.) to use on a medium-term to operate the plant according to what specified at the strategic production planning level. Finally, the short-term control problem, or operational control, deals with managing the plant on a short-term basis, that is, managing the details of its operations according to what was set at the tactical planning level. The short-term production scheduling of a FMS plays a key role in achieving the desired flexibility (Fanti et al. [16]). This study will focus on the short term control, or scheduling, problem.

The most appropriate control of an FMS requires real time control and state feedback. This way all the dynamic characteristics of the system are taken into account. This means that job schedules cannot be predetermined (i.e., predictive scheduling) but they need to be adapted to the current state and goals during the evolution of the system (i.e., reactive scheduling). Moreover, the best decisions can be taken by using all the available information on the state of the system. This approach makes the problem quite complex. Apart from the intrinsic difficulty of the problem, once a solution for a specific configuration is found, the large variability existing across FMS plants makes it hard to easily transfer and adapt the solution to another plant. That is, there is no unique solution applicable to every plant. Moreover, the *optimal* solution must be characterized by robustness and speed necessary for correct operation of the manufacturing plant. Another issue is given by the uncertainty on state measurements (i.e., the information that is used for controlling the plant) and by the necessity of interfacing the FMS with human operators.

The short-term control of an FMS, i.e., scheduling, can be divided in four logic macro components:

- *timing*: that is, deciding when to insert an order into the system;
- *sequencing*: that is, deciding the ordering of orders to be inserted in the system;
- *routing*: that is, deciding where to send a job for an operation in case of multiple choices;

- *priority setting*: that is, deciding the priority for jobs, workstations and resources in general, such that a decision is implicitly made based on priorities.

Another important issue is defining what the "goal" is. Indeed, it is not only important to know qualitatively what is the production objective, but also to be able to quantify it with something measurable. The "goal" in a FMS is described by several objectives that in general are, or may be, conflicting and vaguely formulated. As such objectives we can consider (Viswanadham and Narahari [53], Tempelmeier and Kuhn [51], Young-On [61]):

- low time in system for each job (i.e., total time that a job spends into the system);
- low work in progress (WIP, i.e., number of jobs in the system);
- high resource utilization (i.e., percent of time that the resource is utilized);
- low tardiness (i.e., delay in producing an order with respect to the due date);
- high throughput (i.e., number of parts that exit the system).

A very common approach to scheduling is to use heuristic rules. This approach offers the advantage of good results with low effort but is very limited since it fails to capture the dynamics of the system. Moreover, there is no established set of rules that is *optimal* for every FMS since the success of these rules obviously depends on the particular FMS at hand. Thus, it is known that some set of rules gives good results, but deciding which particular rules are the best for a particular configuration has to be done by trial and error.

In practice, human experts are the ones that, by using practical rules, make an FMS work to the desired objective. This leads to the idea of a scheduling approach that mimics the behavior of human experts, that is the emerging field of *intelligent manufacturing* (Parsaei and Jamshidi [4]). The literature offers a wide variety of *intelligent* techniques for the scheduling of manufacturing systems. Namely, fuzzy logic systems (FLS), artificial neural networks (ANN), genetic algorithms (GA), artificial intelligence (AI) and hybrid systems are used in scheduling. AI based systems (i.e., more precisely expert systems) are useful in scheduling because of their ease in using rules captured from human experts, even though presenting some limitations.

ANN based systems offer the main advantage of learning from historical data of the system. Moreover, they can be implemented via hardware, thus offering excellent speed characteristics. FLSs easily deal with uncertain and incomplete information, and human experts knowledge can be easily coded into fuzzy rules. GAs are a good optimization tool for "black box" type optimization problems that, if correctly used, can be instrumental in a successful solution of the scheduling problem. Finally, hybrid systems present the advantage of combining all the features of the approaches described above. From a general point of view these intelligent techniques give a increased flexibility in scheduling operations in a FMS, thus giving better overall performance. The limit of most of these techniques lies in either the absence of a design procedure or the use of a limited number of objectives in the design. The same holds for the evaluation of these approaches, that is generally done considering only a few management objectives, thus failing to take into account the whole complexity of the problem. Finally, most of the techniques presented in the literature have the disadvantage of not having adaptive properties, even from a potential point of view.

This thesis addresses the scheduling problem in a FMS by using fuzzy techniques. Chapter 2 presents the state of the art on intelligent techniques for scheduling of FMS, along with a short description of commonly used scheduling rules. In Chapter 3 the problem assumptions are described along with the proposed fuzzy scheduler (FS). This scheduler addresses three scheduling problems: sequencing, routing and priority setting for jobs in a machine buffer. The design procedure for such a scheduler is heuristic in nature. In fact discrete event simulations of the particular system are used to determine all the parameters (membership functions and rules) characterizing the FS. To generalize the FS design, a reinforcement learning approach (Dadone et al. [12]) based on evolutionary programming (EP) is introduced, thus leading to what was called an evolutionary fuzzy scheduler (EFS). Apart from the obvious advantage of being a general design procedure, the reinforcement learning approach is promising in terms of future extension to methodologically similar techniques for real time adaptive control of FMSs.

Chapter 4 describes the particular FMS considered in this study, along with the developed FS and EFS and their performances, compared to those of heuristic rules. Finally, Chapter 5 presents the conclusions of this study.

1.2 DEFINITIONS

This Section introduces the definitions of some of the terms that will be used in the following chapters.

A *part* is an unfinished piece of material that will be subjected to processing and eventually assembly with other *parts* to produce a finished product.

An *order* is a commission to supply finished products. In general an *order* is said to "enter the system" when the production to supply that *order* starts.

A *batch* is a number of parts generally moving altogether in the shop-floor.

A *job* is the collection of activities necessary to do the work required, i.e., manufacture a final product. Many times parts, orders, or batches are called *jobs* to indicate the underlying set of activities.

The *total processing time* of a job is the sum of all the processing times of the job.

The *time in system* of a job is defined as the time elapsed from the instant the job is in the loading buffer of the system and the time the job exits the system.

The *tardiness* (or *lateness*) of a job is the minimum between zero and the difference between the time a job leaves the system and its due date. Every negative value of the difference shows a job finished before its due date, thus a job with zero *tardiness*.

The *slack* of a job is the difference between the job due date and the sum of actual time and all the remaining processing times for the job. The *slack* of a job is the idle time margin that the job can afford without being late.

A *resource* is anything that can be used by jobs in the plant. Examples of resources are machines, AGVs, conveyor belts, etc.. A *resource* can be *busy* (i.e., serving a job), *idle* (i.e., not serving any job) or *down* (i.e., not working, a failure happened). A resource is *up* whenever it is not *down*. The *up-time* of a *resource* is the time the *resource* is *up*, that is, the time the *resource* is either *busy* or *idle*.

A *terminating simulation* is a simulation of a system starting from a given initial event and ending with a given final event. Examples of *terminating simulations* are simulations of a bank, of stores or in general of systems that have a opening and closing time.

A *non-terminating simulation* is a simulation of a system where no terminating event exists and the analyst is interested in the steady state behavior of the system. Examples of *non-terminating simulations* are simulations of a completely automated manufacturing plant or of 24 hours a day activities in general. In a *non-terminating* simulation the simulation is ended whenever a certain number of observations deemed enough to estimate the steady state behavior of the system are collected.

CHAPTER 2
INTELLIGENT SCHEDULING: STATE OF THE ART

2.1 INTRODUCTION

This chapter presents some of the many applications of *intelligent control*[1] of FMSs. The following Section §2.2 presents a short review of the heuristic rules used in scheduling. This section is important for understanding the results discussed in the surveyed papers, as well as in this work. The literature has many studies on these heuristic rules, thus the interested reader is referred to references for further details. The following sections present the *intelligent control* of FMS depending on the type of techniques that were used. Therefore, Section §2.3 presents the application of fuzzy techniques, while Section §2.4 presents the application of artificial neural networks to the scheduling problem. Finally, Sections §2.5 and §2.6 present the application of AI techniques, hybrid systems and genetic algorithms.

2.2 HEURISTIC RULES

Heuristic rules are often used in real time scheduling of FMSs. Such rules are effective in practice, but they are also short-sighted in nature. They do not have any predictive and/or adaptive properties (nor can they by nature), thus their success depends on the particular plant that is under study and on the control objectives. Heuristic rules do not account for the dynamic characteristics of the controlled system. The following rules refer only to particular aspects of the scheduling problem, that is, to the ones of interest for the present study. These rules will be briefly presented, more

[1] Narendra and Mukhopadhyay [40] define an intelligent control system as something able to: capture information from the environment and process them reducing uncertainty, planning, generate and execute control actions based on what is learned from examples and from actual plant operations. Antsaklis [3] points out some factors as being essential to intelligent control: learning capabilities (from examples and from occurred situations), autonomous behavior and adaptive characteristics non typical of traditional control systems. These features characterize a completely automatic control system having good performance even under the effect of changing operating conditions and environment. This definitely requires adaptive features that allow for significant changes in the operating region of the control system.

precise descriptions can be found in Young-On [61], Yao [59] and Joshi and Smith [31].

The rules presented in the following are concerned with:

1. *sequencing*: that is, deciding the ordering of orders to be inserted into the system;
2. *routing*: that is, deciding where to send a job for an operation in case of multiple choices;
3. *priority setting* for a job in a machine buffer: that is, deciding which will be the next job to be served by a machine.

Some sequencing rules are:

- EDD[2]: the first order that enters the system is the one with the earliest due date;
- FIFO[3]: the first order that enters the system is the one that arrived first;
- LTPT[4]: the first order that enters the system is the one with the longest total processing time;
- STPT[5]: the first order that enters the system is the one with the shortest total processing time.

Some routing rules are:

- RAN[6]: the next workstation is randomly chosen;
- SQL[7]: the next workstation is the one with the shortest queue length;
- SQW[8]: the next workstation is the one with the shortest queue workload (the queue workload is defined as the sum of the processing times required by all the jobs waiting to be processed);

Finally, some priority setting rules for jobs in a machine buffer are:

- EDD: the first job to be processed is the one with the earliest due date;

[2] **E**arliest **D**ue **D**ate.
[3] **F**irst **I**n **F**irst **O**ut.
[4] **L**ongest **T**otal **P**rocessing **T**ime.
[5] **S**hortest **T**otal **P**rocessing **T**ime.
[6] **RAN**dom.
[7] **S**hortest **Q**ueue **L**ength.
[8] **S**hortest **Q**ueue **W**orkload.

- FIFO: the first job to be processed is the one that arrived first;
- HPFS[9]: the first job to be processed is the one that gives the highest profit;
- LIFO[10]: the first job to be processed is the one that arrived last;
- LS[11]: the first job to be processed is the one with the least *slack*;
- MDD[12]: it is a modified version of the EDD;
- MODD[13]: it is another modified version of the EDD;
- SPT[14]: the first job to be processed is the one with the shortest processing time (on that operation);
- SPT/TPT[15]: the first job to be processed is the one with the lowest processing time (on that operation) to total processing time ratio.

2.3 FUZZY LOGIC SYSTEMS

Fuzzy set theory was introduced in 1965 by Zadeh [62]. Fuzzy sets and their extension to dealing with linguistic variables (Zadeh, [63]) were later successfully employed in many engineering applications. Fuzzy sets are also particularly useful in control problems, due to the development of fuzzy logic systems (FLS), widely described in the literature (e.g., Kosko [34], Mendel [38]). Using fuzzy logic to control flexible manufacturing systems seems very appropriate due to its easiness in dealing with uncertain data, along with the multi-objective nature of the problem.

Hintz and Zimmermann [25], are probably the first to propose a production planning and control system that uses fuzzy set theory. The interesting part of their work consists in the application of approximate reasoning techniques to both the

[9] **H**ighest **P**rofit **F**irst **S**erved.
[10] **L**ast **I**n **F**irst **O**ut.
[11] **L**east **S**lack.
[12] **M**odified Job **D**ue **D**ate.
[13] **M**odified **O**peration **D**ue **D**ate.
[14] **S**hortest **P**rocessing **T**ime.
[15] **S**hortest **P**rocessing **T**ime/**T**otal **P**rocessing **T**ime.

sequencing and the priority setting problems. The authors develop a hierarchy of elements that are important to make a decision in both cases. Those elements become the antecedents of the fuzzy rules in the scheduling procedure. This methodology is quite general, thus it can be easily modified and extended by changing the antecedents. The consequent of the rules are the next job to be entered into the system (sequencing) or to be processed (priority setting). The antecedents of the fuzzy rules for sequencing are:

- slack time;
- waiting time;
- utilization uniformity across machines;
- unguarded (i.e., with no personnel) utilization of machines;
- external priority (i.e., priority given by external sources such as management, etc.).

Each antecedent is divided into three fuzzy sets with trapezoidal membership functions. All the sequencing rules are stored in a knowledge base, along with their "strength". In the priority setting problem, the same mechanism but different antecedents are used, that is:

- slack time;
- waiting time;
- machine up-time;
- alternative machines up-time;
- queue length in alternative machines.

The performance of this fuzzy controller is compared to common heuristics by means of discrete event simulations of a particular FMS configuration (Hintz and Zimmermann [25]). As a result, fuzzy expert systems seem to perform better then heuristics in terms of:

- mean waiting time;
- number of in-time (i.e., not late) parts;
- mean machine utilization.

This approach is very innovative for introducing a fuzzy expert approach to scheduling, but it also suffers from being an early approach, in that it only considers sequencing and priority setting. Moreover, the scheduling rules are predetermined with human expert aid and no explicit design procedure is presented. The multiple objective nature of the problem is also not thoroughly investigated, because the comparison with heuristic approaches is done on a limited number of production objectives, whereas tardiness, throughput and work in progress are not considered.

Choobineh and Shivani [10] approach the priority setting and routing problems using fuzzy set theory along with possibility theory. Fuzzy sets are used to model the uncertainty of data and the vagueness involved with human planning. For every possible routing of a part, an aggregate possibility distribution is determined according to the possibility distributions of single attributes of a resource. These possibility distributions are combined into one aggregate possibility distribution, by means of a weighted average, with weights being (trapezoidal) fuzzy numbers expressing the importance of the given attribute (i.e., indifferent, not important, somehow important, important, very important). From the aggregate possibility distribution an appropriateness index of the given routing is determined with a center of gravity defuzzifier. Basically, appropriateness indices are determined according to the comparison of the time a part would spend in a machine with the time it would spend in alternate machines. Therefore, a kind of *minimum time* routing is determined, with an additional constraint given by the different importance of each machine. In the priority setting, highest priority is given to parts that have no alternative route. For parts that have at least one alternative, an appropriateness index for every possible route is determined eventually re-routing parts to alternatives with higher appropriateness index.

In this study no comparisons with standard heuristics are presented, moreover the multiple objective nature of the problem is not accounted for, since WIP, due dates, utilization and tardiness are not explicitly considered.

Hatono et al. [22] develop an *intelligent* modular controller for a multiple objective scheduler. This scheduler is composed by three modules, each one having its own fuzzy logic system and a particular goal (function):

1. schedule evaluation module;
2. schedule policy module;
3. dispatching module.

The schedule evaluation module evaluates a given schedule by means of fuzzy rules having as antecedents:

- resource utilization;
- estimated waiting time at a resource;
- earliness of a batch;
- parts waiting time.

The consequent for these rules is an evaluation of the schedule. The second module (schedule policy) determines the influence of each factor (utilization, waiting time, etc.) on the schedule evaluation. The output of this module are the weights to give to rules in the last module in order to increase the next schedule evaluation. The dispatching module uses some predetermined fuzzy rules to dispatch parts. Each rule effect is amplified or attenuated through the weights determined in the schedule policy module. Each module uses singleton fuzzification, max-min inference and max defuzzifier. No comparison results for this approach were presented.

This modular architecture has the advantage of allowing for objective modification by simply changing the membership functions in the schedule evaluation module. Some production objectives are missing in this formulation but the type of structure seems to allow for further enhancement and addition of objectives. The idea of a scheduler that adjusts itself on-line in order to increase the quality of the schedule is very interesting. Unfortunately it might be hard to determine the correct influences of each factor on the production objectives in real problems. Indeed no way to automatically determine the rules in the schedule policy module is given, and in this case they were determined through the aid of experts. In moderately complex

problems determining the way to adjust the weights of rules in the dispatching module in order to increase the schedule quality might not be a trivial task. Thus, this seems to be the weak point of this approach, or at least the point that needs further study.

Watanabe et al. [55] propose a fuzzy scheduling mechanism for job shops, that they name FUZZY. The only problem that they actually attack is the priority setting problem for a free machine choosing in its buffer the next job to serve. The authors consider clients demands and divide the orders into three categories: normal, express and just in time (JIT). A different profit curve corresponds to every type of order. Two parameters are determined from this profit curve and from an estimated time to manufacture a piece. The first one is the velocity index, that measures how fast a piece should produced by the manufacturing plant. The second parameter is a profit index which measures the profit that the order will generate. Two fuzzy rules use these indices as antecedents in the determination of parts priorities. The rules are as follows:

Rule 1: If the velocity index is *LOW* and the profit index is *HIGH*
then the priority is *HIGH*

Rule 2: If the velocity index is *HIGH* and the profit index is *LOW*
then the priority is *LOW*

The proposed fuzzy scheduler employs non-singleton fuzzifier, max-min inference and center of gravity defuzzifier. All the membership functions are triangular.

The fuzzy scheduler was then tested through computer simulations and compared to common priority setting heuristics, i.e., SPT, LS and HPFS. In all the benchmark tests FUZZY produced the highest profit, but only average tardiness performance. Watanabe's work is limited to one particular aspect of scheduling and does not consider some important objectives like WIP, throughput and utilization. The proposed fuzzy technique is very limited in that it only uses two rules and two fuzzy sets for each antecedent.

Grabot [20] proposed a routing mechanism that embodies expert knowledge and that reacts to resource failures by using fuzzy logic and possibility theory. The proposed method allows a reaction to order changes and resource failures with better satisfaction of production objectives by changing a predetermined schedule. Indeed, this technique is based on the (reasonable) assumption that the FMS already has a predictive scheduler. Thus, in case of unforeseen changes the schedule has to be modified to react to the changes, while maintaining satisfactory performance. The first step is obviously the modeling of production objectives. The second step consists in finding different ways to utilize resources, taking into account the unforeseen changes that happen. Finally, the third step evaluates the various feasible solutions according to the satisfaction of production objectives.

Since objective satisfaction is naturally dealt with in terms of degrees of satisfaction, fuzzy logic and possibility theory seem to be the natural tools to deal with it. A global complex objective is divided into several "atomic" objectives. The degrees of satisfaction with respect to every single objective are evaluated and connected through an AND connector. In this procedure every single objective is also weighted depending on its importance with respect to the global objective. This produces the degree of satisfaction of a feasible choice with respect to the global objective. The process is repeated for every possible choice, and the one with the highest satisfaction of the overall objective is chosen. This technique is very similar to Saaty's criterion [46] in his fuzzy extension, as in Yager [57,58]. The only difference being that the weights for every "atomic" objective are given in an absolute way by human experts and are not determined through a pairwise comparison matrix. This is a limit of the present method that can be easily overcome by changing the setup for the objective weights.

In case of a resource failure there are several actions that can be taken due to the several degrees of freedom still available to the system. These actions are represented through precedence diagramming methods (PDM) that use imprecise fuzzy duration. Every possibility is then evaluated in terms of fuzzy possibility and necessity indices

determined according to the system objectives. Once these indices are determined for every alternative, a decision can be taken.

This structure, a decision support system (DSS), was implemented in a software called the ENIT flexible cell. No comparison with traditional heuristics was presented. In the given example, the only production objectives that were considered are cost and schedule length. Adding and modifying the objective structure should be straightforward. The interesting aspect of this work is the fact that it considers the multiple objective nature of the scheduling problem in evaluating different possibilities. This concept will be considered again later in this thesis.

Angsana and Passino [1,2] seem to be the first to have a more systematic approach to the problem. They present a fuzzy controller for the priority setting problem along with a procedure that can be used for both design and adaptation. This constitutes the real novelty of their work, even though at a very preliminary stage. The authors at first consider the problem of a single machine and build a fuzzy controller (FC) for it. Considering an FMS where every machine has such a controller a distributed fuzzy controller (DFC) is obtained. It is assumed that each machine has a different buffer for every part type. By using the buffer levels they implement a fuzzy version of the clear largest buffer (CLB) heuristic policy. This policy tries to empty the fullest buffer giving priority to the parts it contains. This is done in a fuzzy way, that is, using some fuzzy sets that define buffer levels (at least three of them, low, medium and high). The universe of discourse of these fuzzy sets is the set of real nonnegative numbers. The number of fuzzy sets to use is one of the arguments of study. The authors conclude that it is not always better to use a large number of fuzzy sets. The membership functions are triangular, and to be more specific, isosceles triangles. They are always built in such a way that some kind of symmetry between all the membership functions is kept. Indeed there are only two parameters that unequivocally determine the membership functions: their number and the maximum level for saturation (M). This last parameter fixes the maximum level after which the membership function

value for "high buffer" will be constantly one. The interval [0,*M*] is then used to fit *N* membership functions completely symmetrical and with α-cuts of 0.5. If *P* different parts and *N* fuzzy sets are used, a fuzzy controller for a machine will employ P^N fuzzy rules. This might be a problem in terms of memory requirements for problems of large size. As previously mentioned, the fuzzy rules implement a fuzzy version of the CLB policy. Thus, the antecedents are the levels of each buffer and the consequent is the buffer number that is given the highest priority. For example, if we consider three part types with buffers b_1, b_2 and b_3 and corresponding buffer levels x_1, x_2 and x_3 and three fuzzy sets for each buffer, there will be 27 rules. Each rule will have three antecedents (the three buffer levels) and one consequent (the index of the part type that is given the highest priority, 1, 2 or 3). Some sample rules will be:

If x_1 is *SMALL* and x_2 is *MEDIUM* and x_3 is *SMALL* then $p^*=2$

If x_1 is *SMALL* and x_2 is *SMALL* and x_3 is *LARGE* then $p^*=3$

If x_1 is *LARGE* and x_2 is *LARGE* and x_3 is *SMALL* then $p^*=1$

In these rules p^* indicates the part type to be sent to the machine. The first and second rules show the fuzzy-CLB policy. The last rule shows that in case of ties (there is more than one buffer with the same fuzzy level) priority is given to parts (or buffers) with the lowest index. This is a sort of biasing of the rules with only a minor influence on the overall behavior for different operating conditions. These rules are also accompanied by a universally stabilizing supervisory mechanism (USSM, Kumar and Seidman [35]). This mechanism will override the fuzzy controller if and when the buffer levels become greater than a level fixed in the design stage. The USSM is very useful in terms of theoretical guarantees of stability. The USSM parameters are chosen in such a way that it is never called unless for high instabilities that with their approach do not occur.

The performances of the FC in the case of a single machine problem are compared to those of conventional rules (Perkins and Kumar [42]). More precisely, the FC is compared with the following heuristics: CLB, *clear a fraction* (CAF), CPK

(from the authors names Perkins and Kumar). The ratio of the average buffer length with the theoretical minimum buffer length is used as a performance index. The conclusion that emerges from the simulation study is that the CPK is the best among the heuristics and the FC performs similarly. The FC performs better than CPK for high machine workload, while the opposite holds for low machine workload. Another important conclusion is that the choice of the *M* parameters (one for every buffer, since the universe of discourse of buffer levels differs in general from buffer to buffer) has a major influence on the FC performance. This calls for the determination of good values of those parameters in the design phase. The authors propose an iterative approach to this problem. An initial value for those parameters is chosen. A simulation is run with these values and the maximum levels on each buffer are used as new values of the *M* parameters for a successive simulation. This procedure is iterated until convergence for the *M* parameters is found. A similar mechanism can also be used to make the FC adaptive. Indeed an initial value of the *M* parameters can be chosen and then updated during the simulation (or real time operation) according to the observations of buffers levels in a sliding window. This way leads to an adaptive fuzzy controller (AFC) where the membership functions parameters change with the evolution of the system and with varying operating conditions adapting to them. This adaptive property is verified with simulations where the workload changes. The AFC is able to adapt itself to these variations, giving very good performance. On the other hand, the CPK exhibits very good performance only for a fixed workload. With varying machine workload its performance ungracefully degrades. Simulation results also point out that the choice of the sliding window size (i.e., number of observations contained in the sliding window), that is generally done in some empirical way, has a major impact on the AFC performances. Indeed a large sliding window implies slower changes and thus slower adaptation along with more stability of the control system. On the other hand, a small window size implies quick changes, thus fast adaptation with potential stability problems. A trade-off between these two opposites must be

found. The AFC requires the use of the USSM since there are no theoretical guarantees of its stability without it.

A distributed fuzzy controller for a non-acyclic flexible manufacturing system (i.e., a manufacturing system where each part can visit a machine more than once) is then presented. Such a DFC is obtained by controlling each machine with an AFC, supervised by the USSM. The performance of the DFC is compared with the previously mentioned heuristics and with another heuristic named *distributed clear a fraction* (DCAF). In this comparison the performance of the controlled system is measured by means of several performance indices that in different ways measure the buffer levels and the backlog of each machine. Maximum values as well as time and machines averages are considered. Several tests are run for different FMS structures and the conclusions are the following:

- the DFC has always good performance, comparable or better than conventional techniques;
- sometimes the DFC is able to achieve smaller maximum buffer levels than conventional techniques, at the cost of higher average buffer levels;
- DFC performance is very sensitive to:
  1. Sliding window dimension: increasing it generally gives better performances at the cost of slower adaptation;
  2. Initial values for M: using an initial value for M coming from a previous simulation increases the performances most of the time;
  3. Number of fuzzy sets: five fuzzy sets generally give good results, sometimes having more than five sets gives better results in terms of some performance indices.

Angsana's conclusions are that the results given by the DFC are promising and that further study on the DFC itself, as well as on the adaptation mechanism, could lead to significant results also in terms of theoretical conditions to aid a systematic parameter selection process for the DFC. The work of Angsana and Passino [1,2] is very valuable as it seems to be the first one to introduce a design procedure along with

an extension to online adaptation of the controller. On the other hand several limitations affect their preliminary work:

- Only the priority setting problem for parts waiting for a machine is considered;
- A fixed workload along with deterministic arrivals (fixed arrival rate) are hypothesized;
- The routing of each part in the FMS is fixed and there is no option for multiple routings;
- The performance indices are always a measure of the buffer levels and do not consider due dates, throughput, time in system and utilization. This way the problem looses one of its main characteristics, that is, a decisional process with multiple objectives.

Sentieiro [47] and Custodio et al. [5] use fuzzy set theory in a non-classic approach called FLAS (fuzzy logic applied to scheduling) for short term planning and scheduling. The proposed approach is hierarchical, with three decisional levels (high, medium, low) each one associated with production problems at different time horizons. Each level targets the short term planning and scheduling problem in a non stationary fashion. The highest level determines safety inventory levels such that accidental machine failures can be compensated for. These safety levels are determined from heuristics according to demand, mean time between machine failures and mean time to repair a machine. The medium hierarchical level calculates the loading rate seeking to minimize the difference between cumulative production and cumulative demand, still maintaining a low WIP. The loading rate is determined according to WIP, difference between production surplus and safety levels determined by the highest module and rate of change of this same difference. The lowest hierarchical level control parts flow through resources using the multiple attributes decision making technique of Yager [58], as modified by Buckley [6]. This method offers the useful advantage of being able to flexibly use several decision criteria. The two highest decision levels are concerned with tactical planning, while the last one is the one

dealing with scheduling. Thus, this one will be discussed in more detail in the following.

The control actions of the scheduling module react to three events. The first event is a job processing completion. Two decisions need to be taken. First, if the part needs further processing and multiple routings are possible, a resource is selected according to the following criteria:

- number of parts in the input buffer of each candidate resource;
- number of parts of the same type waiting in the input buffer of each candidate resource;
- failure vulnerability of each resource;
- transport time to each candidate resource.

Once a resource for the processed part is selected a part has to be chosen for processing by the machine that was freed by the routed part. A part is chosen among those in the input buffer of the resource using the following criteria:

- high processing time;
- high waiting time in the resource buffer;
- high part priority.

The second event that triggers the scheduling module is a resource failure. Two decisions need to be taken in this case. First, the part that was currently processed needs to be either discarded from the system, put back in the input buffer and processed from the beginning once the machine is repaired, or put back in the input buffer and the processing is resumed as soon as the machine is repaired (standby part). Moreover the parts waiting in the input buffer of the failed machine need to be either redistributed in alternative resources (if possible) or kept in the input buffer waiting for the machine to be repaired.

Finally the third event is the end of repair of a failed machine. If there is a standby part its processing is resumed. Otherwise, the next part to be processed is chosen as explained before.

This methodology was tested by simulations with both constant and order (variable) demand. In the presented results a high throughput was noticed, together with a low WIP level. The results obtained with the FLAS were compared to the ones of another similar hierarchical structure not using fuzzy multiple attribute decision making procedures (GAC). The FLAS often performs better than the GAC in terms of throughput and WIP. These results show the advantage of using fuzzy techniques for scheduling, along with the advantage offered by fuzzy logic's ability to embody human experts aid, in changing rules and in dealing with uncertainty. The authors also propose a future development in terms of added adaptivity through automatic on-line fuzzy parameters adjustments.

The FLAS approach takes into account machine failures and reacts to them, something that is very important in a dynamic environment. Moreover, the scheduling structure based on fuzzy multiple attribute decision making has the important feature of being easily modified to change the number of objectives involved in a decision as well as their relative importance. The comparison results are limited to the fact that only throughput and WIP are considered as performance measures, while due dates, lateness and utilization are not.

Ben-Arieh and Lee [4] present a fuzzy logic controller (FLC) for part routing. The fuzzy approach uses some fuzzy rules to determine the selectibility factor for each possible alternative route for a part, the route with the highest selectibility factor is chosen. Every rule has one consequent, the selectibility factor of a possible machine, and four antecedents:

- processing time of the part in the machine;
- queue length in the input buffer of the machine;
- slack of the part going to the machine;
- failure rate of the machine.

These crisp variables are transformed in fuzzy variables through a singleton fuzzifier and then passed to the inference mechanism. This is a max-min mechanism that determines the output for each rule using min inference and then compose all the fuzzy outputs with a max operator. The fuzzy output is then converted to a crisp number through a center of gravity defuzzification. This process gives the selectibility factor of a particular possible machine in the route of a part. By repeating this process for all the possible alternative machines, a selectibility factor is determined for each one. The machine with the highest selectibility factor is chosen as the next machine for the part. Every antecedent can also be weighted with *ad-hoc* weights so that the relative importance of different factors is taken into account. All the rules were created with the observation or aid of human experts. The antecedents have from five to seven membership functions each, and the consequent has five membership functions. All the membership functions are triangular where the parameters are heuristically determined from discrete event simulations of the FMS.

The effectiveness of the FLC was then tested with simulations by comparison with common routing heuristics such as RAN, SPT and SQL. The FLC is always superior to those heuristics in terms of:

- average time in system;
- fraction of late parts;
- average parts lateness;
- number of parts completed.

Moreover, decreasing the time between arrivals all the techniques experience a performance degradation that is less relevant for the FLC than for the heuristics. Thus, the FLC behaves better than heuristic rules especially in critical situations like a congested system. The WIP and utilization of resources are not considered as performance indices, and only the routing problem is considered in this work. Nonetheless, the presented results are promising in terms of application of fuzzy logic to the routing problem.

2.4 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANN) are currently widely used in several engineering applications. These connectionist structures try to mimic the human brain with a distributed neural-synaptic-cognitive structure. Artificial neural networks have greatly matured since the early perceptron and associative memories. In some way they can be regarded as a "overly parametrized" nonlinear function whose weights can be determined by optimizing some measure of performance of the network (generally its "distance" by a set of given test points). For more detailed readings on ANNs, Kosko [34] and Haykin [23] are suggested. ANNs offer advantages like the possibility of learning, the existence of several structures for the attaining of particular objectives, high speed (in the utilization phase) and eventual hardware implementation. On the other hand they might be slow to train and the set of "weights" (parameters) that they finally have do not have a real physical meaning to the user. These, along with the fact that fuzzy systems can be regarded as adaptive networks and thus trained with the same paradigms used for neural networks, make fuzzy logic systems a more suitable means for engineering applications. Anyway, some interesting applications of ANN to the scheduling problem exist in the literature and they are briefly reviewed in the following.

Lo and Bavarian [37] use a Hopfield neural network to predictively solve the assignment problem of parts to resources. This neural network is extended to a three dimensional structure called neuro box network (NBN) where the three axes correspond to time, machine and part. The authors minimize an energy function corresponding to the time needed to execute the schedule with the addition of some terms corresponding to the feasibility of the given schedule. Thus, their approach consists in using an Hopfield neural network to solve a constrained minimization problem having as an objective function the length of the schedule. The results are presented in term of convergence of the method but no comparison with common

heuristics is given. Moreover, only one production objective is considered and the approach is predictive. While generally real-time approaches are preferred, in this case an important factor is the learning speed of the network. Indeed for very high learning speeds (high compared to the time horizon that is considered in the schedule) this predictive approach could work on very small time horizons, thus turning into a good approximation of a real-time scheduling process.

Cho and Wisk [9] present an intelligent workstation controller (IWC) focused on the priority setting problem. The IWC uses a multi-layer perceptron (MLP) with two hidden layers. The MLP has the task of assigning a "grade" to distinct heuristic scheduling policies given a state of the system. The input of the MLP is the state of the system while the output consists of the score, with respect to a selected performance index, of each heuristic scheduling rule. Thus, there is one output node for each heuristic rule. The output of each output node is a real number in [0,1], zero being the lowest score for the correspondent heuristic, and 1 the highest. The MLP was trained using past data. The two best scoring scheduling rules are selected for being passed to a selection module. In this module the rules are tested on a multi-pass simulator where the best one is chosen for execution. Some simulation results show how the IWC is superior to conventional heuristic rules in terms of:

- total processing time;
- throughput;
- mean lateness;
- root mean square lateness;
- number of late parts.

The proposed technique gives good results; it is dynamic and was evaluated on several production objectives. Nonetheless, only one scheduling rule is considered and objectives like WIP and resource utilization were left out in the comparison with heuristics. Moreover, the MLP is structured so that the performance index (*PI*) according to which the scoring of heuristics is done can be selected among some. Only

one *PI* at a time can be considered, though, loosing the completeness of the multiple objective nature of the problem. A final remark regards the training for the MLP. Indeed a lot of data is needed for training purposes and moreover in the scheduling problem it is very hard to determine what is the exact benefit of an action at a certain point in time. Thus, even though not specifically mentioned, the training calls for some judgment on what is the time horizon for the influence of a control action on a given *PI*. This is probably the biggest problem in trying to use conventional MLPs for scheduling. Overcoming this problem would be very useful in terms of added adaptivity to training-based solutions where the training could extend on-line.

Watanabe et al. [55] propose a modified version of the least slack heuristic rule for priority setting among pieces waiting for processing. According to this rule the slack (time to the due date minus further processing time needed) of every part in the input buffer of a machine is calculated and the part with the least slack (least time available for waiting and still being on-time) is selected for processing. Most of the times the slack is not a good measure of the "lateness margin" a part has, since an eventual congestion of the system, or the system state in general, can make a difference on how big a slack can assure avoiding late parts. For this reason the authors propose a modified version of the LS rule where a "effective" slack is estimated by a neural network trained on historical data of the plant. By testing this approach with discrete event simulations it is noted how the mean lateness of parts is decreased by the ANN approach, but the maximum lateness is not. This seems to be caused by inaccuracy in "effective" slack estimation in some apparently non-correlated cases. Thus, a second neural network is used to give a measure of the reliability of the slack estimation of the first network. This solution exhibits a better behavior than the traditional LS rule, in terms of mean and maximum lateness of parts. Both neural networks have one hidden layer with twenty neurons and a single node on the output layer. The first network has an input layer with ten neurons, corresponding to the variables that define the state of the system. The second network has an input layer

with eleven neurons: ten corresponding to the same inputs as in the first network and one corresponding to the output of the first network, that is, the slack estimation. Even though this neural network approach performs better than the heuristic LS in terms of lateness of parts, the difference between the performances of the two methods decreases with increasing problem dimensions. Moreover, for a more complex system the number of variables needed to the describe the system state could dramatically increase thus making this approach slow and hard to train. This work is also limited in considering only the priority setting problem and lateness minimization only and not a multiple objective problem.

Hao et al. [21] propose a three phase decisional structure for the routing problem as well as the selection of the transportation unit (i.e., AGV) to use. The first phase is a filtering stage where among all the routing possibilities the unfeasible ones (e.g., routing to a failed machine) are excluded. For every alternative a neural network will determine an output that can be: unfeasible, feasible with low priority and feasible with high priority. One neural network with one hidden layer is used and no mention is given to its training, besides a "right" choice of weights. In the second phase the results of the first phase are used to determine the best one among all the feasible alternatives. An optimizing modified Hopfield-Tank neural network is used. The stable output of this network corresponds to the routing most appropriate for the current system state. The criteria to determine the best routing are:

- queue length in the input buffer of each alternative;
- workload in the input buffer of each alternative;
- slack of the part on each routing;
- minimum distance;
- machine quality.

This criteria may be also weighted differently. Finally, the third phase determines the proper sequence of actions needed to follow the selected route. A self-organizing Kohonen network is used. This network has the advantage of being initialized with a

single node and to automatically evolve through processes of addition and deletion of nodes.

Hao et al. do not present any comparison result or testing of the approach, even though the implementation on a given FMS configuration is discussed. Within its own limitation this work is interesting because of the consideration of the phase division of the problem. Indeed such a structure is open to modification of one or more of its phases, keeping the others the same. In the selection phase multiple criteria are used but no production objectives are explicitly considered and no objective function or "goal" is explicitly mentioned, thus not considering the multiple objective nature of this decision process with objectives differing from plant to plant. Indeed a relation between the selection criteria is not clearly defined in itself.

2.5 ARTIFICIAL INTELLIGENCE AND HYBRID SYSTEMS

The use of artificial intelligence techniques (more precisely knowledge based techniques) for the scheduling problem is possible even though limited by some factors. First of all, to achieve real time control the speed characteristics of the control technique are fundamental. Moreover, a methodology for scheduling should also be able to take into account and manipulate vague and uncertain problem data. Finally an easy interpretation and "maintenance" of the rules that are employed is necessary. For these main reasons AI based techniques often have strong limits in their applicability, even though suitable modifications can be made to take into account the above factors. From this perspective it is apparent how beneficial combining different techniques in a hybrid system can be.

Rabelo and Alptekin [45] present an intelligent flexible manufacturing system scheduler (IFMSS) that uses an knowledge based hybrid methodology to achieve high performance in the priority setting problem. The IFMSS consists of three modules:

1. knowledge controller;
2. real-time scheduler;
3. high performances scheduler.

The first module uses several knowledge bases and neural networks to control the scheduling decisional process. Structurally minimized neural networks are used to:

- estimate the behavior of dispatching and scheduling rules;
- support the temporal reasoning of the integrated system, that is, estimating the minimum time necessary to compute a solution.

In the same module knowledge base systems are used to:

- interpret the objectives and the commands from the different elements in the hierarchical structure of the FMS;
- interact with the user;
- monitor performance and develop strategies for neural networks training;
- implement complex scheduling procedures.

The real-time scheduling module embodies several strategies to make the scheduling process quicker. This module has a high priority in case of strong time limitations that prevent the use of the high performance scheduling module. Whereas this last module uses several knowledge sources (knowledge based systems, optimization algorithms and neural networks) to find optimal solutions. Given the time consuming nature of this knowledge sources they are used in the decisional process whenever there is enough time.

The IFMSS was tested through simulations in several cases and has performance better than heuristic rules in terms of parts' mean tardiness. In this approach only the priority setting problem is considered and multiple objectives are not considered, since only the tardiness is. The knowledge bases could probably be augmented in such a way that the multiple objective nature of the problem is considered. The rules could be eventually fuzzy, or in general should be able to deal with vague data. The two very interesting features of this work are the constant re-training of the neural networks as well as the existence of two scheduling modules. The constant training of neural

networks on-line, if correctly done, should confer adaptivity characteristics to this approach, thus taking into account one of the most important elements of the problem. On the other hand, having two scheduling modules (eventually parallel) could be very beneficial in finding at least a good solution and eventually an "optimal" or "pseudo-optimal" solution in case time is available for that.

Shaw et al. [48] develop an knowledge based approach to the priority setting problem that uses machine learning capabilities in order to minimize the mean part lateness. To refine and acquire heuristic rules they develop a pattern directed scheduling (PDS) that uses an inductive learning module. The scheduler classifies distinct productive characteristics and generate a decisional tree made of heuristic policies. In this way the best rule according to the current system state is dynamically chosen. The following heuristic rules are initially considered: EDD, SPT, MDD, MODD. The attributes used to classify the distinctive production characteristics (i.e., state space partition) are:

- number of machines in the system;
- buffer dimension;
- maximum machine workload;
- variability in machine workload.

The methodology is articulated in three steps:

1. generation of training examples, by using several dispatching rules;
2. determination of heuristics through inductive learning;
3. execution of adaptive pattern-directed scheduling.

This approach was tested through simulations in different conditions and for several parameters values and it always performs better than the best heuristic, in terms of mean part tardiness. Only one objective is considered and vague data are not taken into account. The adaptive feature of this approach, achieved through inductive learning, is very interesting since it allows for good performance even in the presence of on-line parameter variation. The main idea of the inductive learning used in this

approach is to partition the state space in classes of states equivalent with respect to the action to take. In other words, whenever a *good* action is found for a given state of the system, this information is stored into a learning module, so that the next time the system is in the same state the action to take is already known. The inductive learning approach through partitioning the state space might suffer from exponentially increased complexity with the problem dimensions. Moreover the state of the system must be defined and available.

Rabelo et al. [43,44] present a control architecture using neural networks and heuristic rules for the priority setting problem of a free machine from a central buffer. Indeed, seven different part types are considered, each able to go to any of five machines. All the parts waiting for processing wait in a central buffer, thus when a machine is free a selection must be made for a next piece to process, among the ones in the central buffer. The controller is modular. The most interesting module is the optimization module. This consists of a modular neural network, a simulator, a genetic algorithm and an induction mechanism. Given some initial heuristic policies these are the input of each module of the neural network as well as of the gating module. The network modules determine an estimate of the performances of the given heuristic, given the actual state of the system. Each module takes care of one performance measure. Seven modules are used to estimate:

- maximum time in system;
- mean time in system;
- maximum lateness;
- mean lateness;
- WIP;
- utilization;
- production level.

The performances estimated by each module are then combined in a scalar performance index through the gating network. Indeed this *PI* is formed as a linear combination of the estimates with weights given by the gating network[16]. This modular network was previously trained and structurally optimized. Once all the heuristics are evaluated for the current state according to a scalar multiple objective performance index the best are selected and passed to a parallel simulation for further evaluation. Finally, a genetic algorithm, is used to generate new schedules according to the techniques previously chosen. In a short time the GA determines scheduling policies that are superior to the initial ones. The best among these is chosen as the next action. An induction mechanism memorizes the action that was taken, along with the system state, in order to partition the state space in equivalence classes of states with respect to actions.

This technique is adaptive and possesses learning capabilities. Moreover, as Rabelo et al. claim, it has small execution times. This last statement seems rather strange since parallel simulations and genetic algorithms are involved. Moreover the problem that is addressed has small dimensions. Thus, increasing problem dimensions would probably be detrimental in terms of execution times and exponential growth of the state space with consequent increased partition complexity.

Smith et al. [50] conduct what they claim to be the first comparative study between intelligent reactive (real-time) and predictive (*a priori*) scheduling. Two scheduling systems are developed based on the model of a real FMS plant in Aiken (South Carolina, USA): the intelligent real-time scheduling system (IRTS) and the intelligent a priori scheduling system (IAPS). Both systems embody the knowledge of experts from the Aiken FMS plant. The time horizon for the IAPS is one week. Comparing the two systems, the IRTS results superior to the IAPS in terms of part lateness. This study is very practical and grounded to reality, thus it is a good starting

[16] Note that the global *PI* extraction employed here is very similar to Yager's [57,58] fuzzy MADM approach described in Section §3.2.3.

point for comparison between intelligent reactive and predictive scheduling. Moreover, the study is still in an early phase since more work needs to be done in terms of:

- multiple performance indices for evaluation;
- effect of the time horizon in the IAPS design;
- predictive schedulers design methods;
- effect of failures, tools shortenings and quality on IAPS and IRTS;
- effect of product mix variety on IAPS and IRTS.

2.6 GENETIC ALGORITHMS

Genetic algorithms are an optimization technique that is efficient for complex and high-dimension problems with *irregular* objective functions where generally gradient-based techniques fail. They were introduced by Holland [26] in 1975 and later developed by Goldberg [19] among others. These algorithms conduct a random search starting from an initial population that iteratively *evolves* by means of certain operators. This *evolution* corresponds to moving towards areas in the search space corresponding to the maximum of a given objective function that represent the *fitness* of a particular individual (solution). Because of their characteristics GAs seem to be particularly suited for scheduling problem, as also remarked by Tsang in a comparative study of scheduling approaches [52]. Given their nature, GAs are used for predictive scheduling, that is, to determine an optimal schedule at the beginning of a fixed time horizon. This is probably the limit in the use of GAs for scheduling purposes. With the increasing computational power available at decreasing costs GAs might become particularly suited for a predictive scheduling that approaches reactive scheduling. Indeed by decreasing their targeted time horizon they could be used in a predictive fashion on very small time steps, thus approximating a real-time approach. The very key of this evolution of the role of GAs stands in the objective function evaluation time. If very small (compared to the time horizon) evaluation times can be achieved then a quasi real-time solution can be found. Constraint representation and expression

is another problem with GAs, even though some solutions exist. In the following some examples of use of GAs in scheduling are listed very briefly to show some of the existing approaches.

Gen et al. [18] and Kim and Lee [32] use a GA to determine a schedule for a job-shop. The objective function is the schedule length.

Falkenauer and Bouffouix [15] use a GA to determine a schedule for a job-shop where the objective is lateness minimization and earliness maximization.

Sittisathanchai et al. [49] present a GA for job-shop scheduling. The objective is the minimization of the schedule length along with its cost. The cost of the schedule is defined in terms of lateness and operations advance.

Dorndorf and Pesch [13] use a GA as a training source for some standard heuristics. A job-shop scheduling problem is considered, where the objective is the minimization of the schedule length.

Holsapple et al. [27] use a GA to present random examples to a predictive AI based FMS scheduler. The scheduler learns autonomously from these examples.

2.7 CONCLUSIONS

In this chapter a review of fuzzy techniques for scheduling in flexible manufacturing system was made. Elements of neural, AI based, hybrid and GA based techniques were presented too. Some characteristic features of the reviewed techniques are summarized in Table 2.1. Every fuzzy approach, besides the one presented by Angsana and Passino [1,2], lacks of a systematic design procedure that could hold for different FMS configurations. On the contrary, all neural network based techniques have a design procedure that basically consists in the training of the network. This type of training has drawbacks in terms of speed and data collection. AI-hybrid based solutions possess a design procedure. Indeed they also have adaptive properties that in fact constitute their design mechanism too. Those properties are

achieved through inductive learning capabilities, thus enabling the user to design the scheduler off-line by means of simulations and/or past data and to have it adapting on line. Unfortunately these seem to be quite complex solutions and a high potential for exponential performance degradation with increasing problem size is foreseen. Another drawback that these type of solutions share with neural network based approaches consists in deciding how to assign the reward or a penalty to a given action. Indeed, an action has an effect that can be seen after some time steps and eventually for some time steps. Deciding how to evaluate the reward for a given action is basic in implementing any of these techniques and can be a quite complex task. This constitutes one of the main obstacles in developing design and adaptation solutions.

Every fuzzy approach, besides the one of Hatono et al. [22], implement rules either based on some fuzzy version of existing heuristics or based on expert knowledge. This way already working and tested solutions can be embodied in the fuzzy framework and optimized. Both neural and AI-hybrid based approaches are based on some production objectives, generally only one of them (besides Rabelo et al. [43,44]). This is easily explained given the type of neural network training or inductive learning.

Some fuzzy approaches are not adaptive and no potential for their eventual adaptation is foreseen. On the other hand the parameters that characterize fuzzy logic systems and their membership functions make them a good ground for adaptation that can thus be achieved by some of the fuzzy approaches. Only the approach from Angsana and Passino [1,2] uses an adaptation technique that, even though very simple, is successful in decreasing WIP levels. Unfortunately this adaptation technique is heuristic and not based on any production objective. The discussed neural techniques are not adaptive but can be if the off-line training can be, even partially, taken on-line to modify their weights. AI-hybrid techniques present adaptive properties given by the inductive learning methods employed and within the limits discussed above.

Not all the reviewed techniques were tested and compared to heuristic or already existing solutions. Moreover if they were, they were only compared in terms of a limited number of production objectives.

This picture of the state of the art in intelligent techniques for scheduling in FMS shows the definite need for a systematic design procedure based on multiple objectives. Moreover the design procedure should also account for the stochastic and dynamic nature of the system. Some general modifiable structure for designing according to multiple production objectives with different degrees of importance is absent. Such a framework could be the first step towards a truly adaptive solution to the scheduling problem. On these premises the use of fuzzy logic seems very suitable. Indeed fuzzy multiple attribute decision making techniques could offer the advantage of being able to deal with multiple and contrasting objectives. Fuzzy logic systems could be used to deal with uncertain and vague data and to code experts knowledge. Moreover, FLSs could be trained like neural networks and the parameters characterizing its membership functions, eventually along with the rules, could be modified off-line (design) or on-line (adaptation). Fuzzy techniques can offer the same advantages offered by neural networks in terms of training, since they can be regarded as adaptive networks (Jang [30], Jang and Sun [29] and Wang [54]).Thus, gradient methods based on the backpropagation of training errors can be set for FLSs, giving them learning capabilities. Along with this, the final FLS appeals more to intuition than the trained weights of a neural network. Fuzzy techniques can also take advantage of rules, as expert systems, and deal with vagueness. Looking at fuzzy systems as some kind of bridge between neural and AI based solutions it can be concluded that a fuzzy-hybrid solution should be sought.

Table 2.1 Comparison of intelligent scheduling approaches

	Design procedure?	**Rules based on m.o.?**	**Adaptive? Eventually?**	**Comparison with heuristics?**	**Evaluated on m.o.?**
Fuzzy					
Hintz and Zimmermann [25]	NO	NO	NO, NO	YES	waiting time on-time parts utilization
Choobineh and Shivani [10]	NO	NO	NO, NO	NO	NO
Hatono et al. [22]	NO	utilization waiting time	NO, YES	NO	NO
Watanabe et al. [55]	NO	NO	NO, NO	YES	lateness time in system
Grabot [20]	NO	NO	NO, NO	NO	NO
Angsana and Passino [1,2]	YES	NO	YES	YES	only buffer levels
Sentieiro [47] Custodio et al. [5]	NO	NO	NO, YES	YES	WIP throughput
Ben-Arieh and Lee [4]	NO	NO	NO, YES	YES	time in system lateness parts completed
Neural					
Lo and Bavarian [37]	YES	minimize schedule length	NO, YES	NO	NO
Cho and Wisk [9]	YES	one selected objective	NO, YES	YES	processing time throughput lateness
Watanabe et al. [55]	YES	lateness	NO, YES	YES	lateness
Hao et al. [21]	YES	NO	NO, YES	NO	NO
AI-hybrid					
Rabelo and Alptekin [45]	YES	lateness	YES	YES	lateness
Shaw et al. [48]	YES	lateness	YES	YES	lateness
Rabelo et al. [43.44]	YES	time in system lateness WIP utilization production level	YES	NO	NO

CHAPTER 3
EVOLUTIONARY FUZZY SCHEDULER

3.1 FLEXIBLE MANUFACTURING SYSTEMS

3.1.1 INTRODUCTION

The present industrial trend of manufacturing low cost low-to-medium volumes of modular products with high variability demands manufacturing systems with flexibility and low delivery times. This led to manufacturing systems with small batch productions, low setup times and many decisional degrees of freedom. Those systems are flexible manufacturing systems (FMS). They are highly automated systems with many redundancies, thus allowing for many degrees of freedom in the decision process. Even though there are no universally accepted definitions of FMSs, according to what is proposed by Tempelmeier and Kuhn [51] and Viswanadham and Narahari [53] an FMS is composed of:

- Numerically controlled (NC) multipurpose machine, with automated tool exchange;
- Automated materials and tools handling system (MHS), made by conveyor belts, automatic guided vehicles (AGV), industrial robots, etc.;
- Load and unload stations that manage the loading and unloading of parts (loaded parts are fixed on pallets);
- Inspection stations (for quality control);
- Storage areas like input, output and input-output buffers for every machine, or centralized buffers;
- Tools storage areas;
- Hierarchical control system that manages the MHS, all the parts and tools movements and loading and unloading of parts in stations and machines.

3.1.2 THE CONTROL PROBLEM

Flexible manufacturing systems are characterized by a hardware structure that enables them to be flexible as needed to maintain competitiveness. The weak point in

this kind of structure is in the control system that does not allow for exploiting the full potential of FMSs. The control problem in a FMS is divided in three hierarchical levels corresponding to the time horizon they refer to, thus having long, medium and short-term control. In this study the short-term control problem is considered. In order to take into account all the dynamic characteristics of an FMS real-time control is required, along with modern control techniques such as state feedback. Unfortunately, conventional modern control techniques cannot be directly applied to FMSs since they do not belong to the class of continuous (or discrete) time systems but are discrete event dynamic systems, for which, in general, a closed form mathematical model does not exist. In addition, the intrinsic complexity of the control system design the high variability existing across FMS plants makes it difficult to develop a *good* control solution, easily transportable to several plants, without the use of a systematic design procedure. Moreover the *optimal* solution must satisfy robustness and speed characteristics necessary to the correct operations of an FMS. Other difficulties in approaching the control problem come from the uncertainty inherent to data collected for control and from the necessity of interfacing with human operators.

A standard approach to scheduling (short-term control) problems consists of using heuristic rules like the ones described in §2.2. Those rules are effective in practice but they do not exploit the full potential of a FMS. Indeed they generally fail to capture the dynamic characteristics of FMSs and cannot be ranked by *effectiveness* since their success depends on the particular FMS typology at hand. For this reason a set of given *good* heuristics is generally known, but the best one for the given FMS has to be sought. Thus, in current practice of FMS operations, human experts are the ones that really make an FMS work with practical rules. It is then appropriate to think of a control system that tries to mimic human behavior.

The complexities involved in this problem make it easy to recognize how hard it is to make an FMS work properly in an automated fashion, thus the necessity of a good and robust control system. Based on these premises, along with the very high costs involved in FMS control software, Fanti et al. [16] present a structure for the

software for a generic FMS. Somehow that can be regarded as an *operating system* for a FMS. Indeed short-term control problems of an FMS are really similar in nature to the ones involved in the design of operating systems for multitasking environments. A major difference, though, consists of the *physical* nature of FMS problems, where jobs are physical entities (i.e., material) rather than machine-language code.

The scheduling problem consists of several *decisional* points. A first division into four parts can be made:

- *Timing*: that is, when to insert a part into the system;
- *Sequencing*: that is, defining the order with which different parts (batches, orders) are inserted into the system;
- *Routing*: that is, defining the route (machine, AGV, etc.) for a part in presence of alternatives;
- *Priority setting*: that is, defining a priority for parts, machine and resources in general so that a choice is directly implied.

Fanti et al. [16] indicate a set of thirteen scheduling rules, even though in their work only AGVs are considered as a transport system and tool movements and pallet constraints are not considered. This shows how the scheduling problem is a complex one, even in the presence of restrictive assumptions. Some preliminary and restrictive assumptions are needed in order to limit the complexity of the problem while still maintaining its reality. Another interesting aspect in the study proposed by Fanti et al. [16] is that the thirteen scheduling rules are themselves a part of the system state and thus subject to changes in the form of state transitions.

A final consideration on the control objective is due. Indeed a measurable control objective must be stated. Viswanadham and Narahari [53], Tempelmeier and Kuhn [51] and Young-On [61] describe several performance measures for a FMS that were discussed in Chapter 1. These objectives are often contrasting and vaguely formulated, thus adding complexity to the control problem. Indeed, some way to satisfy these multiple objectives with different degrees of importance must be sought.

This must be done also keeping in mind the vague nature of the objectives and the need for interface with management directions.

3.1.3 PERFORMANCE EVALUATION: SIMAN

A common way to evaluate the performance of a manufacturing system (along with its control system) is through simulations. This requires a previous modeling phase of the problem. Flexible manufacturing systems do not belong to the typical classes of systems with which a control engineer is generally acquainted. Indeed a FMS is not a *time-driven* system but it is an *event-driven* system; that is, its evolution (i.e., state transitions) is not dictated by the flow of time but by the occurrence of *events*. Manufacturing systems belong to the class of discrete event dynamic systems (DEDS). DEDS were first studied by Zeigler [66,67,68] who introduced a formalism for their specification called discrete event system specification (DEVS). The DEVS formulation of a DEDS is really interesting in that shows how DEDSs belong to the general class of systems studied by systems theory. Unfortunately, there is no universally accepted modeling paradigm for DEDS (e.g., like differential equations for continuous time systems) thus introducing a certain level of uncertainty in the performance evaluation. Some of the most common modeling paradigms for DEDS are (Cassandras [7]):

- Markov chains;
- automata;
- queuing networks;
- computer-based models;
- Petri nets.

As pointed out in Fanti et al. [16] the DEVS formalism seems to be the most appropriate for FMS modeling. Indeed DEVS is an easily understandable formalism that is also able to include all the characteristics of the system without forcing unrealistic simplifications. The DEVS formalism is also interesting for having a strong connection to the system simulation. For the purpose of this study the controller

performance was evaluated using the SIMAN simulation package, one among many commercially available packages. SIMAN (simulation analysis) is described in Pedgen [41]. This simulator is organized such that every element in the system is an entity with its own attributes (Chen [8]). Entities move into the system through block functions having several effects and purposes in the dynamic of the entire system. The block functions can be divided into the following types:

- general utility functions: used for variable value assignments, introducing delays, sending signals, etc.;
- resource related functions: used to seize or release a resource or change its capacity;
- transport system related functions: used to request, block, and use AGVs or conveyor belts;
- file related functions: used to manipulate files, that is, to act on queues, since queues are generally stored in files;
- statistical functions: used to record variables observation samples;
- other functions: used to combine entities, wait for a condition to hold, etc.

Every SIMAN simulation is built using two files. The first file (*mod* extension) contains the model of the simulated system in terms of the path of an entity through block functions. The second file (*exp* extension) contains the experimental conditions for the simulation of the system (parameters defining statistical distributions, simulation length, transient length, number of replications, etc.). Fortran subroutines can also be linked with the simulator in order to easily add other features that otherwise could be hard to add in the SIMAN framework. This gives many degrees of freedom in the system simulation, since the power of the DEDS simulation package is linked to the versatility of a procedural language.

In the present study a FMS was modeled and simulated using SIMAN. Thus, the performance of the proposed control techniques was compared to those of standard techniques by means of extensive simulations.

3.1.4 PROBLEM STATEMENT AND WORKING ASSUMPTIONS

In the present study the following assumptions on the FMS are made:

1. Tool management is not considered, i.e. it is supposed that all the tools are available where needed.
2. Failure of workstations and/or transport systems is not considered, i.e., the machines and/or transport subsystem are not subject to failure.
3. Orders arrive to the FMS as Poisson processes with a fixed inter-arrival time.
4. Production of orders occurs in batches, and the movement of the whole batch is considered, so that batch dimensions are not important.
5. Setup times are independent of the order in which operations are executed, i.e. they are constant and embodied in the operation times of each job (batch).
6. There are as many pallets and fixtures as are needed (this assumption is mitigated by the fact that the number of jobs in the system is constantly controlled).
7. The routing of every job is random and directly defined as a sequence of workstations the job has to go through. Thus, the route of a job is not defined in terms of the operations needed by the job. In other words, every operation corresponds directly to the workstation that will execute it, i.e., the routing is defined as a sequence of workstations (i.e., workstation 1, 5, 6, 2).
8. There can be multiple routing choices, i.e. at a certain point a job can be equivalently sent to different workstations (as specified in its routing plan) having different processing times.
9. Loading, unloading and processing times are random.
10. Due dates are assigned according to the total processing time of a job.
11. Each workstation can work only one job at a time.
12. The transport system is composed of automated guided vehicles (AGVs) and each AGV can transport only one job at a time.
13. Neither the weight of a piece nor the dimension of a batch affects the speed of AGVs, which is assumed to be constant.

14. Every workstation has one input buffer and no output buffer, therefore it will be free as soon as there is one free AGV that can transport the processed job to another workstation.
15. Delays in accessing the state information are neglected.
16. Among all the possible scheduling rules (Fanti et al. [7] contains a list of rules for a quite general FMS), the following are considered:
    - sequencing;
    - priority setting for a job (selection of a piece among those waiting to receive service from a machine);
    - routing decisions concerning the next required workstation.

3.2 FUZZY LOGIC

3.2.1 INTRODUCTION

Fuzzy set theory was introduced in 1965 by Zadeh [62]. In this framework every element of a *universe of discourse* (i.e., analogous to the "environment" set in the classical, *crisp*, set theory) can belong to a fuzzy set with a membership degree (μ) being any real number in [0,1]. In *crisp* logic a certain element of the environment set either belongs to a given set or it does not, that is:

$$\text{given } A \subseteq E \text{ and } a \in E \;\Rightarrow\; \text{only one of the following holds } \begin{cases} a \in A \Leftrightarrow \mu_A(a) = 1 \\ a \notin A \Leftrightarrow \mu_A(a) = 0 \end{cases}$$

This means that the element has respectively either a 1 or a 0 membership degree in the given set. On the contrary in fuzzy sets the degree of membership does not need to be either 0 or 1 but can be any real number between them. Loosely speaking, this is like excluding the possibility of only black or white, but accommodating all the different shades of gray in between. In fuzzy set theory a set is completely identified by its membership function as well as in crisp set theory. Indeed while in crisp set theory

the membership function can be any function defined in the environment set and having values in {0,1}, in fuzzy sets the membership functions is still defined in the universe of discourse but it assumes values in [0,1]. Thus, for every element in the universe of discourse the membership function of a fuzzy set gives its membership degree.

Fuzzy set operators can be defined in terms of operations between membership functions the same way it can be done for *crisp* sets, thus showing how *crisp* set theory can be regarded as a subset of fuzzy logic theory. Indeed in *crisp* set theory the union of two sets, *A* and *B*, is the set containing all the elements of *A* and *B*. This means that the membership function corresponding to the union of *A* and *B* ($\mu_{A\cup B}$) is one for all the elements having unit membership degree in at least one of *A* and *B*, and is zero otherwise (i.e., for elements having zero membership in both *A* and *B*). Thus $\mu_{A\cup B}$, for *crisp* sets, can be defined as:

$$\mu_{A\cup B} = \max\{\mu_A, \mu_B\} = \mu_A + \mu_B - \mu_A \cdot \mu_B \tag{3.1}$$

The union of two fuzzy sets can be defined in terms of a *t-conorm* (or *s-norm*) operator, indicated with the symbol $\oplus$, applied to the sets' membership functions. Commonly used *t-conorms* are the maximum and the *sum-minus-product* operators, that is, the same operators that characterize the union of *crisp* sets in (3.1).

Something analogous holds for the intersection of two sets. The intersection of two *crisp* sets is the set containing the common elements between them. That is, a set having zero membership function for all elements except for the ones having unit membership in both sets. Thus, the membership function of the intersection of two sets *A* and *B* can be defined by:

$$\mu_{A\cap B} = \min\{\mu_A, \mu_B\} = \mu_A \cdot \mu_B \tag{3.2}$$

The intersection of two fuzzy sets can be defined in terms of a *t-norm* operator, indicated with the symbol $\otimes$, applied to the sets' membership functions. Commonly used *t-norms* are the minimum and the product operators, that is, the same operators that characterize the intersection of *crisp* sets in (3.2).

Finally the complement of a fuzzy set *A* is defined as the set characterized by the complement to unity of the membership function of *A*, that is:

$$\mu_{\overline{A}} = 1 - \mu_A \tag{3.3}$$

It can be easily observed how even this last operation can be defined the same way for both *crisp* and fuzzy sets. It is important to note that with *crisp* sets the use of each one of the possible operators corresponding to a set operation will always produce the same set, that is, the same membership function. On the contrary, for fuzzy sets each operator will generally produce a different membership function. This difference is obviously caused by the different nature of the possible membership degree values in the *crisp* and fuzzy case.

Fuzzy set theory is characterized by its capability of handling linguistic variables (Zadeh [63]); this makes it a paradigm very close to the way a human thinks. For the above reason fuzzy logic is frequently used in those problems where it is necessary to mimic the behavior of some human expert. This is one of the main reasons that makes fuzzy logic a useful approach to the scheduling problem. Moreover, fuzzy logic has been extensively and successfully applied to many engineering problems, raising a growing interest in it that led to theoretical as well as practical developments. Among those developments fuzzy logic systems (FLS) and fuzzy multiple attribute decision making (MADM) techniques present characteristics useful in approaching the scheduling problem. Indeed FLS are rule based system, thus offering the advantage of more sophisticated rules than heuristics, along with the fuzzy nature of the rules themselves. On the other hand fuzzy MADM techniques are characterized by handling multiple imprecise (fuzzy) objectives. Fuzzy logic systems are briefly described in the following §3.2.2 while some fuzzy MADM techniques are discussed in §3.2.3.

3.2.2 FUZZY LOGIC SYSTEMS

Fuzzy logic systems are a natural extension of fuzzy set theory to relations between fuzzy sets and rules. A comprehensive introduction on FLSs can be found in Mendel [38] or Kosko [34], among others. A FLS is characterized by four modules:

- fuzzifier;
- defuzzifier;
- inference engine;
- rule base.

A schematic representation of a FLS is presented in Fig. 3.1. A FLS is based on the rules contained in the rule base. Each rule has the following form:

$$\textbf{If } x_1 \text{ is } B_1 \textbf{ and } x_2 \text{ is } B_2 \textbf{ and } \ldots\ldots \textbf{ and } x_m \text{ is } B_m \textbf{ then } y \text{ is } C \tag{3.4}$$

The first *m* terms are called the *antecedents* of the rule while the last term (the one after the "then") is the *consequent* of the rule. Rules with multiple consequents might exist but it can be seen that they are equivalent to more rules with only one consequent each, thus rules with only a single consequent will be considered in the following. The terms x_i are fuzzy variables and the terms B_i are linguistic variables. It can be noted that the inputs to a FLS somehow correspond to the antecedents of the rules in the rule base. A difference exists though. Indeed the inputs to the FLS, as can be seen in Fig. 3.1, come from the outside (i.e., controlled process) and are *crisp* variables in general. On the contrary the antecedents of the fuzzy rules are always fuzzy sets. The role of the fuzzifier in a FLS is now apparent. Indeed it converts a *crisp* input variable into a fuzzy set that is ready to be processed by the inference engine. The fuzzification can be performed in several ways. Typically the use of a *singleton* fuzzifier is common. A *singleton* fuzzification consists in creating a fuzzy set with a membership function that is always zero in the universe of discourse, except for a point, where it is one. Non-*singleton* fuzzifiers are also used, typically in the case of uncertain measurements. Indeed in this case the input crisp value is affected by some uncertainty, thus the corresponding fuzzy set (after fuzzification) can be some kind of function that is one at the particular value of the measurement and then decreases to zero. This way the existence of noise in the measurement is somehow accounted for. When a non-*singleton* fuzzification is used, the corresponding fuzzy set is generally characterized by being centered in the measurement value and having a spread that is somehow

proportional to the amount of noise affecting the measurement. Figure 3.2 shows an example of *singleton* and non-*singleton* fuzzification.

Once the inputs are fuzzified they are passed to the inference engine that processes the rules retrieved from the rule base. Given a rule like the one in (3.4) all the antecedents are processed at first. That is, every term of the type x_i is B_i, corresponding to the intersection of the two fuzzy sets x_i and B_i, is calculated by applying the *t-norm* operator to the membership function of both x_i and B_i as in (3.2). Note that in the case of singleton fuzzification the particular *t-norm* used does not affect the result. Afterwards, the outcomes of all the antecedent terms are put together using again an and operator (*t-norm*) as is also apparent from the structure of the rule. The strength with which the particular rule is fired is then given by the result of the antecedents' processing. This is combined with the consequent fuzzy set through a *t-norm* operator, to finally produce the fuzzy set output of the given rule. This process is summarized in Fig. 3.3. This process is repeated for every rule in the rule base thus producing as many fuzzy outputs as the number of rules. All these fuzzy sets are then combined into one fuzzy set that constitutes the output of the inference engine to the given input. The output fuzzy sets are combined by making the union of the outcome of all the rules. This means that all the single fuzzy sets are combined by the use of *t-conorm* operators, finally producing the fuzzy output.

In order to be used the fuzzy output needs to be interfaced to the *crisp* domain by the defuzzifier. The output fuzzy set indicates what the output is in fuzzy terms. This fuzzy output will be the membership function of a fuzzy set that gives the degree of membership of several possible crisp outputs. Thus, the point corresponding to the highest degree of membership in the fuzzy output has to be sought. This operation would correspond to a type of defuzzification, called *max* defuzzification. Unfortunately in most cases the situation is not so simple, since there might be many points having the same maximum degree of membership to the fuzzy output and an indecision on which one of these points to choose arises. Moreover, choosing the maximum point of the membership function is an operation that looses most of the

information contained in the membership function itself. It is then appropriate to think at the use of a technique that *summarizes* the information contained in the membership function. The *crisp* output corresponding to a certain fuzzy output set should be a number that takes into account the points with high membership degree more than the ones with small or no membership degree in the fuzzy output set. This corresponds to a center of gravity operation. Thus, one of the most popular defuzzifiers is the center of gravity defuzzifier that transforms a fuzzy output set into a number that is the x-coordinate of the set center of gravity. One drawback of this kind of defuzzification is the complexity involved with finding the center of gravity (integration). For this reason, easier defuzzification schemes are generally employed for reduced computational burden. Thus max, mean-max, height and modified height defuzzifiers are often used.

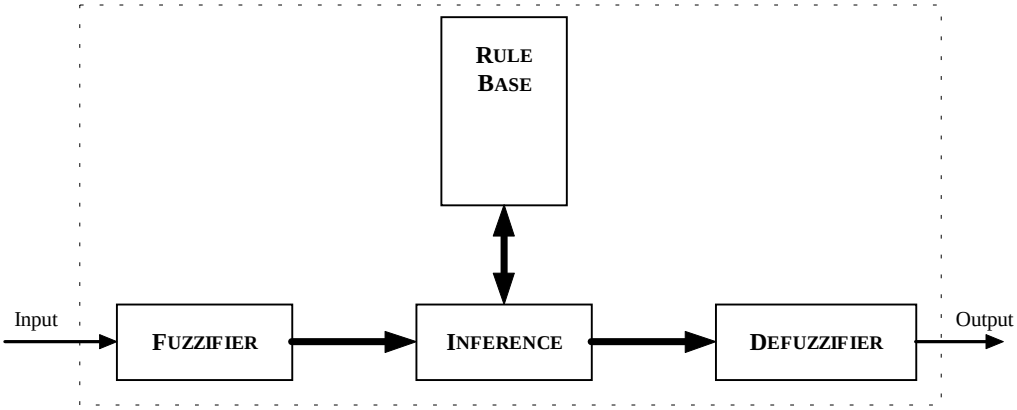


Figure 3.1 Structure of a fuzzy logic system

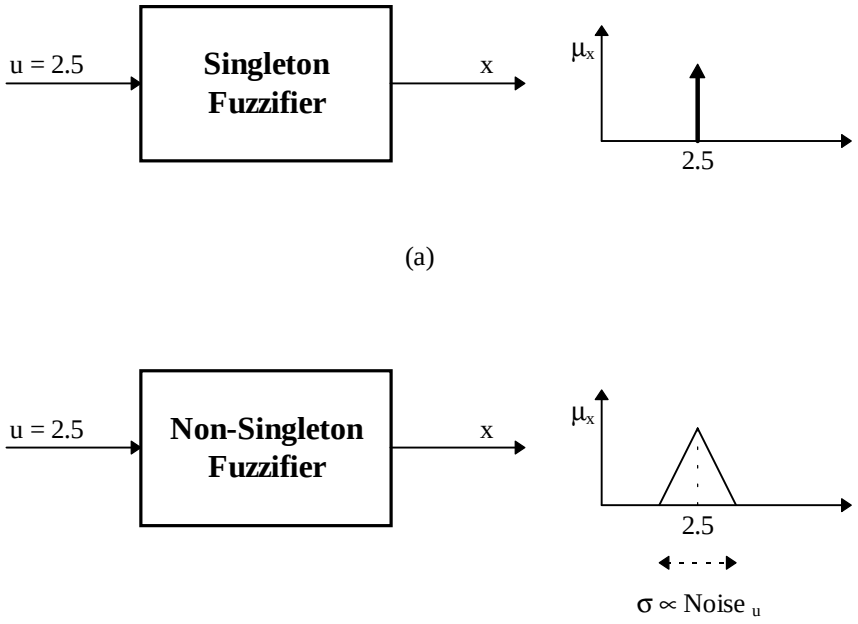


Figure 3.2 Singleton (a) and non-singleton (b) fuzzifiers

Example:

Problem: Control the acceleration of a car to reach a fixed point in minimum time

One rule could be:

If *distance* is *big* and *velocity* is *small* then *acceleration* is *big*

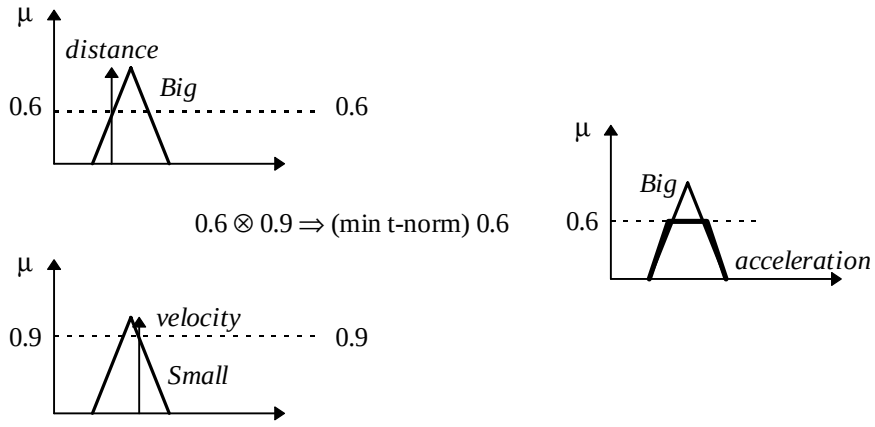


Figure 3.3 Inference engine

3.2.3 MULTIPLE ATTRIBUTE DECISION MAKING

One of the main characteristics of the scheduling problem for an FMS is its multiple objective nature. Fuzzy set theory can help even in this case, through the use of fuzzy MADM. The fuzzy MADM techniques that will be discussed in the following are all extensions of what was presented by Saaty [46] and that is briefly reviewed in the following.

Saaty [46] presents a decision methodology for cases where multiple decision criteria with different degrees of importance exist and a decision must be made among several alternatives. It is assumed that an alternative can be evaluated versus each criterion and that a final decision must be made, thus requiring the weights of each criteria. The objective of Saaty's technique is mainly to determine the weights (e.g., importance degrees towards the final decision) of each criterion. In order to evaluate the importance of each criterion towards the overall objective the aid of experts is needed. Instead of asking the experts a difficult evaluation of the absolute numeric importance of each criterion, a pairwise linguistic comparison of the criteria is asked. All the possible pairwise comparisons are performed and then converted to numeric comparisons through the use of Table 3.1. The numeric results are then ordered in a pairwise comparison matrix *A* that is square in the number of criteria. The a_{ij} (i.e., row *i*, column *j*) element of this matrix will be the numeric equivalent of the linguistic comparison of criterion *i* and criterion *j*. An interesting property of this matrix is that $a_{ij}\, a_{ji} = 1$ since they are the same comparison only looked at from criteria *i* or criteria *j*. The *A* matrix can then be used to determine the degree of importance of each criterion through relatively simple operations. Indeed it turns out that the eigenvector corresponding to the maximum eigenvalue (λ_{max}) contains the degrees of importance of each criterion. Moreover λ_{max} needs to be reasonably close to the number of criteria in order for the pairwise comparisons matrix (*A*) to be *consistent*. In this context by *consistent* is meant the fact that the pairwise comparisons between criteria have some degree of coherence. An example of incoherent comparisons would be:

1. Criterion 1 is strongly more important then criterion 2;
2. Criterion 2 is strongly more important then criterion 3;
3. Criterion 3 is strongly more important then criterion 1.

Obviously, the consistence only implies coherence of judgments in making the comparisons but it does not necessarily mean that the determined weights are exactly the desired ones. Indeed the comparisons could be coherent but a wrong estimate of the true compared importance of the criteria. Indeed, given some true values of the degrees of importance of each criterion, the pairwise comparisons provided by experts are a linguistic estimate of the ratios of degrees of importance of two criteria. The consistency property of the pairwise comparison matrix reflects only the coherence of the pairwise comparisons estimates but not their distance from the true values. Thus, consistency of the pairwise comparison matrix is a necessary but not sufficient condition for the degrees of importance estimates to be close to the true ones.

This technique has an interesting intuitive explanation that follows. Assume to have n criteria and that their degrees of importance (the quantities we are seeking with Saaty's technique) are known. Let α_i denote the degree of importance of the i-th criterion. Then the A matrix built using the true degrees of importance is:

$$A = \begin{bmatrix} \frac{\alpha_1}{\alpha_1} & \frac{\alpha_1}{\alpha_2} & \cdots & \frac{\alpha_1}{\alpha_n} \\ \frac{\alpha_2}{\alpha_1} & \frac{\alpha_2}{\alpha_2} & \cdots & \frac{\alpha_2}{\alpha_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\alpha_n}{\alpha_1} & \frac{\alpha_n}{\alpha_2} & \cdots & \frac{\alpha_n}{\alpha_n} \end{bmatrix} \quad \text{thus:} \quad A \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{bmatrix} = n \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{bmatrix}$$

and then we can conclude that n is an eigenvalue of the pairwise comparison matrix with corresponding eigenvector being the vector of the degrees of importance of each criterion. It can also be proven that n is the maximum eigenvalue of the pairwise comparison matrix. This explanation also shows why the distance of λ_{max} from n is a measure of the matrix consistency. Indeed, in a practical case the true degrees of

importance would not be known and an estimate of the *A* matrix would be used. Thus, if the estimate reflects some underlying degrees of importance (not necessarily the true ones) then the λ_{max} will be close to *n* as showed in the above discussion.

Table 3.1 Linguistic to numeric equivalence of criteria comparisons

Numeric value	Linguistic definitions
1	equal importance of both criteria
3	weak importance of a criterion over another
5	strong importance of a criterion over another
7	proven importance of a criterion over another
9	absolute importance of a criterion over another
2,4,6,8	intermediate values between adjacent judgments

Yager [57,58] proposed an extension to Saaty's method that employs fuzzy variables to express the degree of satisfaction of a certain criterion from a given alternative. Consider an overall objective *O* consisting of *n* criteria C_i with degrees of importance α_i determined from experts pairwise comparisons through Saaty's λ_{max} technique. Also associate a membership function $\mu_{C_i}(x)$ to C_i, this membership function gives the membership degree to which a given alternative satisfies the *i*-th criterion. Suppose the *j*-th alternative is characterized by *n* measurements x_{ji} (*i*= 1,2,…,n) relative to the *n* criteria, then the degree of satisfaction of the overall objective by the *j*-th alternative is given either by:

$$\mu_O(x_j) = [\mu_{C_1}(x_{j1})]^{\alpha_1} \otimes [\mu_{C_2}(x_{j2})]^{\alpha_2} \otimes ... \otimes [\mu_{C_n}(x_{jn})]^{\alpha_n} \quad (3.5)$$

or by:

$$\mu_O(x_j) = [\mu_{C_1}(x_{j1}) \otimes \alpha_1] \oplus [\mu_{C_2}(x_{j2}) \otimes \alpha_2] \oplus ... \oplus [\mu_{C_n}(x_{jn}) \otimes \alpha_n] \quad (3.6)$$

Equation (3.5) corresponds to applying expansion (power $\alpha_i < 1$) and contraction (power $\alpha_i > 1$) operators to the *n* fuzzy degrees of satisfaction of the alternative before intersecting them with a *t-norm* (and operation) in order to determine the overall degree of satisfaction of the objective. This appeals to intuition since the exponential function with base smaller than unity (e.g., the membership degree) is a strictly decreasing function of the power. That is, the higher the degree of importance of the given criterion, the higher the power the membership degree is raised to, and the lower the raised number. Thus, criteria with high degree of importance will correspond to small terms (i.e., $[\mu_{C_i}(x_{ji})]^{\alpha_i}$) that dominate the overall objective degree of satisfaction. Equation (3.6) has a slightly different interpretation. The degree of satisfaction of each criterion is *intersected* through a *t-norm* operation with the criterion absolute importance (i.e., $[\mu_{C_i}(x_{ji}) \otimes \alpha_i]$). The *union* (*or*) of all these terms is done through *t-conorm*, finally obtaining the degree of satisfaction of the overall objective. This approach appeals to the intuition of a weighted sum of the degrees of satisfaction of each criterion, with weights being the degree of importance of each criterion. The

author prefers the approach underlined in (3.5) because it is important for all the criteria to be satisfied at the same time (and) after an opportune expansion or contraction due to the criterion absolute weight.

Buckley [6] and Laarhoven and Pedrycz [36] subsequently extended Saaty's criterion to fuzzy weights. Not only is the degree of satisfaction of each criterion fuzzy, but the weights can also be fuzzy numbers. Thus, the pairwise comparison matrix is defined as a matrix of fuzzy numbers. More matrices, corresponding to different experts, can be available and also be incomplete in some parts. The information contained in each matrix is put together in order to determine the weights of each criterion as fuzzy numbers. The fuzzy weights are then used to determine the degree of satisfaction of the overall objective as a fuzzy number. The comparison among alternatives is finally done by comparing the fuzzy numbers and eventually finding a *best* alternative. The advantage of this approach consists in offering the capability of handling vague pairwise comparisons and multiple experts at the expense of increased computational complexity. This approach will not be further explained because it will not be used in the present work. It is worth mentioning, in order to show the possibility of increased accuracy and complexity in the treatment of the problem.

3.3 FUZZY SCHEDULER

3.3.1 INTRODUCTION

This section will describe the structure of the fuzzy scheduler (FS) that was implemented for the present study. The FS considers three particular rules in the scheduling problem: sequencing, priority setting (for a piece in front of a machine), and routing. In this work the sequencing problem can be viewed as the choice among orders waiting to be loaded into the system. For this reason it can be considered a priority setting problem. Sequencing and priority setting were addressed with FLSs

containing rules to define the priority of waiting jobs. Because of the similarity of the two problems' approaches the FLSs for sequencing and priority setting are described in the next Section §3.3.2. The following Section §3.3.3 describes the approach to the routing problem using fuzzy MADM techniques.

3.3.2 SEQUENCING AND PRIORITY SETTING

The sequencing and priority setting problems were approached using fuzzy controllers having rules with two antecedents and one consequent. The consequents of both FLSs always correspond to the job's priority while the antecedents are:

- total processing time (TPT) and time margin to the due date (DD-TN, i.e., due date minus actual time) for the sequencing FLS;
- slack (SL) and processing time of the job in the given machine (PT) for the priority setting FLS.

The FLSs determine the priority of each job waiting for loading or in a machine buffer, so that whenever the load station or the machine are free the job with the highest priority among those waiting is chosen.

The universe of discourse for the antecedents is covered by three triangular membership functions corresponding to the linguistic variables small, medium and large time. The membership functions corresponding to small and large always present a *saturation* at the unit value beyond a given minimum or maximum limit respectively. The consequent membership functions are three singletons (this way the defuzzification process is greatly simplified). Figure 3.4 shows the shape of the membership functions for antecedents and consequent. The membership functions for each antecedent are completely defined by seven parameters (i.e., two for each small and large set and three for the medium fuzzy set) while the consequent ones are defined by one parameter each. Thus, each FLS is completely determined by 7+7+3=17 parameters and nine rules. All the parameters defining the antecedents' membership functions of the FLSs are determined through discrete event simulations of the FMS plant under commonly used heuristics. The *central* points of each

membership function (i.e., small, medium and large) for a given variable are defined through the averages of minimum, medium and maximum values of the same variable in several *independently seeded*[1] simulations of the plant. The dispersion of each membership function is determined from the variance of the simulation results. The preceding statements were used as a guideline in fixing the antecedents' membership functions in a heuristic way more than in an algorithmic one. The rules were determined as fuzzy hybrid versions of commonly used heuristics. It can be observed from Table 3.2 that the fuzzy rules for sequencing are a fuzzy version of the STPT and LS heuristics. Analogously the rules for priority setting (Table 3.3) are a fuzzy version of the SPT and LS heuristics.

A singleton fuzzifier was used, along with *max-product* inference (i.e., a max *t-conorm* and a product *t-norm* are used) and *modified-height* defuzzifier. Note that the choice of singleton consequent membership functions along with the *modified-height* defuzzifier is motivated by limiting the computational burden associated with the FS. Indeed a simple weighted sum substitutes an integration operation as is also discussed above. A complete set of rules was used, that is, $3^2 = 9$ rules were used for each FLS. Indeed the choice of a low number of antecedents and of a low number of membership functions on the antecedents' universe of discourse was motivated by keeping low the number of rules. Indeed the number of rules can otherwise exponentially increase (for a complete set of rules) thus resulting in increased computational and storage requirements.

Given the structure of the FLSs their computational procedures are analyzed in the following. The *i*-th rule in the rule base will be:

$$\boldsymbol{R_i}\text{: If } x_1 \text{ is } B_1 \text{ and } x_2 \text{ is } B_2 \text{ then } y \text{ is } C_i$$

[1] Every simulation is started by using a random number seed that is then used as a starting point for subsequent random numbers generation. Obviously different seeds lead to different simulations of the same plant. Independent seeds offer the advantage of independence of the collected samples. In other words statistical independence of the seeds implies statistical independence of the observed realizations.

Assume that the inputs to the FLS are z_1 and z_2. Given the type of fuzzification, inference, *t-norm* and *t-conorm* that were used the fuzzy set output of the *i*-th rule will be given by:

$$\mu_{Ri}(x) = \mu_{Ci}(x) \times [\mu_{B_1}(z_1) \times \mu_{B_2}(z_2)] \tag{3.7}$$

The calculation in (3.7) is performed for each of the *M* (nine in this case) rules in the rule base. The fuzzy outputs for each rule are thus determined. Given the type of defuzzifier, no rule output composition is necessary. If the position of the consequent membership function for the *i*-th rule is denoted by y_{Ci} then the output of the FLS after *modified-height* defuzzification is given by:

$$p = \frac{\sum_{i=1}^{M} y_{C_i} \cdot \mu_{R_i}\left(y_{C_i}\right)}{\sum_{i=1}^{M} \mu_{R_i}\left(y_{C_i}\right)} \tag{3.8}$$

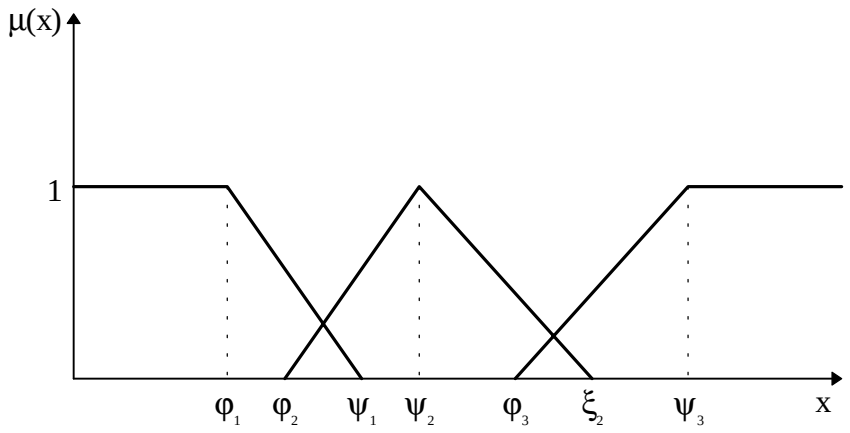
Observing (3.8) it can be noted how the shape of the consequent membership functions is not really important while their center of gravity is. Indeed the concept behind the *modified-height* defuzzification consists of making a weighted average of the center of gravity of each rule consequent membership function, having as weights the membership degree with which the rule is fired. Therefore, the consequent membership functions are completely specified through the coordinate of their center of gravity. For this reason singleton consequents can be used in order to stress this point and not add useless details.

As previously mentioned, the parameters defining all the membership functions were chosen heuristically through the observation of discrete event simulations of the FMS plant under commonly used heuristics. The rules in the rule base were chosen the same way, that is, observing simulations of the plant under different heuristics and creating fuzzy versions of *good* heuristics. The rules for sequencing and priority setting are presented in tables 3.2 and 3.3 respectively. In those tables rows and columns correspond to antecedents, while each element in the table is the consequent

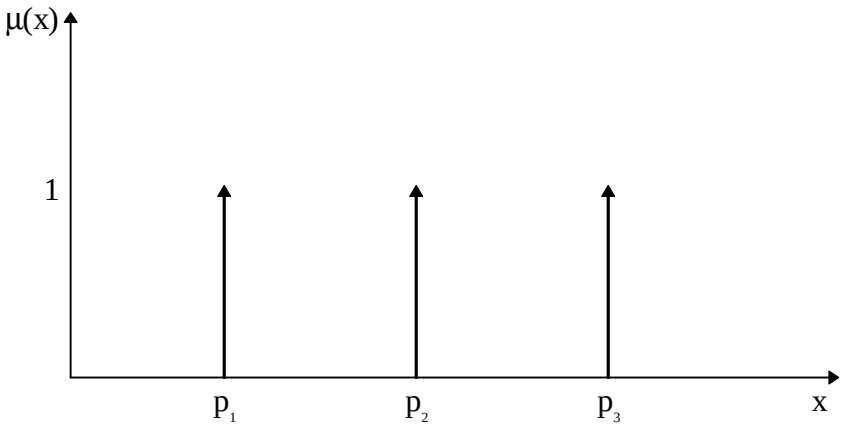
for the corresponding antecedents. For example, considering element (2,1) in Table 3.2 the underlining rule is:

If *TPT* is *medium* and *DD-TN* is *small* then *priority* is *big*

It is important to remark at this point how the heuristic process used for selecting the FLSs parameters and rules is not important for the purpose of this work, since in Section §3.4 a systematic design procedure for the fuzzy scheduler will be presented.



(a)



(b)

Figure 3.4 Membership functions for antecedents (a) and consequent (b)

Table 3.2 Sequencing rules

	DD-TN		
TPT	**S**	**M**	**L**
S	L	M	M
M	L	M	S
L	L	S	S

Table 3.3 Priority setting rules

	SL		
PT	**S**	**M**	**L**
S	L	L	M
M	L	M	S
L	L	M	S

3.3.3 ROUTING

The last *decisional point* that was considered is the routing problem, that is, the choice of one among many possible routes. In the problem considered this is equivalent to choosing the machine for next processing of a job, among the possible alternatives for that job. In this case fuzzy MADM techniques were preferred. Indeed the type of problem is more complex, in terms of the number of input variables involved in the decision. This would require a high number of rules if a FLS were used. Saaty's criterion [46] as modified by Yager [57,58] was employed. Therefore, the criteria necessary to reach the goal of a good choice were first analyzed. A pairwise comparison matrix of these criteria was then built by the author (i.e., that would correspond to a plant human expert) and the absolute importance of each criterion was then determined using Saaty's λ_{max} technique. Finally, the fuzzy sets corresponding to the satisfaction of each criterion were built. The corresponding controller chooses the alternative corresponding to the highest degree of satisfaction of the overall objective calculated as in (3.5). Thus a machine is chosen and the job is sent to this machine for processing.

The criteria for this decision (Fig. 3.5) are:

- small machine workload, i.e. it is better to send a job to the machine with the lowest workload waiting in its buffer so that the job has a smaller waiting time;
- small processing time (of the given machine on the job), i.e. it is better to send the job to the machine that will process it faster;
- small distance of the destination machine from the source machine, i.e. it is better to send the job to a nearby machine so that the minimum amount of time is wasted in job movement and the AGV is free earlier for another job.

A pairwise comparison matrix is then formed by expressing linguistical judgements on the pairwise comparisons of the above criteria. This matrix is converted to a numerical form using Table 3.1 and Saaty's λ_{max} technique is applied, also measuring the consistency of the pairwise comparison matrix. In case of verified consistency the

degrees of importance of the three criteria are assumed to be the ones obtained through Saaty's λ_{max} technique.

The membership functions expressing the satisfaction of the objectives need to be defined. The type of membership functions corresponding to the satisfaction of the first two criteria is shown in Fig. 3.6. This function is characterized by an initial plateau at unit membership degree (full satisfaction of the criterion). A linearly decreasing part follows, corresponding to slowly worsening alternatives. And a final exponential part ends the membership function, for the last and very unattractive alternatives. The exponential behavior was preferred to taking the function to zero membership degree in order to maintain decisional capabilities in the presence of very bad alternatives. Indeed having the membership function for the satisfaction of a decision criterion going to zero after a certain breakpoint would imply that, given two alternatives that are very bad in terms of the specified criterion they would be ranked the same. Indeed zero membership degree on a criterion means that the degree of satisfaction of the overall objective will be zero as well. Therefore there would be no way to discern between two *very bad* alternatives, thus negating the possibility of recovering from accidental trouble in operations. It can be seen that four parameters are necessary to specify each membership function of the type of Fig. 3.6. Three parameters are obvious from the figure, while the fourth parameter is the one that characterizes the exponential part. Indeed the fourth parameter is the *time constant* of the exponential part, that is, θ if the last part is expressed by:

$$\mu(\mathrm{x}) = \mathrm{h} \cdot \mathrm{e}^{-\frac{\mathrm{x}-\gamma}{\theta}} \tag{3.9}$$

The third criterion is the low distance of source and destination machine. In this case, since the set of all the possible distances among machines is discrete and finite, the *small-distance* fuzzy set corresponding to the third criterion will be characterized by a discrete membership function. This membership function can be described by simply listing the possible distances and their membership degree towards small distance. For example, suppose there are three possible distances d_1, d_2, and d_3 with

memberships μ_1, μ_2, and μ_3 respectively. The small distance membership function will be indicated as:

$$\text{small distance fuzzy set: } \left\{ \frac{\mu_1}{d_1} \quad \frac{\mu_2}{d_2} \quad \frac{\mu_3}{d_3} \right\}$$

Obviously the number of possibilities (elements needing to be defined in the membership function) increases with the number of machines in the considered FMS.

Once everything described above is set the controller is ready to operate. Whenever a job ends being processed at a machine its routing plan is examined. If there are no alternatives routes, or the job does not need further processing, the next machine is immediately determined and an AGV is requested. In case the route plan presents different alternatives (machines where the job can equivalently be processed) every alternative is tested. The degree of satisfaction of each criterion for a given alternative is determined using the appropriate membership functions. The degree of satisfaction of the overall objective is determined using (3.5) and the process is repeated for all the alternatives. Once the degree of satisfaction of the overall objective is known for each and all the alternative routes, then the one with the highest degree is chosen and the job is sent to the corresponding machine.

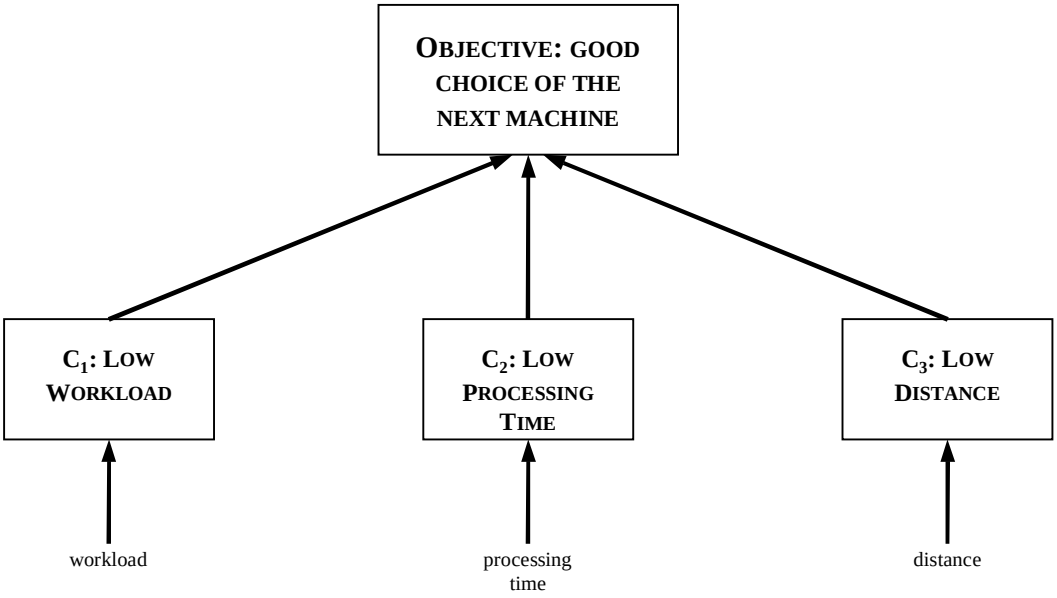


Figure 3.5 Routing hierarchy

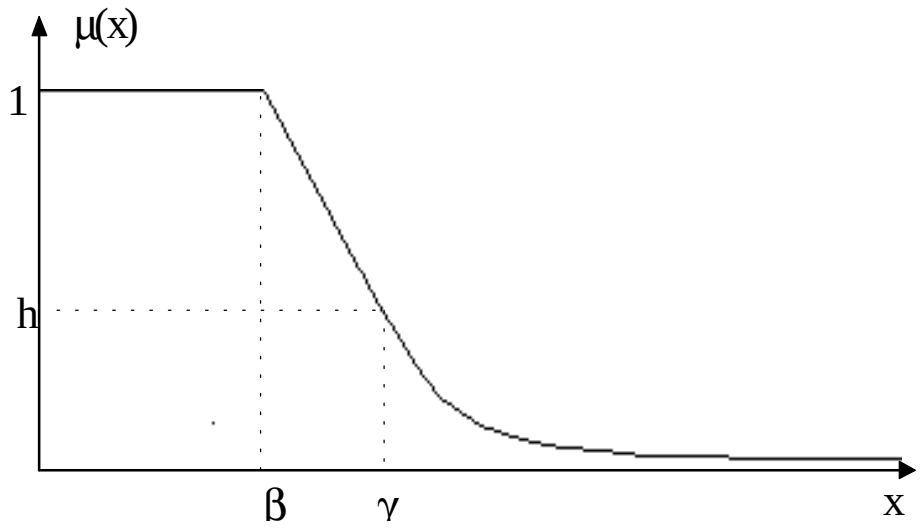


Figure 3.6 Membership function for the first two routing criteria, small workload and small processing time

3.4 EVOLUTIONARY FUZZY SCHEDULER

3.4.1 INTRODUCTION

In the preceding section a fuzzy scheduler for a FMS was described. The setup of this scheduler involves the definition of a number of parameters that give us a lot of degrees of freedom in the design phase. Moreover, defining the value of each parameter in a systematic fashion is not a simple task mainly because of the lack of theoretical criteria that could guide the design process. The proposed FS consists of two FLSs for sequencing and priority setting and a fuzzy MADM approach to the routing problem. In general, the design of a FLS can be divided into structural and parametric design. The structural design of a FLS consists of defining the type (shape) of membership functions, the *t-norm* and *t-conorm* operators and eventually the rules. The parametric design consists of defining the parameters of an FLS, that is, all the parameters that univocally determine a membership function (given its shape). Generally, the structural design of an FLS is considered less important than the parametric design. That is, the shape of membership functions along with the *t-norm* and *t-conorm* operators does not seem to significantly affect the performance of a FLS. In setting up the FS both structural and parametric designs were performed heuristically, using several FMS simulations as a guidance. This type of approach lacks generality; indeed the FS adopted for a given FMS plant probably will not be the same as another FMS plant. This type of problem is significant, given the wide variety of existing FMS plants (Tempelmeier and Kuhn [51]). Moreover a heuristic design approach might lead to a *good* solution to the scheduling problem, but no quantitative measures of how *good* is the solution are embedded in the design. Thus, leaving space for eventual improvements unknown to the designer or potentially leading to *bad* solutions.

In the field of parametric design of an FLS Jang [30], Jang and Sun [29] and Wang [54] propose a very interesting approach. In this approach FLSs are embodied in the more general framework of adaptive networks, the same framework that neural networks belong to. The advantage of this approach is given by the added possibility of training an FLS using the wide number of existing training techniques already developed for neural networks. From this perspective the difference between FLSs and artificial neural networks becomes blurred. Indeed both paradigms are characterized by a number of parameters that can be trained with the same methodologies. Fuzzy logic systems offer the advantage of giving a meaning to its parameters, a characteristic that neural networks do not possess. Indeed the weights of a neural network do not have a particular meaning for the designer, while the parameters of a FLS do. For this reason training convergence for a FLS should be in general easier to achieve than for a neural network, because the initial parameters do not need to be randomly initialized (as is typically done with neural networks) but some *educated guess* can be made.

Unfortunately, the type of problem considered in the present work does not allow for guided learning (supervised learning), given the absence of a *teacher* for the FLS. Thus unsupervised learning must be sought. A learning methodology effective for both a design phase and an adaptation phase is desirable in order to achieve some kind of intelligent control (Antsaklis [3]). In the following the focus will be on the design phase and on a framework that could be extended to on-line adaptation of the scheduler. Indeed, adaptivity is currently under study, but is quite complex and would require a considerable additional amount of work. Moreover, the design problem itself is an important one, on which the current state of the art is lacking as emerged from the analysis of Chapter 2. The use of genetic algorithms (GA) in the context of the design of an FLSs seems promising. Examples of GA based design of membership functions (shape and parameters) and rules can be found in the literature (Kim et al. [33], Homaifar and McCormick [28]). This will be discussed in more detail in Section §3.4.3.

A direct approach can be used to determine improvements in the control action. With this method the configuration space is *explored* by modifying the control actions and observing how the performance changes, retaining beneficial changes. This type of trial-and-error learning has been called *reinforcement learning* because of its similarity to animal behavior modification used by psychologists. Assuming that the controller is a parametrized structure, the exploration of the parameter space can be performed either on-line with the actual system, or off-line with the use of a simulator. In the following Section §3.4.2 a *reinforcement learning* approach presented by Dadone et al. [12] is discussed along with its application, with evolutionary programming (EP) techniques (Fogel [17]), to the design of a FS that will thus become an evolutionary fuzzy scheduler (EFS).

3.4.2 REINFORCEMENT LEARNING

Dadone et al. [12] present a simulator-based reinforcement learning approach whose general architecture is presented in Fig. 3.7. It is assumed that the plant exhibits a behavior that can be simulated. A reinforcement learning mechanism adjusts the adaptive controller parameters by evaluating and trying to increase a performance index (*PI*). The *PI* provides an evaluation of the performance of the controller in a certain simulation run (in the case of a simulation, i.e. off-line adaptation) or in a particular sliding time-window (in the case of on-line adaptation). Thus, the reinforcement learning mechanism changes the controller parameters based on past and present values of the *PI* and seeking to maximize it. This approach can be applied off-line and on-line. On-line reinforcement learning is a more complicated task that will not be considered in the present work. Indeed adaptation would require controller's parameters changes quickly enough to adapt to the changing environment but still remaining within stability boundaries. Computing parameters' changes would require the computation of a *good* direction to follow (e.g., gradient) that is a complex problem for discrete event systems. Moreover, determining stability boundaries is

another complex task for DEDS, since no commonly accepted definitions of stability exist, except some intuitive ones (e.g., increasing buffer levels).

The design phase consists of finding, by means of simulations, some values for the parameters of a given parametric controller that optimize a performance index. Therefore, this phase can be regarded as a simulation response optimization problem where several approaches can be taken. Even though the field of simulation optimization is quite mature, there is no general *good* algorithm for simulation optimization, but there are several techniques which give good results depending on the particular problem that is considered. Therefore, there is no fixed method for the controller design, but a set of a methods to choose from. This makes the design phase, even though apparently simple, something far from a simple task.

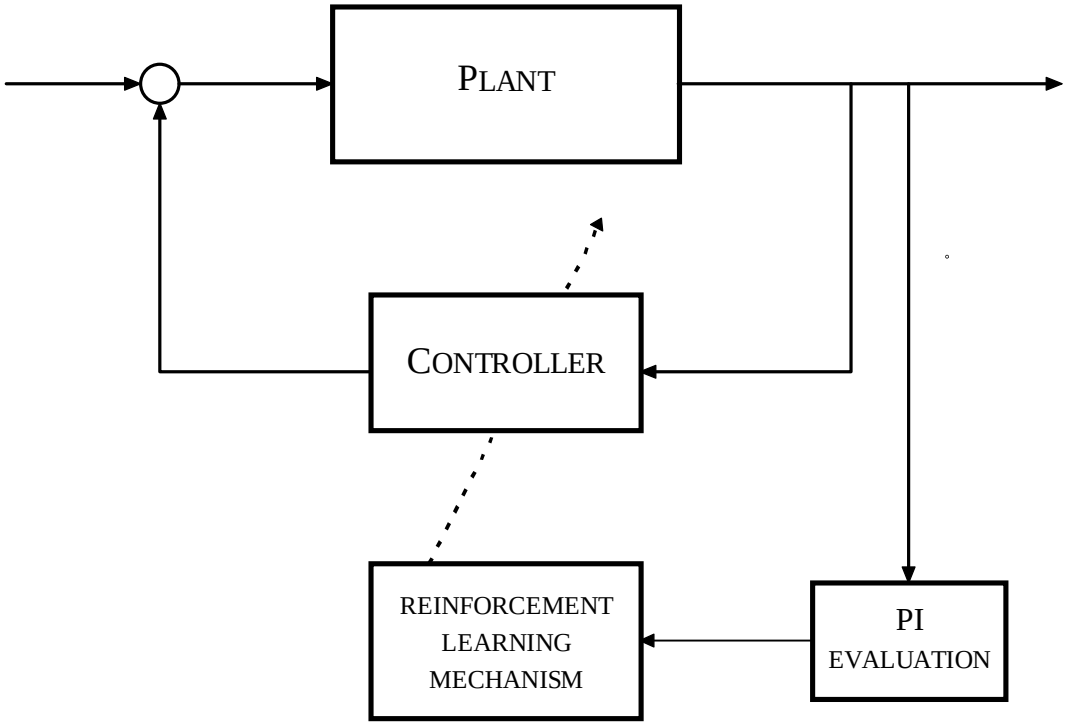


Figure 3.7 Reinforcement learning approach

The presence of noise on the *PI*, the number of variables involved and the necessity of numerically evaluating the gradient of the *PI* with respect to all the design parameters makes a non-gradient based approach more appealing. Therefore, the simulation response optimization (controller design) will be performed by using evolutionary programming techniques (EP). This is also motivated by the previously mentioned GA based approaches to FLS design and by the results in Dadone et al. [12]. Indeed evolutionary programming appears to be the most computationally expensive, but also the most robust and reliable technique.

In this study only the design problem (i.e. off-line adaptation) will be considered. Therefore, the plant of Fig. 3.7 is a simulation of a FMS. The adaptive controller in Fig. 3.7 is the FS described in Section §3.3. The *PI* should take into account the relevant performance measures for the FMS at hand, that is, the different and possibly contrasting objectives whose satisfaction gives the overall objective satisfaction.

3.4.3 THE EVOLUTIONARY PROGRAM

The design of the fuzzy scheduler consists of determining the parameters and rules that unequivocally characterize a FS. Such parameters are:

- Parameters that define the membership functions of the antecedents of sequencing rules: $(\mathbf{p_1}, \mathbf{p_2}) = (\varphi_{11}, \psi_{11}, \varphi_{12}, \psi_{12}, \xi_{12}, \varphi_{13}, \psi_{13}, \varphi_{21}, \psi_{21}, \varphi_{22}, \psi_{22}, \xi_{22}, \varphi_{23}, \psi_{23})$;
- Parameters that define the membership functions of the antecedents of priority setting rules: : $(\mathbf{p_3}, \mathbf{p_4}) = (\varphi_{31}, \psi_{31}, \varphi_{32}, \psi_{32}, \xi_{32}, \varphi_{33}, \psi_{33}, \varphi_{41}, \psi_{41}, \varphi_{42}, \psi_{42}, \xi_{42}, \varphi_{43}, \psi_{43})$;
- Parameters that define the membership functions giving the degree of satisfaction of each criterion in the routing problem: $\mathbf{p_5} = (\beta_1, \gamma_1, \theta_1, \beta_2, \gamma_2, \theta_2)$.
- Rules for sequencing and priority setting: they are stored in the matrices $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$ respectively. Both matrices are square and of order three. They are obtained from the rules shown in Tables 3.2 and 3.3 substituting the integers 0,1 and 2 instead of S, M and L (small, medium and large).

Note that in defining the design parameters for the fuzzy MADM of the routing problem the *h* parameter for low workload and low processing time was not considered. Indeed it is fixed at a certain value and is not considered in the design phase assuming a minor influence of this parameter on the controller performances because the corresponding parameter γ is already a design parameter. Moreover, the discrete membership function for small distance in the fuzzy MADM is not considered in the design phase too because of the minor importance attributed to the small distance criterion in the routing decision. Also note that the parameters defining the singleton membership functions for the consequents of the FLSs rules are not considered here. Indeed in both cases those consequents consist in the priority of jobs, therefore their value per se is not important but their order is. Thus three equally spaced values can be freely chosen without altering the performances of the scheduler.

A significant element in the design phase is the determination of other rules than the ones heuristically chosen, thus embedding the rules in the design process. An individual in the evolutionary program will be a vector with 52 components and given by:

$$\mathbf{I} = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5, \mathbf{R}_1, \mathbf{R}_2) \tag{3.10}$$

In the EP approach the algorithm starts with an initial population of individuals as (3.10) and the population evolves based on a fitness evaluation. After a sufficient number of generations the population will be composed mainly of *strong* individuals, i.e. solutions characterized by high fitness. In our case the algorithm starts from the FS that was previously heuristically designed and creates other individuals either randomly or by random perturbations of the FS. Once the initial population of, say, *k* individuals is initialized every individual is tested. That is, its fitness is determined by simulation of the FMS with FS with the parameters corresponding to the particular individual. The *PI* corresponding to a FS, that is to an individual, is the fitness of the individual. Once the fitness of all the individuals in the population is determined, the best *k*/2 individuals (i.e., the ones with the highest fitness) are chosen to be part of the next population. The remaining individuals for the next generation are either *offsprings* of the best

chosen or are randomly chosen. An *offspring* of an individual is simply defined as a perturbed version of the original individual. The perturbations are generally random with specific statistical characteristics.

In the EP used for this study the perturbations to the FS parameters ($\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\mathbf{p_4}$, $\mathbf{p_5}$) are additive independent random variables, normally distributed with zero mean and fixed standard deviation (around 5-15% of the parameter value range). The rules are perturbed according to a discrete probability distribution. What is changed in a rule is the consequent. Consequents, in our case, can only be small, medium or large. Therefore a probability is defined for every possible transition and the perturbation is performed according to this discrete probability distribution. The transition probability from consequent $x \in$ {small, medium, large} to consequent $y \in$ {small, medium, large} will be denoted with p_{xy} and will obviously satisfy:

$$\forall x \in \{\text{small,medium,large}\} : p_{x,small} + p_{x,medium} + p_{x,large} = 1.$$

Once a new *perspective* individual is determined (through perturbations or completely randomly) it must satisfy some feasibility constraints that make it an individual. Thus, a *perspective* individual will be inserted into the new population if it satisfies the feasibility constraints; otherwise it will be rejected and a new *perspective* individual; will be generated and again tested until a new feasible individual can be inserted in the population. The feasibility constraints on the parameters arise from the consideration that for the parameters to be representative of membership functions that keep their original meaning they must respect some *order*, that is, the end of a triangular membership function cannot occur before the mid-point of the triangle and so on. These constraints are expressed by:

$$\begin{gathered} \varphi_{i1} \le \psi_{i1} \qquad \varphi_{i2} \le \psi_{i2} \le \xi_{i2} \qquad \varphi_{i3} \le \psi_{i3} \\ \varphi_{i1} \le \psi_{i2} \le \psi_{i3} \qquad \beta_i \le \gamma_i \\ i = 1, 4 \end{gathered} \tag{3.11}$$

The EP evolves choosing from each generation the best fit individuals and building new generations by perturbations of the strongest individuals of the past one.

After a fixed number of generations the program is stopped. This technique is very similar to a GA on all the evolution type aspects. It differs from GAs though, in the fact that is less random, and neither mating nor mutation are considered. Thus offsprings in an EP are on average closer to their *parents* than in GAs. In fact, an offspring in an EP context is obtained by a small perturbation of its parent while in a GA approach is obtained by reproduction and mating. Thus the distance of an offspring from its parent in the EP approach is the small perturbation applied to its parent. On the other hand in a GA approach the offspring*s* are obtained by mating of two parents, thus they could fall in a region of the search space that is completely different from the ones where the parents were. The EP approach seems more appropriate for our case where an initial heuristically determined solution already exists. Anyway the use of a GA instead of an EP would not change anything in the structure of this approach. In order for the algorithm to evolve it is vital for it to be able to evaluate the fitness of each individual. In this study that computation corresponds to determining the *PI* corresponding to a particular FS. Unluckily, given the multiple objective nature of the problem, the *PI* evaluation is not a really straightforward task. This topic will be discussed in the next Section §3.4.4.

3.4.4 PERFORMANCE INDICES

The evaluation of a FS in a design context can be performed through simulations of the FMS plant with the given controller. As already stated before, the FMS control objective is not easily formulated in terms of a single performance measure but it consists of the satisfaction of several objectives possibly contrasting. A way to merge the objectives together in a single *PI* is sought. In this study a fuzzy MADM is employed for this purpose. More specifically the same fuzzy MADM technique used for routing is adopted. That is, fuzzy MADM as presented by Yager [57,58] is used. The details of this type of approach were previously presented in Section §3.2.3. This type of approach is preferred because not only does it allow for multiple objectives with different degrees of importance, but it is also very easy to change at a later time

to accommodate other objectives without changing the structure of the overall approach. The use of linguistic pairwise comparisons coming from human experts is also a very important aspect of this technique. Indeed linguistic management directives can be easily translated into degrees of importance of criteria forming a *PI*, with no further efforts or *interface* needs. As was already noted, this fuzzy MADM technique is basic and if necessary it could be complicated to accommodate for multiple experts, more hierarchical levels in the decision making progress and so on, without altering the overall structure of this approach. In the present study it is considered enough to consider this simpler approach.

The control objective consists in the satisfaction of four objectives (Tempelmeier and Kuhn [51], Viswanadham and Narahari [53]). The degree of satisfaction of each objective is determined through membership functions that are fixed *a-priori* from observations of FMS simulations under commonly used heuristics. In general the membership functions are fixed so that the degree of satisfaction corresponding to average heuristics performances is around 0.5~0.6. This way there is some space for performance improvement for the EFS, if possible. The degrees of satisfaction of the four objectives are then combined according to their importance to give the degree of satisfaction of the overall objective, that is the *PI*. The decision hierarchy is shown in Fig. 3.8. The four objectives that were considered in this study are:

- **O_1**: *Low time in system*: The goal is to minimize the excess time that jobs spend in the system. Thus the ratio between time in system and total processing time for a job is calculated and we seek to minimize it to a unit value, or at least close to that. The mean value of this ratio on a simulation of the FMS is measured. In case of multiple routings the minimum processing time is considered to form the total processing time. The universe of discourse for the corresponding membership function is the real interval [1, +∞]. The infinite upper bound obviously corresponds to a mathematical condition and not to a real physical condition, since it would mean that a job stays in the system for an infinite time. The membership function

giving the degree of satisfaction for this objective is shown in Fig. 3.9-a. Its functional form is:

$$\mu(x) = e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \tag{3.12}$$

This membership function is characterized by one parameter σ that was heuristically chosen to be $\sigma = m_{obs}$ where m_{obs} indicates the average ratio value for simulations under heuristic rules. With this choice of σ the average performance of heuristic rules in terms of this objective have a membership degree of approximately 0.6, leaving enough space for improvement for the evolutionary fuzzy scheduler.

- **O_2:** *Low WIP*: A high value of the work in progress means monetary resources blocked on the shop floor. Thus a low value for the WIP is sought. Ideally a WIP as low as the number of machines is sought. This way there would be one job per machine in the system. Thus the ratio between WIP and number of machines is measured and its average value over a simulation is a measure of how low the WIP is. Unit value of this ratio is desired. The universe of discourse corresponding to this ratio is the real interval [0, +∞]. Obviously the upper infinite bound corresponds to a mathematical condition but not a real physical condition. Indeed it would mean that an infinite number of jobs is in the system. The membership function giving the degree of satisfaction for this objective is shown in Fig. 3.9-b. Its functional form is:

$$\mu(x) = \begin{cases} x & 0 \le x \le 1 \\ e^{-\frac{1}{2}\left(\frac{x-1}{\sigma}\right)^2} & x \ge 1 \end{cases} \tag{3.13}$$

This membership function is characterized by the parameter σ set to $\sigma = m_{obs} - 1$ for the same reasons explained in the above objective. The given shape is motivated by the search of a unit value of the WIP over number of machines ratio. Thus, high values of WIP correspond to exponentially decreasing degrees of satisfaction of the objective. Low WIP values (i.e., below unity) are penalized with a linear function because it is the easiest decreasing function to fit in the interval [0,1] and passing

through the origin and the point (1,1). Moreover, it will be very difficult to encounter situations where the WIP is smaller than the number of machines thus the shape of the membership function between zero and unity is not deemed very important.

- **O_3:** *High resource utilization*: It is desirable to utilize most of the available resources, avoiding inactivity. For this purpose the ratio between the time a resource is busy and its total operating time is measured. The maximization of this value to unity is sought. The mean value of this ratio over all the machines and over a simulation is used as a measure of high utilization. The universe of discourse for this ratio is the real interval [0,1]. The ratio is converted to a degree of satisfaction of this objective using a membership function as the one shown in Fig. 3.9-c. Its functional form is:

$$\mu(x) = \frac{A}{1 + e^{-a(x-m)}} - \delta \tag{3.14}$$

This membership function is characterized by the parameters *A*, *a*, *m* and δ. *A* and δ are chosen so that the function passes for the origin, i.e. zero utilization correspond to zero degree of satisfaction of the high utilization objective. Thus: $A = 1 + \delta$ and $\delta = e^{-am}$. Moreover, to achieve an approximately 0.5 degree of satisfaction of the objective by the heuristic rules *m* must be chosen as $m = m_{obs}$. Note that this choices guarantee the 0.5 membership degree only in the realistic case that $m_{obs} > 0.5$, in problems with low resource utilization a different choice would be required. Finally, *a* needs to be chosen. The membership function of (3.14) will asymptotically assume the unit value, thus negating the possibility of a unit membership function for any utilization value. Using a pragmatic approach the parameter *a* can be chosen in such a way that for unit utilization the membership degree is very close to 1. Since the slope of the membership function for $x = m$ is proportional to *a* we choose *a* so that *five time constants* correspond to the distance from *m* to 1, that is:

$$a = \frac{5}{1 - m_{obs}} \tag{3.15}$$

With this choice for *a* unit utilization will correspond to an objective satisfaction degree of approximately 0.99, i.e. sufficiently close to unity.

- **O_4:** *Low tardiness*: It is desired to deliver finished jobs with no delays respect to the due date. The average value of the tardiness on a simulation constitutes a measure of this objective satisfaction. The universe of discourse for the tardiness is the real interval [0, +∞]. The infinite upper bound corresponds to a mathematical but not physical condition of at least one job with infinite tardiness, thus an infinite time in system. The degree of satisfaction of this objective is measured with the membership function shown in Fig. 3.9-a, i.e., the same of the first objective. Thus the functional form is the one in (3.12) and the parameter σ is set to $\sigma = m_{obs}$ for the same considerations discussed above.

A pairwise comparison matrix is defined, ***A***. This matrix embodies the judgment of a human expert on what is the real production objective. This matrix can be formed by having the plant experts compile some prepared questionnaires. In case of multiple experts, Buckley's technique [6] can be used. The matrix ***A*** is a square matrix of order four and contains the pairwise comparisons of the four objectives listed above. To determine the degrees of importance of each objective Saaty's λ_{max} technique is used.

At the end of a long simulation (or batch of simulations) performance measures corresponding to the given objectives are collected. Namely the following performance measures are obtained as the output of a simulation:

- average ratio of time in system over total processing time;
- average ratio between work in progress and number of machines;
- average resource utilization;
- average job tardiness.

The degree of satisfaction of each objective is determined from the corresponding membership functions. The degree of satisfaction of the overall production objective is then determined from the degrees of satisfaction and degrees of importance of each objective according to (3.5). This procedure leads to a *PI* that is a satisfaction degree

of the overall objective. Thus it will be a number defined in the real interval [0, 1]. The biggest is the *PI* the better the FS performed.

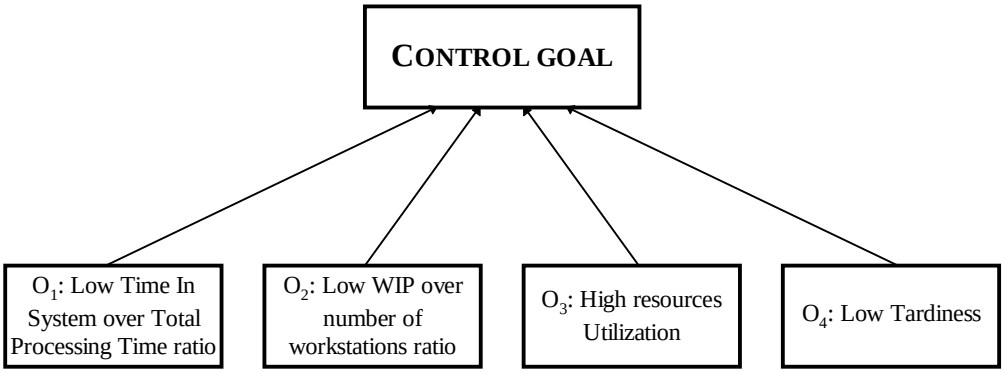


Figure 3.8 PI evaluation hierarchy

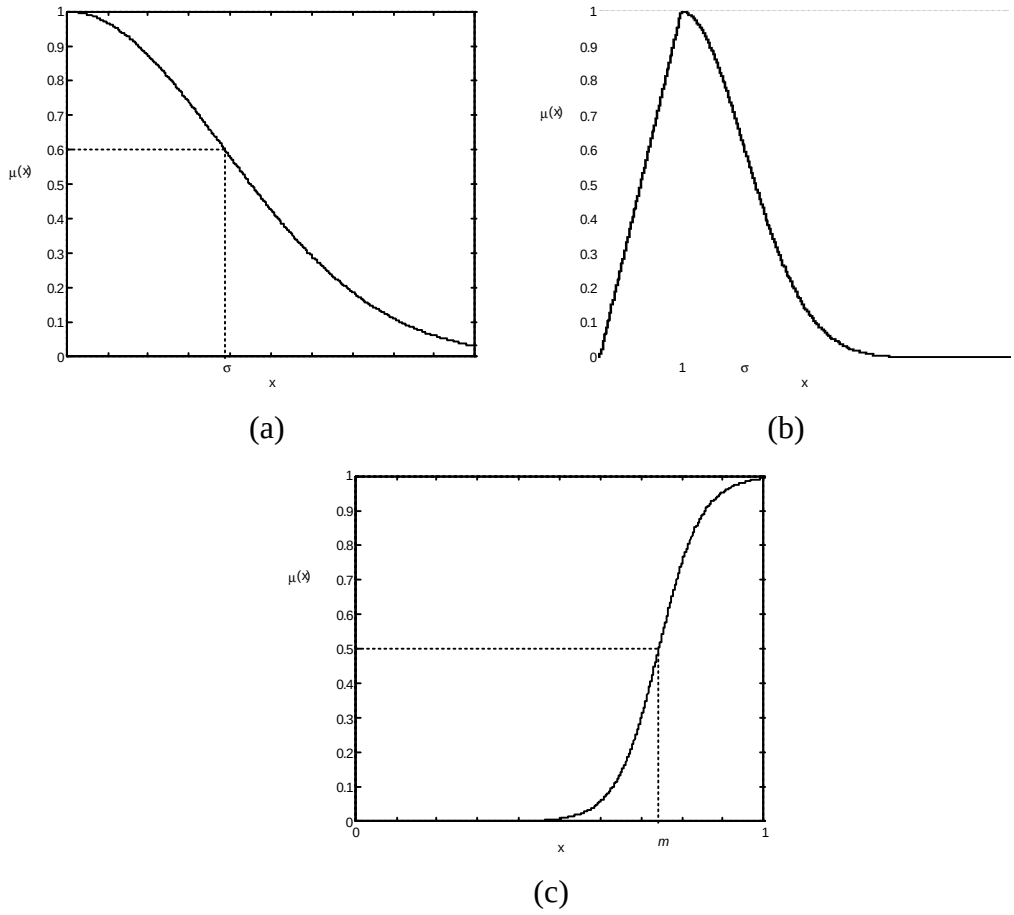


Figure 3.9 Membership functions of the four objectives: low time in system (a), low work in progress (b), high resource utilization (c), and low tardiness (a)

3.5 PERFORMANCE MEASURE ESTIMATION AND COMPARISONS

The underlying assumption of the proposed design technique is the availability of a simulation of the plant. Simulations are useful in estimating performance measures for design and also to make comparisons between different alternatives. In the following some statistical background is given, regarding how the comparisons of Chapter 4 were made.

Consider one performance measure and denote it by x. The mean value of this measure is μ_x and, since we are addressing a non-terminating simulation, is defined by:

$$\mu_x = \lim_{j\to\infty} E[x_j] \tag{3.16}$$

where x_j is the j-th observation of the performance measure and $E[\bullet]$ denotes the expectation operator. The simulation must be long enough to give a reasonable approximation of the limit, and to minimize the residual effects of the initial transient bias. Consider the case of n simulations and l steady-state observations of a performance measure for each simulation. Let x_{ij} denote the j-th observation in the i-th simulation (i=1, 2, …, n; j=1, 2, …, l). If l is sufficiently big the sample mean of the observations in the i-th simulation will be a random variable with mean that is approximately μ_x, that is:

$$m_i = \frac{\sum_{j=1}^{l} x_{ij}}{l} \quad \text{and} \quad E\left[m_i\right] \approx \mu_x \tag{3.17}$$

Thus, an approximately unbiased point estimate of μ_x will be given by averaging the m_is, that is:

$$\overline{m} = \frac{\sum_{i=1}^{n} m_i}{n} \quad \text{and} \quad E[\overline{m}] \approx \mu_x \tag{3.18}$$

If the simulations are independently seeded, the m_is are independent and identically distributed random variables (IID). Indeed, they will be normally distributed according to the central limit theorem. Therefore, an approximate (1-α) confidence interval for μ_x can be found as:

$$\overline{m} \pm t_{n-1,1-0.5\alpha} \frac{\sigma_m}{\sqrt{n}} \tag{3.19}$$

where t_a is the Student's *t* distribution with *a* degrees of freedom, and σ_m^2 is the sample variance of the m_is.

Consider two different system configurations (i.e., two possible structures of the same system, e.g., different controllers, different layout, etc.) generating observations x_{ij} and y_{ij} of a given performance measure. Comparing the two configurations in terms of the given performance measure consists in trying to compare their mean performances μ_x and μ_y. A first approach could be to simply compare the point estimates of μ_x and μ_y. This is a very crude approach since it does not take into consideration the variance in the measurements, thus it might eventually fail and no theoretical guarantees exist that the conclusions that are drawn from this comparison are exact with some probability. A better approach consists of comparing the confidence intervals for μ_x and μ_y concluding that one technique is better then the other if the two intervals do not overlap. This approach relies on the assumption that the observations corresponding to the two alternatives are independent, which is not always the case, but is a good first approximation comparison. A better way to compare μ_x and μ_y consists of finding a confidence interval for (μ_x - μ_y). If the interval contains zero then there is not enough evidence to conclude that the two alternatives have a different behavior, otherwise a conclusion can be reached. Thus the sample mean and confidence interval are formed for the data (x_{ij} - y_{ij}) with the same considerations as were used before. In particular, it is necessary to remember that even in this comparison there is one approximation involved, that is, the one expressed by (3.18). Since we are not running the simulation for infinite time, all the sample means that we are calculating are not exactly unbiased estimators of the corresponding mean.

They are asymptotically unbiased estimators, thus making the simulations long enough this approximation will be practically insignificant. In Chapter 4 comparisons between fuzzy scheduling and heuristics based scheduling will be performed using confidence intervals for (μ_x - μ_y). That is, confidence intervals for the mean performance difference will be formed and a difference will be noted in case this interval does not include zero.

3.6 CONCLUSIONS

In this chapter the scheduling problem for flexible manufacturing systems was discussed. Fuzzy set theory was introduced as a viable tool for approaching the control problem. Namely, fuzzy logic systems and a fuzzy multiple attribute decision making technique were used for the sequencing, priority setting and routing problems. The outcome of this synergy was a fuzzy scheduler, characterized by 34 parameters and 18 rules. A reinforcement learning approach based on evolutionary programming techniques was presented as a way to achieve a systematic design procedure for the fuzzy scheduler of a generic FMS plant. The underlying realistic assumption on this technique is that a simulation of the plant is available along with linguistic directives from human experts.

As already stressed before, the innovative aspects of this approach are the possibility of merging linguistic human directives (e.g., management directions) together with a systematic design procedure that accounts for multiple and possibly contrasting production objectives. Moreover, an adaptive extension of this approach could be achieved within the same framework.

CHAPTER 4
RESULTS

4.1 INTRODUCTION

This chapter focuses on a particular FMS for which a simulation model was built. The FS and the EFS were tested through simulations, and their performances were compared to commonly used heuristics. The details of the FS and of the EFS are discussed along with the simulation results.

4.2 THE FLEXIBLE MANUFACTURING SYSTEM

In the present study the FMS that was considered relies on the assumptions listed in Section §3.1.4. Moreover, the FMS used to test the performance of the FS and of the EFS is precisely characterized by:

- A FMS with six machines, three AGVs, one load and one unload station (Fig. 4.1) is considered. Each workstation has an input buffer. The AGVs are initially positioned at the load station. The AGV chosen for moving a job is always the closest one to the job among the AGVs that are free. When an AGV ends moving a piece, it stays where it is (i.e., at the destination workstation) until it receives another request.
- Every order has a random routing. The number of different processing operations a job has to go through is a random variable characterized by a discrete probability distribution. A job will need 4 different operations with 30% probability, 5 operations with 40% probability and 6 operations with 30% probability. The route plan for an order is determined in the following way. At first the number of processing operations needed by a job is determined. Afterwards the machine necessary for each operation is determined according to a discrete probability distribution with same probability for each workstation to be chosen. A first route plan with no alternatives is thus generated. The different routing possibilities (i.e., a piece can go to another machine for the same type of processing) are then added to the first route plan. This is done according to a discrete probability distribution.

There will be only one alternative processing in the route plan with 10% probability, there will be two different alternatives with 50% probability and three alternatives with 40% probability. Once the number of alternatives to add to the route plan is determined they are randomly positioned on the initial routing plan according to equal probabilities for each operation in the plan. This is done by randomly choosing an operation in the plan and adding another possible machine for processing, choosing from all the machines with equal probability. The only constraint is that the same machine of the first plan, or the next or previous one in the plan cannot be chosen. This process is repeated for all the alternatives that need to be added (from 1 to 3). In case the number of alternatives to add is three, the alternatives cannot end up all on the same operation. Thus every operation can be performed by at least one machine but by no more than three machines (i.e., one from the initial plan and at most two alternatives).

- Loading and unloading times are randomly distributed with a triangular probability distribution starting at 1 time unit, going to unit probability at 10 time units and then going back to zero at 11 time units. Processing times for every operation in every machine (i.e., different alternatives have different processing times) are normally distributed with a mean of 20 time units and standard deviation of 5 time units. Moreover the processing time cannot be smaller then 5 time units, therefore in the very unlikely case it turns out to be so it is *clipped* at 5 time units.
- Orders arrive as Poisson processes with mean inter-arrival time of 40 time units. The due date of an order is assumed to be the double of the total processing time for that order, modulated by an almost unit factor. That is, a random variable normally distributed with unit mean and 0.05 standard deviation. The total processing time is defined as the minimum total processing time, that is, in case of multiple alternatives the one corresponding to the minimum processing time is chosen for total processing time calculation.
- When a job is actually processed by a workstation its processing time is the one previously calculated plus a small perturbation. This perturbation is a random

variable with triangular probability distribution characterized by (-2,0,5) for machines from 1 to 4 and by (-1,0,1) for machines 5 and 6.

- The mutual distances between machines are summarized in the following ***D*** matrix. The element d_{ij} corresponds to the distance of machine *i* from machine *j*. The values 7 and 8 of the indices correspond to the load and unload stations respectively. The distance from every machine, or station, in the plant and the two adjacent ones is of 5 length units.

$$\mathbf{D} = \begin{bmatrix} 0 & 5 & 10 & 15 & 20 & 15 & 5 & 10 \\ 5 & 0 & 5 & 10 & 15 & 20 & 10 & 15 \\ 10 & 5 & 0 & 5 & 10 & 15 & 15 & 20 \\ 15 & 10 & 5 & 0 & 5 & 10 & 20 & 15 \\ 20 & 15 & 10 & 5 & 0 & 5 & 15 & 10 \\ 15 & 20 & 15 & 10 & 5 & 0 & 10 & 5 \\ 5 & 10 & 15 & 20 & 15 & 10 & 0 & 5 \\ 10 & 15 & 20 & 15 & 10 & 5 & 5 & 0 \end{bmatrix}$$

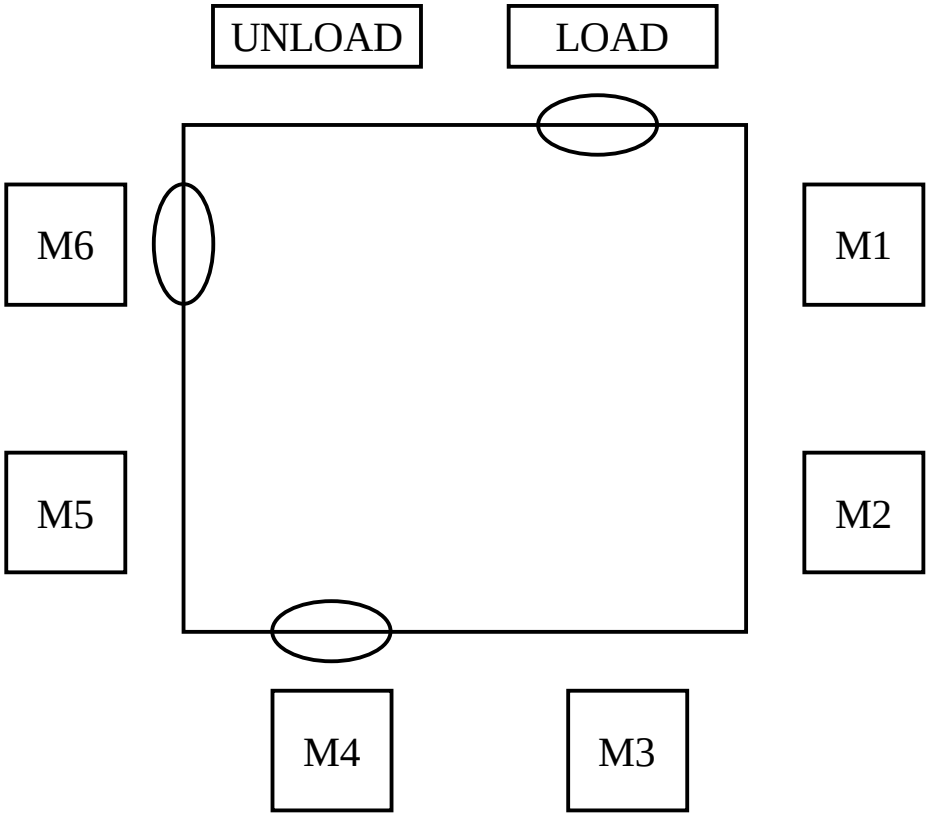


Figure 4.1 Layout of the FMS

4.3 SIMULATIONS

The FMS described above was simulated using the SIMAN simulation package. The simulation is a non-terminating simulation, that is the interest is not focused on the behavior of the system from an initial time to an ending time but on its steady state behavior. For this purpose a long simulation time is needed and the initial transient observations of the variables of interest are discarded. From observations of the simulations the total simulation time was set to 100 000 time units and the transient time was conservatively assumed to be 30 000 time units. Thus, the variables of interest (time in system over total processing time ratio, WIP over number of machine sratio, utilization and tardiness) are collected only after 30 000 simulation time units. Under these conditions the simulation time for this system is of approximately 12 seconds using a PC Pentium II at 233 Mhz.

For the purposes of *PI* evaluation for the evolutionary fuzzy scheduler, only one simulation is used. Indeed the times concerned with the evolutionary program would otherwise increase enormously. Moreover, the simulation is run for long enough to give accurate measures of the *PI*. Obviously, increasing the number of simulations would decrease the uncertainty in the measurement at the cost of added simulation time. In this study a pragmatic approach of searching for a *good* but not necessarily optimal solution in a reasonable time motivated the choice. In other cases where more precision is demanded and more time is allotted, replications are recommended. For the purpose of comparison of the FS, the EFS and commonly used heuristics 100 independently seeded simulations were used.

4.4 FUZZY SCHEDULER

The *heuristically designed* FS is characterized by the following parameter values:

- parameters defining the membership functions of the antecedents for the sequencing FLS:

$$(\mathbf{p_1}, \mathbf{p_2}) = (\varphi_{11}, \psi_{11}, \varphi_{12}, \psi_{12}, \xi_{12}, \varphi_{13}, \psi_{13}, \varphi_{21}, \psi_{21}, \varphi_{22}, \psi_{22}, \xi_{22}, \varphi_{23}, \psi_{23}) =$$

$$= (40, 90, 50, 100, 150, 130, 180, 80, 180, 100, 200, 300, 260, 360)$$

- parameters defining the membership functions of the consequents for the sequencing FLS:

$$(p_{11}, p_{12}, p_{13}) = (10, 20, 30)$$

- rules for the sequencing FLS as in Table 3.2;
- parameters defining the membership functions of the antecedents for the priority setting FLS:

$$(\mathbf{p_3}, \mathbf{p_4}) = (\varphi_{31}, \psi_{31}, \varphi_{32}, \psi_{32}, \xi_{32}, \varphi_{33}, \psi_{33}, \varphi_{41}, \psi_{41}, \varphi_{42}, \psi_{42}, \xi_{42}, \varphi_{43}, \psi_{43}) =$$

$$= (-200, -100, -150, -50, 50, 0, 180, 10, 20, 15, 20, 25, 20, 30)$$

- parameters defining the membership functions of the consequents for the priority setting FLS:

$$(p_{21}, p_{22}, p_{23}) = (10, 20, 30)$$

- rules for the priority setting FLS as in Table 3.3;
- parameters that define the membership functions giving the satisfaction degree for the routing fuzzy MADM:

$$\mathbf{p_5} = (\beta_1, \gamma_1, \theta_1, \beta_2, \gamma_2, \theta_2) = (10, 20, 50, 10, 20, 5)$$

$h = 0.5$ small distance fuzzy set: $\left\{\frac{1.0}{5}, \frac{0.8}{10}, \frac{0.4}{15}, \frac{0.2}{20}\right\}$

- pairwise comparison matrix for the routing criteria:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 1/3 & 1 & 3 \\ 1/5 & 1/3 & 1 \end{bmatrix}$$

The maximum eigenvalue of **A** is $\lambda_{max} = 3.0385$ (it is very close to 3, thus the matrix is consistent) and the corresponding eigenvector (normalized to a value of 3 in the 1-norm) is $\mathbf{w} = [\ 1.911, 0.7749, 0.3142]^T$, thus giving the absolute weights of the three criteria.

4.5 EVOLUTIONARY FUZZY SCHEDULER

The EP approach that was used in this study does not use any completely randomly generated individuals, because the underlying assumption is that the FS is already a good scheduler and fine tuning is sought more so than a complete random search. Therefore, only offsprings generation through additive perturbations of the *parents* (best individuals of the last generation) is done. The EFS is completely characterized by the parameters defining the probability distributions for the perturbations and by the parameters that completely define the *PI* formation mechanism. All those parameters are given in the following:

- perturbations on $\mathbf{p_1} \sim N(0,5)$ (i.e., normally distributed with zero mean and standard deviation of 5);
- perturbations on $\mathbf{p_2} \sim N(0,10)$;
- perturbations on $\mathbf{p_3} \sim N(0,2)$;
- perturbations on $\mathbf{p_4} \sim N(0,1)$;
- perturbations on $\mathbf{p_5} \sim N(0,1)$;
- transition probabilities for the consequents of the rules of the FLSs: with 50% probability the consequent will stay the same, with 50% probability it will change to another one with 25% probability each;
- parameters defining the low time in system over total processing time ratio membership function: $\sigma = 3.82$;
- parameters defining the low WIP over number of machines (6) ratio membership function: $\sigma = 0.77$;

- parameters defining the high utilization membership function: m = 0.74, a = 19.3, $\delta = 5.7\times10^{-7}$ and $A = 1 + 5.7\times10^{-7}$;
- parameters defining the low lateness membership function: σ = 206;
- pairwise comparison matrix for the production objectives:

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 2 & 1/5 \\ 1/4 & 1 & 1/3 & 1/7 \\ 1/2 & 3 & 1 & 1/5 \\ 5 & 7 & 5 & 1 \end{bmatrix}$$

In this case λ_{max} = 4.1478, which is not so far from 4 (therefore the consistency property is not violated for **A**), and the weights are given by the corresponding normalized (i.e., to a value of 4 in the 1-norm) eigenvector **w** = [0.7646, 0.2273, 0.4976, 2.5105]^T.

4.6 HEURISTICS USED FOR COMPARISON

In order to test the FS and the EFS they were compared with commonly used heuristics. Some of the heuristics already discussed in §2.2 were considered, more precisely:

- for sequencing: EDD, FIFO, LTPT, STPT;
- for priority setting: EDD, FIFO, LIFO, LS, SPT, SPT/TPT;
- for routing: SQL, SQW.

There are 48 possible combinations of these rules, each combination giving a different heuristic scheduler. All the combinations were tested for 100 simulations and the five best ones were considered for detailed comparison with the FS and the EFS. The five best combinations are:

- **H_1:** EDD, EDD, SQW;
- **H_2**: FIFO, EDD, SQW;
- **H_3**: STPT, EDD, SQW;
- **H_4**: FIFO, LS, SQW;
- **H_5**: STPT, LS, SQW.

From this ranking it can be noted that in the particular FMS under consideration the best rules for sequencing are either the ones that leave the orders the way they arrived or let the orders with the smallest processing time to go first or (in one instance only, H_1) let the orders with the earliest due date go first. This means that either the ordering is untouched (i.e., FIFO), or precedence is given to the fastest orders in order not to congest the system with high processing time parts that otherwise would penalize fast orders with long waiting times. In the priority setting problem the best heuristics are EDD and LS, that is, those heuristics having a more complete past, present and future picture of a job. Finally in the routing problem the best heuristic is the SQW, thus showing that it is better to choose among alternatives the one with the shortest queue workload and not queue length. The results corresponding to the use of these sets of heuristics are discussed later in comparison with the results obtained with the FS and the EFS. The *PI* for those heuristic sets of rules is approximately 0.12.

4.7 RESULTS AND COMPARISON

In order to determine an EFS a reinforcement learning approach using EP techniques was used. The EP that was employed was previously described in Chapter 3 and in Section §4.5. Each generation consists of ten individuals. The starting generation is determined by perturbations from the FS of Section §4.4. The program evolves from generation to generation by keeping the best five individuals and generating their offsprings. Running the evolutionary optimization for 400 generations leads to the convergence history shown in Fig. 4.2, where the *PI* of the best and worse individuals among the five best of each generation is shown. Considering the final generation of the evolutionary algorithm, the best five individuals are considered, each of them corresponding to an EFS.

Heuristic rules, FS and EFSs were then compared in terms of their mean performances over 100 independent simulations. The comparison for the *PI* is shown

in Fig. 4.3 (95% confidence intervals are also indicated), where EFS_1 and EFS_2 represent respectively the best and the worse of the evolutionary fuzzy schedulers. From the figure it can be noted that the FS has a *PI* that is around 15% better than that of the heuristic rules while the EFS has a *PI* that is 20% better than that of heuristic rules. The small difference between the EFS *PI* and the FS *PI* seems to imply that the FS is robust to parameter changes, confirming the assumption of fine tuning characteristics for this case. Through the analysis of performance on single objectives (Figs. 4.4 to 4.7, 95% confidence intervals are indicated) it is seen that the FS and the EFSs are slightly better than the heuristic rules with respect to time in system and work in progress. Moreover, FS and EFSs are a bit worse than heuristic rules with respect to resources utilization, and better by about 10% than heuristic rules with respect to tardiness. Both the FS and the EFSs are mostly effective in decreasing job tardiness. This is in complete agreement with the weights for the objectives. Indeed low tardiness is the most important objective (i.e., has the highest weight), and is therefore the one for which the performances are optimized the most. It can also be observed how heuristic rules and fuzzy scheduling in this case act following two different strategies. Specifically, higher values of utilization and tardiness for heuristic rules suggest that more pieces are placed into the system in order to utilize the resources as much as possible, but with a high risk of congestion problems. On the other hand, fuzzy scheduling sends fewer pieces into the system so that the system is less utilized, but no congestion problems appear.

More precise comparisons can be done by building confidence intervals for the performance difference between fuzzy and heuristic techniques as explained in Section §3.5. The best among the heuristics (H_2) was compared to the three fuzzy schedulers FS, EFS_1 and EFS_2. Confidence intervals at 95% confidence level were calculated and normalized to H_2 performances. Figures 4.8 to 4.12 show the comparisons between H_2 and FS, EFS_1 and EFS_2 in terms of the *PI* and of the single objectives. The same conclusions as before hold. Moreover we can also conclude that with approximately 95% confidence level all the fuzzy solutions are better then the heuristics. It is also

interesting to compare the results of the pairwise comparisons. Indeed the EFS_1 results are better than the FS (in comparison to H_2) in terms of low time in system to total processing time ratio. While on other performance measures EFS_1 is slightly better then FS. The results for EFS_2 (the worse evolutionary solution) are *somehow* in between FS and EFS_1. Indeed EFS_2 is in between FS and EFS_1 in terms of *PI*, time in system and utilization. The EFS_2 is worse then the FS (even though a significant statistical difference cannot be proven) in terms of average work in progress and tardiness.

The evolutionary fuzzy scheduler was then tested under different operating conditions than the ones it was designed for. Indeed the EFS_1 (designed for order inter-arrival time of 40 time units) was tested along with H_2 for order inter-arrival time varying between 31 and 50 time units. The results of these comparisons are shown in Figs. 4.13 to 4.16. In these figures the dark areas represent the performances of the FMS controlled by H_2 and the light colored areas represent the performances with EFS_1 control. It can be noted that for all the performance measures, except utilization, both the fuzzy and heuristic based control have what it appears to be an exponential type degradation. The evolutionary fuzzy scheduler shows a slower degradation than the heuristic, even though no term in the EFS takes into account the arrival rate of orders. This confirms what emerged from the previous discussion of the results. That is, the fuzzy approach is effective in reducing system congestion.

The different behavior of the utilization (Fig. 4.15) can be intuitively explained, along with the exponential type behavior of the other performance measures. Indeed, decreasing the order inter-arrival time means increasing the rate that orders arrive to the system, thus increasing congestion in general. This is reflected by increasing time in system (Fig. 4.13), increasing WIP (Fig. 4.14) and increasing tardiness (Fig. 4.16) in an exponential fashion. The machine utilization on the other hand will increase too, up to the limit of unity but with a non-exponential behavior because it is converging to a finite value. Indeed in a situation where the system is congested, the machines will be working at a very high rate that eventually will end in a unit utilization.

These results are very promising because they show that even though the evolutionary fuzzy scheduler is designed for a certain operating condition, its performance degrades slower than those of commonly used heuristics. Moreover, the EFS does not have any term that takes into account the inter-arrival time of orders. Therefore, eventually adding some rules or mechanism to take the inter-arrival time into account would make the approach more robust. It should also be noted that the fuzzy approach has the potential for taking into account the inter-arrival time and other sources of variability, while heuristic rules in their common versions do not.

It can be concluded that the proposed fuzzy techniques exhibit better performance than commonly used heuristic rules. Moreover, the adopted design technique is successful in improving the fuzzy scheduler performances where possible.

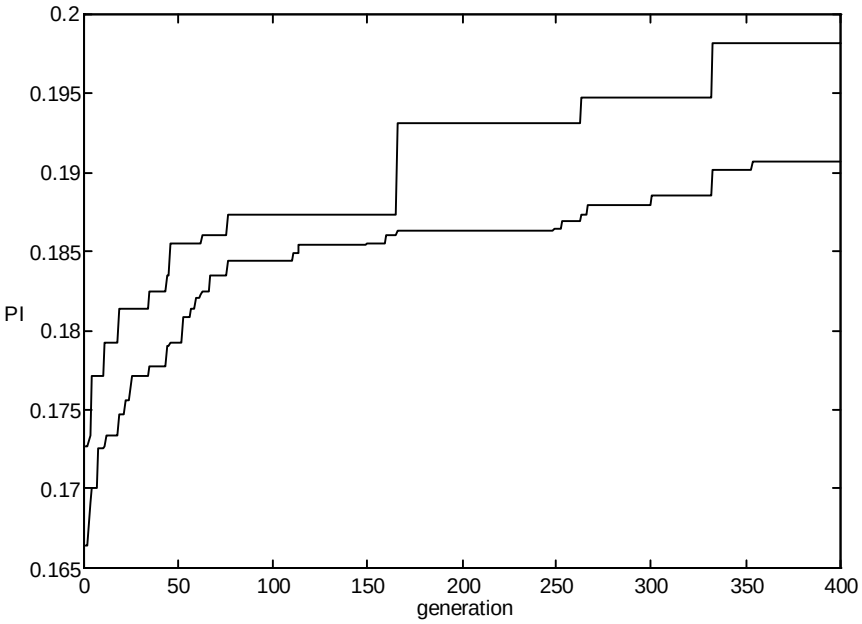


Figure 4.2 Convergence history for the evolutionary algorithm

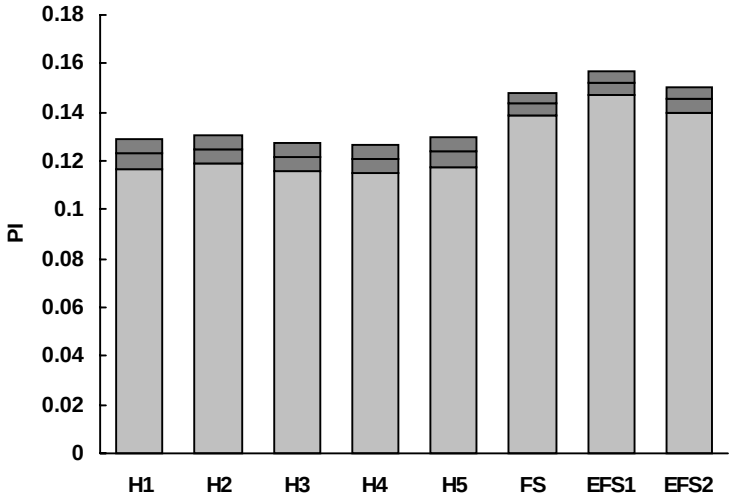


Figure 4.3 *PI* for heuristics and fuzzy rules

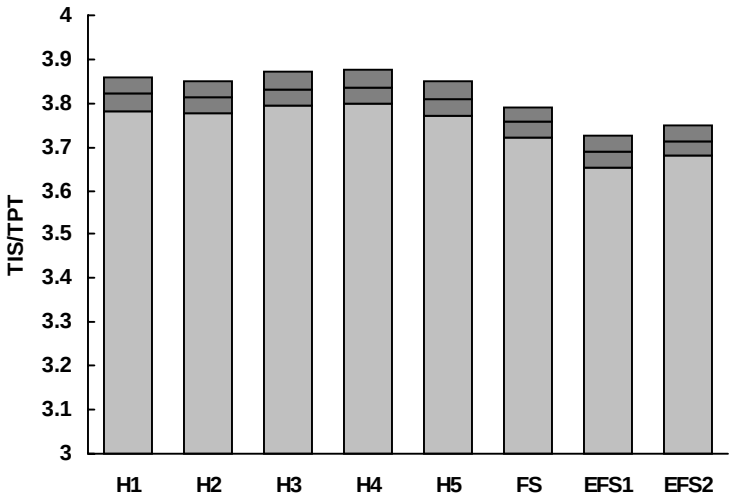


Figure 4.4 Time in system over total processing time ratio for heuristics and fuzzy rules

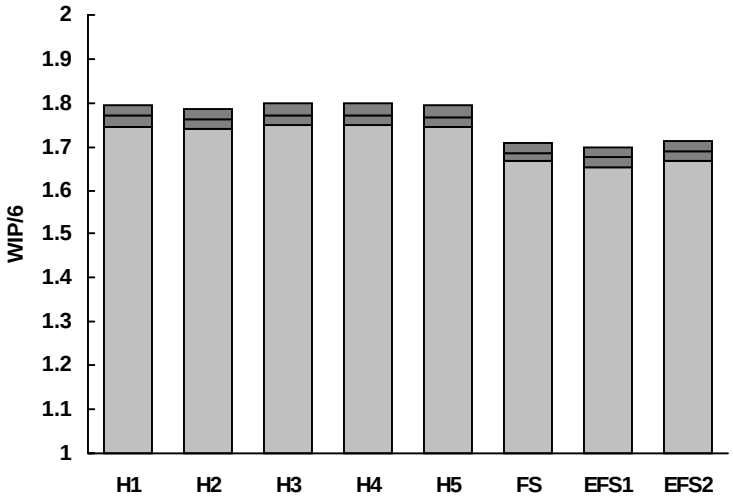


Figure 4.5 WIP over number of machines ratio for heuristics and fuzzy rules

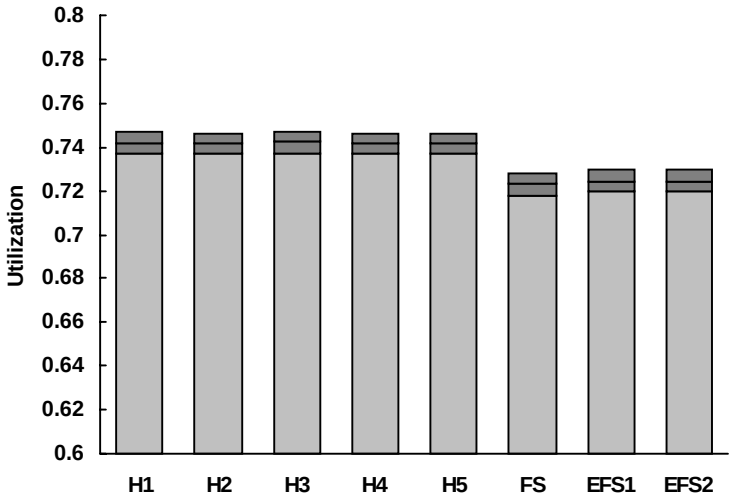


Figure 4.6 Utilization for heuristics and fuzzy rules

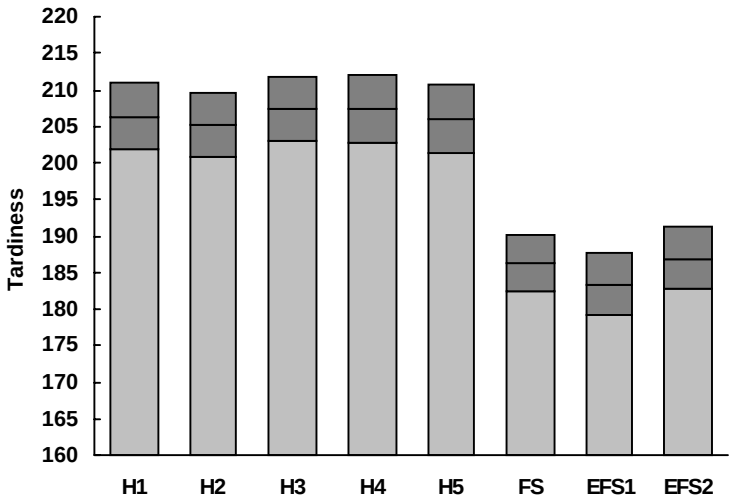


Figure 4.7 Tardiness for heuristics and fuzzy rules

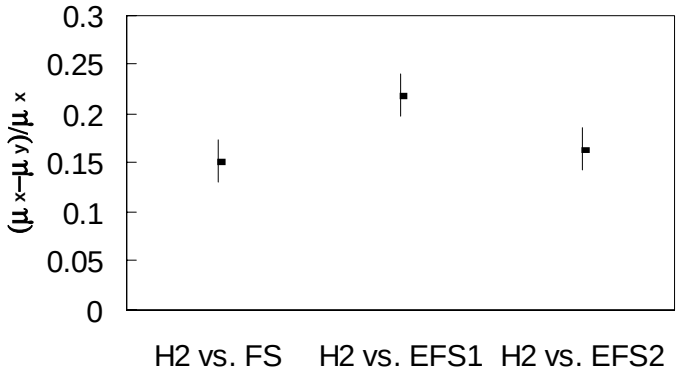


Figure 4.8 Confidence intervals (95%) for *PI* comparisons

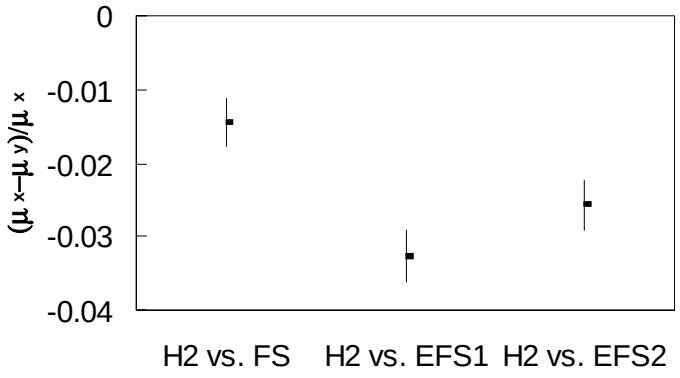


Figure 4.9 Confidence intervals (95%) for time in system over total processing time ratio comparisons

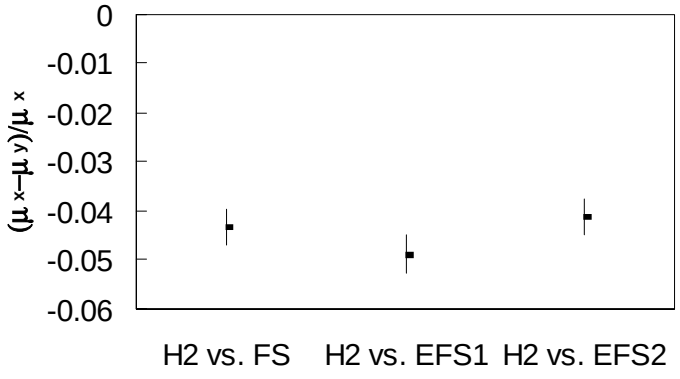


Figure 4.10 Confidence intervals (95%) for WIP over number of machines ratio comparisons

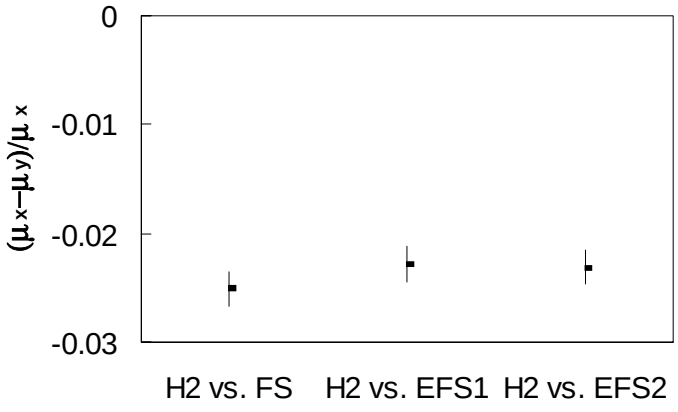


Figure 4.11 Confidence intervals (95%) for utilization comparisons

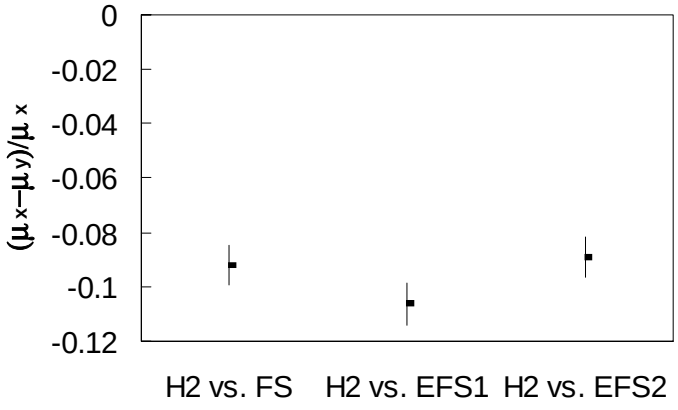


Figure 4.12 Confidence intervals (95%) for tardiness comparisons

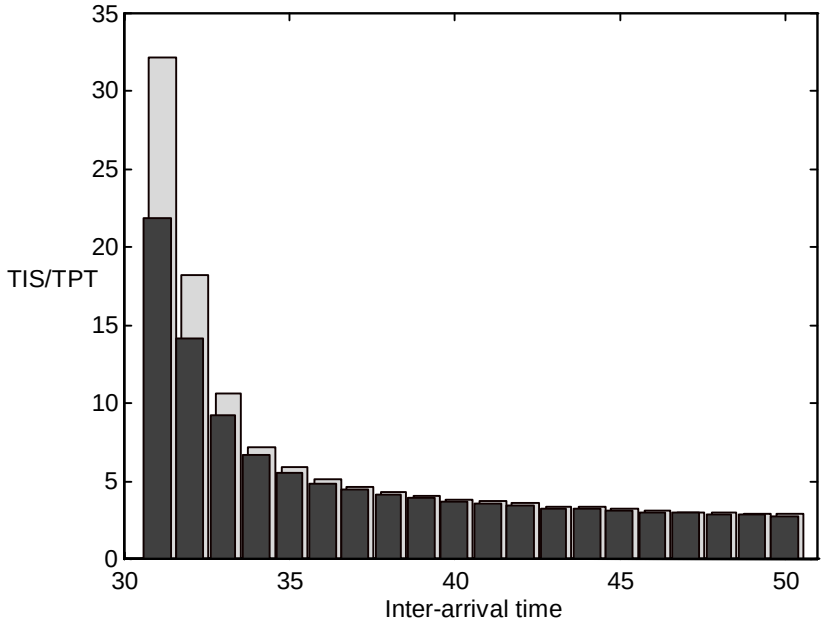


Figure 4.13 Comparison on time in system over total processing time ratio of H_2 (light) and EFS_1 (dark) for varying inter-arrival time

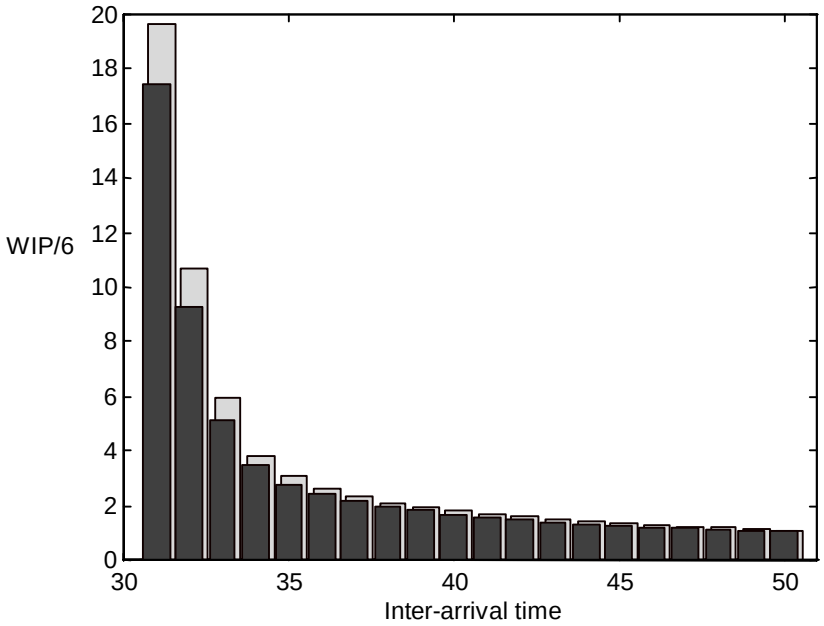


Figure 4.14 Comparison on work in progress over number of machines ratio of H_2 (light) and EFS_1 (dark) for varying inter-arrival time

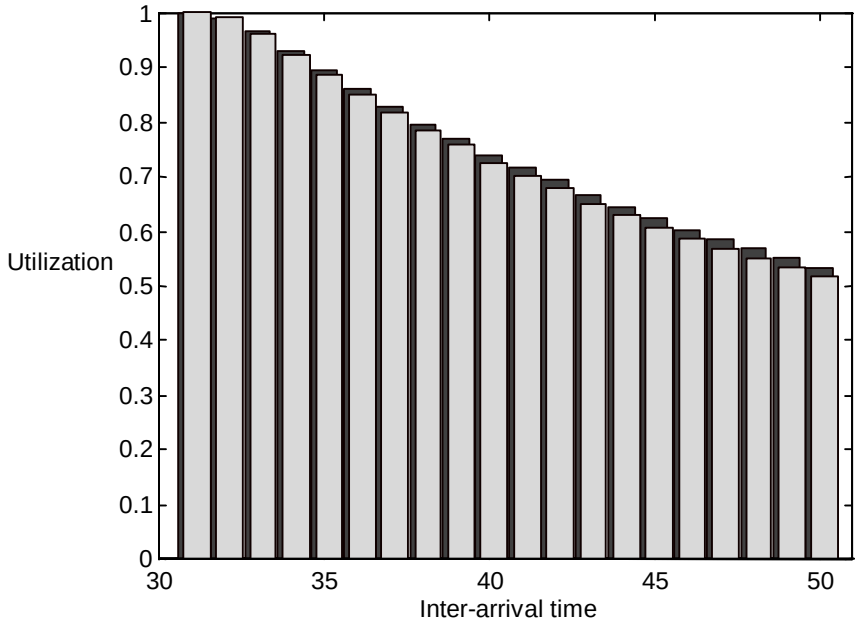


Figure 4.15 Comparison on utilization of H_2 (light) and EFS_1 (dark) for varying inter-arrival time

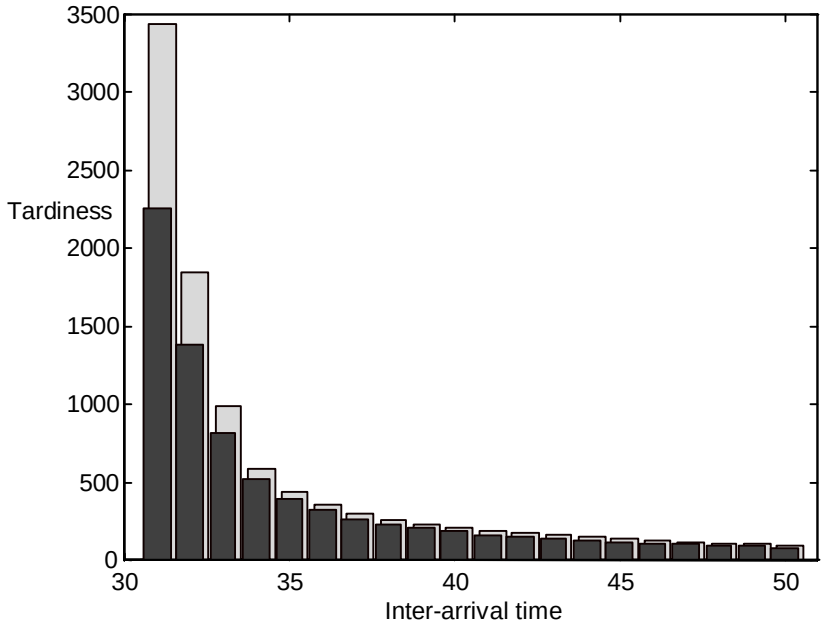


Figure 4.16 Comparison on tardiness of H2 (light) and EFS1 (dark) for varying inter-arrival time

4.8 CONCLUSIONS

In this Chapter a particular FMS plant with several stochastic components was presented as the target for the proposed fuzzy control technique. The details of the fuzzy scheduler and of the evolutionary fuzzy scheduler were given. The simulation results for commonly used heuristics and fuzzy and evolutionary fuzzy techniques were also presented and discussed, concluding that fuzzy techniques are successful in increasing some given performance measures. Moreover the presented design technique was successful in fine tuning the fuzzy scheduler for higher levels of performance. Finally, the evolutionary fuzzy scheduler that was designed shows slower performance degradation than heuristic rules for varying order inter-arrival time.

CHAPTER 5
CONCLUSIONS

The work presented in this thesis was directed towards investigating the applicability of fuzzy techniques as a decision aid in the short-term control of flexible manufacturing systems. For this purpose a flexible manufacturing system composed of six machines, three AGVs, one load and one unload station and with routings and arrivals with fixed statistical characteristics was considered. A fuzzy scheduler for sequencing, priority setting and routing was developed. This scheduler uses fuzzy logic systems as well as fuzzy multiple attribute decision-making techniques. Finally, a reinforcement learning technique for the fine-tuning of the scheduler parameters was employed. This methodology consists of learning guided by the value of a performance index that takes into account the different and conflicting objectives of an FMS, aggregating them in a fuzzy fashion. In this approach the importance of each objective can be extracted from linguistic directives given by experts. This thesis addressed only the off-line learning of the fuzzy scheduler (design). Simulations of the controlled plant were used to evaluate the objective function to be maximized with an evolutionary programming approach. The outcome of this optimization was called an evolutionary fuzzy scheduler (EFS).

Both fuzzy schedulers were compared to the best combinations of heuristic rules in terms of time in system, work in progress, utilization and tardiness. They show better performance than heuristic rules in terms of system congestion, that is, time in system, work in progress and tardiness. The evolutionary approach was successful in increasing the performance of the fuzzy scheduler (where possible), thus leading to a systematic design approach that takes into account the multiple objective nature of the problem. Thus, the thesis was successful in showing increased performance by using fuzzy techniques and also in giving a systematic design procedure (lacking in the literature) that takes into account multiple objectives and needs no interface with linguistic directions from human experts (e.g., management). Some objectives and their corresponding importance have been considered, but the technique used here to extract the *PI* and optimize the controller can be easily adapted to any plant requirement. Indeed, any number of objectives (or objective hierarchy) can be used and the importance of each objective can

be specified by human experts in terms of linguistic variables (e.g., high utilization is "much more important" than tardiness). There is no need for an interface between the FMS experts and the fuzzy experts, since the FMS experts need only to state the priorities for the plant, for these statements to be transformed into degrees of importance needed by the PI extraction module.

The present study employed a reinforcement-learning approach to the off-line design problem. This same framework could be used for on-line adaptation too, which is one of the requirements for a truly optimizing controller for an FMS. Indeed, a high performance controller should be able to adapt itself to all the unforeseen events on the shop-floor (i.e., machine failures, priority changes, etc.). Studies on adaptivity of the fuzzy scheduler are in progress and are strongly suggested for future developments on the application of fuzzy control to manufacturing systems.

The possibility of controller adaptation is strongly connected to the optimizing paradigm that is employed, along with computational power. Indeed, an evolutionary approach like the one used in this work would probably be too expensive in terms of computations. Thus, modifications of the optimization approach are also suggested as future research. Probably the integration of random search approaches (i.e., evolutionary programming, genetic algorithms) together with (approximate) gradient based methods could speed the optimization and thus eventually lead to adaptive properties. From this perspective, studies regarding gradient approximation techniques in discrete event simulations (e.g., infinitesimal perturbation analysis, harmonic gradient techniques, etc.) are of great importance. Also, trying to reduce the number of design parameters through sensitivity analysis and other approaches is desirable.

Another field of expansion of the present work is to try to investigate the type of performance degradation experienced with varying inter-arrival time. Indeed, finding the function approximating the collected data and theoretically explaining it for both the heuristics and the fuzzy approach would help in finding some kind of qualitative relation between the performance degradation behavior and the controller parameters. This way the slower performance degradation experienced by the evolutionary fuzzy scheduler could be explained and eventually improved.

Finally, further studies regarding the performance index formation technique are important. Indeed, the effectiveness of fuzzy MADM techniques should be tested in comparison to other techniques. Moreover, the employed fuzzy techniques could be extended in order to account for more complex, thus more real, situations (multiple experts, incomplete and fuzzy comparisons, etc.).

REFERENCES

[1] A. Angsana and K.M. Passino, "Distributed fuzzy control of flexible manufacturing systems," *IEEE Transactions on Control Systems Technology*, vol. 2, n. 4, pp. 423-435, December 1994.

[2] A. Angsana and K.M. Passino, "Distributed intelligent control of flexible manufacturing systems," *Proceeding of the 1993 American Control Conference*, pp. 1520-1524, 1993.

[3] P.J. Antsaklis, "Intelligent learning control," *IEEE Control Systems Magazine*, pp. 5-7, June 1995.

[4] D. Ben-Arieh and E.S. Lee, "Fuzzy logic controller for part routing," *Design and Implementation of Intelligent Manufacturing Systems* (H.R. Parsaei and M. Jamshidi, Eds.), pp. 81-106, PTR Prentice Hall, 1995.

[5] F. Biennier, J. Favrel, and G. Belson, "Integration of information and knowledge from the engineering activity to the workshop control," *Proceedings of the IEEE International conference on Robotics and Automation*, vol. 1, pp. 840-845, 1993.

[6] J.J. Buckley, "Fuzzy hierarchical analysis," *Fuzzy sets and systems*, n. 17, pp. 233-247, 1985.

[7] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*, Richard D. Irwin and Aksen Associates Inc. Publishers, Boston, MA, 1993.

[8] P.P. Chen, "The entity-relationship model toward a unified view of data," *ACM Transactions on Data Base Systems*, vol. 1, n. 1, pp. 9-36, March 1976.

[9] H. Cho and R.A. Wysk, "Robust adaptive scheduler for an intelligent workstation controller," *International Journal of Production Research*, vol. 31, n. 4, pp. 771-789, April 1993.

[10] F. Choobineh and M. Shivani, "Dynamic process planning and scheduling algorithm," *Proceedings of the Japan/USA Symposium on Flexible Automation*, pp. 429-432, ASME 1992.

[11] L.M.M. Custodio, J.S.S. Sentieiro, and C.F.G. Bispo, "Production planning and scheduling using a fuzzy decision system," *IEEE Transactions on Robotics and Automation*, vol. 10, n. 2, pp. 160-168, April 1994.

[12] P. Dadone, H.F. VanLandingham, and B. Maione, "Modeling and control of discrete event dynamic systems: a simulator-based reinforcement-learning paradigm," to appear in *International Journal of Intelligent Control and Systems,* 1998.

[13] U. Dorndorf and E. Pesch, "Evolution based learning in a job-shop scheduling environment," *Computers and Operations Research*, vol. 22, n. 1, pp. 25-40, 1995.

[14] G.W. Evans and J. Haddock, "Modeling tools for flexible manufacturing systems," *Production Planning and Control*, vol. 3, n. 2, pp158-167, April-June 1992.

[15] E. Falkenauer and S. Bouffouix, "A genetic algorithm for job-shop," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 824-829, April 1991.

[16] M.P. Fanti, B. Maione, G. Piscitelli, and B. Turchiano, "System approach to design generic software for real-time control of flexible manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, vol. 26, n. 2, March 1996, pp. 190-202.

[17] D.B. Fogel, *Evolutionary Computation*, IEEE Press, 1995.

[18] M. Gen, Y. Tsujimura, and E. Kubota, "Solving job-shop scheduling problems by genetic algorithms," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1577-1582, 1994.

[19] D.E. Goldberg, *Genetic Algorithms in search, optimization and machine learning*, Addison-Wesley, 1989.

[20] B. Grabot, "A Decision support system for variable routings management in manufacturing systems," *Fuzzy Sets and Systems*, vol. 58, n. 1, 25 August 1993, pp. 87-104.

[21] G. Hao, J.S. Shang, and L.G. Vargas, "A Neural network approach for the real time control of a FMS," *Proceedings of the Hawaii International Conference on System Sciences*, vol. 3, pp. 641-648, 1994.

[22] I. Hatono, T. Suzuka, M. Umano, and H. Tamura, "Towards intelligent scheduling for flexible manufacturing: application of fuzzy inference to realizing high variety of objectives," *Proceedings of the Japan/USA Symposium on Flexible Automation*, vol. 1, pp. 433-440, ASME 1992.

[23] S. Haykin, *Neural Networks*, Macmillan/IEEE Press, 1994.

[24] K.S. Hindi, "On knowledge-based industrial scheduling systems," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 725-730, 1993.

[25] G.W. Hintz and H.J. Zimmermann, "A method to control flexible manufacturing systems," *European Journal of Operational Research*, vol. 41, pp. 321-334, 1989.

[26] J. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[27] C.W. Holsapple, V.S. Jacob, R. Pakath, and J.S. Zaveri, "A Genetics-based hybrid scheduler for generating static schedules in flexible manufacturing contexts," *IEEE Transactions on Systems Man and Cybernetics*, vol. 23, n. 4, July-August 1993, pp. 953-972.

[28] A. Homaifar and E. McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 3, n. 2, pp. 129-139, May 1995.

[29] J.S.R. Jang and C.T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, n. 3, pp. 378-406, March 1995.

[30] J.S.R. Jang, "Self-learning fuzzy controllers based on temporal back propagation," *IEEE Transactions on Neural Networks*, vol. 3, n. 5, pp. 714-723, September 1992.

[31] S.B. Joshi and J.S. Smith, *Computer Control of Flexible Manufacturing Systems*, Chapman & Hall, 1994.

[32] G.H. Kim and C.S.G. Lee, "An evolutionary approach to the job-shop scheduling problem," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 501-506, 1994.

[33] J. Kim, Y. Moon, and B.P. Zeigler, "Designing fuzzy net controllers using genetic algorithms," *IEEE Control Systems Magazine*, pp. 66-72, June 1995.

[34] B. Kosko, *Neural Networks and Fuzzy Systems - a dynamical system approach to machine intelligence*, Prentice Hall, 1992.

[35] P.R. Kumar and T.I. Seidman, "Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems," *IEEE Transactions on Automatic Control*, vol. 35, n. 3, March 1990, pp. 289-298.

[36] P.J.M. Laarhoven and W. Pedrycz, "A fuzzy extension of Saaty's priority theory," *Fuzzy Sets and Systems*, n. 11, pp. 229-241, 1983.

[37] Z.P. Lo and B. Bavarian, "Scheduling with neural networks for flexible manufacturing systems," *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1991, pp. 818-823.

[38] J.M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, n. 3, pp. 345-377, March 1995.

[39] L. Monostori and D. Barschdorff, "Artificial neural networks in intelligent manufacturing," *Robotics and Computer Integrated Manufacturing*, vol. 9, n. 6, December 1992, pp. 421-437.

[40] S.K. Narendra and S. Mukhopadhyay, "Intelligent control using neural networks," *IEEE Control Systems Magazine*, pp. 11-18, April 1992.

[41] C.D. Pedgen, *Introduction to SIMAN*, Systems Modeling Corporation, 1984.

[42] J.R. Perkins and P.R. Kumar, "Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Transactions on Automatic Control*, vol. 34, n. 2, February 1989, pp. 139-148.

[43] L. Rabelo, A. Jones, and J. Tsai, "Using hybrid systems for FMS scheduling," *2nd Industrial Engineering Research Conference Proceedings*, pp. 471-475.

[44] L. Rabelo, Y. Yih, A. Jones, and J.S. Tsai, "Intelligent scheduling for flexible manufacturing systems," *Proc. IEEE International conference on Robotics and Automation*, vol. 3, pp. 810-815, 1993.

[45] L.C. Rabelo and S. Alptekin, "A hybrid neural and symbolic processing approach to flexible manufacturing systems scheduling," *Hybrid Architectures for Intelligent Systems* (A. Kandel and G. Langholz, Eds.), Chapter 18, pp. 380-401, CRC Press, 1992.

[46] T.L. Saaty, "Exploring the interface between hierarchies, multiple objectives and fuzzy sets," *Fuzzy Sets and Systems*, n. 1, pp. 57-68, 1978.

[47] J.J.S. Sentieiro, "Planning and scheduling: non-classic heuristic approaches," *IFAC 12th triennial World congress*, pp. 479-483, 1993.

[48] M.J. Shaw, S. Park, and N. Raman, "Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge," *IIE Transactions*, vol. 24, n. 2, May 1992, pp. 156-168.

[49] S. Sittisathanchai, C.H. Dagli, and H.C. Lee, "A genetic scheduler for job-shops," *Proceedings of the Artificial Neural Networks in Engineering Conference*, pp. 351-356, 1994.

[50] A.E. Smith, T.D. Fry, P.R. Philipoom, and J.R. Sweigart, "A comparison of two intelligent scheduling systems for flexible manufacturing systems," *Expert Systems with Applications*, vol. 6, pp. 299-308, 1993.

[51] H. Tempelmeier and H. Kuhn, *Flexible Manufacturing Systems*, John Wiley and Sons, 1993.

[52] E.P.K. Tsang, "Scheduling techniques - a comparative study," *BT Technology Journal*, vol. 13, n. 1, pp. 16-28, January 1995.

[53] N. Viswanadham and Y. Narahari, *Performance modeling of Automated Manufacturing Systems*, Prentice Hall, 1992.

[54] L.X. Wang, *Adaptive Fuzzy Systems and Control - Design and stability analysis*, PTR Prentice and Hall, 1994.

[55] T. Watanabe, H. Tokumaru, and Y. Hashimoto, "Job-Shop scheduling using neural networks," *Control engineering practice*, vol. 1, n. 6, pp. 957-961, 1993.

[56] T. Watanabe, H. Tokumaru, Y. Nakjima, and Y. Hashimoto, "Job shop scheduling using fuzzy inference to take profit into account," *Proceedings of the Japan/USA Symposium on Flexible Automation*, vol. 1, pp. 423-427, ASME 1992.

[57] R.R. Yager, "Fuzzy decision making including unequal objectives," *Fuzzy Sets and Systems*, no. 1, pp. 87-95, 1978.

[58] R.R. Yager, "Multiple objective decision-making using fuzzy sets," *International Journal of Man-Machine Studies*, n. 9, pp. 375-382, 1977.

[59] D.D. Yao, *Stochastic modeling and analysis of manufacturing systems*, Springer-Verlag, 1994.

[60] Y. Yih, T.P. Liang and, H. Moskovitz, "Robot scheduling in a circuit board production line: a hybrid OR/ANN approach," *IIE Transactions*, vol. 25, n. 2, March 1993, pp. 26-33.

[61] H. Young-On, "FMS performance versus WIP under different scheduling rules," *Master's Thesis VPI & SU*, 1994.

[62] L.A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338-353, 1965.

[63] L.A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, n. 1, pp. 28-44, January 1973.

[64] B.P. Zeigler, "Devs representation of dynamical systems: event-based intelligent control," *Proceedings of IEEE*, vol. 77, n. 1, pp. 72-80, January 1989.

[65] B.P. Zeigler, "Hierarchical, modular discrete-event modelling in an object oriented environment," *Simulation*, vol. 49, n. 5, pp. 219-230, November 1987.

[66] B.P. Zeigler, "System-theoretic representation of simulation models," *IIE Transactions*, pp. 19-34, March 1984.

[67] B.P. Zeigler, *Multifaceted Modelling and Discrete Event Simulation*, Academic Press, 1984.

[68] B.P. Zeigler, *Theory of Modelling and Simulation*, Krieger, 1976.

Vita

PAOLO DADONE was born in Torino, Italy, the 14th of July of 1971. He received the *Laurea* degree with honors in electronic engineering in 1995 from the Politecnico di Bari, Italy, and the M.S. degree in electrical engineering in 1997 from the Virginia Polytechnic Institute and State University, USA, where he currently is a Ph.D. student. His research interests are in intelligent control algorithms development and implementation, theory and control of discrete event dynamical systems and manufacturing systems. Mr. Dadone is a member of several honor societies as well as IEEE technical societies. He was the recipient of the 1996 IEEE VMS graduate student best paper contest and of the 1996 Politecnico di Bari fellowship for studying abroad.