

Using Goal Interactions to Guide Planning

Caroline Hayes

Robotics Institute, Carnegie Mellon University
Pittsburgh, Pa.

Abstract

The Machinist program extends domain dependent planning technology. It is modeled after the behavior of human machinists, and makes plans for fabricating metal parts using machine tools. Many existing planning programs rely on a problem solving strategy that involves fixing problems in plans only after they occur. The result is that planning time may be wasted when a bad plan is unnecessarily generated and must be thrown out or modified. The machinist program improves on these methods by looking for cues in the problem specification that may indicate potential difficulties or conflicting goal interactions, before generating any plans. It plans around those difficulties, greatly increasing the probability of producing a good plan on the first try. Planning efficiency is greatly increased when false starts can be eliminated. The machinist program contains about 180 OPS5 rules, and has been judged by experienced machinists to make plans that, are on the average, better than those of a 5 year journeyman. The knowledge that makes the technique effective is domain dependent, but the technique itself can be used in other domains.¹

I. Introduction

Machinist is a planning program that works on machining problems, and produces feasible plans for manufacturing individual metal parts. **Machining** is the art of producing metal parts using a variety of power tools to shape the metal. It is a highly skilled task requiring 10 to 15 years to become fairly accomplished.

The program works by first scanning the problem specification (a set of shapes to be cut in a metal block, and some information on raw material, dimensions, etc.) for cues or patterns that indicate potential problems. It also looks for other types of patterns that provide salient information: what set of tools and processes can be used for specific cuts, as well as information on the details and restrictions on those processes. Using this information as the building blocks, the program constructs a plan for

producing the part.

This approach is more efficient than traditional planning methods, for domains that have many interactions between the goals. Traditional planners typically work by first generating a plan then using "critics" to check the resulting plan for problems and correct them [Sussman 75, Scacerdoti 75]. The critic method uses much more time in generating and fixing bad plans.

The ideas for **Machinist's** planning technique are taken from observations of the behavior of human machinists. Protocol analysis was used to collect this information. The resulting program consists of about 180 OPS5 rules, and it runs on a DEC-20, a UNIX VAX, and a SUN workstation.

The main emphasis of this paper is to explain the program's planning methods and to examine how these methods can be used in other domains. The way in which this planning technique is implemented is domain dependent: the ability to identify a goal interaction efficiently by looking at a problem specification requires intimate knowledge about that problem domain. This knowledge, in the form of patterns which identify interactions, together with operators that tell how to avoid the interactions, takes many years for the expert to build up and years for the knowledge engineer to extract. As used here, a pattern together with an associated composite operator will be referred to a *macro-operator*. Unfortunately, the planner *must* have these macro-operators to find these interactions in complex domains, otherwise the search would be tremendous. This does not lend hope for domain independent planners to be successful in large domains, but perhaps we must reconcile ourselves to the fact that efficiency may require expertise [Sussman 75].

II. Interactions Make Planning Difficult

A major problem that the machinist confronts in planning is interactions between the different features that are cut into the part. Cutting one feature first may make it difficult or impossible to cut subsequent ones. One can view the collection of features as subgoals to be achieved in the machining plan. The difficulty in making a plan is finding an order in which none of the subgoals interferes too seriously with achieving the others.

This type of problem is not isolated to the

¹This research was sponsored by Cincinnati Milacron and Chrysler

machining domain; interactions between subgoals have been observed in many planning domains by many researchers: Stefik [Stefik 81], Hammond [Hammond 86], Sussman [Sussman 75], Tate [Tate 76], and Carbonell [Carbonell 81] to name a few. Sussman noted it as early as 1973 in HACKER: "interactions between steps, (are) a common cause of bugs."² Stefik perhaps, expressed it best: "In planning problems, there are typically many goals to be achieved in some order. The goals interact with each other in many ways which depend both on the order in which they are achieved and on the particular operators which are used to achieve them."³ A *feature interaction* happens when cutting one collection of features affects the way in which others can be made. Cutting one set of features may make it difficult to make other features later on in the process. The methods used to make those subsequent features may have to be changed, or all the steps in the plan may have to be reordered so it is possible to cut all of the features.

Feature interactions have several different causes. Most commonly they result from clamping problems; producing one feature destroys the clamping surfaces needed to grip the piece while cutting another feature.

A feature interaction is shown below in figure 1. This part has two features: an angle, and a hole. The angle has been cut and the hole is about to be drilled, but when the drill touches the angled surface it will slip sideways and cause the hole to be placed inaccurately. The angle can be said to interact with the hole. The solution is to drill the hole first while the end of the part is still flat. Since the hole does not affect how the angle is made, a simple reordering prevents the features from interacting.

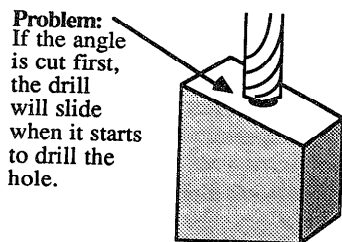


Figure 1: *Feature interaction*: the hole must be cut before the angle

III. How the Program Works

The Machinist program is modeled after the human's planning process but it only implements a part of that

²Gerald J. Sussman, *A Computer Model of Skill Acquisition*, American Elsevier Publishing Company, New York, 1975, MIT AI Technical Report TR-297, August 1973, p. 119

³Mark Stefik, "Planning and Meta-Planning (MOLGEN: Part 2)," *Artificial Intelligence*, vol 16, no. 2., 1981, p. 141.

process. The human's planning process is described in [Hayes 87]. The most important omission is that there is no verification phase at the end of the program's planning.

To demonstrate how the program works, let us suppose one wanted to make the part in figure 2 from the metal stock shown in figure 3. There are five features that need to be cut into this part: three holes, an angle, and a shoulder (a shoulder is any ledge-like shape cut out of a side). The part is represented in the program as a rectangular block from which features are subtracted. The block of metal that it will be made from, the *stock*, is saw cut and irregular on all sides.

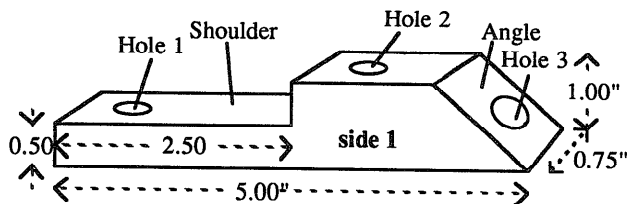


Figure 2: A part with 5 features: three holes, a shoulder, and an angle

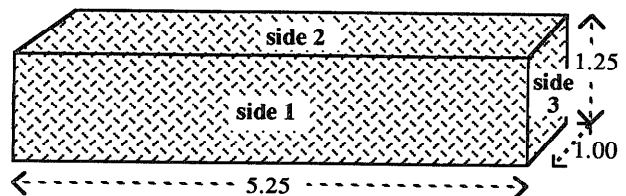


Figure 3: The stock from which the part will be made: saw cut on all sides

The first task to be done is that the program must identify the problems and interactions that occur in the part. This gets the program oriented to the basic structure and difficulties of the problem. Macro-operators are used to identify the interactions and produce the corresponding restrictions that they cause.

In this part there are three interactions. The first is between Hole 3 and the angle. If the angle is made first it will interact with the hole, by causing the drill bit to slip on the slanted surface. This will make the hole placement inaccurate, as shown in the previous section II. The restriction that this interaction puts on the plan is that Hole 3 must be made before the angle.

The second interaction is between Hole 3 and the shoulder: the hole *must* be made before the shoulder. If the shoulder is made first, the part will be too thin and floppy when it is clamped to cut the hole. The result of the third interaction is that the angle must be made before the shoulder, for similar reasons.

These three interactions: Hole 3 before Angle, Hole 3 before Shoulder, Angle before Shoulder, all restrict the order in which the features can be cut. They can be put

together into one *interaction graph* (shown in figure 4). Each arrow represents one interaction.

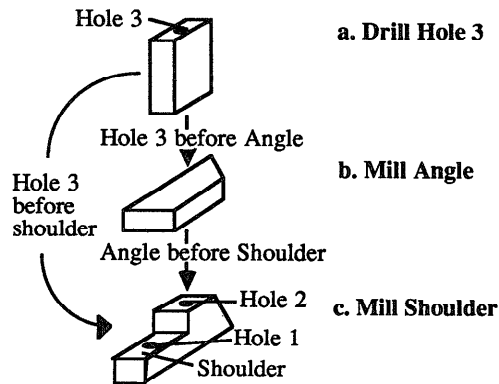


Figure 4: Interaction Graph the order in which the features may be cut

The next task is to retrieve a *squaring graph* from memory. A squaring graph outlines all methods for getting the raw material into a square and accurate shape with the minimum waste of material. It represents the constraints on the order in which each of the sides may be "squared off." It serves as a framework from which the feature constraints can be hung.

The squaring graph for this example is shown in figure 5. In each step, the shaded surfaces will be machined smooth. Steps that are shown side by side as branches in the graph can be done in either order: it does not matter which side of a branch is done first.

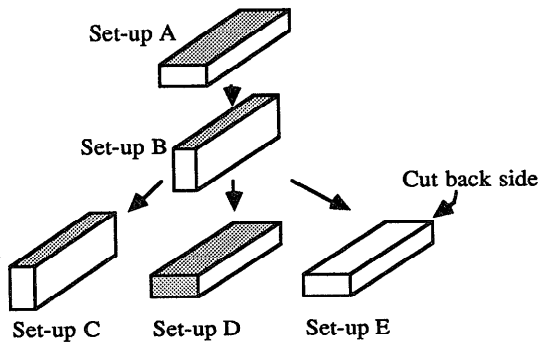


Figure 5: The Squaring Graph for squaring up a block that is sawn on all sides

We now have a graph showing the orders in which the features can be produced, and a graph showing the orders in which the sides may be cut. Each graph represents a separate set of constraints on the plan. The two must be merged with as much overlap between the steps as possible, so that we get a compact sequence. The more overlap the better, because the plan will be more concise. The merging of the two graphs is shown in figure 6.

Observe that between the Interaction Graph and the Squaring Graph there are 8 steps, but in the final plan there

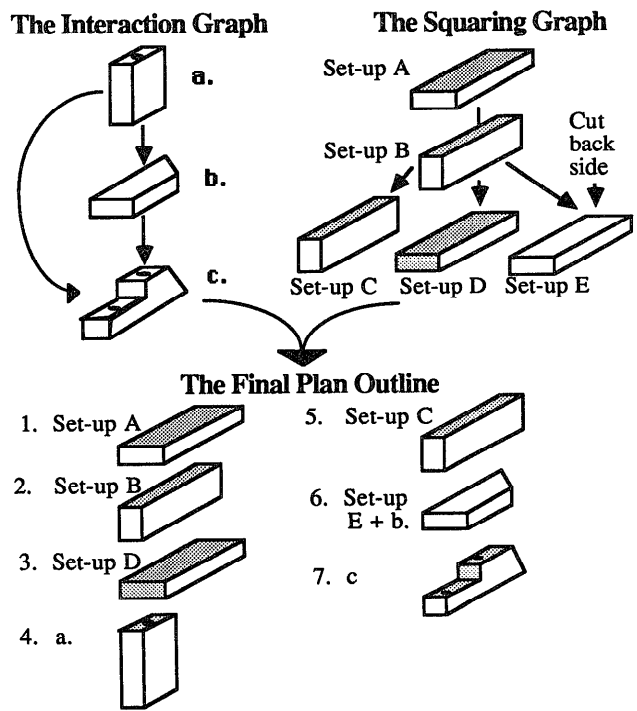


Figure 6: Merging the Interaction Graph with the Squaring Graph

are only 7. This is because we were able to combine step b from the Interaction Graph with Set-up E from the Squaring Graph. The details on the processes by which squaring plans are chosen and the two graphs are merged is described in [Hayes 87].

After producing the plan, the program does not go through the final verification phase as the human does. If all problems and goal interactions have been properly identified, the plan will be correct and the verification step unnecessary.

However, the program would obviously be more robust if it used a verification step as the human does. It is not always possible to identify all problems beforehand: neither the machinist nor the program can have a complete set of patterns to identify absolutely all possible problems and goal conflicts. Therefore, the plans produced will not always be good the first time: there needs to be some sort of a safety net to catch problems that initially escape notice. Human machinists also use a "critic" approach, to check the final plan for errors. They may reorder steps, or replan to fix them. Future versions of the *Machinist* program will also be able to do this.

VI. Discussion

Out of the 180 productions that comprise this system: 10 productions identify feature interactions and construct the feature interaction graph, 39 identify other problems and generate constraints not caused by interactions, 13 choose the squaring graph, 44 merge the interaction graph with the

squaring graph, 11 generate the final plan from the merged constraint graphs, and 63 enter and check data, infer missing data, group features, push and pop goals, etc. The first two categories which identify interactions and generate constraints, are the ones that have the most room to grow. Productions can be added to these two categories, greatly increasing the range of parts that the system can handle, while the rest of the system remains the same.

How much do the heuristics implemented by these rules cut down the search space? There are several categories of heuristics used by the program: feature interactions, squaring graphs, and graph merging. If the total effect of all the heuristics on the example used in this paper is taken together, we find that they reduce the number of plans that must be examined by a minimum factor of 1,663,200 compared to search using no heuristics.

Let us now consider only the feature interaction heuristic by itself. For the example part there are 5 features but only 3 interactions. For this case, the feature interaction heuristic alone cuts down the number of plans that must be examined by a factor of 10. If we look at a more complicated example taken from [Hayes 87] that has 14 features and 5 interactions, the feature interaction heuristic cuts down the number of plans examined by a factor of 630,000.

Essentially, the more features and the more interactions there are, the more difficult it is to find a good plan. The problem is not that the search space gets larger as more interactions are added, it is that the density of good solutions in that space goes down. The machinist's knowledge of feature interaction helps him to zero-in on only those good solutions.

V. Comparison of the Program to Human Performance

The program was tested against four machinists at various experience levels: two second year apprentices, one third year apprentice, and one journeyman with 5 years experience including the apprenticeship. Each of these subjects was asked to create a machining plan for the same series of three parts. Each part was apparently simple, but contained difficulties when examined more closely.

Their resulting plans were judged by two very experienced machinists, each having more than 15 years experience. The average rating given to each of the four subjects and the program are shown in figure 7. The program's average performance was better than that of the apprentices or the journeyman. In fact, Machinist 1 declared the program's plan for Part III to be "Almost the perfect plan. Who ever did this is a man after my own heart."

The judging was done in the following way: for each of the three parts there were five plans generated, one from each of the four young machinists, and one from the program. All information indicating who (or what) created the plan was removed, and the plans were presented to the two experienced machinists. Independently, they

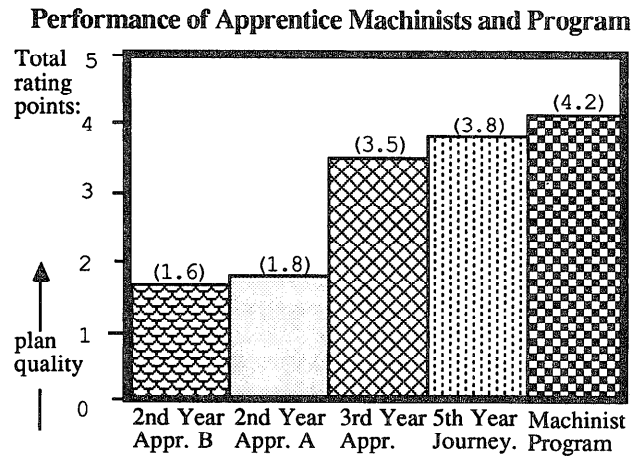


Figure 7: Average Plan Rating for Each Subject

ordered each set of five plans, rating them from best to worst. The best plans were given a score of 5, and the worst, 1.

The machinists' ratings agreed exactly for 8 of the plans, differed by 1 point for 3, and more than one for 4. However, neither machinist felt that the other was wrong in his ratings. Both felt that the plans which they rated differently were actually very close in quality.

VI. Previous Work

Many pieces of this planning process have been described before but not as one cohesive method. Virtually all of the planners referenced in this paper recognize the importance of goal interactions in planning, but their method of dealing with this problem is different than Machinist's. Typically they do not foresee problems in the problem specification and avoid them. Instead they make plans with mistakes in them and use critics to recognize and correct them after the fact [Sussman 75, Scacerdoti 75]. Time is wasted fixing and replanning.

TWEAK [Chapman 85], and GARI [Descotte 81] both work by successively adding constraints to the description of the solution. The interaction and squaring graphs used by Machinist are also constraints, but Machinist's advance over this approach is to obtain the constraints as the result of feature interactions.

A number of chess strategy planners use macro-operators. They use patterns associated with plans to make search more efficient. Interestingly, many of them have been modeled, at least indirectly, from human behavior. Berliner and Campbell [Berliner 83], and De Groot [De Groot 65] all use some variant of this method but none of them seem to consider the effect of goal interactions on planning.

There are only a few programs that take goal interactions into account before attempting a plan. One of the earliest, Tate's [Tate 76] planner for house construction, does take interactions into account before it makes a plan.

However, these interactions must be entered by a human since the planner itself cannot determine what tasks interact. A new set of interactions must be entered for each new task. This type of solution is not practical for machining problems since each new problem contains a different set of interactions. One cannot reuse the same set of interactions over and over again for a large class of problems.

Wilenski's planner, PANDORA [Wilensky 80], and to some extent Wilkins planner, SIPE [Wilkins 84] specifically look for goal interactions before planning (which is a great advance in domain independent planning). However, since it is domain independent, it can not make use of domain knowledge (in the form of patterns) to help identify goal interactions quickly and to find a way around them. Consequently, its performance on complex tasks such machining problems would be impractically slow.

Chef [Hammond 86] is a planner that generates recipes for Chinese cooking. It is one of the few planners that looks at the problem description for cues to potential problems and interactions. However, it does not use the interaction information to generate the plan as *Machinist* does but only to retrieve and modify plans. This is a good approach for many problems but it will not do for machining. Small differences in the shape or size of a part may make big differences in the plan so it is not good enough to index a past plan for a part that looks similar, and modify it. The plans may have so little similarity that it is easier to construct a new plan from scratch.

VII. Conclusion

The difference between *Machinist* and other planners is that it has *all* of the following properties *together*:

1. a pre-planning step in which it scans the problem specification for signs of possible goal interactions,
2. macro-operators to identify goal interactions, and to suggest ways to restrict the plan so as to avoid them,
3. a plan constructed from collected information rather than a plan that is indexed from memory and modified.

In particular, the pre-planning identification of problem areas can greatly increase planning efficiency within a particular domain. The macro-operators that identify problem areas and suggest solutions are the key to planning efficiency. The set of operators used must be domain dependent, but the general strategy can be applied to other domains.

Acknowledgements

I would like to extend a special thanks to Jim Dillinger, Dan McKeel, Ken Pander, Steve Klim, Dave Belotti, and

Phyllis Huckestein for contributing their machining expertise to this project; to Paul Wright, Jaime Carbonell, Herb Simon, Irene Skupneiwicz, Paul Englert, Brack Hazen, Gregg Lebovitz, Barbara Wright, Mark Perlin and Mike Parzen for their advice, comments, and proof reading; and to Ken Mohnkern for the art work. This research was funded by Cincinnati Milacron and Chrysler.

References

- [Berliner 83] Berliner, Hans; Murray Campbell. *Using Chunking to Solve Chess Pawn Endgames*. Technical Report CMU-CS-83-122, Carnegie Mellon University, April, 1983.
- [Carbonell 81] Carbonell, Jaime G. *Subjective Understanding, Computer Models of Belief Systems*. UMI Research Press, Ann Arbor, Michigan, 1981. PHD Thesis, Yale University, 1979.
- [Chapman 85] Chapman, David. *Planning for Conjunctive Goals*. Technical Report 802, Massachusetts Institute of Technology, May, 1985.
- [De Groot 65] De Groot, A. D. *Thought and Choice in Chess*. Mouton & Co., The Hague, Netherlands, 1965.
- [Descotte 81] Descotte, Y., J. Latombe. GARI: A Problem Solver that Plans how to Machine Mechanical Parts. *Proceedings of IJCAI* :766-772, 1981.
- [Hammond 86] Hammond, Kristian J. CHEF: A Model for Case-Based Planning. *AAAI-86* :267-271, 1986.
- [Hayes 87] Hayes, Caroline C. Planning in the Machining Domain: Using Goal Interactions to Guide Search. Master's thesis, Mellon College of Science, Carnegie Mellon University, April, 1987.
- [Scacerdoti 75] Scacerdoti, Earl D. The Nonlinear Nature of Plans. *IJCAI4* :206-214, 1975.
- [Stefik 81] Stefik, Mark. Planning and Meta-Planning (MOLGEN: Part 2). *Artificial Intelligence* 16(2):141-170, 1981.
- [Sussman 75] Sussman, Gerald J. *A Computer Model of Skill Acquisition*. American Elsevier Publishing Company, New York, 1975. MIT AI Technical Report TR-297, August 1973.
- [Tate 76] Tate, Austin. *Project Planning Using a Hierarchic Non-Linear Planner*. Technical Report D.A.I. Research Report No. 25, University of Edinburgh, University of Edinburgh, August, 1976.
- [Wilensky 80] Wilensky, Robert. Meta-planning. *AAAI* :334-336, 1980.
- [Wilkins 84] Wilkins, David E. Domain-independent Planning: Representation and Plan Generation. *Artificial Intelligence* :269-301, 1984.