Classical Nonlinear Planning in Complex Domains

Mark Boddy
boddy@src.honeywell.com
Honeywell Systems & Research Center, MN65-2100
3660 Technology Drive
Minneapolis, MN 55418

Abstract

Classical nonlinear planning has been studied in the abstract since the early 1970's. More recently, interest has grown in applying these techniques to real problems, or at the least to simplified versions of real problems. In this paper, we explore some of the implications and results of those applications, with an eye to addressing the question of where the main problems lie. What successes have been achieved? What remains to be done? In particular, what fundamental representational or algorithmic issues, if any, need to be addressed before classical planning becomes a useful tool for large, complex, real-world problems?

Introduction

This paper is primarily a discussion of classical nonlinear planning in complex domains. Our concern is with how the planning problem is changed when we attempt to model more of the characteristics encountered in these domains. Abstraction has both benefits (refining away details to reveal the essential nature of a problem) and disadvantages (some of those details have a profound impact on the nature of the problem). For example, the Blocksworld has proved to be a useful abstraction in which to isolate and examine certain issues in planning. However, as the limitations of the Blocksworld have became apparent, more recent abstract domains (e.g., Tileworld [Philips and Bresina, 1991]) added such complexities as duration and deadlines, events not under the planner's control, and uncertain action outcomes.

Section provides some definitions and historical perspective. We then investigate the problems caused for planners by the introduction of partially-ordered plans, action durations and deadlines, simultaneous actions, and events not under the agents control. This is not a list of arguments for abandoning classical planning. Quite the contrary: for every difficulty raised here, there is at least a partial solution to the problem. We close with a summary and some recommendations for future work.

Definitions

We begin by defining some terms and providing a limited historical background. A *plan* is an internal representation of actions an agent may perform to bring about changes in the environment. Representing a plan requires *operators*: abstract representations of the actions available. Predicting the effects of executing a plan (i.e., performing the corresponding actions) requires some representation of the environment's state and how that state is modified by the performance of the actions corresponding to the various operators. This prediction is commonly called *projection*. The effects of actions are frequently modelled as operator *postconditions*, sometimes specified as add and delete lists of propositions modified by performance of the action. Operator *preconditions* encode the states in which a given action may be taken and how the effects of that action depend on the state in which the action is performed. [1]

The *planning problem* starts with an initial state description, and a *goal* to be achieved. The goal consists of a set of conditions to be achieved at the end of the plan's execution. [2] Work on replanning or generative planning adds a *partial plan* to be modified so that the goal is achieved. For this paper, I take "classical planning" to be the study of generative methods for solving the planning problem. Included in this definition are specializations such as hierarchical or constraint-posting planning.

The planner cannot make perfect predictions about the effects of a given plan. This true is independent of any consideration of dynamic domains (e.g., malicious infants knocking over towers of blocks). The issue is purely one of abstraction: any practical model of a physical process omits some details. In order to plan in the classical sense of the term, the planner needs to manipulate models of both the environment and the

[1] For the moment, we finesse the frame problem and related issues by assuming a STRIPS-rule semantics on a linear sequence of completely specified operators.

[2] This definition of a goal can be extended in a straightforward way to include descriptions of states to be avoided or maintained throughout the plan's execution.

agent's actions. The more details omitted from these models, the more errors will occur in the planner's predictions. As planners are applied to more complex domains, the domain and action models required to maintain a reasonable degree of fidelity become more complex as well.

In STRIPS [Fikes and Nilsson, 1971], states are represented as lists of predicates and actions are described by operators including preconditions, add and delete lists. These operators make no provision for context dependent effects, simultaneous actions, or action durations. More recently, STRIPS operators have been extended to model context-dependent effects [Pednault, 1988] and additional inference mechanisms have been added to derive context-dependent effects, e.g. [Wilkins, 1984, Allen and Koomen, 1983]. Some work has been done to add duration [Vere, 1983, Dean *et al.*, 1989, Tate *et al.*, 1992].

The advent of *nonlinear* planning [Sacerdoti, 1977, Tate, 1977] introduced new complications to the problem of determining the effects of a given plan. We follow [Minton *et al.*, 1991] in defining nonlinear classical planning to be generative planning that manipulates partially-ordered sets of operators, whether or not the final plan produced is required to be completely ordered.

Projection and Partial Orders

The original intuition behind nonlinear planning was the happy thought that we could plan by a process of adding orderings only as required to ensure that our plan has the right effects. This "least commitment" approach was entended to other forms of constraints as the notion of constraint-posting planning was generalized to include other aspects of a developing plan (e.g., [Stefik, 1981]. Constraint-posting planning can be viewed as an incremental process of eliminating possible plans from a set defined by the current set of constraints by adding additional constraints. For this process to be effective, it must be the case that the planner can determine whether or not there is a plan in the current set of possibilities that will achieve the goal, or that can be augmented so that it will achieve the goal. If the planner cannot construct a nontrivial description of the remaining possible plans and how that set would be changed by additional constraints, it is essentially adding constraints arbitrarily so as to obtain a plan sufficiently constrained that it can be analyzed. Making such arbitrary choices is exactly the kind of behavior that the least commitment approach was intended to circumvent.

Generating a correct and non-trivial description of the effects of a nonlinear plan is computationally expensive [Chapman, 1987, Dean and Boddy, 1987]. One way around this difficulty is to treat planning as a purely constructive activity. For example, Chapman's Modal Truth Criterion focuses only on the effects of operators: if there is a preceding operator establishing a goal, and no possible intervening defeater, the goal is declared to be satisfied. This is not projection, because we are making only a local determination. No attempt has been made to determine whether the preconditions for the operator satisfying the goal are true. This approach suffices for an environment in which the planner has complete control over what happens: goals can be satisfied by adding a new operator or moving existing operators around. If there are events not under the agent's control, it may not be possible to add operators where desired or to force the necessary orderings. This puts our planner back in the position of needing projection in the more complete sense.

This result also implies that nonlinear planners doing projection (i.e., those not using some local criterion like the MTC) are either solving a hard problem in some cases, or computing incorrect results in some cases. If computationally expensive problem instances are sufficiently rare, this may be acceptable. The possibility of incorrect results is more troubling.

In previous work [Dean and Boddy, 1988], we have demonstrated that it is possible to compute an approximation to projection for nonlinear plans in polynomial time. The algorithm is sound, meaning that if it says that a fact is necessarily true at a point, that determination is correct. The approximation involved is that the algorithm is not complete: it may fail to detect the fact that some fact is necessarily true. [3] Since the algorithm handles negated propositions, it is possible to compute projections involving a fact and its negation so as to determine whether a proposition is necessarily true at a point, necessarily false at that point, or indeterminately one or the other. Thus, even for planning problems involving events beyond the planner's control, projection can be regarded as a solved problem, though one that requires a little care in implementation. [4]

Duration and Deadlines

The fact that actions take time was abstracted out in the earliest domain models. Planners using these models will be of limited use in domains where synchronization with other events or processes is important. This may include such domains as manufacturing planning and scheduling, spacecraft operations, robot planning in any but the most simplified domains, and scheduling distributed problem-solving or other processing.

As mentioned in Section , several planners include representations for metric time and action durations. In addition, various people have implemented "temporal reasoning systems" in an attempt to isolate and optimize reasoning about the relations among actions, or between actions and other processes [Allen, 1983,

[3]The circumstances under which this algorithm is incomplete are described in [Schrag *et al.*, 1992].

[4]Assuming nonlinear plans consisting of STRIPS operators. It is certainly possible to complicate projection by adding other features to the representation.

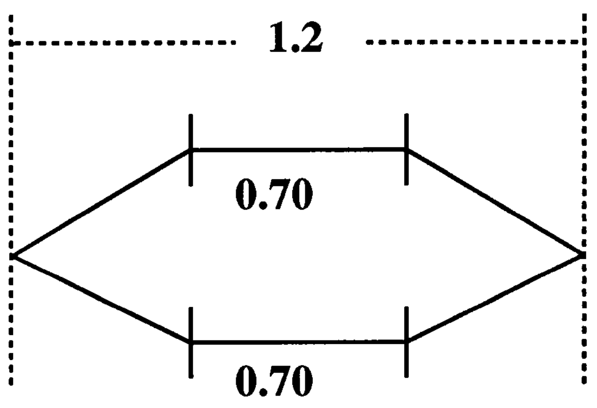

Figure 1: A simple problem with duration and partial orders

Dean and McDermott, 1987, Arthur and Stillman, 1992]. Oplan-2 contains a separate reasoning module (the *time point network manager*) with similar functionality. This kind of reasoning tends to be computationally expensive. Forbin, Deviser, and Sipe all suffer from performance problems limiting the size of the problems to which they can be applied. Oplan-2 appears to be able to handle larger problems than the other planners mentioned here.

Implementing an efficient temporal reasoning system is not the sole hurdle, however. Adding duration to nonlinear plans increases the difficulty of determining whether or not the current partial plan can be refined into a plan that will have the desired effects. In fact, it becomes difficult to determine simply whether the actions described in the current partial plan can even be executed.

Consider the simple plan fragment in Figure 1. There are two unordered tasks, each annotated with an estimated duration. If actions can only be taken in sequence, the two tasks depicted must eventually be ordered. When the planner tries to order them, it will discover that neither ordering will work, because there simply isn't room for them to be performed in sequence. In general, determining whether there is an ordering for a set of actions constrained in this way is a hard problem.

There are two proposed techniques for addressing this problem. The first is *simulation*: the planner maintains a partial order, and after every modification expends some effort exploring the corresponding set of total orders to ensure that there is some feasible total order [Miller, 1985, Muscettola, 1990]. As generally employed, this is a heuristic method: the planner gives up before exploring the complete set of consistent total orders.

Another possible approach is proposed by Williamson and Hanks in [1988]. In this approach, the partially ordered set of operators is organized into a hierarchy of abstract operator types, known as *Hierarchical Interval Constraints* (HIC). Each HIC type has a function defined for calculating bounds on its duration. For the example in Figure ??, the two activities would be contained in an HIC whose duration was calculated by summing the duration of the included operators.

The problem with this approach is the requirement of a rigid hierarchy. If actions must be ordered for reasons that are not locally determinable (e.g. because of resource conflicts, not because they are sequential steps in some task reduction), this representation will break down. It may be possible to augment Williamson and Hanks' representation to cope with a limited number of special structures representing such nonlocal information.

Simultaneous Actions

The general problem of reasoning about the effects of actions, when those effects may depend on what other actions occurred at the same time, is a computational nightmare. Potentially, you need to construct a description of simultaneous effects for the power set of the set of operators available to the planner. There are a couple of possible approaches to representing and reasoning about simultaneous effects more efficiently:

- Replace projection with a more general representation of causality and the evolution of a domain in terms of state variables [Muscettola, 1990].
- Restrict the kinds of operator interaction modelled, for example considering only resource usage [Wilkins, 1988].

Reasoning about resource usage is simpler than the general problem of simultaneous effects because the interactions compose. In other words, it doesn't matter why the present current demand is 5 amps. If your task requires 3 amps, the resulting demand is 8 amps. We can certainly construct resource models for which this property does not hold, but from an applications viewpoint, the aim is to expand the set of domain characteristics we can model, not to prove that the general problem is hard.

Summary

In this paper, we briefly explore some of the modelling and computational problems introduced by augmenting our domain models to include additional characteristics of complex domains. The particular characteristics we consider include unordered and simultaneous actions, and duration and deadlines. We briefly survey some proposed methods for dealing with these issues. It is not accidental that in doing so we stray in the direction of scheduling terminology and techniques. It has been clear for some time (certainly the earliest papers on Forbin in 1985 made this point) that effective "planning" for real-world domains will require techniques drawn from both the planning and scheduling communities. More people than we have room to cite here