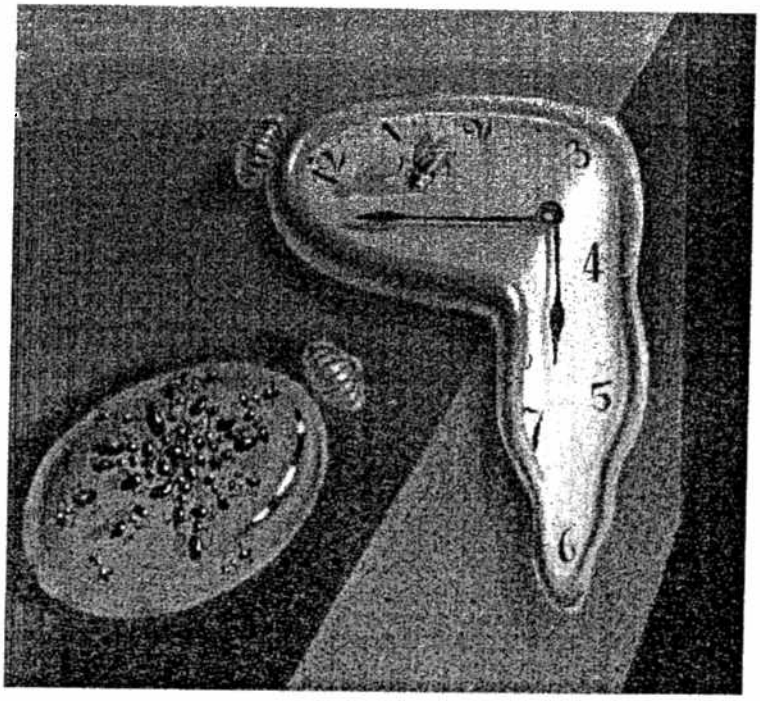
Institut für Automation
Abt. für Automatisierungssysteme

Technische
Universität
Wien

Projektbericht Nr. 183/1-31
April 1992

A Case-Based Reasoning Approach to Dynamic Job-Shop Scheduling

A. Bezirgan



Ausschnitt aus: Salvador Dali, "Die Beständigkeit der Erinnerung"

A Case-Based Reasoning Approach to Dynamic Job-Shop Scheduling

Extended Abstract

Atilla Bezirgan
Institute for Automation, Technical University Vienna
Treitlstrasse 3 / 183 / 1, A-1040 Vienna, Austria
Tel.: 00 43 - 1 - 588 01 - 81 84
Fax: 00 43 - 1 - 56 32 60
e-mail: aberziga@email.tuwien.ac.at

Introduction

This section gives a brief introduction to the dynamic job-shop scheduling problem and to case-based reasoning.

The problem

Scheduling is the task of assigning resources to operations over a period of time in order to achieve certain goals. The job-shop environment has the following characteristics

- the goals to be achieved are to process some discrete parts,
- there are several different kinds of goals, e.g. different products,
- usually, there are several different plans to achieve one goal, and
- different plans consist of partially different operations requiring different resources.

Scheduling in such an environment involves selecting a plan to achieve a goal. The term dynamic job-shop scheduling refers to a scheduling process in job-shop environment which is able to deal with unforeseen events in the execution of a schedule. Unforeseen events can be new orders which arrive during the execution of a schedule or events like the break down of a machine.

Why is this problem complex?

There are several reasons for the complexity of this problem.

- There is a combinatorial explosion in the number of possible schedules in each problem dimension such as the number of machines and operations. This makes it necessary to use implicit representations of the search space such as with constraints instead of enumerating the elements of the search space.
- There are a number of constraints which a valid schedule must satisfy such as due dates and constraints concerning operation sequences. Interactions of these constraints make scheduling decisions difficult.
- Scheduling decisions very often depend on context. This makes it difficult to write down an explicit goal function.
- Dynamic scheduling involves reacting in time to changes in the environment.

What is case-based reasoning?

To reason from cases means to use experience made earlier in solving a certain problem to solve the current problem rather than to solve the current problem from scratch using first principles. A case usually consist of the description of a problem, a solution to the problem as well as of knowledge used in solving the problem. A case memory contains cases organised in a way to make efficient retrieval possible. In case-based reasoning, given a new problem a retrieval component retrieves the case most similar to the current problem. Using this case an adaptation component tries to generate a solution to the new problem. See [BEZI92] for a detailed discussion.

Why use case-based reasoning for dynamic job-shop scheduling?

Since case-based reasoning does not involve exploring the whole search space, the problem of combinatorial explosion is avoided. One strength of case-based reasoning lies in its ability to deal with

context dependent information. Thus constraint interactions and scheduling decisions depending on context may be modelled efficiently in a case-based manner. One weakness of case-based reasoning is that it is not well suited for optimisation problems since it does not explore the whole search space. However, the impact of this weakness becomes smaller in a real-time environment in which a timely, good solution is often better than the strive for a perfect solution that misses deadlines. Further, case-based reasoning is usually faster than other enumerative methods. Thus case-based reasoning seems to have the characteristics needed for a method suitable for dynamic scheduling tasks.

Has case-based reasoning been used for scheduling yet?

There are some successful applications of case-based reasoning to scheduling tasks though no application to dynamic job-shop scheduling for job-shops of considerable size is known to me. [BAHE89], [MARK89], and [HEHI91] describe an application of case-based reasoning to autoclave scheduling. The system is reported to be used in real-world environment. [MISY??] describes an interactive scheduling assistant for job-shop schedule repair. [KOTO89] describes plans to build a system to schedule large-scale airlift operations.

Further contents of this paper

The first steps in building a case-based reasoning system are

1. define what cases contain and how they are represented,
2. define case memory organisation,
3. define retrieval and how similarity of cases is assessed, and
4. define the adaptation process.

This list does not imply a strict linear development process. As will be seen later, there are mutual dependencies between these steps.

The rest of this paper represents the results of these steps for the dynamic job-shop scheduling problem. Among the issues not discussed in this paper is "how cases get into the case memory", i.e. knowledge acquisition is not a topic here. Assuming a non-empty case memory, an outline of case-based scheduling will be given.

Assumptions

There are a few assumptions which have been made in the rest of this paper.

- For the following discussion, a production environment is assumed. So problems of assembly or other kinds job-shop environments are not regarded.
- Beside the case-based reasoning specific components described below, the scheduling system possesses static knowledge of machines, parts that can be produced, and plans for the production of parts. Usually, there are several plans per part. These knowledge structures are not part of the case memory. However, they are connected by pointers to the case memory. For example, plans have pointers to instances of their use. As will be seen further below, this helps in retrieval and adaptation.

Cases

Since the scheduling of incoming orders is influenced by the schedules of earlier and later orders, cases do not have sharp boundaries. A case can be said to start at the point in time when an order comes in and to end when the ordered product leaves the production hall. However, such a case has pointers to earlier and later cases which influence it and which it influences. These pointers and the cases pointed in some sense belong to the case, thus leading to unsharp boundaries.

The components of a case are described below. Their representation and organisation is described in the section titled "Case Memory Organisation".

The problem

The problem description in a case consists of the following components:

- the goals,
- the constraints on the goals, and
- the problem environment.

An internal model of the factory and its current status makes up the problem environment. This model is kept as part of the case memory (see "Case Memory Organisation") and updated when necessary, e.g. when events like machine breakdown occur. Thus it does not have to be specified explicitly in every problem description.

The main goal is always the generation of a schedule. However, this goal can be triggered by a number of events, e.g. new orders coming in or tardi-

ness of an operation. In the following, only scheduling new orders will be regarded as a goal.

The constraints give restrictions on the kinds of solutions to be generated, e.g. cost or quality requirements. The different types of constraints which can be specified here are much like the ones in the ISIS system. However, the language used in expressing constraints must be a restricted one since retrieval and adaptation must be able to check equality and subsumption relations between constraints which would not be possible if unrestricted first order predicate calculus expressions were allowed for defining constraints. This language has not been developed yet.

The solution

The solution consists of an assignment of resources to operations over time. This represents a prediction of the development of the activities on the shop-floor. The output of the system is an event-driven schedule and a set of commands to machines and personnel ensuring the execution of the schedule.

Justification structures for scheduling decisions

A case further contains, for each scheduling decision, a justification structure that gives the reasons for this decision. In this justification structure a rich causal vocabulary is used which facilitates adapting previous schedules. This vocabulary makes it possible to express causal dependencies like "scheduling operation O_1 on machine M_1 at time T_1 made scheduling operation O_2 on machine M_2 at time T_2 impossible". This way the relations of the scheduling decisions and constraints which were important in a certain decision process can be modelled. The language of causal configurations does not exist yet. It will contain relations like "A enables B", "A causes B", "A disables B", "A is a side-effect of B", "A is the desired effect of B".

Case Memory Organisation

The case memory is organised around a time-line. This time-line provides a model of time which is so important in scheduling. Any suitable model of time - interval-based or time-point-based, continuous or discrete - is possible. The case memory also contains a model of the factory and its current status as well as pointers to the static knowledge sources of the system. Orders are registered at the time points at which they come in and have pointers to other components of a case, e.g. solution, belonging to them. Past and current schedules are also recorded using this time-line. Each scheduling decision has pointers to objects, orders, plans, machines, and - over a justification structure - to other scheduling decisions and constraints related to it.

The case memory is one large semantic network in which the boundaries of cases are fuzzy and floating. Events on the shop-floor such as machine breakdown are registered in the model at the time of their occurrence and connected by pointers to the schedules they invalidated as well as to the cases representing the rescheduling processes. Note that objects, events, and constraints common to two cases are represented once and pointed to by both cases. This makes it possible to reach one case from the other using their common features.

Retrieval

When a new order arrives the retrieval component searches the case memory for a previous order that is most similar to the current one. Similarity of orders is judged by comparing the constraints on the orders, i.e. similarity of orders is assessed using the similarities of the constraints. One golden rule of case-based reasoning says "do not retrieve what you cannot adapt". Thus similarity of two constraints is judged by the ease of adapting a solution satisfying one constraint to become a solution satisfy the other constraint. This means that the definition of similarity - and hence the whole of retrieval - depends crucially on the capabilities of the adaptation component.

The most similar old order is not found by comparing all old orders to the new one but by exploiting the semantic net to get from one similar case (case sharing one or more constraints with the new order) to another one.

For example, given a new order we can easily retrieve all previous orders for the same product. This is possible since all cases have pointers to the description of the products produced as well as to the plans used. The associated cases are similar to the new order in the sense that they deal with the same product. Using further constraints on quality or quantity we can further differentiate between these cases. Following several links we can get to other cases which may be of use. For example, having decided to use a certain plan we can take a look at previous usages of that plan and the problems that were encountered then. Further, using

some deeper features such as constraint looseness may help in retrieving more useful cases. Finally, the adaptation component is not bound to use one case in generating a new schedule. It can as well use several cases, one in solving a specific scheduling decision problem. In the following, adaptation from a single case is assumed.

After the most similar old order is selected adaptation is performed. The result of the adaptation is a new case including a solution to the new problem. Next it is checked either the new solution satisfies all constraints. If it does, we are done. Else the next similar old order is tried. If after several adaptation tries no solution is found then the system is unable to deal with the given problem and the user must be consulted. For example, the adapted unsuccessful solutions may be presented to the user or he may be requested to relax some constraints.

Adaptation

Given a new order and an old case the adaptation component proposes a solution for the new problem. In doing so it utilises the justification structure of the old case, analyses the old solution and the differences between the old and the new order. It uses rules that map the above pieces of information to changes that must be made to the old solution to make it fit the new order. Further, the adaptation component can incorporate some algorithms for doing local search and partial scheduling. It can also use local dispatch rules in adaptation.

For example, a simple adaptation process would involve using another machine if the new order has higher quality requirements than the old order and these requirements cannot be met by the old solution.

The justification structures in the case memory play a central role in adaptation. They make it possible to make "plausible and rational" changes to old cases. For example, if the justification structure in an old case states that a certain machine M had to be used because M is fast and the due dates were tight then the adaptation component can check if these conditions are valid for the new order and if not select another machine which may be slower but also cheaper in operation.

Conclusion

This paper tried to outline the beginnings of a case-based reasoning approach to dynamic job-shop scheduling. The outlined research is being conducted at the Institute for Automation in co-operation with the Interuniversitary Centre for Computer Integrated Manufacturing (IUCCIM). The task at hand is to solve the dynamic job-shop scheduling problem for the production of toy cars (Ferarri Testarossa).

Much remains to be done. Languages for defining constraints and for building justification structures must be defined. Retrieval and adaptation algorithms must be formulated. The incompleteness of the discussion of these two components in this extended abstract is indeed due to the fact that these algorithms do not exist yet. Further, a prototype has to be built and the approach has to be evaluated using this prototype and more theoretic evaluation methods. Though not much can be said about the success of the approach at this stage of the project, using case-based reasoning in conjunction with justification structures seems promising in getting good schedules quickly.

References

[BAHE89] Case Adaptation in Autoclave Layout Design, Ralph Barletta, Dan Hennessy, in Proc. Case-Based Reasoning Workshop, Morgan Kaufmann Pub., Inc., 1989, p. 203-207.

[BEZI92] Case-Based Reasoning Systems, Atilla Bezirgan, Technical Report, Christian Doppler Lab. f. Exp. Sys., Vienna, 1992, forthcoming.

[HEHI91] Initial Results from Clavier: A Case-Based Autoclave Loading Assistant, Daniel Hennesy, David Hinkle, in Proc. Case-Based Reasoning Workshop, Morgan Kaufmann Pub., Inc., 1991, p. 225-232.

[KOTO89] SMARTplan: A Case-Based Resource Allocation and Scheduling System, Phyllis Koton, in Proc. Case-Based Reasoning Workshop, Morgan Kaufmann Pub., Inc., 1989, p. 285-289.

[MARK89] Case-Based Reasoning for Autoclave Management, William Mark, in Proc. Case-Based Reasoning Workshop, Morgan Kaufmann Pub., Inc., 1989, p. 176-180.

[MISY??] CABINS: Case-Based Interactive Scheduler, Kazuo Miyashita, Katia Sycara, in ??, p. 47-51.