Research in Concurrent Planning

William Cushing
Dept. of Comp. Sci. and Eng.
Arizona State University
Tempe, AZ 85281
william.cushing@asu.edu

Overview

Required concurrency is hardly a novel concept. PDDL2.1 specifically aims at allowing models of such problems, including a toy problem (the light-match domain) demonstrating the notion in its specification (Fox & Long 2003). The International Planning Competition dedicates a track solely to temporal planning, above and beyond classical (sequential) planning, which uses this language. Many planners, publicly available and otherwise, parse durative actions and output plans with concurrency: CRIKEY, LPGP, ZENO, IxTeT, VHPOP, SAPA, Prottle, FPG, SGPlan, MIPS, TGP, and so forth; many of which further compete in the IPC.

Our major contribution lies in emphasizing the importance of the role that required concurrency plays in planning and explicating the true nature of required concurrency. The key observation is that planners which attempt to fully support required concurrency (CRIKEY, LPGP, VHPOP, ...) are quite slow relative to those which use *decision epochs* (SAPA, Prottle, FPG, ...), which are again still slow relative to those planners which cannot handle required concurrency at all (SGPlan, MIPS, TGP, ...). So even though inherently sequential domains and required concurrency domains are both in PSPACE under reasonable assumptions, there is ample empirical evidence that problems requiring concurrency are harder. For that matter, if the problems are not more difficult, then clearly we need better theories for guiding such planners — there is no planner that supports required concurrency which can compete with, say, SGPlan. Our first paper attempts to analyze the nature of required concurrency and how a lack of required concurrency can be exploited for a computational advantage (Cushing *et al.* 2007a).

One could argue that the deepest issue in temporal planning is that which, supposedly, drove Zeno to suicide — the nature of continuous time. However, there are many answers to Zeno's paradox, and ultimately, the nature of time is completely independent of whether or not actions can be executed concurrently. Interpreting STRIPS actions as unit duration actions which induce changes in state in the manner of TGP shows that even Graphplan is an example of a concurrent planner (in discrete time), which TGP extends to the non-unit and non-uniform duration case. However, the TGP/Graphplan mutual exclusion rule means that neither accepts actions with *temporal gap*, that is, neither can even parse a problem potentially requiring concurrency.

SGPlan can be understood as a highly optimized version of TGP — it parses PDDL2.1, but internally enforces an equivalent mutual exclusion rule. It, however, operates in more or less continuous time, so in fact, it is clear that continuous time poses no special burden for plan synthesis above and beyond discrete time; there is no intuition that restricting SGPlan to integer/discrete times would make it run any faster. Our position is that the planning competition should have separate tracks for inherently sequential and required concurrent planning: not classical and temporal planning. It is not the nature of time which is in fact the problem, but rather the causal structure of plans. In our second paper on this topic we develop support for this position by analyzing the temporal competition benchmarks to show that, as given, they are inherently sequential, so that techniques for exploiting inherently sequential domains end up performing exactly the same on the classical and temporal versions of the benchmarks (e.g. SGPlan on versions of Rovers) (Cushing *et al.* 2007b). That is, the temporal competition is in the odd position of having developed a language aimed at domains requiring concurrency, further having entrants which to greater and lesser degree support such domains, and yet running only thinly disguised classical benchmarks, so that an inherently sequential planner walks away with first place.

Part of our explanation for this dearth of benchmarks is that PDDL2.1 makes modeling so difficult that it is easier to simply adapt classical benchmarks than it is to find real world problems which really need a rich durative action model instead of a TGP-like language. Current and future work involves formalizing a richer modeling language to support finding and expressing such problems, and to develop tools or at least formal methods for rewriting such problems into PDDL2.1 proper. This then allows a meaningful empirical evaluation of techniques for planning in the context of required concurrency. For example, we aim to identify correct pruning rules: conditions under which concurrency is definitely not required. This is an idea pioneered in CRIKEY (Halsey, Long, & Fox 2004), however, there is much more to the story of required concurrency than just *envelopes*, so that despite appearances, developing such pruning rules is still an open question. The following are excerpts of the work to date.

When is Temporal Planning *Really* Temporal?

William Cushing, Subbarao Kambhampati,
Mausam, Daniel Weld. IJCAI 2007.

Abstract

While even STRIPS planners must search for plans of unbounded length, temporal planners must also cope with the fact that actions may start at any point in time. Most temporal planners cope with this challenge by restricting action start times to a small set of *decision epochs*, because this enables search to be carried out in *state-space* and leverages powerful state-based reachability heuristics, originally developed for classical planning. Indeed, decision-epoch planners won the International Planning Competition's Temporal Planning Track in 2002, 2004 and 2006.

However, decision-epoch planners have a largely unrecognized weakness: they are incomplete. In order to characterize the cause of incompleteness, we identify the notion of *required concurrency*, which separates *expressive temporal action languages* from *simple* ones. We show that decision-epoch planners are only complete for languages in the simpler class, and we prove that the simple class is 'equivalent' to STRIPS! Surprisingly, no problems with required concurrency have been included in the planning competitions. We conclude by designing a complete state-space temporal planning algorithm, which we hope will be able to achieve high performance by leveraging the heuristics that power decision epoch planners.

Introduction

Although researchers have investigated a variety of architectures for temporal planning (e.g., plan-space: ZENO (Penberthy & Weld 1994), VHPOP (Younes & Simmons 2003); extended planning graph: TGP (Smith & Weld 1999), LPG (Gerevini & Serina 2002); reduction to linear programming: LPGP (Long & Fox 2003); and others), the most popular current approach is progression (or regression) search through an extended state space (e.g., SAPA (Do & Kambhampati 2003), TP4 (Haslum & Geffner 2001), TALPlan (Kvarnström, Doherty, & Haslum 2000), TLPlan (Bacchus & Ady 2001), and SGPlan (Chen, Hsu, & Wah 2006)) in which a search node is represented by a world-state augmented with the set of currently executing actions and their starting times. This architecture is appealing, because it is both conceptually simple and facilitates usage of powerful reachability heuristics, first developed for classical planning (Bonet, Loerincs, & Geffner 1997; Hoffmann & Nebel 2001; Nguyen, Kambhampati, & Nigenda 2001; Helmert 2004). Indeed, SGPlan, which won the International Planning Competition's Temporal Planning Track in both 2004 and 2006, is such a progression planner.

There is an important technical hurdle that these temporal state-space planners need to overcome: each action could start at any of an infinite number of time points. Most of these planners avoid this infinite branching factor by a (seemingly) clever idea: restricting the possible start-time of actions to a small set of special time points, called *decision epochs*. Unfortunately, the popularity of this approach belies an important weakness — decision epoch planners are *incomplete* for many planning problems requiring concurrency (Mausam & Weld 2006).

Seen in juxtaposition with their phenomenal success in the planning competitions, this incompleteness of decision epoch planners raises two troubling issues:

1. Are the benchmarks in the planning competition capturing the essential aspects of temporal planning?
2. Is it possible to make decision epoch planners complete while retaining their efficiency advantages?

In pursuit of the first question, we focused on characterizing what makes temporal planning really temporal — i.e. different from classical planning in a fundamental way. This leads us to the notion of *required concurrency*: the ability of a language to encode problems for which *all* solutions are concurrent. This notion naturally divides the space of temporal languages into those that can require concurrency (*temporally expressive*) and those that cannot (*temporally simple*). What is more, we show that the temporally simple languages are only barely different from classical, non-temporal, languages. This simple class, unfortunately, is the only class for which decision epoch planners are complete.

In pursuit of the second question, we show that the incompleteness of decision epoch planners is fundamental: anchoring actions to absolute times appears doomed. This leaves the temporal planning enterprise in the unenviable position of having one class of planners (decision epoch) that are fast but incomplete in fundamental ways, and another class of planners (e.g., partial order ones such as Zeno and VHPOP) that are complete but often unacceptably slow. Fortunately, we find a way to leverage the advantages of both approaches: a temporally lifted state-space planning algorithm called TEMPO. TEMPO uses the advantage of lifting (representing action start times with real-valued variables and reasoning about constraints) from partial order planners, while still maintaining the advantage of logical state information (which allows the exploitation of powerful reachability heuristics).

Evaluating Temporal Planning Domains

William Cushing, Daniel Weld, Subbarao Kambhampati,
Mausam, Kartik Talamadupula. ICAPS 2007 (to appear).

Abstract

The last eight years have seen dramatic progress in temporal planning as highlighted by the temporal track in the last three International Planning Competitions (IPC). However, a recent paper, (Cushing *et al.* 2007a), claims that most of these "temporal" IPC domains are essentially indistinguishable from STRIPS, requiring only a linear-time, scheduling, postprocessing step to optimize plans produced by a classical planner. This paper takes a critical look at these claims, asking "What makes a meaningful benchmark for temporal planners?" We argue that (Cushing *et al.* 2007a)'s claims (while correct) were unjustified, and show that it is actually quite difficult to prove that the IPC domains are *inherently sequential*. In order to do this, we develop a set of increasingly powerful analytic methods for domain analysis. We conclude that temporal planners should be evaluated on both inherently sequential domains as well as those requiring concurrency. We suggest some real-world domains with required concurrency,

and use a compilation argument to show domains with required concurrency are harder in the sense that they correspond to longer sequential plans.

Introduction

Since its inception in 1998 the bi-annual International Planning Competition (IPC) has led to dramatic improvements in planner speed. Starting in 2002, the IPC has included a track dedicated to temporal planning, whose language (PDDL) was specifically designed to describe domains with concurrent actions (c.f. the "Match" domain of (Fox & Long 2003)). Much to general satisfaction, the performance of temporal competition winners tracked improvements in classical planner performance, suggesting that advances discovered in the classical context were transferring smoothly to more complex problem sets.

However a recent paper, (Cushing *et al.* 2007a), challenges this conclusion. In a nutshell, Cushing *et al.* divide temporal domain description languages into two categories, "temporally simple" and "temporally expressive", and show that simple languages (e.g. TGP) can not even *express* problems whose solution plans *require concurrency* (though concurrency may lead to a shorter makespan). In contrast, expressive languages (e.g., PDDL) *can* describe problems where concurrency is required by all solutions, but *not all* domains written in an expressive language will have problems whose solutions require concurrency. Furthermore, if a language is temporally simple (i.e. every problem always has sequential solutions), then Cushing *et al.* show the much stronger property that the entire language is *inherently sequential* in the sense that optimal solutions of problems in the languages always correspond to linear-time rescheduling of sequential solutions. Approximately, temporally simple languages are isomorphic to STRIPS in the sense that one could augment a classical planner with such a rescheduling step at every search node. Indeed, it is now clear that SGPLAN (Chen, Hsu, & Wah 2006), which won the temporal track in 2004 and 2006, uses this strategy.[1]

Cushing *et al.* also point out that most of the top entrants of the planning competition, which are based on decision epoch planning, are not complete for domains and problems that require concurrency. Based on this, they speculate that most IPC temporal track domains/problems probably do not require concurrency. If true, this would suggest that the perceived progress in temporal planning may be illusory — we are simply seeing the result of faster classical planners on inherently sequential domains.

These discoveries leave researchers in a quandary, with several questions unanswered.

- Are the IPC temporal domains inherently sequential?
- If so, then is this a sign that real world domains rarely require concurrency?

And most importantly:

- Can one automatically analyze a PDDL domain description to determine if it is inherently sequential or has required concurrency?

[1] related in a personal communication by an author of SGPLAN.

In this paper we answer these questions, showing that the IPC domains are, in fact, sequential; that required concurrency is both important and common in real-world domains, and providing powerful analytic tools for determining the nature of PDDL domains. In addition, our analysis of the well-studied IPC domains reveals several modeling errors which lead us to suggest better ways of describing temporal domains.

Acknowledgements

I would like to thank my advisor Subbarao Kambhampati, coauthors Mausam and Daniel Weld, and peers within the Yochan group, J. Benton, Menkes van de Briel, Daniel Bryce, and Kartik Talamadupula, as well as the NSF and DARPA for funding this (and other) planning research.

References

Bacchus, F., and Ady, M. 2001. Planning with resources and concurrency: A forward chaining approach. In *IJCAI*.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism for planning. In *AAAI*.

Chen, Y.; Hsu, C.; and Wah, B. 2006. Temporal planning using subgoal partitioning and resolution in SGPlan. *JAIR*.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. 2007a. When is temporal planning *really* temporal? In *IJCAI*.

Cushing, W.; Weld, D.; Kambhampati, S.; Mausam; and Talamadupula, K. 2007b. Evaluating temporal planning domains. In *ICAPS (to appear)*.

Do, M. B., and Kambhampati, S. 2003. SAPA: A multi-objective metric temporal planner. *JAIR* 20:155–194.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *JAIR* 20:61–124.

Gerevini, A., and Serina, I. 2002. LPG: A planner based on local search for planning graphs. In *AIPS*.

Halsey, K.; Long, D.; and Fox, M. 2004. CRIKEY - a temporal planner looking at the integration of scheduling and planning. In *WIPS, ICAPS*, 46–52.

Haslum, P., and Geffner, H. 2001. Heuristic planning with time and resources. In *ECP*.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *ICAPS*, 161–170.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Kvarnström, J.; Doherty, P.; and Haslum, P. 2000. Extending TALplanner with concurrency and resources. In *ECAI*.

Long, D., and Fox, M. 2003. Exploiting a graphplan framework in temporal planning. In *ICAPS*, 51–62.

Mausam, and Weld, D. S. 2006. Probabilistic temporal planning with uncertain durations. In *AAAI*.

Nguyen, X.; Kambhampati, S.; and Nigenda, R. 2001. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *AIJ* 135:73–123.

Penberthy, S., and Weld, D. 1994. Temporal planning with continuous change. In *AAAI*.

Smith, D. E., and Weld, D. 1999. Temporal planning with mutual exclusion reasoning. In *IJCAI*.

Younes, H., and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *JAIR* 20:405–430.