Planning in Manufacturing Domains Controlled by Case-Based Reasoning

József Váncza

Computer and Automation Research Institute
Hungarian Academy of Sciences
H-1518 Budapest POB 63, HUNGARY
vancza@sztaki.hu

Abstract

In this paper a class of planning problems is dealt with where resources have a central role: they can allow for many alternative courses of actions, but, at the same time, they have to be used economically. Such problems are typical in manufacturing domains, e.g., in manufacturing process planning or inspection planning. The problem of constrained plan optimization is exposed in such resource-rich domains, and an appropriate planning method is outlined. The method that performs task grouping is supported by case-based reasoning. Finally, it is shown how the approach was applied in robotic inspection planning.

Introduction

Planning in manufacturing domains brings about serious difficulties for a planner: it must reason about complex objects, handle various, sometimes even contradictory technological constraints and deal with resources in a parsimonious way. In such domains, thanks to the abundance of alternative resources, finding a plan is rarely a problem. However, generating plans with at least close-to-optimal resource utilization is already very troublesome (Kis and Váncza 1996). The constrained plan optimization problem can be solved only if one is able to exploit domain knowledge in the quest for the best plans.

Below the problem of planning in resource-rich domains is outlined and a new planning method is presented. The method enables us to apply domain knowledge directly in the control of the search process. The most critical step of the planning process is performed by case-based reasoning.

Planning in Resource-Rich Manufacturing Domains

This section characterizes planning domains where resources play a central role. Such domains are typical in manufacturing. For further details see our detailed models on manufacturing process (Horváth et al. 1996, Váncza and Márkus 1996) and inspection planning (Váncza et al. 1998).

In manufacturing domains a planning problem consists of many *atomic tasks*. The tasks are, however, not independent. For instance, in process planning a problem is specified basically by a part model with form features, but further information on general part properties (material, sizes) as well as feature relations (e.g., perpendicularity) are also substantial.

The tasks are consolidated local subproblems with well-proven *solution patterns*. E.g., it is well understood which machining processes—and under what conditions—are applicable for drilling (or inspecting) deep holes.

Tasks can be performed by *operations* that use various *resources*. Operations usually require combinations of resources: e.g., a milling operation needs a machine tool, a fixture and a milling tool. In manufacturing domains the set of resources is typically large, the resources are modular and often have overlapping capabilities. As a consequence, the same task can be performed by a large number of alternative operations, all using different resource combinations. Resource changes are costly and may deteriorate the accuracy of machining (or inspection) operations.

The order of performing tasks is very important. Typically, the tasks are nonserializable (Barrett and Weld 1994): they cannot be achieved without interleaving subplans for other tasks. Beyond *ordering constraints*, there can be also *resource binding constraints* that require two or more tasks to be performed with the same resources (quite a common requirement in inspection planning). The constraints stem from technological considerations.

Plans should be *complete*, *consistent* (i.e., executable), and *optimal*. Optimal plans contain a minimal number of resource changes and use the cheapest resources. Further on, plans should cover as many alternative solutions as possible. Typically, partial-order plans are needed with alternative resource assignments. However, all solution instances must satisfy the relevant constraints and utilize resources at least in a close-to-optimal way.

Planning in manufacturing domains is a *constraint satisfaction* problem on one hand: all the relevant technological constraints must be observed. This is not always possible, since constraints which are mostly based on local solution patterns may contradict each other when put together. Hence, planning needs the application of constraint checking and relaxation methods, too. On the other hand,

planning is an *optimization* problem as well: in manufacturing domains optimal resource usage is a key issue.

The Planning Method

We have developed a planning method which can cope with large planning problems in resource-rich manufacturing domains, even if optimization is a major concern. The overall planning process goes like this:

1) For each task, generate operation alternatives with all applicable resource combinations.
2) Generate ordering and resource binding constraints for the tasks. Resolve conflicts, if any.
3) Find maximal groups of tasks so that they have common resources, and do not violate the constraints.
4) Propagate constraints to groups and perform detailed planning within the groups.

First, supposing the independence of elementary tasks, we generate alternative ways for achieving each of them. Then, with a somewhat wider scope, constraints are generated on the feasible combinations of local plan fragments. Here conflicting constraints, if any, have to be resolved by relaxing or removing some of the constraints. After this, when we know already a lot about how the various tasks could be achieved, and what constraints have to be satisfied by the combinations of local solutions, we suspend completely the assumption of independence and try to define groups of tasks that may be strongly interdependent. In resource-rich domains interdependence means first of all the use of *common resources.* Thus the optimization step is based on grouping those tasks that can be performed by operations using common resources. Finally, the groups are ordered and detailed planning is performed within each group.

This structuring of the planning process is not without good reasons: Steps 1) and 2) can both be performed by using local pieces of knowledge that are strongly related to the tasks. E.g., what processes and resource combinations are applicable for machining a particular slot, or measuring its dimensions; how should these operations be ordered relative to other operations that perform some—but definitely few—other tasks. Potential conflicts between the constraints are resolved in a mixed-initiative way: combinatorial algorithms check the satisfiability of constraint sets and suggest what constraints to relax or remove, but the decision is left to the human planner. Once task groups have been established, further planning is quite easy, because (1) groups factorize the search space, and (2) within groups the number of applicable resource alternatives may be greatly reduced, too.

Hence, the computational complexity of planning is concentrated into step 3). This strategy is similar to some recent domain-independent planning approaches that build up the space of disjunctive solutions relatively easily by way of putting the burden of complexity on the process that extracts solutions from this space (Kambhampati 1997). In our case, this solution extraction is directed toward close-to-optimal plans by *task grouping.*

Task Grouping

Task groups have to meet the following requirements:

- Tasks in the same group must be executable by operations that can use the same resources.
- Any grouping must be, just like the plans, complete and consistent with the plan constraints.
- The grouping should be as close to the optimization objectives as possible. Primarily, the plan should have a minimal number of groups. A secondary concern is that the common resources of the groups be as cheap as possible.

Finding the best grouping of tasks is not easy because one has to keep one eye on the resource alternatives, and one eye on the ordering and resource binding constraints. For instance, consider the simple example in Fig. 1. Nodes of the graph are tasks, while arcs show ordering constraints (resource binding constraints are not considered here). Let R(x) denote the set of resource combinations that are applicable by operations aimed at performing a task (ti) or a task group (Gi). Assume that certain tasks have overlapping resource sets; e.g., $R(t1) \cap \ldots R(ti) \ldots \cap R(t6) = R1 \neq \{\}$, and similarly, $R(t7) \cap \ldots R(tj) \ldots \cap R(t14) = R2 \neq \{\}$. From the point of view of resource usage alone the groups G1 and G2 seem to be ideal, but they introduce a cycle in the graph of orderings. Hence, such a grouping is not feasible.

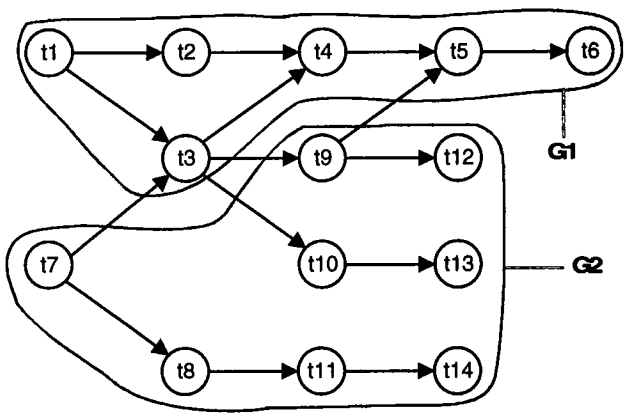

Figure 1. A task grouping that results in ordering conflict between the two groups.

How tasks are grouped has a critical impact both on the feasibility and optimality of plans, and on the efficiency of the planning process. When suggesting a particular grouping one can but anticipate that it will result in a close-to-optimal and consistent plan. The solution of this problem calls for the application of domain knowledge. We claim that such knowledge is available, and it can best be utilized by *case-based reasoning.*

In manufacturing domains planning experts do a kind of task grouping. They often perform so-called *setup planning*: collect elementary manufacturing tasks into groups so that all the tasks can be performed in the same setup and possibly, with the same tool or instrument (Sarma

and Wright 1996, Halevi and Weill 1995, Farago and Curtis 1994). Setup planning is accomplished, however, either before or after sequencing the tasks and the corresponding operations. Since alternative resources and orderings are not considered simultaneously, the search problem is greatly simplified. Of course, this can be done only at the risk of having the plan optimization process halted at some local optimum.

All in all there are a lot of pieces of advice on how to form setups, but if we remove the order and/or resource related restrictions, then the conditions of their applicability become rather blurred, and our decision problems become ill-structured.

Task Grouping by Case-Based Reasoning

For capturing and utilizing essential domain knowledge with somewhat ambiguous application conditions, we attempted to use case-based reasoning.

Cases and their representation

Cases are high-level descriptions of plan sets together with suggestions on how to modify the sets. The modifications are aimed at restricting the set of plans possibly to those plans that are both feasible and close-to optimal. However, it must be emphasized that, contrary to most case-based planning approaches (see Champati et al. 1996, Veloso et al. 1996), cases are definitely not problem statements on the task level with plans as perspective solutions.

The stored cases have a problem description and a solution component:

- The *problem description* is the specification of a search control situation. It is given in terms of a *task set profile*. The profile characterizes a problematic set of tasks: it may refer to the number, the types and relations of tasks, the characteristics of resource sets as well as to the plan constraints.
- The *solution* contains a *suggestion* on how to form groups from the elements of a task set. Actually, this is an advice for the human planner. It has also a machine executable part for implementing the suggestion whenever it is accepted.

For representing the cases, weighted feature vectors with Boolean, string and numeric attributes suffice. Our experience suggests (see below) that different weights for match, mismatch and absence can well be exploited in case retrieval.

Reasoning with cases

Case-based reasoning is applied when there is a set of tasks G that cannot be performed by operations using common resources; i.e., when $R(G)=\{\}$. If this is the situation, groups G1,G2,...Gn are formed from G so that $G=G1 \cup G2 \ldots \cup Gn$. How groups should be formed is suggested by the stored cases. If for any group Gi the resource set R(Gi) is empty, then the process is repeated for this group.

The steps of the reasoning process are the following:

1) If the tasks have some common resource alternatives, then no grouping is needed; return the input set as one group.
2) Otherwise define the actual case by generating the task-set profile.
3) Retrieve stored cases which are similar to the actual case.
4) Present suggestions of the retrieved cases in a ranked list and let the user select from among them. If there is no (good) choice, then go back to 3) and retrieve cases with a looser match.
5) Execute the group forming action that comes with the accepted suggestion.
6) Check the constraints: if the new task grouping introduces any conflict, then go back to step 4).
7) Recursively apply the same procedure to the new task groups.

In step 2) the profile of the actual set of tasks is prepared by rule-based reasoning. Then this actual case is matched against those stored in the case-base. For retrieval *k-nearest neighbour matching* is applied.

What happens in step 5) is a kind of case *adaptation*. In contrast to most case-based applications, adaptation is relatively easy in our framework because it does not need to tailor a previous solution to an actual problem. Instead, following an accepted suggestion, a particular grouping action has to be carried out. Production rules together with some auxiliary procedures can do this perfectly.

However, there is no guarantee that recalling and reusing previous cases always leads to the best solution. Hence, after the iterative, top-down group forming process a bottom-up merging is initiated: we try to recombine the groups so as to form better groupings, if possible. When merging the Gi and Gj task groups, the new resource set, $R(Gi \cup Gj)$ will be $R(Gi) \cap R(Gj)$. The two groups can be merged only if $R(Gi \cup Gj)$ is not empty. Group merging can be performed in an exhaustive way since due to non-overlapping resource sets most of the groups cannot be merged at all.

Application in Inspection Planning

The above ideas have been applied in the development of a knowledge-based inspection planning system (Váncza et al. 1998). The system had to assist robot-based dimensional quality control in ordinary production environments: it had to support the proper and justified selection of inspection resources (including universal robots) and the generation of inspection plans for machined parts with average tolerance and surface roughness requirements.

Inspection can be carried out by using many types of equipment: there is a wide choice of measuring instruments (like calipers, probes or gauges), specialized measuring stations (e.g., cylindricity or roundness testers), and various coordinate measuring machines. Inspection needs also auxiliary equipment for staging and manipulating the part. A planning problem is specified by a set of dimensional control tasks. Thanks to the universal robots and multi-purpose instruments and setups, the tasks can be performed in many alternative ways. Ordering and resource binding

constraints are numerous and often conflicting. The primary optimization objective is to avoid resource changes as far as possible.

The solution of the robotic inspection planning problem required multimodal representation and reasoning:

- Domain objects (part and its features, instruments, setups, auxiliary instruments and robots) and their relations were represented by means of frames.
- Part analysis, determination of elementary inspection tasks, selection of appropriate resource combinations, generation of constraints, as well as detailed planning were all accomplished by rule-based reasoning.
- Geometric reasoning and constraint checking was done by procedures.
- Finally, task grouping was carried out first of all by case-based reasoning.

A simplified case from the case-base of this application is shown below.

```
(DEFSCHEMA CB-PROF-ROT-TOL-ALL
  (INSTANCE-OF TASK-PROFILE)
  (NUMBER-KEY 13)
  ;;TASK-SET PROFILE (matched attributes)
  ;;part-related attributes
  (PART-TYPE ROTATIONAL)
  ;;tolerance type related attributes
  (DIMENSIONAL-TOLERANCE-CONTROL Y)
  (FORM-TOLERANCE-CONTROL Y)
  (RELATIONAL-TOLERANCE-CONTROL Y)
  (ROUGHNESS-CONTROL Y)
  ;;constraints related attributes
  (MAX-LENGTH-OF-TOLERANCE-CHAINS 1)
  ;;resource related attributes
  (COMMON-SETUPS-FOR-PART-ROTATING Y)
  ;;SOLUTION (non-matched attributes)
  (SUGGESTION "Form task groups on the basis of
   tolerance types.")
  (ACTION FORM-GROUPS-BY-TOLERANCE-TYPES)
)
```

Figure 2. A case applied to form task groups in inspection plans (without match, mismatch and absence weights)

The system has been implemented by using the tool-kit of ART-IM™ (Brightware Corp.) which supports all the above technologies. It was tested with models of commercial robots and contact-type measuring instruments. While solving inspection planning problems of average complexity (rotational parts with about 20 dimensional control tasks), typically 10-50 alternative operations were generated per tasks. The partial order of tasks looked typically like the ordering shown in Fig. 1. The space of potential solutions was built up by thousand or so rule firings. In this space case-based reasoning directed the search process toward optimal solutions always in a reliable and efficient manner. Though CBR was not called for many times, together with the preparation and adaptation of cases (including group merging) it consumed about one third of the total computing time.

Discussion

A planning method was outlined that is able to explore alternative ways of achieving elementary tasks and to find close-to-optimal combinations of plan fragments. The solution space is not restricted by a wired-in decision structure or by heuristics. Instead, plans that have to meet global optimization criteria are generated with a global view of the space of potential solutions.

The method has been verified in manufacturing process planning (Horváth et al., 1996, Váncza and Márkus 1996), and later in robotic inspection planning. The two solutions are in one respect very different: while in manufacturing process planning a powerful, but expensive and "blind" genetic algorithm was applied for finding groups of interrelated tasks, in the inspection planning domain case-based reasoning serves this purpose. Great advantage of the latter approach is that it enables the direct application of optimization-related knowledge, and just in that phase of the planning process where such help is mostly needed.

There are already some case-based planners applied in manufacturing domains (Champati et al., 1996; Muñoz-Avila and Weberskirch, 1996), but these planners do not take much care just of the abundance of potential resources. Hence, they can hardly serve optimization purposes. Another problem with case-based planners is that they can be inhibited whenever too complex adaptation tasks have to be resolved. In the worst case, a planner can save nothing by starting from a previous solution instead of planning from scratch. At least, this is known within the classical framework of planning (Nebel and Koehler 1995). However, these results are relevant also in our case, since, as we have shown in (Kis and Váncza 1996), our manufacturing planning problem can also be formulated within the classical model. All in all, though our system uses cases while generating plans, it has not much in common with case-based planners. The planning method is most similar to that of the CABINS system (Miyashita and Sycara 1995). CABINS works in the domain of job shop scheduling and suggests repair strategies and tactics for improving schedules by case-based reasoning.

In our framework case-based reasoning provides a technology for injecting domain specific knowledge into the search control of the planning process, and supports also, via its soft matching mechanism, the application of this knowledge under somewhat blurred conditions. On the other hand, using CBR in search control instead of directly in planning is also advantageous, since cases with simple structure are suitable, the solution quality and efficiency of the planning process can be enhanced by using relatively small case-bases, and case retrieval and adaptation can be fully automated.

In conclusion, it must be emphasized that case-based reasoning is embedded in a complex planning framework: the power of the approach seems to be just in the integrated use of various knowledge sources and reasoning methods.

Acknowledgments This work has been supported by grant T023305 of the National Research Foundation of Hungary (OTKA) and the IC15-CT96-0708 (MINOS) project. The author is indebted to A. Márkus for many discussions and suggestions.

References

Barrett, A., and Weld, D. S. 1994. Partial-Order Planning: Evaluating Possible Efficiency Gains. *Artificial Intelligence* 67:71-112.

Champati, S., Lu, W. F., and Lin, A. C. 1996. Automated Operation Sequencing in Intelligent Process Planning: A Case-Based Approach. *Int. J. of Advanced Manufacturing Technology* 12:21-36.

Farago, F.T., and Curtis, M.A. 1994. *Handbook of Dimensional Measurement.* Industrial Press Inc.

Halevi, G., and Weill, R.D. 1995. *Principles of Process Planning.* London: Chapman & Hall.

Horváth, M., Márkus, A., and Váncza, J. 1996. Process Planning with Genetic Algorithms on Results of Knowledge-Based Reasoning. *Int. J. of Computer Integrated Manufacturing* 9(2):145-166.

Kambhampati, S. 1997. Refinement Planning as a Unified Framework for Plan Synthesis. *AI Magazine* Summer 1997:67-97.

Kis, T., and Váncza, J. 1996. Computational Complexity of Manufacturing Process Planning. In Ghallab, M. and Milani, A. (Eds.), *New Directions in AI Planning*, IOS Press.

Miyashita K., and Sycara, K. 1995. CABINS: A Framework of Knowledge Acquisition and Iterative Revision for Schedule Improvement and Reactive Repair. *Artificial Intelligence* 76:377-426.

Muñoz-Avila, H. and Weberskirch, F. 1996. Planning for Manufacturing Workpieces by Storing, Indexing and Replaying Planning Decisions. *Proc. of the 3rd Int. Conf. on AI Planning System.*

Nebel, B. and Koehler, J. 1995. Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis. *Artificial Intelligence* 76:427-454.

Sarma, S.E. and Wright, P.J. 1996. Algorithms for the Minimization of Setups and Tool Changes in "Simply Fixturable" Components in Milling. *J. of Manufacturing Systems* 15(2):95-112.

Váncza, J., and Márkus, A. 1996. Experiments with the Integration of Reasoning, Optimization and Generalization in Process Planning. *Advances in Engineering Software* 25(1):29-39.

Váncza, J., Horváth, M., and Stankóczi, Z. 1998. Robotic Inspection Plan Optimization by Case-Based Reasoning. *Journal of Intelligent Manufacturing*. In print.

Veloso, M.M., Muñoz-Avila, H., and Bergmann, R. 1996. Case-Based Planning: Selected Methods and Systems. *AI Communications* 9(3):128-137.

Appendix

1. Integration Category:
Rule-based reasoning and CBR.

2. Performance Task:
Planning, especially in resource-rich domains. Such planning problems are typical in manufacturing, like computer-aided process planning (CAPP), inspection planning.

3. Integration Objective:
Efficiency gain in CPU and session time; solution quality in terms of the value of the generated plans; cognitive fidelity.

4. Reasoning Components:
CBR, RBR, procedural reasoning. Rule-based reasoning sets the stage for CBR and executes what is suggested by CBR; in particular it 1) analyzes the planning problem at hand, 2) builds up a disjunctive representation of alternative plans (an implicit search space), 3) executes plan modifications suggested by CBR. CBR gives advice on how to extract good enough plans from this space. Procedures are needed for geometric reasoning.

5. Control Architecture:
Interleaved. The planning process is segmented to three different phases; CBR is master in the second phase (solution extraction), otherwise RBR dictates.

6. CBR Cycle Steps Supported:
Pre-processing, retrieval and reuse. RBR supports both pre-processing and fully automatic reuse.

7. Representations:
The primary representation is plans given in terms of actions and constraints. Though, there are frames for representing complex domain objects, rules for capturing technological domain knowledge, and also procedures for dealing with geometrical information. Cases represent optimization-related domain knowledge.

8. Additional Components:
User provides input in the course of the case-based reasoning: actually, CBR cyclically offers a ranked list of plan modification advice, the user selects one, and finally RBR implements it.

9. Integration Status:
Under empirical evaluation. Tests are made with real-life inspection planning problems, with detailed models of objects involved. Experiments are carried out with different parts and resource sets (instruments, fixtures, etc.). These can be regarded as independent variables. As for the solutions, not only their value, but also the path to them is concerned. Evaluation is supported by 3D simulation.

10. Priority Future Work:
The addition of learning features, theoretical analysis and generalization.