To appear in AAAI 2007 Senior Member Track

Model-lite Planning for the Web Age Masses: The Challenges of Planning with Incomplete and Evolving Domain Models

Subbarao Kambhampati*
Department of Computer Science & Engineering
Arizona State University
http://rakaposhi.eas.asu.edu/rao.html

Abstract

The automated planning community has traditionally focused on the efficient synthesis of plans given a complete domain theory. In the past several years, this line of work met with significant successes, and the future course of the community seems to be set on efficient planning with even richer models. While this line of research has its applications, there are also many domains and scenarios where the first bottleneck is getting the domain model at any level of completeness. In these scenarios, the modeling burden automatically renders the planning technology unusable. To counter this, I will motivate model-lite planning technology aimed at reducing the domain-modeling burden (possibly at the expense of reduced functionality), and outline the research challenges that need to be addressed to realize it.

Introduction

In the past several years, significant strides have been made in scaling up plan synthesis techniques. We now have technology to routinely generate plans with hundreds of actions. A significant amount of ongoing work in the community (as well as in my own research group [7]) has been directed at building up on these advances to provide efficient synthesis techniques under a variety of more expressive conditions (including partial observability, stochastic dynamics, durative/temporal actions, over-subscribed resources etc.).

All this work however makes a crucial assumption–that a complete model of the domain is specified in advance. In particular, the expected domain model includes preconditions and effects of actions, probabilities of different outcomes (in the case of stochastic domains), and action costs and goal utilities (in the case of domains allowing partial satisfaction).

While there are many domains where knowledge-engineering such detailed models is necessary as well as feasible (e.g. mission planning domains in NASA, factory-floor planning), there are also many scenarios where insistence on correct and complete models renders the current planning technology unusable. Some high-profile examples of such scenarios are:

Web Service Composition: It is now common knowledge that the problem of composing web services has significant relations to automated planning [29]. Within the planning community, this realization has lead to work on supporting planning and composition tasks over richly specified web services (c.f. [25]). Outside of the planning community however, the key realization has been that the main bottleneck in web service composition is getting the specifications of the services, and arguments have been made in favor of planning support for light-weight models. (c.f. [10, 8, 9, 11]).

Workflow Management: Another wide-spread use of "plans" is in terms of specification and management of *ad hoc* workflows generated by lay users (be they scientists using and modifying scientific workflows or software engineers specifying flow-charts for applying patches [23, 6]). Here too, insistence on detailed domain models seems to be a significant liability (c.f. [21]).

Learning to Plan from Demonstrations: There is increasing interest in supporting learning to plan from interactive demonstrations. Two of the high-profile DARPA programs, Integrated Learning and PAL both involve aspects of this. Domain models acquired in this fashion are, of necessity, incomplete and evolving [33].

What is needed in such scenarios seems to be *model-lite planning* that can get by with incomplete domain models. Ideally, we need an *any-knowledge* planning technology that is able to cope with a variety of shallow and incompletely specified domain models and provide automation that is (a) proportional to the level of the available domain knowledge and (b) improves with time and experience.

Although there have been isolated attempts at realizing such model-lite planning technology (c.f. [9, 31, 6, 13]), there has not been any concerted effort to bring such work to the mainstream planning community (which continues to focus on model-rich and interaction-heavy planning). My aim is to thrust it to the foreground by identifying the benefits as well as exciting research challenges that underlie the realization of such model-lite planning technology.

My intent is not to argue against model-intensive planning work that is at the center stage in the community. It clearly continues to have a role. Rather, I want to persuade that model-lite planning also deserves serious consideration and

*For the latest version of this document as well as related resources, please see http://rakaposhi.eas.asu.edu/model-lite

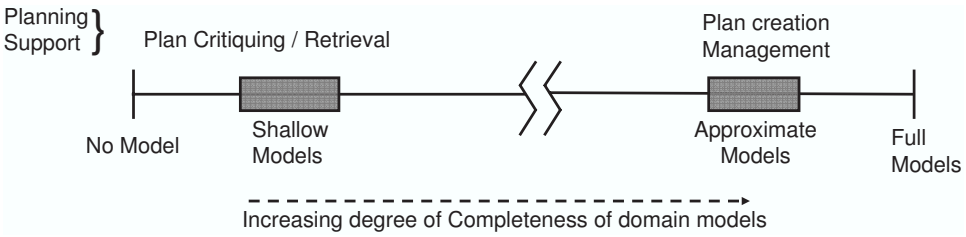

Figure 1: A Spectrum of Incomplete Domain Models and associated planning capabilities

that there are impressive pay-offs as well as exciting technical challenges in realizing it.

In the following, I will first describe related work and my own personal motivations. I will then draw a qualitative distinction between two classes of model-lite planning. This is followed by three sections that provide a sketch of challenges I foresee in developing model-lite planning frameworks, and current progress (in the community at large) towards handling them.

In keeping with the spirit of the title, this short paper does not make any claims on the completeness of its treatment of model-lite planning. I do hope to maintain a page of resources related to model-lite planning as well as any expanded versions of this paper at `rakaposhi.eas.asu.edu/model-lite`

Background & Motivations

In a way, the realization that domain models must, of necessity, be incomplete goes back to the very beginnings of planning research (c.f. qualification and ramification problems [26]). Even after planning community embraced state-variable (STRIPS) representations of the world, some of the earlier work was explicitly done to allow incomplete domain specification.

It is interesting to note, for example, that the original motivation for HTN planning was to handle domains where one doesn't have complete causal theory of the domain. In particular, HTN planning models were supposed to allow effects and/or dependencies between non-primitive actions that cannot be explained in terms of the preconditions/effects of the primitive actions. As time went on however, HTN planning has come to be seen as an add-on over and above complete primitive-action theory of the domain.

A similar shift happened in case-based planning. Originally, case-based planning was supposed to handle domains where the only available domain models were "case-knowledge" and a store of plausible modification rules (c.f. [16, 27]). As time went on however, HTN and case-based planning approaches have come to be seen more as add-ons over complete and correct STRIPS models. So, at one level, this paper can be seen as an argument to restore the importance of handling incomplete models.

My own interest in model-lite planning began with my efforts to adapt planning technology to autonomic computing [30] (the specific application involved reasoning with and managing software patching scripts). I have also noticed this need in web-service composition where we found that available services have at best "textual" descriptions of the service capabilities [11].

I was also motivated by my own involvement in data/information integration research [19, 20, 17]. Traditionally, the database community worked with model-rich scenarios where schemas are pre-specified and the capabilities and statistics of the databases under consideration are readily available. With the advent of web, the community has realized the need for supporting light-weight models (c.f. [12, 15]) and this has, in turn, lead to a robust research program that supplements rather than replaces the traditional model (schema)-rich database research. I believe a similar direction can be beneficial to the automated planning community.

Shallow vs. Approximate Domain Models

I view model-lite planning as planning with incomplete domain models. As I mentioned, the aim here is to reduce the modeling burden (as we shall see, planning itself may well be harder rather than easier in the face of incomplete models). I will assume that a "full" domain model includes enough knowledge to justify the correctness, and optionally, the optimality of the plan. To this end, a full model includes effect and cost models of the actions, inter-relations between the actions as well as background ontologies.

Incompleteness of the domain model is a matter of degree—ranging from no models to full models (see Figure 1). In looking at the challenges for handling incomplete models, we will find it useful to qualitatively distinguish two ends of incompleteness:

- "Approximate" domain models are those that are almost complete, but have some missing details. Examples of missing details could include missing preconditions and effects of actions (c.f. [13]), or cost models. We would like to be able to use approximate models to support plan creation as well as plan critiquing.
- "Shallow" domain models, in contrast, are those that aim to provide knowledge to support critiquing rather than creation of plans. Examples of shallow models include I/O type specifications, task dependency knowledge, case-bases, etc. (see below). Typically, these models do not involve precondition-effect style characterization of the actions.

By supporting approximate domain models, we admit that models may inevitably be faulty and incomplete. This, in turn, reduces some of the model-validation burden from the

domain modeler. By supporting shallow models, we can provide tools for plan critiquing and supporting manual plan generation, for scenarios where the users are unwilling or unable to provide generative models. (It is of course possible to have domain models that are shallow in some aspects and approximate in other.)

Challenge: Planning Support for Shallow Domain Models

The twin challenges of planning with shallow models are:

- Investigating wider variety of domain knowledge that can either be (a) easily specified interactively or (b) can be mined/learned.
- Types of planning support that can be provided with such knowledge.

I have already mentioned in the background section that the original motivation for case-based planning was to support planning when the only model of the domain involved past cases. Given the availability of large databases of workflows (c.f. [23]), it would be interesting to rekindle the original aims of case-based planning. To this, we can add other shallow models including:

I/O Type Specifications: Some work on web service composition has shown that even just the knowledge of input/output types can be gainfully used to support manual plan generation (c.f. [10, 9]).

Task Dependencies: The research on workflows has focused on action/activity dependency specification that cannot directly be explained in terms of the underlying precondition/effect causal theory (c.f. [3]). Such dependencies were found to be useful in managing software patching scripts (c.f. [30]).

Clearly, these different models are not mutually exclusive, and it would be interesting to consider ways in which they can be gainfully combined. In this connection, one attractive possibility that is worth exploring is whether the various types of domain knowledge can be compiled down into some common substrate (e.g. equivalent knowledge in state-variable models; see [30, 31]).

Challenge: Plan Creation with Approximate Domain Models

When the available domain models can be characterized as missing details from an almost correct model, the central challenge is to support plan creation *despite the incompleteness*. Note that the model incompleteness can be either in the domain dynamics or in terms of costs of various tasks. Theoretically, the approximate model can be seen as a stand-in for all the full models that are consistent with it. Ideally, we should then generate robust plans that are guranteed to work for any of the domain models. The plans themselves may be "conformant" or have branches that are conditioned on sub-classes of full models. The challenge of course is doing this efficiently.

Generating Robust Plans: One possible direction is to develop plans that are guaranteed to be robust as minor additional features of the underlying domain model are discovered/specified. Beginnings of such techniques can be found in the work by Garland and Lesh [13] and Ginsberg [14]. Another possibility is to model the incompleteness in the domain model as *uncertainty* in the domain (or the problem initial state) and use stochastic/non-deterministic planning techniques. An example of such an approach can be found in [2].

Generating Diverse or Multi-option Plans: An alternative approach for handling domain/cost model incompleteness is to generate a diverse set of plans that are complete/correct with respect to the possible complete models consistent with the incomplete one. Approaches for generating plans that are diverse with respect to an incompletely specified cost model are described in [28, 24]. It would also be interesting to characterize these plans as a single branching "multi-option" plan where the branching conditions are tests on domain/cost models (that can potentially be evaluated during run-time).

Challenge: Learning to Improve Completeness of Domain Models

Learning plays a central role in model-lite planning–either in terms of acquiring original (shallow) domain models or in terms of improving them through experience. The last time there was significant interest in learning techniques in planning, it was mostly for speedup reasons [18, 34]. Once the community figured out how to scale up search, that motivation partially disappeared. The need to deal with incomplete domain models puts learning for planning back into spotlight. In particular, we need techniques for learning planning knowledge from a variety of sources including textual descriptions, plan traces as well as expert demonstrations.

For shallow models, we expect learning to bootstrap the domain models by mining (or interactively acquiring) task dependencies, cases, and I/O type specifications (c.f. [10, 5]). For approximate models, we expect learning to help in improving the domain models through experience (or interactions with humans).

Here too there is a resurgence of recent interest. There has been work on learning action models either purely from example plans [32, 2] or in the presence of background knowledge [5, 22]. There has also been work on learning cost models indirectly given examples of better and worse plans [1]. Much however remains to be done. An added bonus of this direction is that it will naturally re-invigorate interest in knowledge-based learning, a critical area that has lain dormant in the recent years.

Summary

In this paper, I motivated the need for supporting "model-lite planning." I divided model-lite planning into two categories: planning with shallow models and planning with approximate models—and identified planning and learning challenges in both. I also tried to provide references to exist-

ing work that can be seen as implicitly focusing on model-lite planning.

I believe that in many ways, the time is ripe for the planning community to focus on model-lite planning. The community has already started taking domain modeling issues seriously (as evidenced by the Knowledge Engineering track of the International Planning Competition, being held for the second time in 2007). The need for interactive planning support in the presence of work flows, web services and desktop automation also pulls the community towards model-lite planning. Finally, model-lite planning is also going to gel well with the recent interest in integrated approaches for planning and learning (as evidenced, for example, by the DARPA Integrated Learning program).

Acknowledgements:

I would like to thank Biplav Srivastava for first stoking my interest in this topic, and J. Benton, Dan Bryce, Will Cushing, Sungwook Yoon and Menkes van den Briel for helping me refine my ideas (without laughing at me). Jose Luis Ambite, Jim Blythe, Alon Halevy and Joerg Hoffman provided useful comments and sanity checks on an earlier draft. Support for this work is provided in part by the DARPA Integrated Learning Program (through a sub-contract from Lockheed Martin), NSF grant IIS-0308139, and by ONR grant N000140610058.

References

[1] J. L. Ambite, C. A. Knoblock, S. Minton. Learning Plan Rewriting Rules. AIPS 2000: 3-12.

[2] E. Amir. Learning Partially Observable Deterministic Action Models. IJCAI 2005: 1433-1439

[3] P. C. Attie *et. al.* Specifying and Enforcing Intertask Dependencies. VLDB 1993: 134-145

[4] J. Blythe *et. al.* The Role of Planning in Grid Computing. ICAPS 2003: 153-163

[5] J. Blythe. Task learning by instruction in tailor. IUI 2005: 191-198

[6] J. Blythe, E. Deelman, Y. Gil. Automatically Composed Workflows for Grid Environments. IEEE Intelligent Systems 19(4): 16-23 (2004)

[7] D. Bryce, S. Kambhampati. A Tutorial on Planning Graph Based Reachability Heuristics. AI Magazine. Vol 28, No 1, Spring 2007. Tutorial delivered at ICAPS 2006 and IJCAI 2007.

[8] M. Carman, C. Knoblock. Learning Semantic Descriptions of Web Information Sources, IJCAI, 2007.

[9] M. Carman, L. Serafini, P. Traverso. Web Service Composition as Planning. ICAPS Workshop on Planning for Web Services. 2003

[10] X. Dong *et. al.* Similarity Search for Web Services. VLDB 2004.

[11] J. Fan and S. Kambhampati. A Snapshot of Public Web Services. SIGMOD Record, March 2005.

[12] M. Franklin, A. Halevy and D. Maier. From Databases to Dataspaces: A new abstraction for information management. ACM SIGMOD Record. December 2005.

[13] A. Garland, N. Lesh. Plan Evaluation with Incomplete Action Descriptions. AAAI/IAAI 2002: 461-467

[14] M. Ginsberg. Approximate Planning. Artif. Intell. 76(1-2): 89-123 (1995)

[15] A. Halevy *et. al.* Crossing the Structure Chasm. CIDR 2003

[16] K. J. Hammond. Case-Based Planning: A Framework for Planning from Experience. Cognitive Science 14(3): 385-443 (1990)

[17] T. Hernandez, S. Kambhampati. Integration of Biological Sources: Current Systems and Challenges Ahead. SIGMOD Record 33(3): 51-60 (2004)

[18] S. Kambhampati. Learning Techniques in Planning. Lectures given at 2006 Machine Learning Summer School. Canberra. 2006 `rakaposhi.eas.asu.edu/ml-summer.html`

[19] S. Kambhampati, C. Knoblock. Tutorial on Information Integration on the Web. Offered at AAAI 2002 (and to be offered at AAAI 2007). `http://rakaposhi.eas.asu.edu/i3-tut.html`

[20] S. Kambhampati, G. Wolf, Y. Chen, H. Khatri, B. Chokshi, J. Fan, U. Nambiar. QUIC: Handling Query Imprecision & Data Incompleteness in Autonomous Databases. CIDR 2007.

[21] J. Kim, M. Spraragen, Y. Gil. An intelligent assistant for interactive workflow composition. IUI 2004: 125-131

[22] G. Levine, G. DeJong. Explanation-Based Acquisition of Planning Operators. ICAPS 2006.

[23] B. Ludscher, C. A. Goble: Guest editors' introduction to the special section on scientific workflows. SIGMOD Record 34(3): 3-4 (2005)

[24] K. L. Myers. Metatheoretic Plan Summarization and Comparison, in ICAPS 2006.

[25] M. Pistore, P. Traverso, P. Bertoli. Automated Composition of Web Services by Planning in Asynchronous Domains. ICAPS 2005: 2-11

[26] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press. 2001.

[27] R. G. Simmons, R. Davis. Generate, Test and Debug: Combining Associational Rules and Causal Models. IJCAI 1987: 1071-1078

[28] B. Srivastava, S. Kambhampati, T. Nguyen, M. B. Do, A. Gerevini. Domain independent approaches for finding diverse plans. IJCAI 2007.

[29] B. Srivastava. The Synthy Approach for End to End Web Services Composition. Planning with Decoupled Causal and Resource Reasoning. AAAI 2006

[30] B. Srivastava and S. Kambhampati. The case for automated planning in Autonomic Computing. 2nd Intl. Conf. on Autonomic Computing. 2005.

[31] B. Srivastava, J. Vanhatalo, J. Koehler. Managing the Life Cycle of Plans. AAAI 2005: 1569-1575

[32] Q. Yang, K. Wu, Y. Jiang. Learning Actions Models from Plan Examples with Incomplete Knowledge. ICAPS 2005.

[33] S. Yoon and S. Kambhampati. Hierarchical Strategy Learning with Mixed Representations. AAAI Workshop on Acquiring Planning Knowledge Via Demonstrations. 2007.

[34] T. Zimmerman, S. Kambhampati. Learning-Assisted Automated Planning: Looking Back, Taking Stock, Going Forward. AI Magazine 24(2): 73-96 (2003)