In Working Notes of the AAAI 1994 Fall Symposium Series, Symposium on Planning and Learning: On to Real Applications, November 1994, New Orleans, Louisiana

Applying a General-Purpose Planning and Learning Architecture to Process Planning *

Yolanda Gil
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
gil@isi.edu

M. Alicia Pérez
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
aperez@cs.cmu.edu

Abstract

Process planning poses significant computational requirements due to the variety of alternative processes, their complexity, and their interactions. General-purpose planners are generally not considered a practical approach, and most current research focuses on special-purpose planning systems. Research within the PRODIGY framework aims to provide expressive general-purpose planners together with learning algorithms that can improve their efficiency, the accuracy of their domain model, and the quality of their plans. Process planning is one of the large-scale complex domains that we have implemented in PRODIGY to demonstrate the feasibility of our approach. Our current model of process planning is still far from comprehensive and is limited in many ways, but it reflects many of the complexities involved in the task. This paper describes how PRODIGY learns control knowledge, acquires domain knowledge, and improves the quality of its plans for this application domain using general-purpose planning and learning algorithms.

Introduction

Current research on automation of manufacturing processes includes CAD aids, assembly automation, and process planning tools. The automation of process planning in particular is becoming a serious need in industry, due to the increasing scarcity of experts on technology that is rapidly changing, the need for lowering manufacturing costs, and the desire to make customized products widely available. The variety of alternative processes, their complexity, and their interactions make the planning task very complex. In addition, a good process plan minimizes resource consumption and execution time. These issues are part of the research agenda of the AI planning community. However, the automation of different aspects of process planning (see (Chang & Wysk 1985) for an overview) has focused on special-purpose systems that address the complexity of the task with mechanisms specific to process planning. Some of these systems use AI techniques (Hayes 1990; Descotte & Latombe 1985a; Nau 1987), and there are approaches that use general-purpose problem solvers coupled with special-purpose systems (Kambhampati *et al.* 1993). All this body of work suggests that general-purpose AI planning techniques cannot handle the complexity inherent to process planning tasks.

Research within the PRODIGY framework aims to provide expressive general-purpose planners together with learning algorithms that can improve their efficiency, the accuracy of their domain model, and the quality of their plans. Process planning is one of the large-scale complex domains that we have implemented in PRODIGY as a useful testbed for our planning and learning research.

Our current model of process planning[1] represents machining, joining, and finishing operations. Although this model is still far from comprehensive and is limited in many ways, it reflects many of the complexities involved in the task. We asked an expert job shop machinist to assist in the construction of the domain so it would be as accurate as possible. The machinist also helped with the description of a real machine shop and sample parts for constructing problems. For some problems we used actual requests that were submitted to the job shop that serves the Mechanical Engineering Department of Carnegie Mellon University.

The paper begins with a brief overview of PRODIGY, followed by a presentation of our model of process planning. Finally, we describe how the learning mechanisms applied to this process planning domain improve PRODIGY's performance in several respects.

Planning and Learning in PRODIGY

The PRODIGY system (Minton *et al.* 1989a; Veloso 1989; Carbonell *et al.* 1992) is an evolving general-purpose problem solving architecture that integrates several learning mechanisms to improve performance. Domain knowledge is represented in a set of operators and inference rules, and

* We would like to thank all the members of the PRODIGY group for many years of collaborations and discussions. This research was partially sponsored by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant numbers F33615-90-C-1465 and F33615-93-1-1330. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of Wright Laboratory or the United States Government. The second author holds a scholarship from the Ministerio de Educación y Ciencia of Spain.

[1] The domain is described in detail in (Gil 1991), and is available upon request from prodigy@cs.cmu.edu.

a type hierarchy for the objects in the domain. The operators are models of the available actions and they specify the effects of the actions under different conditions. Inference rules are used to deduce additional information from the state. A problem is given by an internal state, representing the current state of the world, and a goal state. PRODIGY searches for a solution using a casual commitment strategy for every decision in the search process. Decisions include choosing a goal, choosing an operator, selecting bindings to instantiate an operator, and deciding whether to subgoal or apply an operator whose conditions are satisfied. Search control rules that express definitive selections or heuristic recommendations are applied at each decision point. The problem solver has a very powerful language to express both domain and control knowledge.

Learning Control Knowledge

Control rules can be learned automatically by the system by static analysis of the domain operators (Etzioni 1990), analysis of problem-solving traces (Minton 1988; Borrajo & Veloso 1994), or a combination of both (Pérez & Etzioni 1992).

In addition to learning control rules, PRODIGY can also control the search using derivational analogy with similar previously solved problems (Veloso & Carbonell 1993). Search is also more efficient when PRODIGY is used as a hierarchical problem solver that learns to structure the search in multiple abstraction levels automatically (Knoblock 1991).

Learning to Improve Plan Quality

PRODIGY can learn to improve the quality of the plans it generates (Pérez & Carbonell 1994). Given a domain-dependent objective function that can evaluate the quality of plans, the learning algorithm compares the search trace for the planner solution given the current control knowledge, and another search trace corresponding to a better solution (*better* according to the evaluation function). The latter trace is obtained by letting the problem solver search further until a better solution is found, or by asking a human expert for modifications on the first solution or a completely new one, and then building a corresponding search trace. The algorithm explains why one solution is better than the other and its *output* is search control knowledge that leads future problem solving towards better quality plans. The learning algorithm is effectively operationalizing the objective function into knowledge that the planner can use during plan generation by transforming it into control rules.

Learning Domain Knowledge

PRODIGY can acquire new domain knowledge by interaction with the environment and experimentation (Gil 1992). Given an initial description of the domain operators, PRODIGY can acquire additional preconditions and effects autonomously by executing the plans that it builds with the currently available knowledge. The system has expectations that emerge from its current knowledge. Plan execution is monitored, and when the expectations and the observations diverge, learning is triggered. When there are several possible modifications of the domain knowledge that could potentially fix the problem, PRODIGY designs and executes experiments to discern which modification is appropriate. The experimentation process is efficiently carried out with to a set of domain-independent hypothesis-selection heuristics that are available to the learning system.

Domain knowledge can also be acquired directly from a domain expert. PRODIGY can engage in an apprentice-like dialogue (Joseph 1992), or learn from observing the expert solving problems (Wang 1994).

Process Planning in PRODIGY

In this domain, PRODIGY generates plans to produce parts given a request that specifies the material, the shape (rectangular or cylindrical), the size along each dimension, the surface quality (roughness), the surface finish (metal coatings and polishing), and the features (holes that can be reamed, tapped, counterbored, etc). This specification forms the goal state. A description of a shop with machines, tools, and parts forms the initial state of any problem. Parts have six sides, and the location of a feature is determined with x and y coordinates in a given side. Besides the machining operations themselves, a plan consists of operations to secure the part with a holding device in a certain orientation, to clean metal burrs from its surface, and to install an appropriate tool in the machine.

Domain Knowledge

In our model, most operators correspond to machining, joining, and finishing actions, as well as to the steps to prepare the part and tool set-ups. Consider, for example, an operator for face milling a part. We need to represent the fact that if we use a milling cutter on a milling machine the size of the part will change along a dimension corresponding to the part side facing up, that the part must be held by a holding device in such a way that the desired dimension can be machined, and that the new size of the part must be smaller than the current size. Also, any surface properties of the side being machined will disappear, and the part will have dirt and burrs. Figure 1 shows the corresponding operator. The notation means that if the preconditions are true in the current state then we can perform the milling operation, which changes the state according to the effects listed.

The domain implementation makes use of PRODIGY's ability to represent infinite types and to do arbitrary Lisp function calls. Infinite types, i.e. types with infinitely many instances, are used to represent numeric quantities, such as part sizes, hole depths, diameters, and angles. Functions can be used to denote facts that never change in the state, as generators for the infinite types, and to perform numeric calculations. In the FACE-MILL operator, the function `smaller` represents the restriction that the part size never increases after milling. Inference rules are used to specify the availability of machines, parts, tools, tool holders, and holding devices. They are also used to determine which sides should be used to hold a part.

```
(Operator FACE-MILL
 (params <machine> <part> <cutter> <hold-dev>
        <side> <side-pair> <dim> <value-old> <value>)
 (preconds
   ((<machine> MILLING-MACHINE)  (<cutter> MILLING-CUTTER)
    (<hold-dev> (or 4-JAW-CHUCK VISE COLLET-CHUCK TOE-CLAMP))
    (<part> Part)               (<dim> Dimension)
    (<side> Side)               (<side-pair> Side-Pair)
    (<value-old>
     (and Size (gen-from-pred (size-of <part> <dim> <value-old>))))
    (<value> (and Size (smaller <value> <value-old>))))
  (and (shape-of <part> RECTANGULAR)
       (side-up-for-machining <dim> <side>)
       (sides-for-holding-device <side> <side-pair>)
       (holding-tool <machine> <cutter>)
       (holding <machine> <hold-dev> <part> <side> <side-pair>)))
 (effects ((<surface-coating> SURFACE-COATING)
           (<surface-finish> SURFACE-FINISH))
  ((del (is-clean <part>))       (add (has-burrs <part>))
   (del (surface-coating-side <part> <side> <surface-coating>))
   (del (surface-finish-side <part> <side> <surface-finish>))
   (add (surface-finish-side <part> <side> ROUGH-MILL))
   (add (size-of <part> <dim> <value>))
   (del (size-of <part> <dim> <value-old>)))))
```

Figure 1: The FACE-MILL operator.

Some qualitative and quantitative measures of the complexity of this domain are:

- The effects of most operators are not reversible.
- The precondition expression of some operators and inference rules includes negations, disjunctions and universal quantification. Some of the preconditions correspond to predicates derived by inference rules.
- There are context-dependent effects of operators.
- There are 117 rules, that include 73 operators and 44 inference rules. 38 of the operators correspond to machining operations, and 35 to set-ups.
- The average number of parameters for an operator is 7, the average number of preconditions is 5, and the average number of effects is 3.
- There are 41 different predicates. 7 of them are static (i.e., do not change during problem solving). 11 Lisp functions are used to perform numerical computations and constrain variable values.
- There are 85 different types and subtypes of objects in the type hierarchy, 5 of which are infinite types.
- The length of many solutions is over one hundred rules (including operators and inference rules).
- The initial state that represents the machine shop includes more than 500 facts.

Control Knowledge for Process Planning

Learned or handwritten control rules guide the search for solutions along the more promising paths. For example, the rule in Figure 2 rejects certain kinds of cutting fluid for some machining operators according to the material of the part.

Plan Quality in Process Planning

Plan quality is crucial in process planning to minimize both resource consumption and execution time (Doyle 1969; Descotte & Latombe 1985a). For instance: it may be advantageous to execute several cuts on the same machine with the same fixing to reduce the time spent setting up the work on the machines; or, if a hole H_1 opens into another hole H_2, then H_2 should be machined before H_1 in order to avoid the risk of damaging the drill.

```
(control-rule DONT-USE-MINERAL-OIL
  (if (and (current-goal-first-arg <part>)
           (current-ops (DRILL-WITH-HIGH-HELIX-DRILL
                         DRILL-WITH-GUN-DRILL REAM
                         ROUGH-GRIND FINISH-GRIND
                         CUT-WITH-CIRCULAR-FRICTION-SAW ...))
           (or (known (material-of <part> STEEL))
               (known (material-of <part> ALUMINUM)))
           (type-of-object <f> mineral-oil)))
  (then reject bindings ((<fluid> . <f>))))
```

Figure 2: Control rule that rejects bindings for the cutting fluid depending on the part's material.

Sharing parts of the set-ups among operations on one or more parts usually reduces the total plan cost. Plan length is usually not an accurate metric of plan quality, as different operators have different costs. For example, a tool can be switched automatically but holding the part requires human assistance (Hayes 1990). Therefore plans that share set-ups are cheaper than plans that share tools. The next section describes how quality-enhancing control knowledge can be acquired automatically.

Learning to Improve Performance in Process Planning

This section describes how PRODIGY's learning techniques described in the second section can be used to improve the planner's performance in our process planning domain.

Efficient Process Planning through Learning

There are two main approaches to building process planning systems (Chang & Wysk 1985). *Generative* approaches combine elementary process planning operations to produce the final plan. *Variant* approaches retrieve complete plans from a plan library and adapt them to suit the needs of the current problem. In the implementation just described, PRODIGY finds solutions for process planning problems in a generative fashion, i.e., by constructing plans given a set of possible operators. PRODIGY's analogical engine (Veloso & Carbonell 1993) could be used to implement a variant approach using predefined planning episodes associated with families of parts, modifying them for the particular part wanted.

Abstraction planning has been applied to process planning and scheduling domains successfully (Fox & Smith 1984; Nau 1987). PRODIGY's domain-independent techniques (Knoblock 1991) should provide useful abstractions to handle the interactions within subproblems in a process planning application.

PRODIGY currently uses many control rules to guide the search in the process planning domain. These control rules are hand-coded, and continue to grow in number as we continue to understand how to control the search complexity of the domain. Some of the work on automatically learning control knowledge in PRODIGY's has been applied to the process planning domain (Borrajo & Veloso 1994).

Learning to Generate Process Plans of Good Quality

The performance of the planner can also be improved by learning new rules to guide the search towards better quality solutions. The mechanism to learn quality-enhancing control knowledge described previously has been applied to the process planning domain. The following simple example illustrates the learning process. The domain-dependent quality metric used is additive on the cost of the individual operators and the operations to set-up the part on the machine are more expensive than those to switch the tool.

Suppose the goal is to reduce the height of a part and have a spot hole at certain coordinates, and the planner chooses the drill press to drill the spot hole. A domain expert may input modifications to improve that solution so that it uses the milling machine to drill the spot-hole, and shares the same set-up (orientation, machine and holding device) for the drill and mill operations. The learning mechanism comes up with the control rule in Figure 3 and a similar bindings preference rule. Goal preferences are also learned from other problems.

```
(control-rule pref-drill-with-spot-drill-in-milling-machine30
 (if (and
      (current-goal (has-spot <part> <hole> <side> <loc-x> <loc-y>))
      (pending-goal
       (holding <mach> <holding-dev> <part> <side> <side-pair>))
      (type-of-object <mach> milling-machine)))
 (then prefer operator drill-with-spot-drill-in-milling-machine
                       drill-with-spot-drill))
```

Figure 3: Search control rule learned from the example problem.

As the explanation is built from a single example and does not consider all possible hypothetical scenarios, it may be incomplete and the learned rules may be overgeneral. Upon unexpected failures the system refines the learned knowledge incrementally adding new rules if needed, and may set priorities among rules.

Table 1 shows the effect of the learned knowledge on the solution cost over 70 randomly-generated problems. Each column corresponds to a set of 10 problems with common parameters: number and type of goals, parts, etc. The training set consisted of 60 randomly generated problems with the same parameters than for sets 1 to 6 in the table. In many of the training and test problems the planner did not require control knowledge to obtain a good solution. Consequently for each problem set we have only recorded those for which the solution was actually improved. The number of nodes and total CPU time was also reduced due to shorter solution lengths. However we plan to further analyze the possible tradeoff between the learned-knowledge matching cost and the savings obtained by using it. We are exploring the effect of this learning mechanism on other domains and on other types of evaluation functions.

Problem set (10 probs per set)	1	2	3	4	5	6	7
# problems with improvement	3	9	3	10	10	4	9
Without learned control knowledge	107	202	190	431	362	442	732
With learned control knowledge	91	132	166	350	220	409	665
Cost decrease	44%	48%	33%	24%	47%	17%	8%

Table 1: Improvement on the quality of the plans obtained for 70 randomly-generated problems in the process planning domain. The third and fourth rows show solution cost according to the evaluation function.

Learning Domain Knowledge for Process Planning

PRODIGY acquires new preconditions by experimentation while planning using the process planning operators that are initially given to the system (Gil 1992). Table 2 presents some results obtained when PRODIGY learns preconditions that are missing from its initially given specification of the process planning domain. The tests were run in domains with 10% and 30% incompleteness using two training sets and two test sets.

Conclusion

Process planning is often considered too complex to be handled by general-purpose mechanisms. The work presented here illustrates our work on applying PRODIGY's general-purpose planner augmented with learning techniques that improve its performance in a process planning domain along several dimensions.

References

Borrajo, D., and Veloso, M. 1994. Incremental learning of control knowledge for nonlinear problem solving. In *Proceedings of the European Conference on Machine Learning, ECML94*. Sicily, Italy: Springer Verlag.

Carbonell, J. G.; the PRODIGY Research Group: Blythe, J.; Etzioni, O.; Gil, Y.; Joseph, R.; Kahn, D.; Knoblock, C.; Minton, S.; Pérez, A. (editor); Reilly, S.; Veloso, M.; and Wang, X. 1992. PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University.

Chang, T. C., and Wysk, R. A. 1985. *An Introduction to Automated Process Planning Systems*. Englewood Cliffs, NJ: Prentice Hall.

Descotte, Y., and Latombe, J.-C. 1985b. Making compromises among antagonist constraints in a planner. *Artificial Intelligence* 27:183–217.

Doyle, L. E. 1969. *Manufacturing Processes and Materials for Engineers*. Englewood Cliffs, NJ: Prentice-Hall, second edition.

Etzioni, O. 1990. *A Structural Theory of Explanation-Based Learning*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science. Also appeared as Technical Report CMU-CS-90-185.

Fox, M., and Smith, S. 1984. ISIS: A knowledge-based system for factory scheduling. *International Journal of Expert Systems* 1(1).

(a) Cumulative number of unexpected action outcomes during training.

(b) Number of plans successfully executed in the test set.

Table 2: PRODIGY learns new preconditions in the process planning domain.

Gil, Y. 1991. A specification of process planning for PRODIGY. Technical Report CMU-CS-91-179, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Gil, Y. 1992. *Acquiring Domain Knowledge for Planning by Experimentation*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science. Available as technical report CMU-CS-92-175.

Hayes, C. 1990. *Machining Planning: a Model of an Expert Level Planning Process*. Ph.D. Dissertation, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Joseph, R. L. 1992. *Knowledge Acquisition for Visually Oriented Planning*. Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Available as technical report CMU-CS-92-188.

Kambhampati, S.; Cutkosky, M. R.; Tenenbaum, J. M.; and Lee, S. H. 1993. Integrating general purpose planners and specialized reasoners: Case study of a hybrid planning architecture. *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Planning, Scheduling, and Control* 23(6).

Knoblock, C. 1991. *Automatically Generating Abstractions for Problem Solving*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science. Also appeared as Technical Report CMU-CS-91-120.

Minton, S.; Carbonell, J. G.; Knoblock, C. A.; Kuokka, D. R.; Etzioni, O.; and Gil, Y. 1989a. Explanation-based learning: A problem-solving perspective. *Artificial Intelligence* 40:63–118. Available as technical report CMU-CS-89-103.

Minton, S. 1988. *Learning Effective Search Control Knowledge: An Explanation-based Approach*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science. Also appeared as Technical Report CMU-CS-88-133.

Nau, D. 1987. Automated process planning using hierarchical abstraction. In *1987 Texas Instruments call for papers on AI for Industrial Automation*.

Pérez, M. A., and Carbonell, J. G. 1994. Control knowledge to improve plan quality. In *Proceedings of the Second International Conference on AI Planning Systems, AIPS-94*.

Pérez, M. A., and Etzioni, O. 1992. DYNAMIC: A new role for training problems in EBL. In Sleeman, D., and Edwards, P., eds., *Machine Learning: Proceedings of the Ninth International Conference (ML92)*. San Mateo, CA.: Morgan Kaufmann.

Veloso, M. M., and Carbonell, J. G. 1993. Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning* 10:249–278.

Veloso, M. M. 1989. Nonlinear problem solving using intelligent casual-commitment. Technical Report CMU-CS-89-210, School of Computer Science, Carnegie Mellon University.

Wang, X. 1994. Learning planning operators by observation and practice. In *Proceedings of the Second International Conference on AI Planning Systems, AIPS-94*.