Future Directions for First-Order Decision-Theoretic Planning

Research Proposal

Scott Sanner
Department of Computer Science
University of Toronto
ssanner@cs.toronto.edu

September 8, 2005

Contents

1 Overview

MDPs have become the *de facto* standard for modelling decision-theoretic planning problems. Recent work on MDPs has focused primarily on two research areas:

1. *Language extensions for MDP models.* The language used to specify an MDP determines how succinctly a given domain can be described. And succint model specification often has a direct impact on the design of algorithms that can efficiently exploit this structure.

2. *Exploiting MDP structure for efficient solution algorithms.* There are many types of MDP structure that can be exploited by solution algorithms to avoid full state and action enumeration. By exploiting this structure, solution algorithms can scale to MDPs that would be otherwise unsolvable with full state and action enumeration.

In recent years, *first-order MDPs (FOMDPs)* [5] have become a popular formalism for modelling decision-theoretic planning problems, owing to their ability to succinctly represent planning problems stated as stochastic variants of STRIPS [8] or PDDL [11]. In this research proposal, we outline current work and future directions for research in the areas of language extensions and exploitation of structure to efficiently represent and solve first-order decision-theoretic planning problems.

2 Previous and Current Research

To date, previous and current research has focused primarily on exploiting various types of structure that naturally occurs in decision-theoretic planning problems.

2.1 Exploiting Independence for Exact Solutions

Previous work [3, 10] for solving propositionally factored MDPs has used tree and ADD [14] data structures to exploit context-specific independence [4]. Very recent work on affine extensions of ADDs [16] (see attached) has provided a data structure that extends ADDs to compactly exploit additive, multiplicative, and context-specific independence in MDP inference. Empirical results suggest that the AADD outperforms both traditional tabular representations as well as ADDs on a variety of problems containing additive and multiplicative structure.

In another vein, current research on first-order ADDs (FOADDs) has examined methods for generalizing the ADD data structure from propositional to first-order representations. This work allows decision diagram nodes to consist of full first-order formulae that are lexicographically ordered with respect to a fixed relation ordering. Then, ordered resolution can be efficiently applied during a generalization of the standard ADD *Reduce*$(\cdot)$ and *Apply*$(\cdot)$ operations to prune out inconsistent nodes in the resulting decision diagram. With the FOADD data structure and algorithms properly defined, it is then straightforward to extend them to a first-order affine ADD (FOAADD), thus allowing FOMDP solution algorithms to exploit context-specific, additive, and multiplicative structure. Initial empirical results with the FOADD approach for value iteration have been promising, showing a marked increase in performance over a simple enumerated case representation. However, more comprehensive experimentation with both FOADDs and FOAADDs is needed to fully evaluate the impact of each of these data structures on the efficiency of FOMDP inference algorithms.

2.2 Exploiting Basis Function Representations for Approximate Solutions

Given the complexity of solving first-order MDPs, approximation techniques offer an efficient alternative to finding an exact solution. One popular method for approximating MDP solutions has been to represent the value function as a linear combination of weighted basis functions. Previous work [9, 17] has provided efficient solution techniques for finding good sets of basis function weights by exploiting the structure of propositionally factored MDPs.

More recent work on first-order approximate linear programming (FOALP) solution approaches [15] (see attached) has extended these basis function techniques to efficiently solve for approximate value functions in FOMDPs. This work represents the value function as a weighted set of first-order basis functions. It then uses a first-order generalization of cost network maximization in tandem with constrain generation techniques to efficiently solve for settings of these weights. Empirical results have shown that these techniques are relatively efficient in practice and yield policies which outperform both handcoded heuristics and myopically-optimal policies. In addition, these solutions yield error bounds on policy quality that apply equally to all domain instantiations of a problem – a novel result for the relational and first-order MDP literature.

However, despite the successes of the FOALP work, there are many future refinements to this work that could further improve results. One interesting question for future work is whether the uniform relevance weighting of partitions currently used in the FOALP objective is the best approach. It would be informative to explore alternative FOALP objective specifications and to evaluate their impact on value function quality over a variety of domains. In addition, it is an interesting question as to whether dynamic reweighting schemes could improve solution quality by identifying state partitions with large error and adjusting their relevance weights so they receive more emphasis on the next LP iteration. Altogether, such improvements to FOALP could bolster an already promising approach for efficiently and compactly approximating FOMDP value functions.

3 Future Research Directions

Future research directions build on current research directions and additionally look at first-order MDP modelling language extensions and the generalization of solution algorithms to handle such extensions.

3.1 Modelling Language Extensions

Perhaps one of the most important frontiers for FOMDP and general decision-theoretic planning research is that of providing the user with a rich set of language features that enable them to naturally model real-world problems. Following are a number of modelling language extensions that were motivated by actual planning problems along with a brief discussion of the modifications to solution algorithms to handle such extensions.

3.1.1 Sum and Counting Aggregators

In first-order domains, it is often very natural to predicate transition function and reward dependencies on the count of objects satisfying some criteria. For example, in a logistics domain, the probability that a delivery truck leaves on time may depend on the number of packages being loaded on the truck, and the reward might be the count of packages successfully delivered. While counting and inequalities can be stated for *specific* values within first-order logic, *generic* counting for potentially infinite domains cannot be done without augmentations of the FOMDP specification language. For example, using a count aggregator $\#_p\ \phi(p)$ which counts the number of instantiations of p that make $\phi(p)$ true, one can easily state a reward that scales with the number of packages successfully delivered: $\#_p\ [Package(p) \wedge Delivered(p)]$. However, there is no finite representation of such a reward for potentially infinite domains when using only standard first-order quantifiers; Such a specification would need to provide a condition and corresponding reward value for every possible count of packages.

The specification of sum and counting aggregators and the necessary extension of the regression operator is straightforward for FOMDPs. Thus, a general research approach for solving FOMDPs with sum and count aggregators would rely on defining the general backup operators and a full dynamic programming (DP) solution algorithm. However, solving such extensions of FOMDPs with a full DP algorithm presents a number of technical complications. Specifically, *simplification* and *consistency checking* are needed by full DP solution algorithms and the complex interaction between quantifiers and sum aggregators makes both of these operations very difficult.

Consequently, once the full DP solution to FOMDPs with sum and count aggregators has been defined, this definition could be applied to a more tractable approximate solution approach using basis functions. Basis function solution techniques would pose an elegant and tractable solution approach since they do not require that formulae be simplified and since sampling techniques can be used to reduce the sum aggregators to first-order formulae for which consistency checking is straightforward.

Although sampling and the avoidance of simplification considerably blowup the representation, these solution methods discard this representational blowup and project the value function down to a set of basis function weights, thus maintaining compactness.

3.1.2 Handling Quantity

Sum and counting aggregators are useful for counting domain objects satisfying some criteria, but discrete and continuous quantities can also be represented directly as relational attributes, e.g. $hasPackages(Paris, 5)$. While this specification does not allow one to model the specific properties of individual elements contributing to the quantity, it is a commonly used construct in planning domains and is simple to formalize in a FOMDP using the $+$ and $-$ arithmetic functions and equality/inequality predicates. The primary difficulty with reasoning in such domains is introducing relevant rules of inference for performing tractable inconsistency detection with the intended interpretation of these additional language elements.

3.1.3 Topological Structure

Topological structure occurs commonly in a number of planning domains, especially those with underlying location constraints. For example, logistics problems may require that trucks can only travel on certain roads to reach different cities, and that planes can only fly to certain airports in certain cities. While these domains can be formalized in FOMDPs for *arbitrary* underlying topologies, the solutions to such FOMDPs are often intractable as they have to take into account every possible topology for a potentially infinite number of locations.

When considering problems with an underlying topology, it is reasonable to assume that the topology is fixed and to solve the FOMDP with respect to that specific topology. In doing this, one can then make use of efficient graph algorithms in place of first-order reasoning during the FOMDP solution process. Such techniques have the capacity to yield efficient solutions for problems with underlying topologies – solutions which would otherwise prove difficult with first-order reasoning techniques alone.

3.1.4 Concurrent Actions

Many real-world planning domains allow multiple non-interfering actions to be executed simulataneously. While this can currently be done within the FOMDP framework, it involves specifying primitive actions corresponding to all possible joint action combinations that could take place. This approach is inefficient in that it requires an inordinately (if not infinitely) large number of joint actions, but also the specification of positive and negative effects for each of these actions. What is needed for efficient reasoning with concurrent actions is an efficient method for factoring both the effects and the value of simultaneously executed primitive actions in order to tractably deal with the combinatorial explosion of potential action executions. Two potential sources of ideas for this work are ConGolog [7] which specifies transition semantics for situation calculus domains with concurrent and exogenous actions, and work in MDPs [12] that deals with weakly coupled MDPs, each MDP having its own set of actions and local effects.

3.1.5 Program Constraints

Quite often, one has a good idea of the general sequence of actions that an agent should follow and there are simply a few choice points which should be left to the agent to decide according to some decision-theoretic criterion. In this case, it is useful to extend FOMDP solution techniques to handle program constraints such as those specified by DT-GOLOG [6]. The primary approach to solving a FOMDP under such constraints should be a relatively straightforward extension of the hierarchical abstract machine (HAM) framework [13, 1, 2] for solving MDPs. However, modifications will be required to generalize this technique to first-order state spaces.

3.2 Efficient Approximation Algorithms

There appear to be two distinct approaches taken by FOMDP solution algorithms: extensions of value and policy iteration for MDPs and extensions of linear programming techniques for basis function approximations of MDP value functions. Each of these techniques has its own advantages and disadvantages and thus it is worth examining possible extensions that can be made to both classes of algorithms.

3.2.1 Approximation Techniques that Exploit Independence

Value iteration techniques based on FOADDs and FOAADDs hold the promise of yielding efficient exact solutions to FOMDPs. However, it is also interesting to look at approximation extensions of these algorithms in the flavor of the APRICODD [18] extension of SPUDD [10]. In brief, APRICODD approximates an MDP value function by interleaving SPUDD value iteration steps with an approximation step that prunes nodes from the value function ADD in order to maintain a tractable representation. If the approximation is carried out so that it keeps track of the minimum and maximum bounds for the value function, then it is still possible to achieve convergence of the approximated value function under certain conditions. Following this work, APRICODD-style extensions to SPUDD could likewise be generalized to FOMDP value iteration algorithms based on FOADDs and FOAADDs, thus creating a new class of FOMDP approximate solution algorithms.

There are two main research questions for such extensions that have not been considered yet. First, while efficient ADD-based value function approximation techniques have been explored in APRICODD, approximation with AADDs is still an open research area. AADDs pose a number of difficulties for approximation techniques, namely that the properties of the data structure do not permit the direct extension of methods used for approximation with ADDs. Second, while efficient MDP value function approximation techniques have been explored for propositional MDPs, it is an open question as to whether these same techniques will apply to FOMDPs or whether modifications must be made that take into consideration the first-order structure found in FOADDs and FOAADDs. Both of these questions will need to be resolved for efficient FOMDP approximate solution techniques based on FOAADDs.

3.2.2 Alternate Basis Function Approximation Approaches

Currently, only the first-order approximate linear programming (ALP) [15] approach to approximating the FOMDP value function as a linear combination of basis functions has been considered. However there are two alternate approaches to finding basis function weights that have also been considered in the propositional MDP literature. These two methods are approximate value iteration (AVI) and approximate policy iteration (API), the latter having been identified as typically offering higher quality solutions than ALP.

While it is relatively straightforward to define the first-order extensions of the propositional versions of AVI and API in the spirit of the extension for ALP, this leaves a number of computational issues which have posed problems for such extensions. Both AVI and API require that a policy be derived at each step for their respective weight projection tasks.[1] However, naive methods for deriving a policy from a set of Q-functions have turned out to yield extremely large policies and have proved intractable to work with in practice. Before AVI and API can be effective techniques for approximate FOMDP solutions, a compact method for representing the policy and computing it must be derived. For now, one promising approach appears to involve structuring the policy using a FOADD or FOAADD to avoid unnecessary redundancy and to provide as compact a representation of the policy as possible.

[1]While the fact that a policy needs to be derived for API is obvious, this may not be the case for AVI. However it turns out that given the infinite action space of FOMDPs, deriving the max over the Q-functions inherently requires carrying out an operation equivalent to finding the policy for a given value function.

References

[1] David Andre and Stuart Russell. Programmable reinforcement learning agents. In *In Advances in Neural Information Processing Systems*, volume 13, 2001.

[2] David Andre and Stuart Russell. State abstraction for programmable reinforcement learning agents. In *In Proc. AAAI-02*, Edmonton, Alberta, 2002. AAAI Press.

[3] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121:49–107, 2000.

[4] Craig Boutilier, Nir Friedman, Moisés Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *UAI 96*, pages 115–123, Portland, OR, 1996.

[5] Craig Boutilier, Ray Reiter, and Bob Price. Symbolic dynamic programming for first-order MDPs. In *IJCAI 01*, pages 690–697, Seattle, 2001.

[6] Craig Boutilier, Ray Reiter, Mikhail Soutchanski, and Sebastian Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *AAAI 00*, pages 355–362, Austin, TX, 2000.

[7] Giuseppe De Giacomo, Yves Lesperance, and Hector Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.

[8] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *AI Journal*, 2:189–208, 1971.

[9] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venktaraman. Efficient solution methods for factored MDPs. *JAIR*, 2002.

[10] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *UAI 99*, pages 279–288, Stockholm, 1999.

[11] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL—The planning domain definition language, 1998.

[12] Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Solving very large weakly coupled Markov decision processes. In *AAAI 98*, pages 165–172, Madison, WI, 1998.

[13] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In M. Kearns M. Jordan and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 1043–1049. MIT Press, Cambridge, 1998.

[14] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, 1993.

[15] Scott Sanner and Craig Boutilier. Approximate linear programming for first-order mdps. In *UAI 2005*, 2005.

[16] Scott Sanner and David McAllester. Affine algebraic decision diagrams (aadds) and their application to structured probabilistic inference. In *IJCAI 2005*, 2005.

[17] Dale Schuurmans and Relu Patrascu. Direct value approximation for factored MDPs. In *Advances in Neural Information Processing Systems 14 (NIPS-2001)*, Vancouver, 2001. to appear.

[18] Robert St-Aubin, Jesse Hoey, and Craig Boutilier. APRICODD: Approximate policy construction using decision diagrams. In *Advances in Neural Information Processing Systems 13 (NIPS-2000)*, pages 1089–1095, Denver, 2000.