Abstract Number : 002-0039
Title of the paper : Modelling the human scheduler in a FMS: a knowledge-based system approach
Name of Conference : 2nd World Conference on POM, Cancun, Mexico

Author

Dr. James F. O’Kane
Programme Director Research & Enterprise
Newcastle Business School
University of Northumbria
Newcastle Upon Tyne NE1 8ST
Tel 0191-227-3839
Fax 0191-227-4684
james.okane@unn.ac.uk
ABSTRACT

This paper presents a knowledge-based system approach to modelling the human scheduler within a flexible manufacturing system (FMS). The approach and system that was subsequently developed was based on the principle of automated intelligent decision-making via knowledge elicitation from FMS status data, together with knowledge base augmentation to facilitate a learning ability based on past experiences. A simulation model of a real FMS was used to study reactive scheduling policies and from this work a Prolog-based expert system was developed that was capable of interrogating FMS data and offering intelligent decision-making on specific reactive scheduling scenarios. A dynamic database approach for the knowledge-based system is proposed and experiments with a linked simulation model/knowledge-based system are described. This research proposes an approach for the analysis of reactive scheduling in an FMS in which the knowledge-based system attempts to mimic the decision-making of the human scheduler when faced with different types of FMS reactive scheduling problems. A rudimentary “learning” system is described based on the dynamic database concept and these ideas are implemented into the environment to provide decision-making and control across a FMS schedule lifetime.

INTRODUCTION

One of the most interesting and arguably one of the most complex aspects of modern production systems is that of scheduling, which in its simplest form is concerned with the allocation of jobs to resources to satisfy certain constraints such as due date, resource utilisation or cost, Gupta (1989), French (1982). The decision-making requirements for scheduling production systems has been an intriguing research topic for many years and the goal of achieving optimum schedules which will ensure that production objectives are met is an on-going desire of most manufacturing companies. Unfortunately, because of its inherent complexity, scheduling is a dominant issue that can affect whether or not production targets are achieved. In addition to the problems posed by scheduling, the status of a production system may alter at any given instant in time. Therefore, dynamic changes to system status adds considerably to the already difficult scheduling problem and the task of coping with new and unexpected changes is the problem of reactive scheduling.
There are few methods available to engineers which can be used objectively to assist with the analysis of reactive scheduling within manufacturing systems. As manufacturing systems increase in complexity, then more sophisticated support tools will be required to assist human schedulers with their already onerous task.

Effective reactive scheduling requires a human scheduler who has extensive knowledge and experience of the production plant, and who also possesses an instinctive or intuitive approach in deciding the best course of action to pursue when presented with many different scenarios. This decision-making invariably has to be made rapidly and correctly to ensure that system constraints are satisfied. If the expertise and experience of the human scheduler could be captured into a computer program which could analyse the reactive scheduling situation, then automated decision-making could be performed, possibly leading to optimal scheduling solutions. If such a program was configured so that it possessed a human-like reasoning ability together with the capability of learning from experience about its problem domain, then an invaluable advisory aid for scheduling decision-making concerning reactive scheduling issues may be created.

The ability of an intelligent system to provide insights or solutions to the problem of reactive scheduling is an interesting proposal since it offers an opportunity to study and test the degree of reasoning required for a dynamically changing manufacturing environment. With this in mind, therefore, it was the overall aim of this research to conduct an investigation into the merits of using an knowledge-based system (KBS) to assist with decision-making regarding scheduling policies within a Flexible Manufacturing System (FMS).

Numerous authors have studied the simulation of FMS with intelligent systems. These studies have tended to concentrate on either FMS design considerations (Chan et al, 2000, Kovacs et al, 1991, Pflughoeft et al, 1996), or FMS scheduling paradigms (Kazerooni et al, 1997, Smith et al, 1993). In recent years, the merits of linking intelligent systems to simulations to explore in more depth the issue of reactive or dynamic rescheduling has been researched. Priore et al (2003) describe a system based on a neural network approach that is used to analysis dynamic scheduling for dispatching rule selection and Jahangirian at al (2000) developed a simple model based on the premise of machine learning for dynamic scheduling strategies. Rule-based intelligent systems linked to a simulation model to study dynamic scheduling offers an attractive research proposition as the system can be designed to investigate schedule performance via dispatching rule selection (Kunnathur et al, 2004). These types of investigation have led researchers to explore the possibility of utilising a KBS in which the intelligent element could augment the decision-making for dynamic scheduling scenarios and potentially add new knowledge in a evolutionary learning capacity.
By utilising a KBS in the form of an expert advisory system with a simulation model of an FMS, it is possible to attempt to emulate the decision-making process of the human scheduler on the shop-floor, since the knowledge and experience of the scheduler can be embodied within the KBS. A combined simulation/Artificial Intelligence (AI) system capable of interrogating FMS reactive scheduling problems and helping to offer advice and/or recommendations regarding the performance of the FMS represents a sophisticated and invaluable design and planning tool. In order to learn more about the reactive scheduling problem in an FMS it was proposed that an expert advisory tool could be configured so that it not only offered advice as to the optimum scheduling or re-scheduling decision needed, but it could also in some way 'learn' from experience as its knowledge of the reactive scheduling policies grew.

The proposed Expert System (ES) would therefore be capable of increasing its own knowledge base data, hence 'learning', or deleting obsolete information from the database. In a sense, the ES would possess a dynamic run-time database which would alter depending on the FMS data presented to it. Since simulation and AI are both essentially decision-making tools it seems natural that an integrated system consisting of these methodologies is an attractive environment for analysing and developing strategies of the reactive scheduling problem in FMS.

**Reactive Scheduling System Considerations**

Successful reactive scheduling systems require a number of key elements which fundamentally address many of the main problems of dynamic schedule revision. A hierarchical structure, framework, or scheduling architecture is sometimes proposed, in which these key elements are used to handle the predictive scheduling criteria, together with dynamic elements to handle the dynamic conditions. From the research described in this paper, the author has identified the salient or desirable features of a reactive scheduling tool for manufacturing and these features are summarised as the following:-

- **Quick response to dynamic changes**
- **Ability to operate in real-time with efficient temporal reasoning ability.**
- **Embody expert scheduling knowledge and heuristics.**
- **Efficient solution search mechanisms.**
- **Applicable to a wide range of manufacturing systems.**
- **Make use of on-line simulation.**
• Ability to accommodate a wide range of dynamic situations.

• Exhibit effective decision-making and control to ensure production criteria are met.

• Possess an ability to learn from experience to enhance its understanding of the scheduling problem.

• Advise/make a minimum amount of changes to existing schedule to ensure stability.

• Achieve local optimality

It has been the goal for many years to construct a reactive scheduling architecture which will address some or all of the above features. Some of the above features are extremely difficult to implement and most of the research work concerning reactive or dynamic scheduling has tended to focus on a subset of these features, so that some fruitful insight may be gained.

With the Non-deterministic Polynomial (NP) complexity problem and the problems associated with reasoning in real-time with large amounts of data, it is readily accepted that traditional approaches will never succeed in providing efficient and reliable reactive scheduling frameworks. Some of the desirable features outlined above, such as expert knowledge and heuristics, efficient solution search mechanisms and an in-built learning ability all lend themselves to the application of AI-based systems. In the last fifteen years, AI in the form of KBS, ES or intelligent environments have been applied to both the general predictive scheduling problem and the reactive scheduling problem. Using various scheduling architectures many researchers have analysed scheduling problems and have proved that AI-based systems offer promising results in helping to increase our understanding of scheduling across many different manufacturing applications, Watford and Davis (1991). Metaxiotis et al (2002), Meziane et al (2000).

Experiments with Prolog using 'Snapshot Data'

The goal of the first stage of the research was to develop a linked simulation/AI decision-making environment, in which the intelligent element in the form of a Prolog ES could analyse 'Snapshot Data' file information sent from the FMS model and offer advice to the user automatically. To do this, initial experiments were conducted with E.S's which could read a data file and "extract" the relevant bits of information needed to formulate effective and logical decisions concerning reactive scheduling scenarios.

The 'Snapshot Data' file, typically contains information relating to the following:-
Current production plan being run in the FMS model
Current simulation results
Current status of all the pallets

To enable an expert advisory tool to analyse a reactive scheduling condition, the Snapshot Data needs also to contain...

Current reactive scheduling condition prevailing
Percentage of the production plan completed
Performance measure for the FMS plan run

Therefore, the objectives of early experiments were to create simple stand-alone Prolog programs to test the extraction of data and then to extend this to eventually develop a Prolog E.S. which could be linked directly to the FMS simulation model to extract this data and provide automatic decision-making. Initially, some simple programs were created that just read the 'Snapshot Data' files, extracted some data, performed some simple rules and these were used to primarily test the integrity of the data link and the data extraction process.

These experiments culminated in the creation of a simulation-Prolog environment for which the FMS model was run with a specific production plan on one computer and the Prolog program run on another computer with a serial communications link between them. This meant that the user could then simulate different reactive scheduling conditions by halting the model during its run. Once the model was halted, the FMS system status was transferred to the waiting Prolog E.S. via the 'Snapshot Data' file. The E.S. then read this file, extracted the relevant information and using its knowledge base it was able to draw inferences and then suggest a best course of action to follow which was communicated to the user. A diagram showing the structure and data flows in the Simulation-AI environment is given in Figure 1.

**FMS Interruption Analysis**

From the point of view of the dynamic nature of the FMS, it is known that an analysis of the different possibilities for reactive scheduling is centred around the basic structure of the FMS i.e. the resources, pallets etc., and one can analyse an interruption condition in terms of the type, duration and degree of stoppage encountered. As a first step to understanding dynamic interruptions in an FMS, it is appropriate to describe the phenomena in terms of an overall
control horizon of time. For example, once an interruption occurs at say time T1 and ends at time T2, then there exists an interruption time window of duration t, where t = T2-T1. Within this interruption time period the human scheduler and/or FMS control system may attempt to rectify the reactive problem by implementing a policy which hopefully will ensure that the effect on the FMS performance criteria is minimised.

Consider the control horizon scenario as shown in Figure 2. Within the total time to complete a schedule any number of reactive interruptions may occur and for each of these interruptions a decision must be made to take corrective action which will hopefully minimise the problem. Each interruption will last for a finite duration and can occur quite randomly. Within the first interruption (e.g. a machine failure), the scheduler will evaluate the situation and then recommend a course of remedial action. The time taken to come to this decision can be quite random, but hopefully a best course of action decision can be made before the interruption ends. However, this is not always the case and as illustrated in the second interruption that occurs at time T4 (i.e. the worker absenteeism), the decision duration (d) exceeds the interruption period. Therefore, the goal is obviously to maintain a schedule throughout its lifetime so that system performance is not affected to an unacceptable degree. The only way to do this is to ensure that the best policy decision is made as soon as possible within the control horizon, Tayanithi et al (1992).

By considering the effect of multiple interruptions throughout the schedule lifetime and analysing each interruption control horizon, it is possible to judge both the effect the cumulative interruption has on the system and the effectiveness of the decision-making employed within each control horizon.

A knowledge-based system can provide this decision-making at each interruption and a measure of the effectiveness of the advice given by the system can be made. In essence, the knowledge-based system can store this interruption time history by logging it in a separate database. This concept is essentially a description of a Status Database that can provide history logging within the knowledge-based system which can then attempt to 'learn' new facts from each interruption scenario encounter within the total control horizon.

**Configuring a history logging feature to create a Status Database**

Within the lifetime of a schedule, it is not unusual for more than one reactive condition to occur. Each stoppage requires careful consideration and an overall assessment of the degree, type and duration of the interruption will
influence the performance of the FMS and the remedial action adopted. Indeed, a human scheduler's decision-making will be influenced by the frequency of interruptions and the effect on the FMS performance (in particular due-dates).

It is not unreasonable to suggest that from one interruption to the next, a human scheduler will remember the first interruption and the action taken and when the next interruption occurs, the previous experiences might have an influence on the next decision policy. Also, in the event of a schedule encountering multiple interruptions, the total interruption/decision history can be used by the human scheduler to initiate some action or recommendation as a new piece of information which might prove useful in the event of a similar interruption history occurring again.

In essence this is learning by experience. Therefore, if a knowledge-base is capable of storing all of the interruption data and subsequent decision policies, then it too may assimilate all of this information to help 'learn' something new and valid. This is basically the concept of the E.S. utilising a 'Status Database' and a 'History Logging' feature to facilitate these ideas, O'Kane (2003).

Therefore, the next stage of the research consisted of alterations to the existing E.S. to enable multiple interruptions to be simulated during the course of schedule runs and results were collected for various consultation sessions. This was done by simply extracting the relevant pieces of information from each of the "Snapshot Data' files sent during an interruption from the simulation model.

For each reactive interruption, details of the interruption number, resource utilisations, percentage of the schedule completed, time of the interruption and the advice/recommendation given by the E.S. were logged and stored in a separate 'Status Database' file for further analysis. This history log describing the interruption data/decision information etc. was then used to enable a means of emulating the human scheduler, so that the total number of interruptions, their severity and cumulative effects could be judged. Each 'Snapshot Data' file which contained much more information than the 'Status Database' was also stored so that a fuller picture of reactive events could be seen.

**Learning via a Dynamic Database**

The term Machine Learning has been used to describe various methods that can be used to develop algorithms, systems, reasoning mechanisms, etc. which exhibit forms of learning capabilities. The three most popular methods that have been tried by researchers in the past include neural computing, genetic algorithms and A.I.-based reasoning
mechanisms. Almost all of the methods currently known make use of knowledge acquisition and problem solving paradigms. Machine learning is an attempt to teach machines to solve problems, or to support problem-solving by applying historical information.

The basic processes responsible for human learning are not well understood and in the last 40 years the science of A.I. has been applied in an attempt to create a program which will not only 'learn' but also delve deeper into the basic principles underlying human intelligence. In this research programme the author has applied A.I. to the reactive FMS scheduling problem in an attempt to emulate human decision-making based on previous experiences and therefore the intelligent elements of the environment can be classified as an automated advisory and learning tool. In this sense the intelligent system is not configured as a problem solver, but it does exhibit intelligent knowledge acquisition and reasoning abilities and therefore machine learning is possible.

Definitions of the term learning tend to focus on the concept that learning is essentially the process of making adaptive changes until one is able to perform a task or tasks more effectively the next time. In terms of a computer exhibiting program learning behaviour, one can view this process so that computer-based learning is said to occur if the program’s performance improves with experience i.e. "A computer is said to learn a task T, given experience E with respect to a performance measure P if its performance at T improves with E, as measured by P." For example, some of the most celebrated examples of machine learning/A.I. have been the development of intelligent game playing machines that can perform better than human players. In these systems the task T is equivalent to playing the game, the experience E is analogous to the system playing against itself and the performance measure P can relate to the number of games won against human opponents.

The definition of learning as being an adaptive process is one which can be applied to the Prolog-based E.S. developed for the intelligent simulation environment. The 'learning' process that has been developed is based on the premise of storing data for future reference, (i.e. history log), and analysing total schedule interruption history to generate new knowledge that may be worthwhile to the problem. This 'learning' process was developed by configuring the intelligent system so that a Dynamic Database was created in which new knowledge was asserted into the database and old or obsolete data was deleted (retracted) at run-time.

A dynamic database is one which adaptively changes its rules/facts/data over time. It is well known that an FMS is a dynamic system in which the system status may change at any moment in time. The decision-making policies required to analyse dynamic changes to an FMS need to be formulated, executed and monitored swiftly and it is for this reason
that a dynamic database within a knowledge-based system can be utilised to help with reactive scheduling decision-making.

Dynamic databases can take many different forms depending on the problem domain characteristics and the type of knowledge representation mechanism employed etc. In the reactive scheduling problem for FMS, the dynamic database needs to be configured so that the phases of storing, retrieving, analysing, adding and deleting information can be fused together to create a system capable of emulating to a certain extent the decision-making and 'learning' process of the human scheduler. In the Prolog-Simulation environment, the distinction between static data that may be held in a static database and the items in a dynamic database is quite clear. The static elements consist of things like the number of resources, the number of pallets, the tool compliment per machine etc., whereas the dynamic database is typically represented by all of the information in the 'Snapshot Data' file. To complete the picture of the dynamic database, the interruption history of a schedule run i.e. the 'History Log' file was used to represent an accurate account over time of the salient aspects of the knowledge acquisition and expert advice processes.

The basic function of the dynamic database concept is to store important interim results concerning the FMS status and data concerning control decisions. The database can be viewed as a continually changing (increasing and/or decreasing in size) block of data. The term blackboard or scratchpad database is an ideal description of such a database. By definition, a knowledge-based system that comprises a dynamic database does not possess any learning capabilities, but a logical extension of such a system could be the addition of a mechanism capable of rudimentary learning processes based on the concept of adding new information in the form of rules and/or data which may be used by the system again at a later moment in time.

**Learning via Assertion and Retraction of data**

An intelligent simulation environment has been developed in which a dynamic database is used to assist with the creation of a learning tool for reactive scheduling in an FMS simulation model. The learning process was accomplished by augmenting the structure of the intelligent element so that it could apply knowledge acquisition, reasoning and eventually learning schemes to the problem domain. Specifically the learning capability was developed by using the knowledge-based system to analyse interruption history and 'Snapshot Data' information until a schedule was completed (or indeed aborted at some intermediate time) and then a final analysis was performed which resulted in new data/knowledge asserted (added) or old/obsolete data retracted (deleted) from the dynamic database structure.
In this sense, the knowledge-based system is learning new rules and facts from one schedule run to another and this new knowledge may or may not be exploited depending on the unique circumstances of the FMS. This process is, in the authors' opinion, analogous to the learning and decision-making process of the human scheduler in the sense that decisions regarding best plan of action at an instant in time will be formulated from current FMS status knowledge, accumulated general FMS knowledge and past experience from previous schedule executions. The novel features of this approach are "the combinations of automated intelligent decision-making via knowledge elicitation from status information, together with dynamic database augmentation to facilitate a learning process based on previous experiences". Examples of some of the assertion rules and how they were applied in the knowledge base are now described.

**Examples of asserted rules for 'learning'**

Within the framework of the FMS, it was evident that a number of assertion rules could be applied in specific dynamic situations during schedule lifetimes. A human scheduler may in certain circumstances decide to abandon a schedule run if it was felt that continuation of the schedule run, (in which many severe interruptions had occurred), would ultimately lead to excessive and unacceptable increases in makespan. This in turn would lead to missed due-dates and therefore, a rule of thumb that could be applied by the scheduler in practice would be a decision to abandon the schedule run if for instance the number of interruptions were 4 or more, (depending on the type and duration of each interruption).

The percentage of a schedule completed at each interruption could also be used to influence the abandonment/continuation decision for a schedule. Therefore, one of the main asserted rule elements that was implemented in the knowledge base consisted of an analysis of the number of interruptions and the percentage of the schedule completed at each interruption point. In essence, all of the asserted ('learning') rules that were implemented into the knowledge base structure operated in a similar manner as described previously, i.e. from one completed schedule run to the next a pattern may emerge which would cause the knowledge-based system to automatically assert new knowledge into the dynamic database.

Then, in each subsequent schedule run, these new facts and or rules are available to be activated if similar historic reactive status conditions prevailed. This 'learning' process by assertion of new knowledge that may or may not be invoked depending on circumstances is a somewhat simplistic view of learning per se. However, the environment that
has been developed has a rudimentary intelligence capability to analyse many different reactive scheduling scenarios and to use its knowledge of past experiences to help it formulate new best courses of action at anytime in the future.

It is argued that the developed 'learning' process is analogous to the basic learning cycle that human schedulers would employ when faced with new situations on the factory floor, in the sense that they would bring to bear past historical knowledge from their sub-conscious memory. In fact, the intelligent environment is capable of learning new snippets of knowledge, which it can utilise at a later time. Table 1 shows examples of some of the assertion rules implemented into the knowledge. To test the application and effectiveness of these ideas and of the general learning process a series of experiments were performed with the new environment and these issues are discussed next.

Assessing the 'learning' capabilities of the system

The intelligent simulation environment was tested to assess the ability of the system to 'learn' new facts/rules depending on different reactive situations. Trial runs using different production plans (schedules) were done in which different reactive scenarios were simulated throughout schedule lifetimes to observe the intelligent system behaviour from one schedule run to the next. A number of asserted rules for 'learning' were formulated and tested, and these are shown in Table 1.

By simulating different reactive conditions, each of the rules in the table was triggered at least once during a schedule run. Tests were done so that after the first schedule run, the knowledge base was automatically altered in size by the addition of a new data (knowledge). The system was then tested by simulating similar reactive conditions again (for the same production plan) and the intelligent decision-making behaviour was then observed.

It was found that the 'learning' process of the assertion of new data from one schedule run to the next worked well. The main problem encountered with the approach was that experimentation of this kind caused the knowledge base to change continually and it was somewhat difficult to keep track of the amount and scope of the decision-making capabilities of the latest version of the knowledge based system being used.

The author decided to retain a master (unaltered) knowledge base and to do experimentation on a test database (i.e. one in which no new knowledge had been added). The test database was therefore used for different experiments so that its knowledge content continually altered depending on the type of reactive interruption simulated, its duration and other
factors. As an example, a test database was used and an experiment was performed in which, firstly, 5 interruptions were simulated on the first schedule run. This had the effect that a new rule was added to the test database which stated that if in the future more than 4 interruptions occurred in another schedule run then the system would immediately abort the schedule. In the second schedule run, 3 interruptions were simulated and two of these were designated to be AGV failure. In the next schedule run (i.e. the third), the FMS schedule has 4 simulated reactive interruptions. The first and last of these interruptions were simulated as AGV failures. Since the knowledge base had until this point in time contained new rules i.e. it had 'learnt' that if 4 or more interruptions occurred again it would abort the schedule run, it then proceeded to implement this course of action. If in the last schedule run, the 4 simulated interruptions had been (in order of occurrence) m/c #1 failure, AGV failure, m/c #2 failure and AGV failure then the intelligent system would have had two possible courses of action to follow, since it had 'learnt' two new rules.

Depending on the amount of the schedule completed at the time of the final interruption, the intelligent system might have invoked rule #5 (Table 1), instead of abandoning the schedule because 4 interruptions had occurred. Therefore, the intelligent system was configured so that it operated a tie-break decision-making rule in circumstances such as this since it had already 'learnt' two new pieces of knowledge which applied to the current status.

This particular example serves to illustrate the problem of maintaining a dynamic database, no matter what type of problem domain is under consideration. The 'History Logging' and 'Snapshot Data' capture features provided the knowledge base and the user with data to check for logical consistency that appropriate rules were fired and that appropriate new snippets of knowledge were added to the knowledge base accordingly. However, it was found that as the knowledge base grew in size via the 'learning' process, the intelligent system needed to be equipped with a facility to enable the user to assess the current level of reasoning ability of the system. Also, it was felt that a more thorough explanation facility was needed to instil confidence in the recommended decision-making offered by the system.

Conclusions

In building a system in which one is attempting automated decision-making, the scope for accurate reasoning about the problem domain is a function of how well understood the problem domain is itself. FMS scheduling is a very complex problem and the reactive scheduling problem is essentially a matter of digesting current status knowledge of the reactive condition, formulating a best plan of action and monitoring the recommendation to see if it meets specific manufacturing objectives.
An intelligent system has been developed in this research programme which acts in an advisory and control capacity when reactive conditions occur across a schedule's lifetime. The intelligent system was configured to receive information from a simulation of a real FMS, (driven by real schedule data), and to then use its existing knowledge-base data to formulate a recommendation to the user. It was configured to exercise control and influence over the FMS model and also it could 'learn' new pieces of information by asserting data in the form of rules into the knowledge base from one schedule run to the next.

In terms of effectiveness of the decision-making capability, the intelligent system developed was limited by the quality and integrity of the specific FMS problem domain knowledge. It is well known that the main limitations of any knowledge-base system are the amount of knowledge it possesses, its degree of reasoning ability and its explanation abilities.

The author has attempted via the 'History Logging' and 'Snapshot Data' features to extrapolate as much information as possible so that the intelligent system had sufficient data from which to formulate logical advisory decision-making. The intelligent system has been configured with a 'learning' facility which is centred around the premise that assertion of data is analogous to human learning. Therefore, it is argued that this approach is novel and taken within the context of the investigated FMS domain it enabled new possibilities of deductive reasoning to be opened up for investigation and to this end the research presented here has been both fruitful and productive.

References


**Figure 1 : Simulation -A.I. Environment**

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>Assertion rule explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>if number of interruptions ≥4 then Abort schedule run immediately</td>
</tr>
<tr>
<td>2</td>
<td>if number of interruptions &gt;1 and the percentage of the schedule completed is ≤10% then Abort schedule immediately</td>
</tr>
<tr>
<td>3</td>
<td>if number of interruptions ≤3 and two of these are Load #1 failure then check Load #1, re-route pallets to Load #2 and monitor effect on m/c utilisations and projected makespan level.</td>
</tr>
<tr>
<td>4</td>
<td>if number of interruptions ≤3 and two of these are Load #2 failure then check Load #2, re-route pallets to Load #1 and monitor effect on m/c utilisations and projected makespan level.</td>
</tr>
</tbody>
</table>
if number of interruptions = 3 and are all due to AGV failure and if percentage of schedule completed ≤ 10% then abandon schedule run, else if percentage completed > 10% then repair AGV and monitor m/c utilisation level and projected makespan level

TABLE 1 Examples of asserted rules for 'learning' process

<table>
<thead>
<tr>
<th>Table 2: Control horizon for FMS decision-making</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KEY</strong></td>
</tr>
<tr>
<td>T1 = Start of schedule run in FMS</td>
</tr>
<tr>
<td>T2 = Start of interruption #1</td>
</tr>
<tr>
<td>T3 = End of interruption #1</td>
</tr>
<tr>
<td>T4 = Start of interruption #2</td>
</tr>
<tr>
<td>T5 = End of interruption #2</td>
</tr>
<tr>
<td>T6 = Start of interruption #3</td>
</tr>
<tr>
<td>T7 = End of interruption #3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interruption #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e.g. m/c 1 down)</td>
</tr>
<tr>
<td>( t_s )</td>
</tr>
<tr>
<td>( T_1 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interruption #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e.g. worker absent)</td>
</tr>
<tr>
<td>( t_s )</td>
</tr>
<tr>
<td>( T_2 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interruption #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e.g. rush order)</td>
</tr>
<tr>
<td>( t_s )</td>
</tr>
<tr>
<td>( T_3 )</td>
</tr>
</tbody>
</table>

\( t_s \) = interruption duration #1
\( t_s \) = interruption duration #2
\( t_s \) = interruption duration #3
\( d_1 \) = decision duration #1
\( d_2 \) = decision duration #2
\( d_3 \) = decision duration #3