

Meta-Scheduling using Dynamic Scheduling Knowledge

Jürgen Sauer
Universität Oldenburg, FB Informatik
Postfach 2503, D-26111 Oldenburg
e-mail: Sauer@Informatik.Uni-Oldenburg.DE

published in: J. Dorn, K. Froeschl: „Scheduling of Production Processes“, Ellis Horwood, Chichester, 1993.

Abstract

When solving scheduling problems a crucial task is the selection or creation of the appropriate scheduling algorithm. An approach is presented that combines the use of meta-scheduling knowledge to select the appropriate strategy and the use of dynamic scheduling knowledge for the flexible creation of scheduling algorithms. The method is based on the decomposition of scheduling algorithms into their underlying strategies and heuristic rules for selection of appropriate orders, variants, operations or resources. The strategies are represented by means of planning skeletons and the heuristic rules in terms of rules. The refinement of abstract operators within a skeleton with the rules for selection generates again a scheduling algorithm. Therefore a huge library of different scheduling approaches is available due to the various possibilities for the dynamic combination of skeletons and rules. Meta-knowledge is used to select the appropriate skeleton and rules to be used within the skeleton regarding the given problem situation. The implementation leads to a hybrid scheduling system, providing the description and integration of different scheduling approaches and the support in selecting the appropriate ones.

1. Scheduling

The goal of **scheduling** is the temporal assignment of orders (e.g. for manufacturing products) to resources (e.g. machines) where a number of conditions have to be regarded. Scheduling covers the creation of a schedule of the production process over a longer period (**predictive scheduling**) and the correction of an existing schedule (or plan) due to actual events in the planning/ scheduling environment (**reactive scheduling** or **rescheduling**). Nearly the same information is needed for scheduling and rescheduling and the same conditions must hold for the result, e.g. concerning the goals to be fulfilled.

A scheduling problem can be described by the tuple (R, P, O, HC, SC) with:

- a set of resources $\mathbf{R} = \{R_1, \dots, R_r\}$ like machines
- a set of products $\mathbf{P} = \{P_1, \dots, P_p\}$ with information about variants, operations, machines
- a set of orders $\mathbf{O} = \{O_1, \dots, O_o\}$ to manufacture products
- a set of hard constraints $\mathbf{HC} = \{H_1, \dots, H_h\}$ (e.g. production requirements), that have to be fulfilled, and
- a set of soft constraints $\mathbf{SC} = \{S_1, \dots, S_s\}$ (e.g. meeting due dates), that should be fulfilled but may be relaxed.

Result of scheduling is a **schedule** (or **plan**) showing the temporal assignment of operations of orders to the machines that shall be used.

The **problem space** of scheduling can be represented as a heterogeneous AND/OR-tree. The nodes contain a statement about the contribution of the children nodes to the solution and a value of the type corresponding to the layer of the tree. Fig. 1 shows a graphical representation of the scheduling problem - including alternative production variants and machines. By integrating parameters like alternative variants or machines a more general view of scheduling is realized including some of the tasks of process planning. Finding a solution to the scheduling problem is equivalent to finding a solution of the AND/OR-tree where the constraints are met. In Fig.1 a solution is indicated by dark nodes and solid lines.

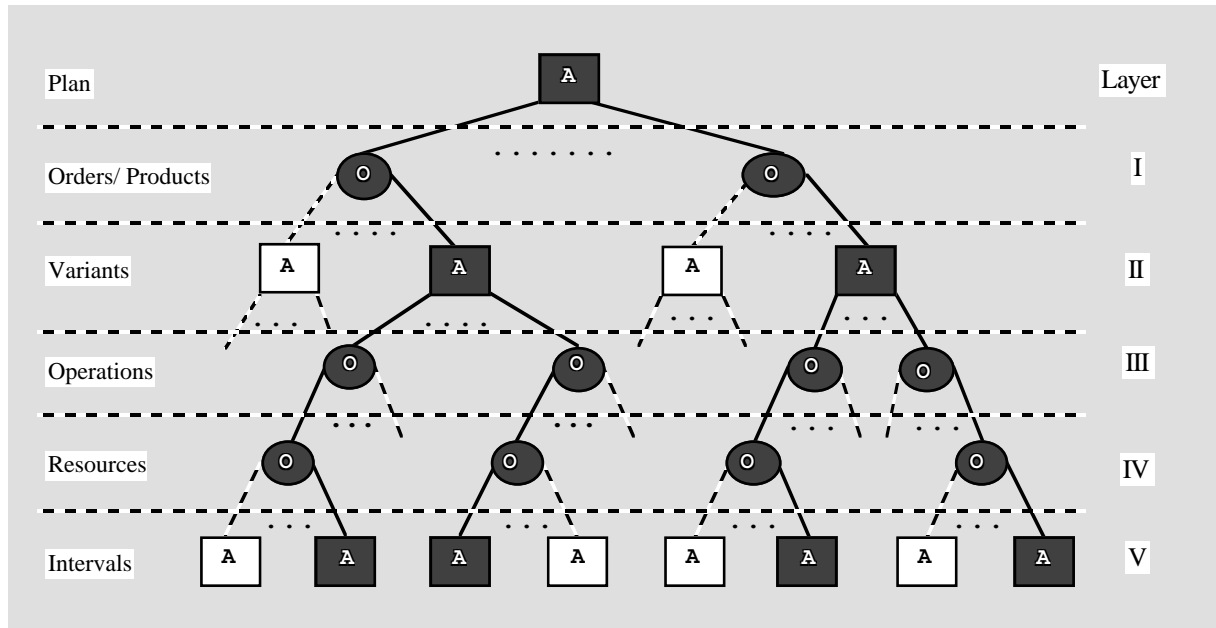


Fig.1: AND/OR-tree: problem space of scheduling

The determination of an optimal solution to a given scheduling problem requires, in the worst case, the generation of all possible solutions. This is not realistic because of the combinatoric size of this kind of problem. Optimal scheduling belongs to the class of NP-complete problems. Therefore, a crucial task of scheduling is the creation of "good" schedules without searching through the whole problem space. Knowledge-based approaches try to reduce the search space by using the experience and (heuristic) knowledge of a human production planner.

Because no general algorithmic approach sufficient for all scheduling and rescheduling problems is known yet, a scheduling system should offer multiple alternative algorithms applicable to varying circumstances in connection with a support in selecting the most appropriate approach for a given problem, e.g. specific rescheduling problems suggest an emphasis on either order- or resource-based algorithms [Ow 88].

This paper describes an appropriate representation of a huge variety of different scheduling approaches, their integration in one planning system and the representation and use of meta-scheduling knowledge to select the appropriate approach in a given situation.

2. Dynamic Scheduling Knowledge

Different kinds of knowledge are used in solving a scheduling problem. This knowledge can be divided into the areas of domain, situation, scheduling and meta-scheduling knowledge:

- The **domain knowledge** contains static information about the application environment, i.e. the structure of the scheduling problem, e.g. the set of orders, the possible products, their recipes, the available resources etc.
- The **situation knowledge** represents the current state of scheduling, e.g. the existing schedule, the remaining capacities of resources etc.
- The **scheduling knowledge** is divided into static and dynamic approaches for scheduling and rescheduling.
 - The **static scheduling knowledge** contains complete algorithms, e.g. operations research algorithms but also knowledge based ones. These algorithms are firmly implemented and are not changeable any more.
 - The **dynamic scheduling knowledge** denotes chunks of heuristic planning knowledge, which can be used together or alternatively in finding solutions to scheduling problems. It provides the possibility for adequate description and integration of multiple algorithmic approaches in one system.
- The **meta-scheduling knowledge** contains the information necessary to determine the "best" algorithm due to the current scheduling problem, i.e. which algorithms are applicable for which tasks (goals) and which ones are appropriate for which events.

Most of the scheduling approaches developed so far are based on a divide-and-conquer technique and can be classified according to their perspective of the problem decomposition:

- *order-based*, i.e. one order is selected from all unscheduled orders and all operations of this order are completely scheduled before continuing with the next order, e.g. [Fox 87].
- *resource-based*, i.e. a resource is selected and then the appropriate operation is chosen out of the set of possible operations on this resource, e.g. [Liu 89].
- *operation-based*, i.e. one operation after another is selected and scheduled, e.g. [Keng 88].

All algorithms work until all orders and their operations are scheduled.

Each of these approaches can be viewed as a scheduling strategy useful to create a solution to a scheduling problem. As the determination of such a solution is equivalent to finding a solution of the AND/OR-tree the ideas of using dynamic planning knowledge can be explained by means of the AND/OR-tree representation of the problem space. A planning strategy describes the traversing of the tree, where several heuristic rules are used to select the next node to be checked. On the different layers of the tree (orders, products etc.), different selection rules are possible, e.g. priority rules for the selection of orders. So the basic components of a scheduling algorithm are the underlying strategy (how to traverse the tree, e.g. select an order and then solve the problem for this order, then the next order etc.) and the heuristic rules used for the selection of the nodes on the different layers. The combination of different strategies and selection rules leads to a huge variety of possible scheduling algorithms. To exploit the power of this approach, an appropriate representation for strategies and rules is needed.

The basic idea for the representation of planning knowledge is the possibility of describing the strategies as skeletons, where special parts (selection of nodes in AND/OR-tree) can be refined with appropriate rules regarding the actual planning problem. The idea of using skeletons is adopted from [Friedland 85], where skeletal plans are sequences of abstract operators describing plans for experiments and the refined skeletal plans are the desired result.

In the approach presented here a **planning skeleton** describes a strategy for searching in the AND/OR-tree (how to create a schedule step by step), but without any heuristic for the selection of the nodes of the tree. For example, a skeleton for an order-based strategy looks as shown in fig. 2 (Roman numbers denote the layer of the AND/OR-tree on which the selection has to be done, *rule* gives an example of a rule that can be used).

```

WHILE orders to plan
  select order (I)
    (rule: combination of bottleneck first, EDD, slack rule und user priority)
  select variant (II)
    (rule: 'stem variant' first, then alternatives)
WHILE operations to plan
  select operation (III)
    (rule: LIFO)
  select resource (IV)
    (rule: 'stem apparatus' first, then alternatives)
  select interval (V)
    (rule: forward from earliest start)
  plan operation or solve conflict
END WHILE
END WHILE.

```

Fig.2: Order-Based Planning skeleton with possible rules

Fig. 2 shows the skeleton of the order-based approach implemented within the EUREKA-project PROTOS (Prolog Tools for Building Expert Systems) [Appelrath 87, Sauer 90, Sauer 91] together with the used rules on the different layers. Fig. 3 shows two other examples of representing planning strategies in terms of skeletons.

operation-based	resource-based
select variant for every order (II) find operations of orders (I) WHILE operations to plan select operation (III) select resource (IV) select intervall (V) plan operation END WHILE	select variant for every order (II) find operations of orders (I) WHILE operations to plan select resource (IV) select operation (III) select intervall (V) plan operation END WHILE

Fig.3: Planning skeletons for scheduling

Rules are used for the selection of nodes at each layer of the tree and thus guide the search for a good solution. For each layer of the AND/OR-tree there exists a set of alternative rules, some of which are listed in fig. 4.

Layer	
I	selecting orders <ul style="list-style-type: none"> - change weights of the 4 combined rules - increasing start days (FIFO) - increasing number of alternatives (critical products first) - increasing planning intervals (critical products in time first) - increasing user priority (critical products of the user first)
II	selecting variants <ul style="list-style-type: none"> - given order with stem first - inverse order - increasing production factor (critical variant first) - decreasing production factor (simple variant first)
III	selecting operations <ul style="list-style-type: none"> - increasing operation number - decreasing operation number - increasing number of alternative resources (critical operation first) - decreasing number of alternative resources (simple operation first)
IV	selecting resources <ul style="list-style-type: none"> - given order with stem first - increasing resource factor (simple resource first) - decreasing resource factor (critical resource first)
V	selecting intervals <ul style="list-style-type: none"> - forward from given starting date - backward from end date to starting date, then forward from end date

Fig.4: Possible rules usable for selection in skeletons

With this kind of knowledge representation a large variety of scheduling and rescheduling algorithms can be created dynamically by combining planning skeletons and appropriate rules for selections, making a library of different scheduling and rescheduling approaches available.

A crucial problem, especially if a library of scheduling approaches is available, is the selection of the "best" approach in a given problem situation. Meta-knowledge can be used to support this task. One first approach was presented by the OPIS system [Smith 90] where one algorithm out of four possible ones is selected due to the values of several parameters describing the actual scheduling situation. In our approach meta-knowledge is used more general and can be used to select

- appropriate strategies for scheduling and rescheduling both as fixed algorithms or as skeletons,
- the rules within the skeletons, and
- conflict solution strategies useful within the skeletons for scheduling and rescheduling.

Different sources of meta-knowledge may be exploited. The meta-knowledge covers the goals which shall be achieved, the events, that have to be tackled, and other information specific to the situation in which an approach is appropriate.

A **goal** is a planning task that shall be met by the plan to be generated. Goals may be global

ones like meeting due dates or local ones like machine utilization or reduction of set-up times. **Events** describe cases of external or internal errors which mainly affect the planning result, in most of the cases this is the existing plan. External events are e.g. cancellations of orders or new high priority orders. Internal events are e.g. breakdowns of machines or other resources.

3. Representation of Scheduling Knowledge by Heuristics

A language called HERA (heuristics for representation of scheduling knowledge) has been developed for the representation of the scheduling and meta-scheduling knowledge. HERA is based on Prolog providing all the features of declarative knowledge representation and the integration of Prolog code as well as calls to other languages. The "classical" representation of heuristics as IF-THEN rules is extended by an explicit representation of goals and events and by control constructions usable in the action part to describe planning strategies. Heuristics then look like

```

HEURISTIC name                % name of the heuristic
IF  SITUATION  situation      % description of the situation to use the heuristic
      GOAL       goal           % description of the goals that can be achieved
      EVENT     event          % description of the events that can be solved
THEN action                    % description of the strategy (the heuristic)
END  HEURISTIC.

```

The IF-part of the heuristic contains the meta-knowledge which is used to select the appropriate strategy. The description of the situation consists of a sequence of Prolog calls (mainly retrieval operations) concatenated by AND. Goals and events are represented by definite Prolog atoms.

```

GOAL <goalname>.                e.g. GOAL meet_due_dates.
EVENT <eventname>.              e.g. EVENT machine_breakdown.

```

The action part (THEN-part) of the heuristics is used to call fixed strategies or to describe the scheduling and rescheduling strategies completely or as planning skeletons. The syntax of the action part is:

```

<actions>          ::= <action> | <action> AND <actions>
<action>           ::= WHILE <w_condition> DO <actions> END WHILE |
                       OR [ <action>, <act_list> ] | <simple_action>
<act_list>         ::= <simple_action> | <simple_action>, <act_list>
<simple_action>     ::= HEURISTIC-CALL(<heuristic_name>, <goals>, <events>) |
                       <rulecall> | <operation> | <prolog_literal>.

```

If a fixed strategy has to be called, the action part only consists of a Prolog call. If a strategy or a skeleton has to be described, control constructs are needed to represent the traversing through the AND/OR-ree:

- **AND**: actions are concatenated by AND
- **WHILE**: used for treating AND-nodes (all descendants have to be considered). While the *w_condition* holds the *actions* are executed.
- **OR**: used to support an intelligent search process. Several rules or heuristics, e.g. for checking different parts of the tree, may be listed and will be applied in the given order.

```

HEURISTIC plan_PROTOS
  IF SITUATION      no_plan
    GOAL            [planning:meet_due_dates]
    EVENT           []
  THEN
    call(protos_algorithm)
END HEURISTIC.

HEURISTIC plan_order_based_PROTOS
  IF SITUATION      no_plan
    GOAL            [scheduling:[meet_due_dates]]
    EVENT           []
  THEN
    create_orderlist
  AND
  WHILE orders_to_plan DO
    select_order_fifo
  AND
    select_interval_earliest_start
  AND
    select_variant_stem_first
  AND
  WHILE steps_to_plan DO
    select_step_fifo
  AND
    select_app_stem_first
  AND
    OR[plan_step, HEURISTIC-CALL(solve_overlap_basic, [], [])]
  END WHILE
  END WHILE
END HEURISTIC.

```

Fig.5: Two examples of heuristics represented with HERA

simple_actions are:

- calls of Prolog-procedures
- calls of predefined rules or simple (update) operations
- heuristic calls: this gives the possibility to call a heuristic explicitly by name (first argument) or to specify goals or events (second and third argument) for the automatic selection of appropriate strategies. Heuristic calls with specified goals and events are used as the abstract operations within planning skeletons.

Fig. 5 shows an example of a heuristic calling a fixed algorithm (HEURISTIC plan_PROTOS), with the meaning: "if a set of orders is given, the goal of scheduling is meeting the due dates, no event has occurred, and no actual plan exists then the PROTOS algorithm may be used", and the PROTOS strategy already mentioned (fig. 2) represented as heuristic where all the control constructions are used.

The scheduling environment (classes of objects), operations on objects like RETRIEVE or MODIFY, and the rules needed are represented in a similar manner, e.g. rules as:

```

RULE name
IF           SITUATION situation
THEN rule_actions
END RULE.

```

name is the name of the rule, *situation* describes the situation in which the rule is to be used and *rule_actions* represents a list of simple actions consisting of basic Prolog-procedures, rule-calls and update- or retrieval-operations.

4. META_PLAN: Using Dynamic Planning Knowledge

Scheduling and in particular rescheduling are such tough problems that no algorithmic approach will solve them in a satisfactory manner in the near future. Therefore, the objective of system approaches should not be the replacement of the human experts, but their support in order to extend their problem solving capabilities. Thus a computer-based scheduling system supporting the human production planner in a user-friendly interactive manner should combine a graphical user interface with multiple algorithmic scheduling methods.

The prototypical system META_PLAN realizes such an approach. The **graphical user interface** provides a comprehensive presentation of information as well as multiple tools for interaction, especially for manual scheduling and user-directed selection of scheduling and rescheduling algorithms. The **algorithmic part** consists of a great variety of knowledge for scheduling and rescheduling and a meta-scheduler for the selection of appropriate approaches. Fig. 6 shows the architecture of META_PLAN consisting of eight components.

The "*User-Interface*" provides a user-friendly and problem adequate graphical presentation of the information currently in the knowledge base, e.g. a Gantt chart representation of the existing schedule, and multiple tools for interaction.

The "*Evaluation*" component offers several functions for evaluating schedules, e.g. sum of delays.

The "*Meta-Scheduling*" component realizes the selection and interpretation of the appropriate scheduling strategies. The new scheduling problem is analyzed, the appropriate skeletons are identified and refined to solution algorithms, the alternative algorithms (static as well as dynamically created ones) are judged and the applicable approaches are ranked with respect to their appropriateness for the current problem.

To identify appropriate heuristics the actual goals and events are compared to the corresponding descriptions in the heuristics. Appropriate are those where the intersection of the goal description and the actual goals is not empty and where all actual events are tackled. The "best one" of these (where most of the actual goals are met) is applied first. If the interpretation of a heuristic succeeds, a new plan has been generated. If the interpretation of a heuristic fails, the next heuristic is chosen until a solution is found or no appropriate heuristic is available. In this case no solution is found.

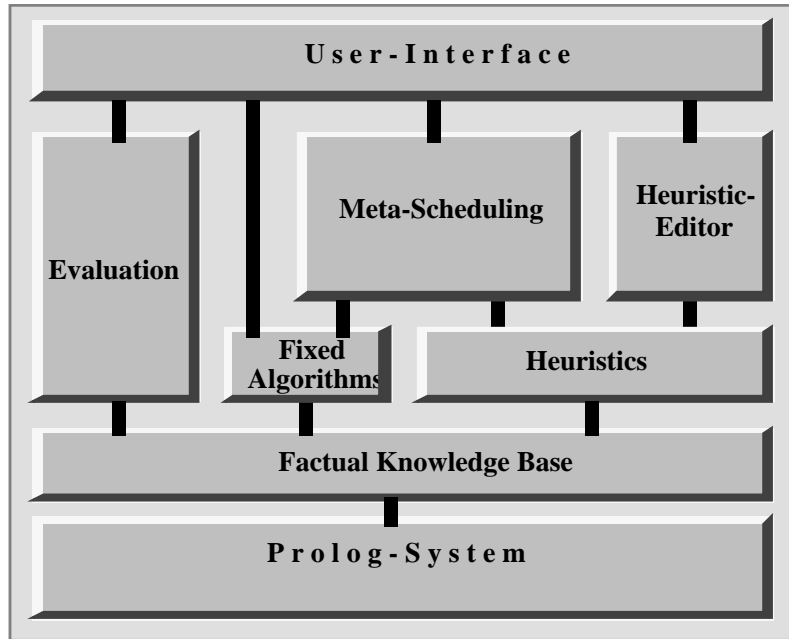


Fig.6: System Architecture of META_PLAN

The interpretation of a heuristic consists of the interpretation of the situation part and of the action part of the heuristic. As the situation part consists of a sequence of Prolog calls concatenated by AND, the interpretation is done by the Prolog interpreter itself. If the situation described in the situation-part is equivalent to the actual situation then the heuristic is used, i.e. the action part is interpreted. Fig. 7 shows some of the rules used for the interpretation of actions.

A rule is interpreted like a heuristic. The main difference is that the *rule_actions* in the action-part of the rule are more simple. Rule calls are leading to an interpretation of the corresponding rules, basic procedures for sorting lists or retrieving information from the database are interpreted by the Prolog system.

```

if the next action is the empty-action then the interpretation succeeds.

if the next action is an OR-action then the OR-connected actions are interpreted in the
given order. If one action succeeds, the OR-action succeeds. If no action succeeds, the
OR-action fails.

if the next action is a WHILE-action then the actions in the body are repeatedly
interpreted as long as the condition holds. If the condition fails at the beginning, no
action of the body is executed.

if the next action is a rule-call or heuristic-call then the corresponding rules or
heuristics are interpreted.

if the next action is a basic procedure then it is interpreted by the Prolog-system.

```

Fig.7: Interpretation rules

In a simulation part the chosen scheduling algorithm is applied to the new scheduling problem and the result is displayed graphically to the user. Additionally the user has the choice of scheduling order after order, eventually changing the algorithms every order, or scheduling the

whole set of orders with one selected approach. The expert can either confirm or reject the result of simulation. This component is a meta-scheduler since it builds schedulers, not schedules [Kempf 89].

The "*Heuristic-Editor*" component enables the acquisition of new and the alteration of current knowledge represented with HERA.

The knowledge base of META_PLAN is spread across three components. The "*Fixed Algorithms*" module contains several fixed scheduling algorithms which can be selected by the meta-scheduler. It also provides an interface for integrating fixed scheduling algorithms (e.g. from operational research) written in other languages together with the meta-knowledge necessary for their application giving META_PLAN features of a hybrid scheduling system. The "*Heuristics*" component contains the dynamic planning knowledge consisting of strategies, skeletons and rules represented with HERA. The "*Factual Knowledge Base*" contains the static and dynamic facts of the scheduling area, e.g. production requirements, orders, and the schedule.

The system is implemented in Prolog using features of the underlying "*Prolog System*" like definite clause grammars and interfaces to user-interfaces (OpenWindows).

Scheduling using META_PLAN then looks as follows:

Starting with the input of some tasks (e.g. scheduling, rescheduling), goals (e.g. meeting due dates) and events (e.g. insertion of new orders, cancellation of already scheduled orders, machine breakdown) a new scheduling problem arises. The new problem is defined by the tasks, goals, events and the current situation, e.g. the existing schedule and the available capacities. The user has the possibility to solve the new problem partly or entirely by manual scheduling. Alternatively the system can analyze the new problem and select the most appropriate algorithm (static or dynamic) to solve it. The user may alter this proposal or confirm it. Now the execution of the chosen algorithm can be simulated and the result is presented graphically. The user can choose whether the result should be entered in the knowledge base as the new schedule or not, and he/she can make manual corrections at all times.

5. Conclusion

A method for the adequate representation and integration of a huge variety of different scheduling approaches together with the meta-knowledge needed to select appropriate ones has been introduced in this paper. The basic concepts of scheduling algorithms are their underlying strategies and heuristic rules for selections. These strategies and heuristic rules are represented separately by means of planning skeletons and rules, respectively. A planning skeleton describes a strategy for searching in the AND/OR-tree, but without any heuristics for selections of nodes. Rules are used for the selection of nodes at each layer of the tree. Based on this representation of planning knowledge a scheduling algorithm is created dynamically by the substitution of the abstract operators within a skeleton with the rules. The various possibilities for combining skeletons and rules dynamically lead to a great variety of scheduling approaches. The meta-knowledge used covers the goals to be reached, the events or disruptions to be solved and specific information about situations.

This method provides high flexibility for scheduling and rescheduling (various different scheduling algorithms are available, each showing selective advantages for specific problems),

the user (he/she is supported in selecting the "best" strategy and can create his/her "own" scheduling algorithm) and extensions (other skeletons and rules can be integrated easily).

In addition, META_PLAN, a hybrid scheduling system providing the flexible use of the dynamic planning knowledge as well as the meta-scheduling knowledge has been described. A prototype of this system has been realized in the context of the PROTOS project.

Literature

- [Appelrath 87] Appelrath, H.-J.: "Das EUREKA-Projekt PROTOS", in: Proceedings of the 2. Int. GI-Conference '87 Knowledge-based Systems, IFB 155, Springer, 1987.
- [Fox 87] Fox, M.: "Constraint Directed Search: A Case Study of Job-Shop Scheduling", Pitman Publishers, London, 1987.
- [Friedland 85] Friedland, P.E., Iwasaki, Y.: "The Concept and Implementation of Skeletal Plans", in: Journal of Automated Reasoning, No. 1, 1985.
- [Kempf 89] Kempf, K.: "Manufacturing Planning and Scheduling: Where We Are and Where We Need to Be", in: Proceedings IEEE 5th Conference on Artificial Intelligence Applications, 1989.
- [Keng 88] Keng, N.P., Yun, D.Y., Rossi, M.: "Interaction Sensitive Planning System for Job-Shop Scheduling", in: Expert Systems and Intelligent Manufacturing, 1988.
- [Liu 89] Liu, B.: "Knowledge-Based Production Scheduling: Resource Allocation and Constraint Satisfaction", DAI Research Paper 436, University of Edinburgh, 1989.
- [Ow 88] Ow, P.S., Smith, S.I., Thiriez, A.: "Reactive Plan Revision", in: Proceedings of AAAI-88.
- [Sauer 90] Sauer, J.: "Design and Implementation of a Heuristic Planning Algorithm", in: Appelrath, H.-J., Cremers, A.B., Herzog, O.(eds.): "The EUREKA-Project PROTOS", Zürich, April 1990.
- [Sauer 91] Sauer, J.: "Knowledge Based Scheduling in PROTOS", in: Proc. of IMACS World Congress on Computation and Applied Mathematics, Dublin, 1991.
- [Smith 90] Smith, S.F., Ow, P.S., Matthys, D.C., Potvin, J.-Y.: "OPIS: An Opportunistic Factory Scheduling System", in: Proc. of 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE, Charleston, USA, 1990.