Discovering and Applying Domain Features in Probabilistic Planning

Jia-Hong Wu
Electrical and Computer Engineering, Purdue University, W. Lafayette, IN 47907
jw@purdue.edu

Abstract

In this paper, we present the current status of my doctoral thesis work on automatic feature discovery in probabilistic planning and discuss future research steps on this topic.

Introduction

Domain features are often used to compactly represent the dynamic structure of planning domains. A common usage of domain features is to form weighted linear-approximated value functions and use such functions to find problem solutions in planning (e.g. greedy policies). Useful domain features are often identified by human experts in actual applications. In my thesis work, which is a joint work with my advisor Dr. Givan at Purdue University, our goal is to devise techniques on finding such useful domain features automatically and applying the learned features in probabilistic planning.

In this paper, we first summarize our most recent work (Wu & Givan 2007), titled "Discovering Relational Domain Features for Probabilistic Planning", which appears in International Conference on Automated Planning and Scheduling (ICAPS) 2007. In (Wu & Givan 2007) we propose a relational feature-discovering framework in probabilistic planning, and the technique is evaluated on Tetris and domains from the first and second international probabilistic planning competitions (IPPCs). We then briefly discuss possible future research directions, where some of these directions may become the rest of my thesis work. A paper containing also a summary of (Wu & Givan 2007) and more detailed future research directions on combining search techniques in feature-discovery planning is appearing in the ICAPS 2007 Workshop on Artificial Intelligence Planning and Learning.

Feature Discovery for Probabilistic Planning

In (Wu & Givan 2007) we consider how relational features for approximated value function representation can be found in highly structured probabilistic planning domains. We use Markov decision processes (MDPs) to model the domains, and use inconsistencies in Bellman equation to identify new features. Features are real-valued and are represented as first-order formulas with one free variable and we use only basic domain predicates in such representations. To evaluate a feature f in a state s, we count the number of objects in s that satisfy f, which is n; we then normalize n to a number between zero and one by dividing n with the maximum number of objects any f can be satisfied in any s in the given problem. For example, in **blocksworld**, a feature may be $(\exists y)$ **on**(x, y). The value of this feature is the number of blocks on top of some other objects, divided by the total number of objects in the state. A beam-search is used to construct candidates for features and the candidates are ranked by how they correlate to the ideal "Bellman residual" feature. Learned features are used in weighted linear-approximated value functions. Starting from a trivial constant or problem-size feature, we use approximate value iteration (AVI) to find weights for the features and add newly learned features to the representation when necessary.

The feature learning method in (Wu & Givan 2007) differs from the one we introduced previously in (Wu & Givan 2005) largely due to how features are represented—the relational feature representation in (Wu & Givan 2007) is richer than the propositional one in (Wu & Givan 2005). In (Wu & Givan 2005) we reformulate the feature learning problem to the standard classification problem–training examples are labelled according to the sign of the statewise Bellman error, and the classification algorithm C4.5 (Quinlan 1993) is used to construct decision-tree features from the training sets. The resulting features are therefore binary-valued, hence another difference between the methods in (Wu & Givan 2007) and (Wu & Givan 2005).

We use the non-deterministic game Tetris and domains from the first and second international probabilistic planning competitions (IPPCs) to evaluate our feature learning method. Value functions are learned in these domains as described above, and we test the performance of the resulting greedy polices associated with the value functions. In Tetris, we outperform our previous propositional approach in (Wu & Givan 2005) in the 8×8 size and also the latest in the relational reinforcement learning (RRL) works (Driessens, Ramon, & Gärtner 2006) in the normal 10×20 size.

In the planning domains we start our learning process from small problem sizes and progress to larger problems when the performance of the planner meets certain thresholds. We also evaluate the progress of learning using certain

large problem sizes during learning. We compare the performance of our planner against two other state-of-the-art probabilistic planners: FF-Replan (Yoon, Fern, & Givan 2007) and FOALP (Sanner & Boutilier 2006). We show superior or similar performance against FF-Replan in blocksworld, boxworld, and tireworld (although not as well in zenotravel, exploding blocksworld, and tower of hanoi); and superior or similar performance in the tested IPPC domains against FOALP, which itself is also a feature-learning planner. The experimental results in those domains show that our technique represents the state-of-the-art for both domain-independent feature learning and for stochastic planning in relational domains.

Future Research Directions

We discuss ideas on how we can further improve our feature learning framework and how features learned in such framework can be applied to problem solving in probabilistic planning.

Learning new features essentially changes the value function representation, and also affects how greedy policies solve planning problems. We discuss several topics on how the learning and representation of features can be changed to improve the problem solving ability of feature-discovering planners. More extensive discussions on these directions are included in a paper to appear in the 2007 AAAI Fall Symposium on Computational Approaches to Representation Change during Learning and Development.

Our construction and application of features is based on information gathered by looking ahead one step into the future to compute the Bellman residual or to greedily select actions. This is a very limited type of search. We investigate how to use common search techniques that are well-known to the planning community in the representation, construction and application of features.

Removing Value Function Plateaus in State Space We observe in our experiments that the existence of large plateaus may be a cause of long solution length. Here we define a plateau to be a state region where all states in the region are equivalent in estimated value, typically because all selected features are constant within the plateau. When greedy policies are used to select actions, there may be no obvious choice in a plateau and the agent may be forced to "guess". This may lead to a sub-optimal plan. To address this problem, we can learn features specifically selected to target the plateaus and describe the dynamics inside the plateaus. One possible feature structure that can address the plateau problem is a two-level one, where the first level identifies the state regions containing the plateaus, and the second level represents the dynamic structure not identified previously in those regions.

Modelling Solution Flows using Features Planning domains may contain general solution "flows" that can be modelled in stages. For example, a solution pattern in blocksworld contains two stages, where in the first stage all blocks are placed on the table, and in the second one the blocks are piled into towers according to the goal description. Again, we can use a two-level feature structure to capture this concept, where the first level models the stages and the second level describes the detail in a certain stage.

Generalization in Feature Learning One issue when learning features in small problems is that the resulting features may "overfit" the small problems and not be able to describe a useful concept in general sizes. How to detect and prevent this type of "over-fitting" automatically may be an interesting research topic.

In some planning domains such as boxworld, problems can be decomposed into subproblems centering on a certain object where each subproblem has a simpler subgoal and such subgoal can be reached without affecting whether other subproblems can be solved. In these domains, features can be learned to represent subproblems instead of the more complex original ones. The feature-learning planner FOALP (Sanner & Boutilier 2006) uses this approach by learning features for generic subproblems for a domain and problems of different sizes in the same domain are solved using the solutions to the generic subproblems. However, some domains such as blocksworld have interacting subgoals and learning features for subproblems may not be desired. How to correctly identify whether and how problems in a domain can be decomposed is therefore critical in using feature learning in problem solving.

Using the Value Function as a Search Heuristic There could be various usages for a feature-based value function representation in constructing problem solutions. Up until now we only use value functions in forming greedy policies, which select actions based on local information. The feature-selection method shown in this paper uses training trajectories that may cover a wide state region. Features learned from using such trajectories represent domain properties in a global fashion. Those properties can be used to direct a general solution "flow" but may not provide local details on how to best act. On the other hand, search techniques can focus on the local details, but may be expensive if the space required to search is large and no guidance (e.g., heuristic) is provided. We can use value functions found by our feature-discovering method as search heuristics in search techniques such as the enforced hill-climbing method used in the deterministic planner FF (Hoffmann & Nebel 2001) or a complete informed search, such as using best-first search (Russell & Norvig 1995). The feature representation for this purpose must be simpler than the one used in our current system to speed up the searching processes. We can modify our feature learner so that ease of computation is part of the scoring of candidate features, a form of regularization.

In probabilistic environments the computation cost for searches is even more sensitive to the growth of search depth due to the need to combine results for different probabilistic outcomes. We can instead create a deterministic version of the probabilistic planning problems and solve the determinized problem instead. Indeed, FF-Replan (Yoon, Fern, & Givan 2007) uses this approach and uses FF (Hoffmann

& Nebel 2001) to solve the determinized plan, and replans if the actual outcome deviates from the predicted one. In our case, we can consider heuristic search in the determinized problem as an approximation to the more expensive direct expectimax search.

Searching for True State Value in Feature Learning In our feature learning method, candidates that are best correlated to Bellman residual are selected, but ideally we would like to use the difference between true value and evaluated value in states as a scoring measure. We may attempt to find a better approximation to such difference than using Bellman residual by searching deep into the state space and discovering more accurate estimates to true state value.

Incorporating Searches in Feature Structure Some regular structure in a planning problem may not be properly represented using features describing individual states. For example, in logistics or boxworld, we may want to learn whether a certain package can be moved correctly toward its destination if it is loaded on a particular truck or plane. This concept is hard to describe if a single state is observed unless the package is already close to the destination. But if search is allowed while evaluating whether a concept is true in a state, we can find out if the package can be advanced toward its destination without a need to backtrack by searching from the current state.

References

Driessens, K.; Ramon, J.; and Gärtner, T. 2006. Graph kernels and gaussian processes for relational reinforcement learning. *MLJ* 64:91–119.

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

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Russell, S., and Norvig, P. 1995. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, New Jersey.

Sanner, S., and Boutilier, C. 2006. Practical linear value-approximation techniques for first-order mdps. In *UAI*.

Wu, J., and Givan, R. 2005. Feature-discovering approximate value iteration methods. In *Abstraction, Reformulation and Approximation : 6th International Symposium*.

Wu, J., and Givan, R. 2007. Discovering relational domain features for probabilistic planning. In *ICAPS*. To Appear.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *ICAPS*. To Appear.