Incorporating Search Techniques in Feature-Discovering Planning

Jia-Hong Wu and Robert Givan
Electrical and Computer Engineering, Purdue University, W. Lafayette, IN 47907
{*jw, givan*}*@purdue.edu*

Abstract

We summarize our paper in the International Conference on Automated Planning and Scheduling (ICAPS) 2007 on relational feature-discovery planning methods and discuss research directions on how to use search techniques for additional performance improvement.

Introduction

We discuss how we can use common search techniques in enhancing the problem solving ability of planning based on feature-discovering techniques. The use of relational features allows solutions found in small problems to be generalized to large problems, and how to learn and use such features in planning is an interesting and ongoing research topic. In this paper, we first summarize our work (Wu & Givan 2007), titled "Discovering Relational Domain Features for Probabilistic Planning", which appears in International Conference on Automated Planning and Scheduling (ICAPS) 2007. We then consider future research directions on how to incorporate search techniques in the construction of features and in the use of features in problem solving.

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 represented as first-order formulas and we use only basic domain predicates in such representations. 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.

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 lastest 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

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.

Using the Value Function as a Search Heuristic There are various ways to use a feature-based value function representation in constructing selecting actions. 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 are intended to represent global domain properties. 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 can be very expensive if the space required to search is large and no guidance (e.g., heuristic) is provided. Here, we consider how these two approaches can be combined to improve problem solving in planning.

We consider combining various search techniques with our feature-discovering method to be a novel research direction. Little previous research has centered around feature-discovering value-function construction, and we know of no previous work that uses such automatically constructed value function as a search heuristic. Search heuristics are usually directly designed by humans, or calculated by following domain-independent rules formulated for this purpose by humans. Some previous techniques (Hoffmann & Nebel 2001; Prieditis 1993) transform given problems to new ones that are generally in reduced form, and automatically finds search heuristics by considering solutions or state structures to such transformed problems. Our feature-discovering method does not consider transformed problems and only works directly on the original planning problems. The implementation and evaluation of the novel combination of our feature-discovery methods with known search methods may lead to the discovery of more effective planners.

We discuss several possible approaches to use value functions found by our feature-discovering method as search heuristics. A greedy policy can be regarded as always looking ahead one step at possible outcomes for each action choice and picking the action that leads to the best expected outcomes. A simple extension to greedy policies is to search more than one step ahead but still to a predetermined horizon. The most effective search horizon may differ from domain to domain or even among the states in a problem. A large horizon makes it easier to escape local optima but can be slow to compute. A small horizon has the reverse effect. A more sophisticated approach is to determine when to terminate a search process by checking the heuristic value. For example, the enforced hill-climbing method used in the deterministic planner FF (Hoffmann & Nebel 2001) repeatedly finds and progresses to states with better heuristic value than the current state, terminating when the goal is reached or no state with better value is found (failed). We can also choose to perform a complete informed search, such as using best-first search (Russell & Norvig 1995).

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. This combination is usually done by computing the expected value of each action from every possible outcome. This means we need to search into all non-trivial outcomes even though we may have already found a solution path through some outcome. This is not always necessary in practice though, as we can choose to ignore the probabilistic information and instead create action aliases for every non-trivial outcome for each actions. The probabilistic planning problems can thus be approximated by deterministic ones, avoiding the need to compute expected values. Indeed, the latest version of FF-Replan (Yoon, Fern, & Givan 2007), which is one of the state-of-the-art probabilistic planners, uses this approach and outperforms all competitors in some probabilistic domains in the fifth International Planning Competition. FF-Replan 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.

What may also need to be addressed when using value functions as search heuristics is how efficient the value functions can be computed. To be used as a search heuristic, a value function should be as computationally efficient as possible; for feature-based value functions, this implies that the features should be evaluated as fast as possible. When designing the feature language for a search heuristic, the representation of the feature must be simpler than the one used in our feature learning system. Those features do not need to perform well when being used in a value function for greedy policies, but should still represent some general dynamic structure in the domain so that an informed search can be as efficient as possible. For example, we can modify our feature learner so that ease of computation is part of the scoring of candidate features, a form of regularization.

Searching for True State Value in Feature Learning In our feature learning method, candidates that are best correlated to Bellman residual are selected. Bellman residual is essentially decided by finding value inconsistencies between neighboring states. Although we have shown that using this measure is useful in finding features to approximate state value, ideally we would like to use the difference between true value and evaluated value in states as a measure. We may attempt to find a better approximation to such difference than using Bellman residual by searching deep into the state space. Informed search can be used here to increase the efficiency.

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 a concept that a truck carrying a certain package can move correctly toward the destination of the package by passing a certain city. This concept is hard to describe if a single state is observed unless the truck 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 truck is driving on a shortest path to the goal by searching from the current state and evaluate that if there is another shorter path to the destination. To speed up this search process, we only need to expand the states that involve changes in objects that we are concerned about.

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.

Prieditis, A. 1993. Machine discovery of effective admissible heuristics. *Machine Learning* 12:117–141.

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.