International Journal of Production Economics, 30–31 (1993) 453–464
Elsevier

An expert system for a local planning environment

G.J. Meester

Laboratory of Production Management and Logistics, Department of Mechanical Engineering, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract

In this paper, we discuss the design of an Expert System (ES) that supports decision making in a Local Planning System (LPS) environment. The LPS provides the link between a high level factory planning system (rough cut capacity planning and material coordination) and the actual execution of jobs on the shopfloor, by specifying a detailed workplan. It is divided in two hierarchical layers: planning and scheduling. At each level, a set of different algorithms and heuristics is available to anticipate different situations.

The Expert System (which is a part of the LPS) supports decision making at each of the two LPS layers by evaluating the planning and scheduling conditions and, based on this evaluation, advising the use of a specific algorithm, and evaluating the results of using the proposed algorithm.

The Expert System is rule-based while knowledge (structure) and data are separated (which makes the ES more flexible in terms of fine-tuning and adding new knowledge). Knowledge is furthermore separated in algorithmic knowledge and company specific knowledge. In this paper we discuss backgrounds of the expert system in more detail. An evaluation of the Expert system is also presented

1. Introduction

Production control has become increasingly complex during the last few decades. On the one hand market requirements tend towards more diversity, a higher quality and reliable due dates. On the other hand the rapid advancements in information technology have increased the possibilities on the shopfloor, thereby leading to more complex planning and control problems.

In this paper we concentrate on discrete parts manufacturing environments. It is commonly accepted that the complexity of the overall planning problems should lead to a hierarchical planning and control framework, where demand management and long term capacity planning are high level decisions, materials coordination is at the medium level, while detailed shopfloor planning and control can be found at the lower levels. For the high and medium level decisions we assume that a Global Planning System (GPS) has been implemented. In practice we often may find an MRP II system (Manufacturing Resource Planning system) at this place.

However, it is well known that Global Planning Systems such as e.g. an MRP II system are not adequate in translating a planned set of orders into detailed work instructions at the shopfloor level. To this end, a Local Planning System (LPS) has recently been developed at the University of Twente (cf. Refs [1–3]). Hence, this LPS acts as shopfloor planning and control system and as such bridges the gap between the Global Planning System and the actual execution of jobs on the shopfloor.

To highlight the functioning of the LPS in some more detail, we assume (reflecting common industrial practice) that the factory shopfloor can be decomposed into relatively

Correspondence to: G.J. Meester, Laboratory of Production Management and Logistics, Department of Mechanical Engineering, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.

autonomous groups of machines (this may be a functional arrangement but also a cell in a cellular manufacturing environment). Now, the Global Planning System organizes the materials flow between these machine groups (to be called Factory Systems (FS) in the sequence). Subsequently, the Local Planning System attempts to match this flow with the critical resources within each FS, such as tools, fixtures, machine capacities and operators. Hence, the FS is the local planning environment for the LPS.

In performing these tasks, a large number of decisions have to be made while facing several, sometimes conflicting, objectives. To support this decision making an expert system may be helpful, in particular to evaluate specific working conditions, to suggest planning and scheduling algorithms and to judge the final results. The objective of this paper is to describe what such an expert system should look like, and to demonstrate that it can support an operator to substantially improve the performance of the LPS. In other words: it is demonstrated how we can formalize knowledge about planning and scheduling and how we can use this knowledge operationally.

Other research on expert systems in the type of production situations discussed here can be found in Refs. [4–7]. For general discussions on expert systems the reader is referred to Refs. [8–12].

In the remainder of this article we will focus on the expert system. First the expert system concept in general will be outlined and some properties of the concept of knowledge are discussed. Next the architecture, its tasks and functions are described for the particular LPS situation. The LPS-Expert system combination has been evaluated in four different situations. The evaluation of the expert system is presented in Section 5. Conclusions and suggestions for further research are given in Section 6.

2. The expert system: principles, tasks and knowledge

The basic idea behind an expert system for an industrial company is that the company should save its investments in knowledge, even if skilled employees should leave the company. Expert systems can be used when we have to deal with problems in such fields as [13] :

(1) interpretation and understanding of a complex data structure,
(2) ordering/scheduling,
(3) the evaluation of situations,
(4) diagnostics,
(5) the tracing of failures.

There are different types of expert systems which can be used to solve these problems [9] such as:

(1) rule based expert systems,
(2) expert systems based on first order predicate logic,
(3) expert systems based on structured objects (e.g. frames and trees).

Looking at the characteristics of the tasks that an expert system should perform in the particular environment, studied in this paper, we have selected a rule based system. Rules can represent a unit piece of knowledge. Advantages of a rule based ES are:

- it is easy to work with in test situations,
- it is easy to implement in a control concept (as will be explained in the next sections).

A more elaborate discussion of arguments leading to the choice of a rule based ES is given in Ref. [14]. The structure of such an expert system is described in detail in Ref. [15]. Knowledge is represented in what we call "knowledge rules" (IF–THEN–ELSE). These rules are mutually independent.

One of the properties of an expert system in general is that the knowledge (structure) and the data are strictly separated. The advantage of an expert system in contrast with a traditional computer program is that if something changes in the problem environment, it will not cause much problems in the ES. One should simply add a new rule or new data. In a traditional program, the entire source code may have to be changed.

The structure of an expert system will be briefly explained now. An expert system is constructed from three basic elements: 1) the database, 2) the knowledge base, 3) the inference engine. In our research framework we

have added some slightly more advanced concepts like a statistics module (and in the future a simulation module). See Fig. 1.

The inference engine matches patterns from the "rule base" or knowledge base, with recommended data from the database. The combination represents the knowledge. In Fig. 1 we have also included the module "statistics". By changing data in the database, it implicitly changes the knowledge. To clarify what is meant by knowledge, how it can be used optimally and how it should be represented in the rule based expert system, some techniques are illustrated by using an example.

Example: Suppose the "expert" knows that computer time will increase exponentially fast when the algorithm "complete enumeration" (this algorithm calculates every possible schedule and gives always the optimal solution) will be used by a set of jobs larger than 20.

We can implement this knowledge as follows:

IF ord# $\leqslant$ 20 {condition}
THEN advise algorithm
= complete enumeration {action},
certainty factor = 100 {certainty}.

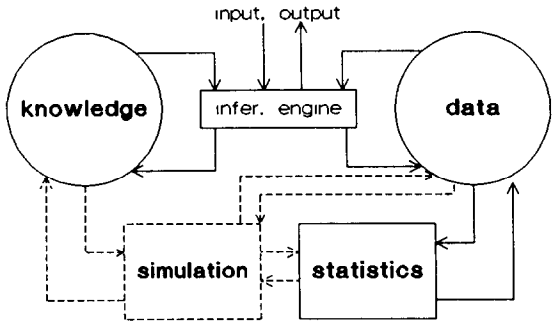

Fig. 1. Expert system, structure.

One can distinguish three main parts in the knowledge rules, which are explained in short:

(1) The condition part: The condition part can be true or false. If it is true, action is taken, if it is false, no action will be taken. The condition part may find the values of the corresponding parameters (constants, variables, etc.) in the database or from other rules in the rule base (tree search). For instance in Fig. 2 action will be taken if $a > b$. The value of the parameter a is found in the database. However, b is not and therefore the inference engine now searches in the knowledge base and finds b in the second rule. Since now d is needed, the inference engine will search for it. In this way a tree of rules can be checked.

(2) The action part: This part of the rule describes the action that has to be taken. An action can be:
- changing data in the database (for control applications),
- placing at inferences disposal of data for tree search activities,
- proposing an algorithm.

(3) The certainty part: This part specifies the value of the so-called *certainty factor*. Basically the value of the certainty factor indicates the probability that a recommended action will lead to satisfactory results. We will come back to the construction and role of the certainty factor in more detail below.

Under complex environmental conditions it is not always obvious what may be expected from the performance of a given rule or algorithm (e.g. priority dispatching rules). Statistical knowledge about the performance of such a rule in the past, under similar conditions, may therefore be helpful to predict its behavior

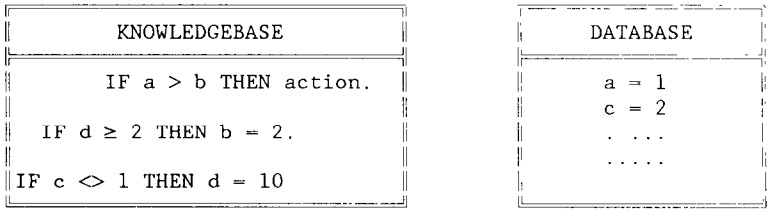

Fig. 2. Example knowledge representation.

in the future. In this way a sort of learning capability is included in the expert system. This idea has motivated the introduction of the *certainty factor*. Basically its value specifies the probability that an algorithm performs satisfactorily. Since any knowledge about recent performances should be included, a reasonable up-date procedure for the value of the certainty factor is as follows:

$$CF_{new} = \frac{(CF_{old} * N + \delta \times 100)}{N + 1}. \quad (1)$$

Here CF_{old} and CF_{new} represent the initial and the updated values of the certainty factor, respectively. Furthermore, we set $\delta = 1$ if the solution determined by the relevant algorithm in the last cycle is satisfactory, and $\delta = 0$ otherwise. N is the number of previously recorded tests upon which the certainty factor has been based so far. For practical reasons the value of N should neither be too low nor too high. When N is very low the certainty factor will be strongly influenced by each additional test. When N is high the influence becomes negligible. When the rule is tested often, N is high and the value will become very stable. As circumstances may change, considering all former tests has its disadvantages. Therefore, both a lower bound and an upper bound are given to (in some cases even a constant value will do).

Having thus motivated the use of the certainty factor (basically to incorporate learning effects) we next explain its use. Given a particular planning or scheduling problem faced by the LPS, we first check the environmental conditions and objectives to find what algorithms or rules might be applicable. Next, we simply choose the rule with the highest CF-value, where ties are broken arbitrarily.

Another way to incorporate knowledge about the performance of certain rules or algorithms is by adjusting the conditions under which it may be applied. These conditions, such as the condition in our example, may be changed as a result of recent performance evaluations. E.g. in our example let us replace the factor 20 by a factor WF. The condition hence becomes ord $\leqslant$ WF. In this case, as a result of statistical analysis one may adapt the value of the factor WF and write this new value in the database. Complex statistical procedures may be required to find good values for such factors.

An expert system will only perform well if the knowledge added to the data and knowledge base is well defined. This means: the process of knowledge acquisition must be carried out thoroughly and accurate. A formal description of the knowledge acquisition process is given in Ref. [16]. For our particular situation we list a few important steps. Note that knowledge acquisition basically concerns the representation of knowledge rules (Fig. 3).

(1) First the relevant relations in combination with the relevant indicators have to be traced. To get these relations one may use a knowledge relation chart (see Fig. 3). In such a chart the relations between the LPS algorithms and the potential environment indicators are given.

After an interview with an expert or analyses done on the LPS algorithms, the relations between the indicators and the LPS-algorithms found can be added into the chart.

(2) After finding the important relations, the effective area of the indicators must be found (i.e. in our example the effective area was the interval [0, 20]. Values of 20 or more are not relevant). Possibilities here are to test the LPS algorithms for different input values. The condition part can be defined in combination with the action part (LPS-algorithm).

		indicators			
		k1	k2	k3	k4
algorithms	p1		*	*	
	p2	*			
	p3		*		*
	p4	*		*	

Fig. 3. Knowledge acquisition chart.

(3) Finally a certainty factor must be set. When there is no information about the performance of the LPS-algorithm an arbitrary value must be added. In the long run the certainty factor converges to a stable value (in a stable environment).

With this information the condition part, action part and certainty part can be projected on the knowledge representation structure (IF–THEN–ELSE structure) which yields the knowledge rules that can be implemented.

3. An expert system for the LPS

A Local Planning System contains several modules, structured in a hierarchical framework. The two most important ones are the "Planning module" and the "Scheduling module".

The planning module performs the following functions:

- It receives a set of orders (see Fig. 4) from the GPS that have to be produced in a certain time period of one to three weeks, say.
- It checks capacity and handles utilization planning on a high aggregation level. This means in particular:
  - moving orders in time,
  - selecting alternative process plans, available in "technical data" (cf. Fig. 4).
- It selects a subset of orders for the next day from the above described order set. This subset forms the input for the scheduling module.

The scheduling module performs the following functions:

- It creates the schedule for the next day by assigning all operations from the selected subset of orders to machines and by sequencing all operations assigned to that specific machine.
- It creates task lists for the resources (e.g. tools, machines, fixtures, operators, etc.).

Both levels contain a range of different algorithms and heuristics. For most particular situations there is an algorithm or heuristic available that performs best under these circumstances. This means that an LPS operator (planner) has to interpret the situation (given by the GPS and the actual situation on the shopfloor) and has to select an algorithm that yields a satisfactory planning/scheduling solution.

The algorithms and heuristics available on "planning" level are:

(1) For utilization planning:
  - moving orders in time:
    - sorting algorithms to select jobs on specific properties,
    - clustering algorithms on technical constraints (for tools and fixtures),
    - clustering algorithms with precedence relations, release and due dates.
  - process plan selection:
    - complete enumeration (workload balancing),
    - mixed integer programming (workload balancing),
    - linear programming, continuous rounded (workload balancing),
    - two-opt heuristic (pair-wise interchange) (workload balancing),
    - sorting on resource load (workload balancing),

(2) For workloading:
  - linear programming.

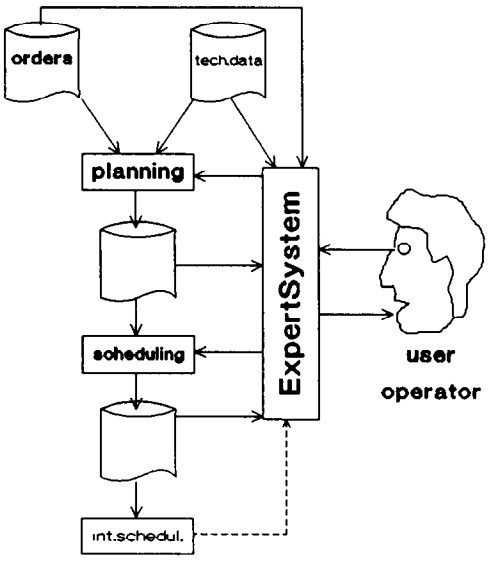

Fig. 4. LPS.

The module scheduling contains algorithms and heuristics to generate schedules which focus on the utilization of resources, short term due-date performance, throughput times, and schedule robustness.

(1) Before using actual dispatching and scheduling algorithms, it is possible to use a preparation tool to let the dispatching and scheduling algorithms perform better. For example:
   - add all operator relations,
   - add relations for a specific fixture type,
   - add all machine-tool relations for a specific machine.

(2) The dispatching may be used to accommodate e.g.:
   - order due dates,
   - operations due dates,
   - priorities.

The actual scheduling procedure is the shifting bottleneck procedure developed by Adams et. al. [17].

(3) After having generated the schedule, there is a possibility to improve the schedule somewhat further by one of the secondary scheduling procedures:
   - make feasible,
   - adapt for gaps.

The heuristics and algorithms are described more extensively in Ref. [3].

The expert system should support the decision making in selecting algorithms on different levels. It may either support the operator or operate as an autonomous program without intervention of the operator. Let us return to the specific expert system structure developed for the LPS environment. The LPS basically consists of two layers: a planning level and a scheduling level. An important reason to split the system in such a hierarchical structure is to be able to deal with the complexity of the problems encountered. Also the ES developed here is used in an experimental environment, with a data and knowledge base on each separate level it seems easier to test. Having just one level where all decisions should be made it may become harder to evaluate the effects of any new rule or alteration.

As mentioned earlier, the expert system has to perform the following tasks. On both, the planning and the scheduling level it has to:
- evaluate the situation in the planning and scheduling environment,
- support the operator by choosing the right algorithm or heuristic in a specific situation (generate),
- evaluate the result of one algorithm or heuristic (are we satisfied or not?) (evaluate).

The generate-evaluation concept (depicted in Fig. 5) is implemented on both LPS levels. On each level the expert system has to advise. All relevant data according to this advice and to the result of this advice is temporarily saved (as best). The evaluation part checks this data.

If "evaluation" is satisfied, the expert system permits to move to a lower level of the LPS.

If "evaluation" is not satisfied, the expert system has to generate a new solution by suggesting a new algorithm.

Remark 1. When "evaluation" is not satisfied, it will explain to the advice-part what's wrong. So the advice part is able to re-advise "intelligently". We call this control loop the *short control loop.*

Remark 2. When "evaluation" continues to return "not satisfied" it may advise to go to

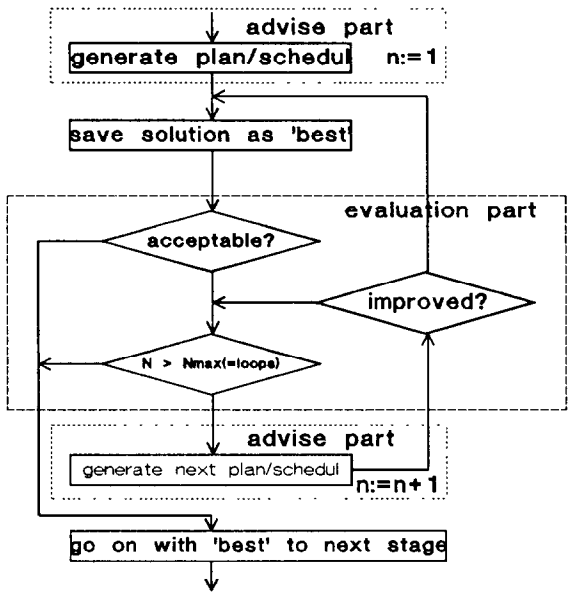

Fig. 5. Control mechanism.

a lower LPS-level after a certain number of loops. This is to protect against infinite loops. (Temporary solution "best" is given to the lower level).

Remark 3. The evaluation system on the LPS-scheduling level has the possibility to send information to the LPS-planning level. Since we deal with a hierarchic system, it may possibly happen that the planning module presents an order subset to the scheduling module which does not allow to meet a specific criterion under any schedule.

Both advice and evaluation part have to be fitted into the expert system (see Section 4).

4. Implementation

The expert system is implemented on a HP 9000 workstation. It operates together with the LPS under an X-window representation manager on a UNIX operating system. Both are written in Standard Pascal, see Ref. [15].

The choice for Pascal is motivated by the fact that in this type of research one has to deal with many new concepts and changes. In Pascal it is easy to develop new ideas. We also have investigated different software packages, like the IBM Expert Shell [9] and the Delfi II Expert Environment. Both of the packages are professional expert environments but the link between these Expert Systems and the LPS is difficult to make. Also extensions on the Expert System framework the hardly to realize.

Once having discussed the control mechanism and the knowledge structure we are now ready to present the complete expert system structure. The structure is depicted in Fig. 6. Observe the knowledge rules embedded in this concept.

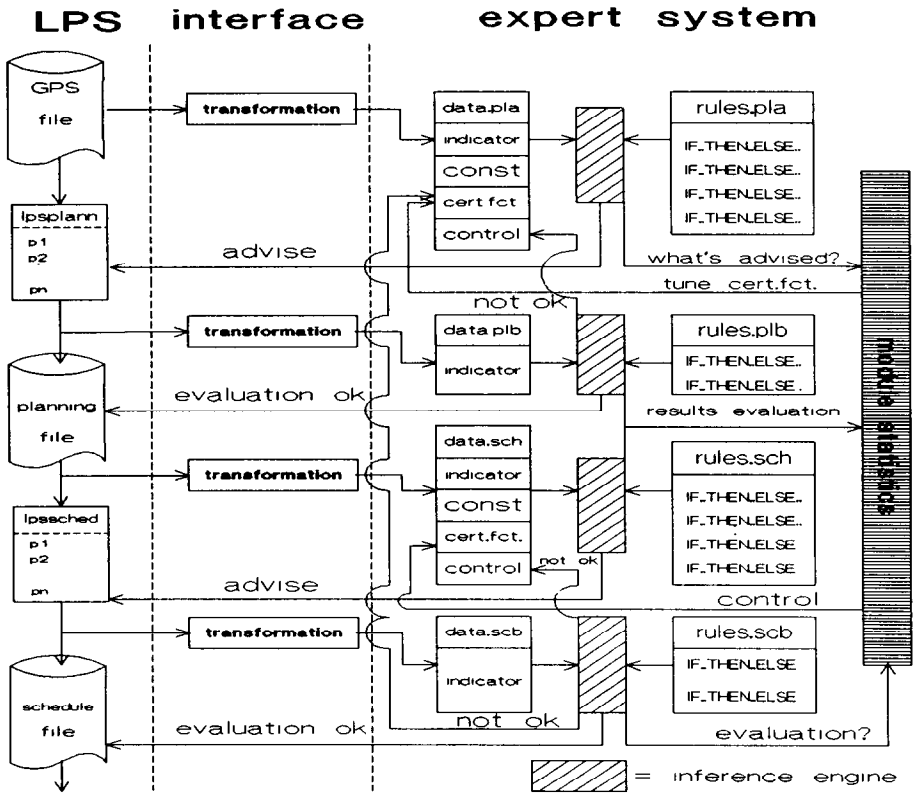

Fig. 6. Control chart.

Not mentioned till now but important for the ES-concept is the interface unit (see Fig. 6). The interfaces are mechanisms which are able to compress files of information in a special way. In particular, a file given by the GPS can be transformed into a vector of indicators. In that way it is easier for expert system to interpret the GPS situation. For a more elaborate treatment of the interfaces we refer to Ref. [14].

To further highlight Fig. 6 one, step of the advisory and evaluation process of the expert system will be described.

Step 1. The GPS file is translated into an indicator vector. The mass of available data will be compressed to a set of indicators which represent the important characteristics of the file. The vector is stored in the database.

Step 2. After a request for advice is received, the inference engine matches the data and the rules and presents the advice. It also reports to the module statistics.

Step 3. The result of the advice is implemented.

Step 4. The planning file (containing the results of following the advice) is transformed in a way similar to the one described in step 1.

Step 5. There are two possibilities. The result of the evaluation is:
"ok": → We continue with scheduling; go to step 1,
"not ok": → a new advice is given. The difference is that the evaluation changes the data in the database on the advice level (see Fig. 6). Go to step 1.

In our expert system we have implemented the knowledge evaluation by means of the certainty factor. The module "statistics" takes care of the handling of the factor N. For every rule which generates an advice a value of N is maintained.

5. Evaluation of the expert system

To evaluate the power and usefulness of the LPS–ES (Local Planning System–Expert System) combination, it is implemented in four different environments.

(1) The combination tested in an operation oriented FS situation. This case concerns an FS which contains 5 rather similar milling/drilling machines (the only difference between the machines is the number of driven rotation and translation axes).
(2) The second case concerns a part oriented FS consisting of 5 machines of three different types. We may think about families of products visiting this machining centre (part flow). For details see below.
(3) The third case is taken from Ref. [18]. Actually this FMS situation concerns a pilot plant in a Dutch research institute (IPL-TNO, Apeldoorn). It consists of a turning centre and a FS. The machines and a tool preparation cell are integrated in a CIM environment (the TNO supervisory control system).
(4) The final case is a comparison made between OPIS/ISIS, a scheduling system developed by Fox et al. [4], and the LPS–ES. The test situation used in this case is described in Ref. [19]. Special in this case is that the FS consists of a number of cells, where each cell contains a number of similar machines.

How the LPS–ES combination works in a particular environment (recall that the ES implementation is situation-specific) is explained in more detail. Although the LPS–ES combination is extensively tested for each of the four cases described above, at this point, for the sake of brevity only the knowledge acquisition and the evaluation of the second case are described.

5.1. *Outline of the second case*

5.1.1. *The factory system*

The FS consists of 5 machines: two of type A, two of type B and one of type C. Each machine has a tool magazine with a capacity of forty tools.

Each job follows a predetermined route along the machine types: A → B → C. The maximum number of different operations within one job is three on machine A and B

and two on machine C. The total number of operations within one job is at most five. Two shifts (16 hours) per day are assumed.

5.1.2. Jobs

80 jobs have to be produced in one week. Every job has to be produced in 1 to 5 days, with a mean of 3. Jobs also can have a priority index ranging from 1 to 3 where 3 means a rush job. Every job consists of one or more products of one available type. For every product type there are two process plans available which can be selected for production. Other job characteristics are:
- the batch size: 1 to 5 with a mean of 3,
- the number of operations: 1 to 5 with a mean of 3,
- the process time: 10 to 50 minutes with a mean of 30,
- special tools per operation: 1 to 5 with a mean of 3.

A file generator has been developed to generate a GPS file with the preferred factory and job characteristics.

5.2. Knowledge acquisition for the second case

At this point the knowledge acquisition process (in analogy of Section 2) for the expert system for the second case is described. We confine with the description of the acquisition process for the LPS-"planning level".

5.2.1. Step 1: definition of useful algorithms, heuristics and solution rules

First step in the knowledge acquisition process is to select a subset of useful algorithms (cf. P1...Pn in Fig. 2) from the nine algorithms available on the planning level.

In this case the objective on the planning level is to balance the workload among the machines. According to this objective we have, based on the knowledge and experience of the LPS-builder, selected the following algorithms:

For the process plan selection:
- full enumeration,
- a heuristic on workload balancing,
- a mixed integer programming formulation,
- a heuristic for sorting on the workload of one specific machine.

For work loading:
- linear programming with a utilization of 95%,
- a heuristic with a tool capacity bound and a utilization of 95%.

5.2.2. Step 2: definition of indicators for the advice part

The expert system has to advise an algorithm, based on the situation given by the GPS and depending on the status of the FS. For this information is required (cf. K1...Kn in Fig. 2). This information must be compressed into a limited set of indicators (see also Fig. 6). Based on the results of step 1 and given the information stored in the GPS data file, we have chosen the following set of indicators for the selection of planning algorithms:
(1) the total workload per machine,
(2) total workload released/total workload,
(3) the number of different parts,
(4) the average number of alternative process plans,
(5) the total number of operations,
(6) the average order priority,
(7) the average release date,
(8) the average due date,
(9) the number of orders.

5.2.3. Step 3: definition of indicators for the evaluation part

The evaluation is an important part of the expert system framework. When an algorithm is selected and executed, the results may be judged to be poor or even not acceptable by the evaluation part of the expert system, according to some specified criteria. These criteria may be simple operational criteria such as the average or maximum lateness but may also concern more "fuzzy" conditions such as the capabilities of operators to deal with the resulting solution. To this end some control indicators are needed. We have

selected:

(10) A parameter indicator whether or not the result is almost satisfactory, what exactly is meant by "satisfactory" is discussed below.
(11) The number of attempts so far.
(12) The algorithms evaluated earlier. These parameters are written into the database. They provide sufficient information to yield a good advice. Generally, the process to find the proper parameters for practical situations will be iterative.

5.2.4. Step 4: definition of the knowledge rules

Now that we have defined a set of important indicators in combination with a set of useful algorithms in the previous steps, we may construct the knowledge rules by using a chart as depicted in Fig. 2. After the determination of the indicator intervals and the certainty factors our knowledge acquisition process is completed.

We implemented two types of knowledge. We can distinguish:

(1) theoretical knowledge: this is knowledge about the actual LPS-algorithms. This means that the relation between the input data characteristics and the output performance of each algorithm and its properties in different situations are stored in knowledge rules. This knowledge is generally usable,
(2) situation specific knowledge which may concern specific company knowledge or FS-configuration constraints (e.g.: tool X is not available for job of type Y). This knowledge is specific for each of the four cases and therefore different in each of the cases.

5.3. Evaluation of the expert system in the second case

5.3.1. How to evaluate an expert system

The expert system is implemented to support the operator in decision making among the different algorithms, rules and heuristics available on the different levels in the LPS. So, what we are interested in, is what the quality is of the advice to select a specific algorithm, as given by the expert system. Did the algorithm, suggested by the expert system, perform well? Before we can give judgement about a satisfactory performance of an advised algorithm, we first must know what exactly is meant by satisfactory. In other words: how can one measure the quality of an advice.

5.3.2. The quality of an advice

To establish the quality of an advice one has to evaluate the actual output of the LPS, which is actually a complete production schedule. A schedule can be evaluated by performance indicators like maximum lateness, tardiness, utilization of machines or throughput times. These performance indicators can be compared with lower bounds. Lower bounds can be found by using e.g the shifting bottleneck procedure (cf. Ref. [17]) for the lateness criterium and a mathematical programming formulation for the utilization criterium. Deviation of the performance indicators from the lower bound gives a measure for the quality of a production schedule.

In this research, the quality of the production schedules is measured for every advice generated by the expert system. The reason for not presenting the data to match, is that the data gives more information about the quality of the used algorithms themselves instead of given straight the information about the quality of the decision of choosing the algorithm. So, in our evaluation we have used the following performance measure: did the advised algorithm lead to a solution (i.e. a workload balancing or a production schedule) which could not be improved by applying any of the other available algorithms according to the performance measures described above (in particular the lateness criterium).

5.3.3. Evaluation results of the second case

The LPS Expert system is tested on 20 different GPS order sets.

In 10 cases the expert system advised these algorithms which lead the LPS to an "optimal" production schedule at once.

In 7 cases the expert system used its evaluation optimization mechanism (cf. Figs. 5 and 6) on planning level only (4 times) on scheduling level only (3 times) and on both levels at the same time (2 times) to find the optimal combination of algorithms leading to the "optimal" production schedule.

In 2 cases the expert system advised to return to the LPS-planning level for a complete new LPS session, after recognizing infeasibility at the LPS-scheduling level. These cases show that the control mechanism as depicted in Figs. 5 and 6, works well.

In one case the expert system did not advise the algorithm that finally led to the "optimal" production schedule.

5.3.4. Remarks

(1) The results of these tests reflects the quality and the completeness of the knowledge gathered during the acquisition phase.
(2) Although only the second case is outlined here, equivalent results have been found in the other three test environments.
(3) The fact that most of the times an optimal production schedule has been found indicates that the situation presented by the GPS is interpreted well by the expert system.

5.3.5. Conclusions extracted from the test cases include

(1) In all four test cases, the LPS performs well on the advices given by the ES. We only once could find a better production schedule by using a different algorithm then the one advised by the expert system in the second case.
(2) In the various, often complex, situations and circumstances the expert system appears to be a useful tool in supporting decision making to control the LPS.
(3) The theoretical knowledge (not presented here but described in Ref. [14]) added till now is functioning well.
(4) It appears to be very easy to add knowledge, and make the universal LPS situation specific (e.g. by adding practical constraints).

6. Conclusions and suggestions for further research

In this paper we have outlined the structure and functions of a rule-based Expert System (ES), which has been developed to assist decision making by a Local Planning and Scheduling System (LPS), operating at a production cell level. In particular, we have underlined the need to recognize and interpret complex situations in the Local Planning Environment, as well as the fact that any advice from the LPS should be based on both theoretical knowledge (algorithms and heuristics) and factory specific practical knowledge. A prototype version of the Expert System has been programmed. Finally, we have evaluated the performance to the ES by considering some typical cases; this evaluation in particular addresses the quality of the advisory function, not primarily the quality of the underlying algorithms (which is quite a different topic).

Further research might be concentrated on a number of aspects, including:

- the influence of alterations in the condition part of the underlying rules. The knowledge evaluation has been based primarily on the values of the certainty factors. Changing the range of values for which a condition of a specific rule should hold may severely influence the ultimate performance of that rule,
- the extension of the ES with a simulation part (cf. Fig. 1). Such an extension yields the possibility to generate additional knowledge by simulating several alternatives. See Ref. [14] for some more elaborate ideas,
- knowledge acquisition in a real manufacturing environment. This will be the next topic in forthcoming research.

Acknowledgement

The author is grateful to Professor W.H.M. Zijm for detailed comments on earlier versions of this paper which have substantially improved the presentation of the material.

References

[1] Bossink, G.J., 1990. A Framework for Planning and Scheduling Manufacturing Systems. In: Proceedings Workshop on Decisional Architectures in Automated Manufacturing, Genova, Italy.

[2] Bossink, G.J., 1990. Planning and Scheduling (Flexible) Manufacturing Systems. In: ISCIE Proceedings, 1990 Japan–U.S.A. symposium on Flexible Automation, Vol. III, pp. 1109, 1112.

[3] Bossink, G.J., 1992. Planning and Scheduling for Flexible Discrete Parts Manufacturing. PhD. Thesis. ISBN 90-9004508-2. University of Twente, Enschede, The Netherlands.

[4] Fox, B.P., Allen, M.S., Smith, S.F. and Strohm, G.A., 1983. ISIS: A Constraint Directed Reasoning Approach to Job Shop Scheduling. Carnegie Mellon Univerity, CMU-RI-TR-83-8, Pitsburg, USA.

[5] Bensana, E., Bel, G. and Dubois, D., 1988. OPAL. A knowledge based system for industrial jobshop scheduling. NATO ASI series, Vol. 49 C.I.M, pp. 295, 330.

[6] Wu, S.D., 1987. An Expert System approach for the control and scheduling of flexible manufacturing cells.

[7] Worden, B. and Weber, E., 1989. Erfurungen eines Expertsystems in Konstructions berreich. In: V.D.I. Berichte 775: Expertsysteme in Entwicklung und Konstruktion, pp. 215, 268.

[8] Bonnet, A., 1987. L'Intelligence artificielle; Promesses et Réalités. InterÉditions, Paris, France, ISBN 90-6789-050-2.

[9] Jackson, P., 1987. Expert Systems. Addison-Wesley, Reading.

[10] Horvitz, E.J., Heckerman, D.E. and Langlotz, C.P., 1985. A Framework for Comparing Alternative Formalisms for Plausible Reasoning. In: Proceedings Fifth National Conference on Artificial Intelligence. Vol. 1: Science. Morgan Kaufmann Publishers, Los Altos, pp. 210, 214.

[11] Turksen, I.B., 1988. An approximate reasoning framework for aggregate production planning. NATO ASI series, Vol. 49, C.I.M, pp. 243, 266.

[12] Levesque, H.J. and Brachman, R.J., 1985. A Fundamental Tradeoff in Knowledge Representation and Reasoning, pp. 42, 70.

[13] Buchanon, B.G. and Duna, R., 1981. Principles of Rulebased Expert Systems. Stanford University Memo, HPP-81-4.

[14] Meester, G.J., 1990. Een Expertsysteem voor Planning- en Schedulingsalgoritmen, Master thesis University of Twente.

[15] Saywer, B. and Foster, D., 1986. An Expert System in Pascal.

[16] Buchanan, B.G., Bartlow, D. and Benet, J., 1983. Constructing an Expert System. (Eds. Hayes-Roth, Waterman and Lenat), chapter 5.

[17] Adams, J., Balas, E. and Zwawack, D., 1988. The shifting Bottleneck Procedure for Job Shop Scheduling. Manage. Sci., 34(3).

[18] Aanen, E., 1988. Planning and Scheduling in Flexible Manufacturing Systems. PhD Thesis, ISBN 90-9002349-6, University of Twente, Enschede, The Netherlands.

[19] Fox, M.S., Chang, W.Y. and Peng, S.O., 1990. Factory Model and Test Data Descriptions: OPIS Experiments. Internal Report. Carnegie Mellon University, CMU-RI-TR-90-6, Pitsburg, USA.

[20] IBM, Knowledge Processing Series, Expert System Development Environment, Expert System Consultation Environment, and Expert System reference manual.