Anytime planning for optimal tradeoff between deliberative and reactive planning

Will Briggs
Department of Computer Science
Lynchburg College
Lynchburg, VA 24501 U. S. A
briggs_w@acavax.lynchburg.edu

Diane Cook
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019 U. S. A.
cook@cse.uta.edu

Abstract

Anytime algorithms are useful when the time available for computation is limited, that is, when there is a tradeoff between the time cost of further computation and the cost of using a solution that is only partially complete. Although machine planning presents this sort of problem, there has not yet been a treatment of anytime planning for the general case, that is, a treatment not tied to specific domains. In this paper, we present a model for general-purpose anytime planning which allows the user to trade off the optimality of plans generated deliberatively with the speed of reactive plan generation. The anytime planner allows an interruption of hierarchical deliberative planning at the completion of any criticality level, and completes the plan at execution time using reactive planning. We illustrate the usefulness of this approach on a manufacturing domain.

Introduction

Automation of planning techniques can potentially save a great deal of design and programming time, and can help robots design plans when human help is not available. Unfortunately, machine planning programs search a space of possible world states (or, for many planners, a space of possible plans) which grows exponentially with plan length. Currently, the computational cost of machine planning algorithms prevents wide-spread use of these systems.

There are various methods for limiting the cost of planning. *Hierarchical planning* (Sacerdoti 1974; Woods 1991; Tennenberg 1099) orders goals or subgoals to plan for the most "critical" goals first, thereby replacing the vast search space with several spaces of reduced branching factor and smaller depth, which (usually) together form a significantly smaller space.

Hierarchical task network (HTN) planning (Wilkins 1984) refines subplans in a predetermined way, thus reducing the branching factor, sometimes to one.

Speedup learning (Minton, Bresina, & Drummond 1991) derives macro-operators to reduce the depth of search.

Each of these methods has the disadvantage of saving an indeterminate amount of plan generation time, and hierarchical planning can actually degrade the quality of the generated plan. *Reactive planning* (Kaelbling 1987; Brooks 1991), which limits the depth of the search space, can be significantly faster than nonreactive, or *deliberative* methods, but may produce suboptimal plans.

Anytime planning (Boddy 1991; Zilberstein & Russell 1996), allows a tradeoff between the solution quality of deliberative planning and the speed of reactive planning. An anytime planner allows the user to decide when to interrupt planning, giving the user a partially completed plan which may then be expanded by the user or by another program.

This type of system has the advantage that a user, not the program, determines how much computation cost is acceptable. Also, an anytime planner could enable an autonomous agent to begin plan execution with a partially specified plan, and refine the plan further during execution.

Existing anytime planning systems, however, are usually specific to particular domains, including path planning (Zilberstein & Russell 1996) and manufacturing (Fox & Kempf 1985). When probabilities and utilities are known and do not fall mostly into relatively high or low ranges, general-purpose decision-theoretic models may provide incremental performance by reducing the state space (Drummond & Bresina 1990) or plan space (Haddawy 1996) under consideration. A model of general-purpose anytime planning is needed if anytime planners are to find general use. The construction of such a model is the purpose of this work.

Anytime planning

Overall structure

For a partial planning solution to have value, there must be some mechanism for completing it. We propose to complete the plan with a (reactive) component that is fast but may not guarantee optimality. Figure 1 shows the structure of the anytime planner.

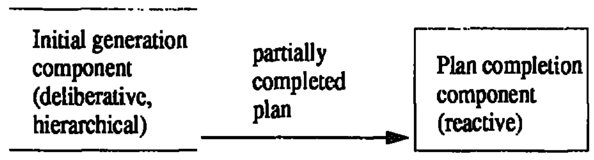
Initial generation component (deliberative, hierarchical)
partially completed plan
Plan completion component (reactive)

Figure 1: An general-purpose anytime planner. When the deliberative component is interrupted, the reactive component takes over.

The initial plan generation component must necessarily be better able to find optimal plans than the plan completion component; if not, the speedier plan completion component would be better to use for all of the plan generation. Our initial plan generator is a hierarchical deliberative planner, which (given enough time), can find solutions to any planning problem; these solutions are guaranteed to be optimal if the planning domain has the ***downward refinement property*** (Bacchus & Yang 1991), that is, if the domain is such that backtracking across criticality levels is not required.

Deliberative planning unfortunately is NP-hard (Chapman 1987). Recent advances in deliberative planning have improved the average-case performance of these systems (Blum & Furst 1997), but run time cost can still be a prohibiting factor. When the planner is interrupted, the partially completed plan is transferred to the plan completion component, which, being reactive, can be expected to complete the plan in polynomial time, and thus more quickly than the deliberative component.

The model allows the integration of alternative deliberative and reactive planners into the architecture, to take advantage of existing or future techniques for reducing plan generation cost.

Deliberative component

A deliberative planner is not necessarily guaranteed to have useful partial planning solutions during any point of its search; any plan being constructed runs the risk of being rendered invalid by interactions between subgoals. To provide partial solutions we use a hierarchical planner, which generates a plan for the most critical goals or subgoals first, and then refines the plans for successively less critical goals or subgoals. A ranking of criticality may be given by the programmer, or derived from the domain (Knoblock 1994). The partially specified plan generated by planning for some criticality level is guaranteed to be correct, if the planning domain exhibits the downward refinement property (Bacchus & Yang 1991) (or if the criticality ranking happens to be appropriate to the given problem), and (given time) the final solution is guaranteed to be optimal. Even without the downward refinement property, the partial solution may require only minor adjustment. In the worst case, a hierarchical planner working in a domain without the downward refinement property may generate a suboptimal plan (Harris 1994).

We interrupt the hierarchical anytime planner after an upper limit on the number of node expansions in the planner's search space has been reached. (We have chosen to consider a node expansion as an operation considered, rather than including constraints on operation variables, for simplicity.) The plan so far for the current criticality level may be unusable, because of interactions between subgoals; so the plan for the previous criticality level, which is complete with respect to goals and subgoals at that criticality level or higher, is sent to the reactive plan completion component.

Plan completion component

We require that the plan completion component incur cost no worse than polynomial in plan length; for general-purpose planning this requires that the second component be reactive. Our reactive planner is a forward-chaining rule-based system, which fires rules in the order given by the designer until one identifies an operator to be applied.

If a solution exists, we assume that the reactive planner can find it eventually; that is, the reactive planner is powerful enough to recognize when it has encountered a world state before, and thus take a different course of action, avoiding infinite loops. If the reactive planner fails to find a useful step, planning is aborted. The utility of the partially completed plan from the initial component is a combination of the percentage of goals achieved and the length of the plan.

Theoretical results

The cost of computation and execution depend on the time of interruption. For deliberative planners, the cost of planning is $O(b^N)$, where b is branching factor of the search space and N is plan length. For hierarchical planners in particular, if the planning domain has the downward refinement property, the cost of planning is $O(Cb^n)$, where C is the number of criticality levels encountered and n is the maximum number of operators added at any level.

In real-world domains, the computation cost for a reactive planner is dominated by the execution cost, and can thus be ignored. Therefore, given the downward refinement property, the cost is approximately the computation cost of hierarchical deliberative planning plus the cost for execution, or,

$$Cost_{anytime} \sim Cb^n + w_1(C_{all}n + R)$$

where w_1 is a user-defined constant or function giving the desired relative weight between cost of plan generation and cost of execution; C_{all} is the total number of criticality levels; and R is the number of steps added by the reactive planner, above the optimal.

If there are goals that may not be achieved because of poor decisions by the reactive planner, this becomes

$$Cost_{anytime} \sim Cb^n + w_1(C_{all}n + R) + w_2G$$

where G is the number of goals so affected, and w_2 is a weighting constant to be determined by the user.

We will have reduced cost over hierarchical planning provided that $Cost_{anytime}$ is less than $Cost_{hierarchical}$, or $Cb^n + w_1(C_{all}n + R) + w_2G < C_{all}b^n + w_1C_{all}n$. This can be reduced to the inequality $C < C_{all} - (w_1R + w_2G)/b^n$, which shows at what criticality level the process should be interrupted to minimize cost. If n is large, this is approximately $C < C_{all}$; that is, any interruption at all will result in savings, as deliberative planning is so time-consuming. For smaller n, the user, or an automated process, can calculate when best to interrupt the planner to minimize cost.

Application

Our anytime planner is implemented as a hierarchical deliberative planner based on GPS (Newell & Simon 1963), modified to use IDS search (Korf 1985) on the plan space, both for efficiency and to ensure that the generated plan will be optimal: the initial depth limit for search (which is the same as maximal plan length) is 1 step, and on failure this is incremented by 1, until a solution is found or the number of node expansions forces the planner to be interrupted.

Manufacturing

In this domain, planning problems relate to altering parts in these ways: altering the shape of the part, by using a roller or a lathe, or using a punch or drill press to create holes; altering the texture of the surface, by polishing, grinding to a smooth surface, or as a side effect of the previous operations; or painting, either with a spray gun or a painter that works by immersion. As the shape must be determined before the surface can be ground or polished, and the surface must be finished before paint is applied, we have goals with three levels of criticality: those related to shape; those related to texture; and those related to painting.

Some operations interfere with others: rolling makes a part hot, which prevents later polishing, punching, drilling, or spray-painting, and fills in any holes previously created by the punch or drill press. Operations related to shape undo operations related to texture or painting.

Examples These simple examples illustrate the problems of this domain related to optimality.

Straightforward expansion. In this example, the goals were to have Part A round and painted blue; Part B smooth and blue; Part C blue, and with a hole. Of the two most critical goals, each may be met in 2 ways: A's shape may be formed with the roller or the lathe, and C's hole may be made by the punch or the drill press. As the initial depth limit for search is 1, each of these four operations was tried and failed at this limit to achieve all goals at this criticality; then, at the next depth limit of 2, 2 operations (roll A, and punch C) are expanded before we have a plan for this criticality level. Therefore, until we allow 6 node expansions, the deliberative planner produces no solution; the reactive planner does all the plan generation. After that, the deliberative planner can meet the above goals; if we allow 7 node expansions, it can also plan for the next level, which involves grinding B. Neither component can paint the parts, as we have no blue paint.

Differences in goals achieved. In this problem, the planner is required to make A round and give it a hole. By arbitrary ordering, the planner considers rolling before lathing to achieve the shape, but is unable to complete a plan this way, as rolling heats the part and makes punching impossible; whereas if the hole is created first, the roller fills in the hole.

For this reason, not only must the deliberative planner consider 4 operations (rolling and lathing, because they can make A round, and punching and drilling, two ways to create the hole) before finding that no plan with only one operator will meet both goals. Then it considers these 3 operators: rolling; adding a punching operator before rolling; drilling rather than punching. It then backtracks, tries lathing rather than rolling, and completes the plan with punching (2 more operators). There are therefore 9 node expansions before plan completion at this (sole) criticality level.

When the allowed node expansions are less than 9, the reactive planner must generate the plan. It first applies the operation of rolling (by the same arbitrary ordering), then finds itself unable to create the hole, as the part is now hot. So in this case, an early interruption means that a goal will not be met.

Differences in length of the final plan. As in the previous problem, Part A is to be made round and given a hole, but now is also to be painted. As before, the deliberative component requires 9 node expansions to finish its first criticality level (relating to the first two goals). If the interruption comes sooner, the reactive plan completion component, given the arbitrary ordering of its choices, paints the part before meeting the other goals. Since the act of changing its shape removes the paint, the part must be painted again after. This results in one extra step in the final plan.

Results Planning was interrupted after one node expansion, after two, and so on, for each of 200 randomly generated problems of 1 - 20 goals each, relating to up to 8 parts. Figure 2 shows the decreasing cost functions for plans of various lengths with later times of interruption. Cost functions dropped more quickly for greater w_1, which is the cost of an executed operation divided by the cost of a node expansion, but were not always monotonically decreasing for sufficiently small w_1 (10 or less). Since an operation executed is generally much more expensive than a node expansion, this should not be a problem in real domains.

Conclusions

The use of general-purpose anytime planning presents an opportunity for user interaction not previously possible in machine planning. Also, autonomous agents

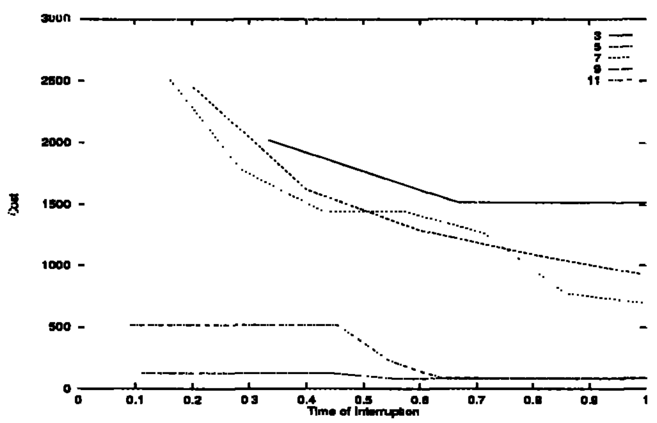

Figure 2: Decreasing cost functions for differing plan lengths. Cost is calculated as the cost of deliberation (nodes expanded), plus execution cost (operators executed multiplied by a weighting factor w_1 of 1000), and is measured in multiples of optimal cost. Time of interruption is measured as a fraction of plan length.

may make use of this process for making tradeoffs between generation cost and plan optimality. This work shows the usefulness of the approach analytically, and provides evidence regarding anytime planning performance in a manufacturing domain.

References

Bacchus, F., and Yang, Q. 1991. The downward refinement property. In *Proceedings of IJCAI-91*. IJCAI.

Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90:281–300.

Boddy, M. 1991. Anytime problem solving using dynamic programming. In *Proceedings of AAAI-91*, 738–743. AAAI.

Brooks, R. A. 1991. Intelligence without representation. *Artificial Intelligence* 47:139–159.

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32:333–337.

Drummond, M., and Bresina, J. 1990. Anytime synthetic projection: maximizing the probability of goal satisfaction. In *Proceedings of AAAI-90*, 138–144. AAAI.

Fox, B. R., and Kempf, K. G. 1985. Opportunistic scheduling for robotic assembly. In *IEEE International Conference on Robotics and Automation*, 880–889. IEEE.

Haddawy, P. 1996. Focusing attention in anytime decision-theoretic planning. In *AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems*. AAAI.

Harris, B. 1994. Hierarchical planning: tradeoffs and alternatives. Master's thesis, University of Texas at Arlington, Arlington, Texas.

Kaelbling, L. 1987. *An architecture for intelligent reactive systems*. San Mateo, CA: M. Kaufmann. Edited by Michael P. Georgeff and Amy L. Lansky.

Knoblock, C. A. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68:243–302.

Korf, R. E. 1985. Depth-first iterative-deepening: an optimal admissible tree search. *Artificial Intelligence* 27(1):97–109.

Minton, S.; Bresina, J.; and Drummond, M. 1991. Commitment strategies in planning: a comparative analysis. In *Proceedings of IJCAI-91*, 259–265. IJCAI.

Newell, A., and Simon, H. A. 1963. *Computers and Thought*. R. Oldenbourg KG. GPS: A Program that Simulates Human Thought. Edited by E. A. Feigenbaum and J. Feldman.

Sacerdoti, E. D. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 5:115–135.

Tennenberg, J. D. 1099. *Abstraction in planning*. Ph.D. Dissertation, University of Rochester, Rochester, NY.

Wilkins, D. E. 1984. Domain-independent planning: representation and plan generation. *Artificial Intelligence* 22:269–301.

Woods, S. 1991. An implementation and evaluation of a hierarchical non-linear planner. Master's thesis, University of Waterloo.

Zilberstein, S., and Russell, S. J. 1996. Optimal composition of real-time systems. *Artificial Intelligence* 82(1-2):181–213.