Discriminative Learning of Beam-Search Heuristics for Planning

Yuehua Xu
School of EECS
Oregon State University
Corvallis,OR 97331
xuyu@eecs.oregonstate.edu

Alan Fern
School of EECS
Oregon State University
Corvallis, OR 97331
afern@eecs.oregonstate.edu

Sungwook Yoon
Computer Science & Engineering
Arizona State University
Tempe, AZ 85281
Sungwook.Yoon@asu.edu

Abstract

We consider the problem of learning heuristics for controlling forward state-space beam search in AI planning domains. We draw on a recent framework for "structured output classification" (e.g. syntactic parsing) known as learning as search optimization (LaSO). The LaSO approach uses discriminative learning to optimize heuristic functions for search-based computation of structured outputs and has shown promising results in a number of domains. However, the search problems that arise in AI planning tend to be qualitatively very different from those considered in structured classification, which raises a number of potential difficulties in directly applying LaSO to planning. In this paper, we discuss these issues and describe a LaSO-based approach for discriminative learning of beam-search heuristics in AI planning domains. We give convergence results for this approach and present experiments in several benchmark domains. The results show that the discriminatively trained heuristic can outperform the one used by the planner FF and another recent non-discriminative learning approach.

1 Introduction

A number of state-of-the-art planners are based on the old idea of forward state-space heuristic search [Bonet and Geffner, 1999; Hoffmann and Nebel, 2001; Nguyen *et al.*, 2002]. The success is due to the recent progress in defining domain-independent heuristic functions that work well across a range of domains. However, there remain many domains where these heuristics are deficient, leading to planning failure. One way to improve the applicability and robustness of such planning systems is to develop learning mechanisms that automatically tune the heuristic to a particular domain based on prior planning experience. In this work, we consider the applicability of recent developments in machine learning to this problem. In particular, given a set of solved planning problems from a target domain, we consider using discriminative learning techniques for acquiring a domain-specific heuristic for controlling beam search.

Despite the potential benefits of learning to improve forward state-space planning heuristics, there have been few reported successes. While there has been a substantial body of work on learning heuristics or value functions to control search, e.g. [Boyan and Moore, 2000; Zhang and Dietterich, 1995; Buro, 1998], virtually all such work has focused on search optimization problems. These problems involve finding "least cost" configurations of combinatorial objects and have a much different flavor than the types of domains encountered in benchmarks from AI planning. To our knowledge, no such previous system has been demonstrated on benchmark domains from AI planning.

Recent work [Yoon *et al.*, 2006] has made progress toward learning heuristics for planning domains. The work focused on improving the heuristic used by the state-of-the-art planner FF [Hoffmann and Nebel, 2001]. In particular, the approach used linear regression to learn an approximation of the difference between FF's heuristic and the observed distances-to-goal of states in the training plans. The primary contribution of the work was to define a generic knowledge representation for features and a features-search procedure that allowed learning of good regression functions across a range of planning domains. While the approach showed promising results, the learning mechanism has a number of potential shortcomings. Most importantly, the mechanism does not consider the actual search performance of the heuristic during learning. That is, learning is based purely on approximating the observed distances-to-goal in the training data. Even if the learned heuristic performs poorly when used for search, the learner makes no attempt to correct the heuristic in response.

In this paper, we consider a learning approach that tightly couples learning with the actual search procedure, iteratively updating the heuristic in response to observed search errors. This approach is discriminative in the sense that it only attempts to learn a heuristic that discriminates between "good" and "bad" states well enough to find the goal, rather than attempting to precisely model the distance-to-goal. In many areas of machine learning, such discriminative methods have been observed to outperform their non-discriminative counterparts. A main goal of this work is to demonstrate such benefits in the context of planning.

Our learning approach is based on the recent framework of learning as search optimization (LaSO) [Daume III and Marcu, 2005], which was developed to solve "structured output classification" problems. Such problems involve mapping structured inputs (e.g. sentences) to structured outputs

(e.g. syntactic parses) and classification can be posed as performing a search over candidate outputs guided by a heuristic. LaSO provides an approach for discriminative learning of such heuristics and has demonstrated good performance across several structured classification problems. However, the search problems corresponding to structured classification are qualitatively very different from those typical of AI planning domains. For example, in structured classification the search problems typically have a single or small number of solution paths, whereas in AI planning there are often a very large number of equally good solutions. Given these differences, the utility of LaSO in our context is not clear.

The main contributions of this paper are to describe a LaSO-inspired algorithm for learning beam-search heuristics, to prove the convergence of the algorithm, and to provide an empirical evaluation in a number of AI planning domains. Our empirical results show that the approach is able to learn heuristics that improve beam-search compared to using the heuristic from the planner FF. In addition, the results show that discriminative learning appears to have an advantage over the existing non-discriminative approach.

In what follows, we first give our problem setup for learning planning heuristics. Next, we give an overview of the LaSO framework for structured classification, followed by a description of our LaSO variant and convergence analysis. Finally, we present experiments and conclude.

2 Learning Planning Heuristics

Planning Domains. A planning domain $\mathcal{D}$ defines a set of possible actions $\mathcal{A}$ and a set of states $\mathcal{S}$ in terms of a set of predicate symbols P, action types Y, and constants C. A state fact is the application of a predicate to the appropriate number of constants, with a state being a set of state facts. Each action $a \in \mathcal{A}$ consists of: 1) an action name, which is an action type applied to the appropriate number of constants, 2) a set of precondition state facts $\text{Pre}(a)$, 3) two sets of state facts $\text{Add}(a)$ and $\text{Del}(a)$ representing the add and delete effects respectively. As usual, an action a is applicable to a state s iff $\text{Pre}(a) \subseteq s$, and the application of an (applicable) action a to s results in the new state $s' = (s \setminus \text{Del}(a)) \cup \text{Add}(a)$.

Given a planning domain, a planning problem is a tuple (s, A, g), where $A \subseteq \mathcal{A}$ is a set of actions, $s \in \mathcal{S}$ is the initial state, and g is a set of state facts representing the goal. A solution plan for a planning problem is a sequence of actions $(a_1, \ldots, a_l)$, where the sequential application of the sequence starting in state s leads to a goal state s' where $g \subseteq s'$. In this paper, we will view planning problems as directed graphs where the vertices represent states and the edges represent possible state transitions. Planning then reduces to graph search for a path from the initial state to goal.

Learning to Plan. We focus on learning heuristics in the simple, but highly successful, framework of forward state-space search planning. Our goal is to learn heuristics that can quickly solve problems using breadth-first beam search with a small beam width. Given a representative training set of problems from a planning domain, our approach first solves the problems using potentially expensive search and then uses the solutions to learn a heuristic that can guide a small width beam search to the same solutions. The hope is that the learned heuristic will then quickly solve new problems that could not be practically solved prior to learning.

Heuristic Representation. We consider learning heuristic functions that are represented as linear combinations of features, i.e. $H(n) = \Sigma_i w_i \cdot f_i(n)$ where n is a search node, f_i is a feature of search nodes, and w_i is the weight of feature f_i. One of the challenges with this approach is defining a generic feature space from which features are selected and weights are learned. The space must be rich enough to capture important properties of a wide range of domains, but also be amenable to searching for those properties. For this purpose we will draw on prior work [Yoon *et al.*, 2006] that defined such a feature space, based on properties of relaxed plans, and described a search approach for finding useful features. In this investigation, we will use the features from that work in addition to using the relaxed-plan length heuristic.

The approach of [Yoon *et al.*, 2006] used a simple weight learning method, where weights were tuned by linear regression to predict the distance-to-goal of search nodes in the training set. While this approach showed promise, it is oblivious to the actual performance of the heuristic when used for search. In particular, even if the heuristic provides poor guidance for search on the training problems no further learning will occur. The main objective of this work is to improve performance by investigating a more sophisticated weight learning mechanism that is tightly integrated with the search process, iteratively adapting the heuristic in response to observed search errors. Below we first describe prior work from structured classification upon which our approach is based, and then describe its adaptation to our setting.

3 Learning Heuristics for Structured Classification

Structured classification is the problem of learning a mapping from structured inputs to structured outputs. An example problem is part-of-speech tagging where the goal is to learn a mapping from word sequences (i.e. sentences) to sequences of part-of-speech tags. Recent progress in structured classification includes methods based on condition random fields [Lafferty *et al.*, 2001], Perceptron updates [Collins, 2002], and margin optimization [Taskar *et al.*, 2003].

A recent alternative approach [Daume III and Marcu, 2005] views structured classification as a search problem and learns a heuristic for that problem based on training data. In particular, given a structured input x, the problem of labeling x by a structured output y is treated as searching through an exponentially large set of candidate outputs. For example, in part-of-speech tagging where x is a sequence of words and y is a sequence of word tags, each node in the search space is a pair (x, y') where y' is a partial labeling of the words in x. Learning corresponds to inducing a heuristic that quickly directs search to the search node (x, y) where y is the desired output. This framework, known as *learning as search optimization (LaSO)*, has demonstrated state-of-the-art performance on a number of structured classification problems and serves as the basis of our work.

LaSO assumes a feature-vector function $F(n) =$

$\langle f_1(n), \ldots, f_m(n) \rangle$ that maps search nodes to descriptive features. For example, in part-of-speech tagging, the features may be indicators that detect when particular words are labeled by particular tags, or counts of the number of times an article-tag was followed by a noun-tag in a partial labeling y'. The heuristic is a linear combination of these features $H(n) = F(n) \cdot w$, where w is a weight vector. LaSO attempts to select a w that guides search to the target solution by directly integrating learning into the search process. For each training example, LaSO conducts a search guided by the heuristic given by the current weights. Whenever a "search error" is made, the weights are updated so as to avoid the same type of error in the future. The process repeats until convergence or a stopping conditions. Convergence results have been stated [Daume III and Marcu, 2005] for certain types of weight updates.

4 Learning Heuristics for Planning

Given the success of LaSO in structured classification, it is interesting to consider its applications to a wider range of search problems. Here we focus on search in AI planning. Recall that our "learning to plan" training set contains planning problems with target solutions. This problem can be viewed as structured classification with a training set $\{(x_i, y_i)\}$, where each $x_i = (s_0, g)$ is a planning problem and each $y_i = (s_0, s_1, ..., s_T)$ is a sequence of states along a solution plan for x_i. We can now consider applying LaSO to learn a heuristic that guides a forward state-space search to find the solution y_i for each x_i.

While in concept it is straightforward to map planning to the LaSO framework, it is not so obvious that the approach will work well. This is because the search problems arising in AI planning have very different characteristics compared to those tackled by LaSO so far. Most notably, there are typically a large number of good (even optimal) solutions to any given planning problem. These solutions may take very different paths to the goal or may result by simply reordering the steps of a particular plan. For example, in the Blocks world, in any particular state, there are generally many possible good next actions as it does not matter which order the various goal towers are constructed. Despite the possibility of many good solutions, LaSO will attempt to learn a heuristic that strictly prefers the training-set solutions over other equally good solutions that are not in the training set. This raises the potential for the learning problem to be impossible to solve or very difficult since many of the other good solutions to x_i may be inherently identical to y_i. In such cases, it is simply not clear whether the weights will converge to a good solution or not.

One approach to overcoming this difficulty might be to include many or all possible solutions in the training set. In general, this is not practical due to the enormous number of possible good plans, though studying methods for computing compact representations of such plan sets and using them in LaSO is of interest. Rather, in this work we continue to use a single target solutions and evaluate an algorithm very much like the original LaSO, noting the potential practical problems that might arise due to multiple solutions. Interestingly, below we are able to derive a convergence result for this algorithm under certain assumptions about the structure of the multiple good solutions relative to the target solution.

Below we describe a variant of LaSO used in our planning experiments. Our variant is based on the use of breadth-first beam search, which is not captured by the original LaSO and that we found to be more useful in the context of planning. We will refer to the modified procedure as LaSO*.

Beam search. In breadth-first beam search, a beam B of beam width b is generated at each search step resulting in a beam of b nodes. At each step, all of the nodes on the current beam are expanded and the top b children, as scored by the heuristic, are taken to be the next beam. This process continues until a goal node appears on the beam, at which point a solution plan has been found. When the beam width is small, many nodes in the search space are pruned away, often resulting in the inability to find a solution or finding very sub-optimal solutions. When the beam width increases, the quality of the solutions tend to improve, however, both the time and space complexity increases linearly with the beam width, leading to practical limitations. The goal of our work is to learn a domain-specific heuristic that allows for beam search with small b to replicate the result of using a large b. This can be viewed as a form of speedup learning.

Discriminative Learning. The input to our learner is a set $\{(x_i, y_i)\}$ of pairs, where $x_i = (s_0, g)$ is a training problem from the target planning domain and $y_i = (s_0, s_1, \ldots, s_T)$ is a state sequence corresponding to a solution plan for x_i. Our training procedure will attempt to find weights such that for each problem the j'th state in the solution is contained in the j'th beam of the search. A search error is said to occur whenever this is not the case. Figure 1 gives pseudo-code for the overall learning approach. The top-level procedure repeatedly cycles through the training set passing each example to LaSO* to arrive at updated weights. The procedure terminates when the weights remain unchanged after cycling through all examples or a user defined stopping condition.

Given a training example (x_i, y_i), LaSO* conducts a beam search starting with the initial beam $\{(x_i, (s_0))\}$, i.e. a single search node with an empty plan. After generating beam j of the search, if $n^* = (x_i, (s_0, s_1, \ldots, s_j))$ is not on the beam then we have a search error. In this case, we update the weights in a way that makes n^* more preferred by the heuristic, ideally enough to remain on the beam next time through the search. We use a weight updating rule, similar to the Perceptron update proposed in [Daume III and Marcu, 2005]

$$w = w + \alpha \cdot \left(\frac{\sum_{n \in B} F(n)}{|B|} - F(n^*) \right)$$

where $0 < \alpha \leq 1$ is a learning rate parameter, $F(n)$ is the feature vector of search node n and B is the current beam. Intuitively this update rule moves the weights in a direction that decreases the heuristic value (increase the preference) of the desired search node n^* and increases the heuristic value for the nodes in the beam. After the weight update, the beam is replaced by the single search node n^* and the search continues. Note that each call to LaSO* is guaranteed to terminate in T search steps, generating training examples as necessary.

HeuristicLearn $(\{(x_i, y_i)\}, b)$
$w \leftarrow 0$
repeat until w is unchanged **or** a large number of iterations
  for every (x_i, y_i)
    LaSO*$((x_i, y_i), w, b)$
return w

LaSO* $((x, y), w, b)$
// x is a planning problem (s_0, g)
// y is a solution trajectory $(s_0, s_1, \ldots, s_T)$
// w current weight vector
$B \leftarrow \{(x, (s_0))\}$ // initial beam
for $j = 0, \ldots, T-1$
  $B \leftarrow$ **BeamExpand**(B, w, b)
  $n^* \leftarrow (x, (s_1, \ldots, s_{j+1}))$ // desired node
  if $n^* \notin B$ **then**
    $w \leftarrow$ **Update**(w, B, n^*)
    $B \leftarrow \{n^*\}$

BeamExpand (B, w, b)
candidates $\leftarrow \{\}$
for every $n \in B$
  candidates $\leftarrow$ candidates$\cup$ **Successors**(n)
for every $n \in$ candidates
  $H(n) \leftarrow w \cdot F(n)$ // compute heuristic score of n
return b nodes in candidates with lowest heuristic value

Figure 1: The discriminative learning algorithm.

5 Convergence of LaSO*

We now prove that under certain assumptions LaSO* is guaranteed to converge in a finite number of iterations to a set of weights that solves all of the training examples. In particular, we extend the convergence results of the original LaSO to the case of "multiple good solutions". The proof is a simple generalization of the one used to prove convergence of Perceptron updates for structured classification [Collins, 2002].

Consider a set of training problems (x_i, y_i), where $x_i = (s_0, g)$ and $y_i = (s_0, s_1, \ldots, s_T)$. For each (x_i, y_i) we denote by $n^*_{ij} = (x_i, (s_0, \ldots, s_j))$ the node on the desired search path at depth j for example i. Also let D_{ij} be the set of all nodes that can be reached in j search steps from n^*_{i0}. That is, D_{ij} is the set of all possible nodes that could be in the beam after j beam updates. In our result, we will let R be a constant such that $\forall i, j, \forall n \in D_{ij}, \|F(n) - F(n^*_{ij})\| \leq R$ where $F(n)$ is the feature vector of node n and $\|\cdot\|$ denotes 2-norm.

Our results will be stated in terms of the existence of a weight vector that achieves a certain margin on the training set. Here we use a notion of margin that is suited to our beam search framework and that is meaningful when there is no weight vector that ranks the target solutions as strictly best, i.e. there can be other solutions that look just as good or better. As defined below a *beam margin* is a triple (b', δ_1, δ_2) where b' is a non-negative integer, and $\delta_1, \delta_2 \geq 0$.

Definition 1 (Beam Margin). *A weight vector w has beam margin (b', δ_1, δ_2) on a training set $\{(x_i, y_i)\}$ if for each i, j there is a set $D'_{ij} \subseteq D_{ij}$ of size at most b' such that*

$$\forall n \in D_{ij} - D'_{ij} \quad , \quad w \cdot F(n) - w \cdot F(n^*_{ij}) \geq \delta_1 \text{ and,}$$
$$\forall n \in D'_{ij} \quad , \quad \delta_1 > w \cdot F(n) - w \cdot F(n^*_{ij}) \geq -\delta_2$$

Weight vector w has beam margin (b', δ_1, δ_2) if at each search depth it ranks the target node n^*_{ij} better than most other nodes by a margin of at least δ_1, and ranks at most b' nodes better than n^*_{ij} by a margin no greater than δ_2. Whenever this condition is satisfied we are guaranteed that a beam search with width $b > b'$ using weights w will solve all of the training problems. The case where $b' = 0$ corresponds to the more typical definition of margin (also used by the original LaSO), where the target is required to be ranked higher than all other nodes. By considering the case where $b' > 0$ we can show convergence in cases where no such "dominating" weight vector exists, yet there are weight vectors that allow search to correctly solve the training problems. The following theorem shows that if LaSO* uses a large enough beam width relative to the beam margin, then it is guaranteed to converge after a finite number of mistakes.

Theorem 1. *If there exists a weight vector w, such that $\|w\| = 1$ and w has beam margin (b', δ_1, δ_2) on the training set, then for any beam width $b > \left(1 + \frac{\delta_2}{\delta_1}\right) b'$, the number of mistakes made by LaSO* is bounded by $\left(\frac{bR}{\delta_1(b-b') - \delta_2 b'}\right)^2$.*

Proof. (Sketch) Let w^k be the weights before the k'th mistake is made. Then $w^1 = 0$. Suppose the k'th mistake is made when the beam B at depth j does not contain the target node $n^* = n^*_{ij}$. Using the fact that for $n \in B$, $w^k \cdot F(n^*) > w^k \cdot F(n)$, one can derive that $\|w^{k+1}\|^2 \leq \|w^k\|^2 + R^2$, which by induction implies that $\|w^{k+1}\|^2 \leq kR^2$. Next, using the definition of beam margin one can derive that $w \cdot w^{k+1} \geq w \cdot w^k + \frac{(b-b')\delta_1}{b} - \frac{b'\delta_2}{b}$, which implies that $w \cdot w^{k+1} \geq k\frac{(b-b')\delta_1 - b'\delta_2}{b}$. Combining these inequalities and noting that $\|w\| = 1$ we get that $1 \geq \frac{w \cdot w^{k+1}}{\|w\|\|w^{k+1}\|} \geq k\frac{(b-b')\delta_1 - b'\delta_2}{b\sqrt{k}R}$, implying the theorem. □

Notice that when $b' = 0$, i.e. there is a dominating weight vector, the mistake bound reduces to $\left(\frac{R}{\delta_1}\right)^2$, which does not depend on the beam width and matches the result stated in [Daume III and Marcu, 2005]. This is also the behavior when $b >> b'$. In the case when $\delta_1 = \delta_2$ and we use the minimum beam width allowed by the theorem $b = 2b' + 1$, the bound is $\left(\frac{(2b'+1)R}{\delta_1}\right)^2$, which is a factor of $(2b'+1)^2$ larger than when $b >> b'$. Thus, this result points to a trade-off between the mistake bound and computational complexity of LaSO*. That is, the computational complexity of each iteration increases linearly with the beam width, but the mistake bound decreases as the beam width becomes large. This agrees with the intuition that the more computation time we are willing to put into search at planning time, the less we need to learn.

6 Experimental Results

We present experiments in five STRIPS domains: Blocks world, Pipesworld, Pipesworld-with-tankage, PSR and Philosopher. We set a time cut-off of 30 CPU minutes and considered a problem to be unsolved if a solution is not found within the cut-off. Given a set of training problems we generated solution trajectories by running both FF and beam search with different beam widths and then taking the best solution found as the training trajectory. For Blocks world, we used a set of features learned in previous work [Yoon *et al.*, 2005; Fern *et al.*, 2003; Yoon, 2006] and for the other domains we

used the those learned in [Yoon *et al.*, 2006; Yoon, 2006]. In all cases, we include FF's heuristic as a feature.

We used LaSO* to learn weights with a learning rate of 0.01. For Philosopher, LaSO* was run for 10000 iterations with a learning beam width of 1. For the other domains, LaSO* was run for 1000 or 5000 iterations with a learning beam width of 10 (this beam width did not work well for Philosopher). The learning times varied across domains, depending on the number of predicates and actions, and the length of solution trajectories. The average time for processing a single problem in a single iteration was about 10 seconds for PSR, 2 seconds for Pipesworld-with-tankage, and less than 1 seconds for the other domains.

Domain Details. Blocks world problems were generated by the BWSTATES generator [Slaney and Thiébaux, 2001]. Thirty problems with 10 or 20 blocks were used as training data, and 30 problems with 20, 30, or 40 blocks were used for testing. There are 15 features in this domain including FF's relax-plan-length heuristic. The other four domains are taken from the fourth international planning computation (IPC4). Each domain included 50 or 48 problems, roughly ordered by difficulty. We used the first 15 problems for training and the remaining problems for testing. Including FF's relaxed-plan-length heuristic, there were 35 features in Pipesworld, 11 features in Pipesworld-with-tankage, 54 features in PSR and 19 features in Philosopher.

Performance Across Beam Sizes. Figure 2 gives the performance of beam search in each domain for various beam widths. The columns correspond to four algorithms: LEN - beam search using FF's relaxed-plan-length heuristic, U - beam search using a heuristic with uniform weights for all features, LaSO* - beam search using the heuristic learned using LaSO* (with learning beam width specified above), and LR - beam search using the heuristic learned from linear regression as was done in [Yoon *et al.*, 2006]. Each row corresponds to a beam width and shows the number of solved test problems and the average plan length of the *solved* problems.

In general, for all algorithms we see that as the beam width increases the number of solved problems increases and solution lengths improve. However, after some point the number of solved problems typically decreases. This behavior is typical for beam search, since as the beam width increases there is a greater chance of not pruning a solution trajectory, but the computational time and memory demands increase. Thus, for a fixed time cut-off we expect a decrease in performance.

LaSO* Versus No Learning. Compared to LEN, LaSO* tended to significantly improve the performance of beam search, especially for small beam widths—e.g. in Blocks world with beam width 1 LaSO* solves twice as many problems as LEN. The average plan length has also been reduced significantly for small beam widths. As the beam width increases the gap between LaSO* and LEN decreases but LaSO* still solves more problems with comparable solution quality. In Pipesworld, LaSO* has the best performance with beam width 5, solving 12 more problems than LEN. As the beam width increases, again the performance gap decreases, but LaSO* consistently solves more problems than LEN. The trends are similar for the other domains, except that in PSR, LEN solves slightly more than LaSO* for large beam widths.

Blocks World

	Problems solved				Average plan length			
b	LEN	U	LaSO*	LR	LEN	U	LaSO*	LR
1	13	0	25	11	6022	-	2444	4254
5	21	0	26	19	3094	-	939	1767
10	22	0	25	19	2589	-	1035	368
20	23	0	27	22	921	-	671	227
50	20	0	27	19	522	-	488	102
100	19	0	24	20	290	-	218	157
500	17	0	17	23	101	-	122	84
1000	16	0	19	20	100	-	103	68

Pipesworld

	Problems solved				Average plan length			
b	LEN	U	LaSO*	LR	LEN	U	LaSO*	LR
1	11	13	13	8	375	4323	1739	3888
5	16	17	28	19	1467	3176	1409	1300
10	17	17	26	21	2192	2252	740	800
20	17	17	27	22	161	1287	173	885
50	18	19	27	21	264	643	84	4111
100	18	16	27	21	83	233	72	165
500	21	18	25	21	39	73	67	74
1000	20	18	22	20	31	47	52	38

Pipesworld-with-tankage

	Problems solved				Average plan length			
b	LEN	U	LaSO*	LR	LEN	U	LaSO*	LR
1	6	4	7	2	139	2990	1491	1678
5	6	8	8	6	466	914	427	1556
10	6	8	8	9	142	2357	390	739
20	9	8	11	9	530	503	273	880
50	6	5	10	6	81	289	417	548
100	5	4	9	5	73	138	135	58
500	5	6	9	4	301	142	159	45
1000	5	6	8	7	69	61	79	100

PSR

	Problems solved				Average plan length			
b	LEN	U	LaSO*	LR	LEN	U	LaSO*	LR
1	0	0	0	0	-	-	-	-
5	0	4	14	9	-	231	275	228
10	1	20	12	13	516	178	194	160
20	7	18	12	17	374	151	183	146
50	13	17	16	16	154	106	105	109
100	13	15	9	13	114	91	86	86
500	4	4	2	2	55	61	53	48
1000	1	2	1	1	39	50	39	43

Philosopher

	Problems solved				Average plan length			
b	LEN	U	LaSO*	LR	LEN	U	LaSO*	LR
1	0	33	0	33	-	363	-	363
5	0	33	24	33	-	363	5469	363
10	0	33	21	33	-	363	3999	363
20	0	32	18	32	-	368	2612	358
50	0	6	10	23	-	255	966	308
100	0	16	9	18	-	287	1348	281
500	0	7	2	7	-	225	254	220
1000	0	1	1	4	-	198	214	204

Figure 2: Experimental results for five planning domains.

LaSO* significantly improves over U in Blocks world, Pipesworld and Pipesworld-with-tankage. Especially in Blocks world, where U does not solve any problem. For PSR, LaSO* only improves over U at beam width 5 and is always worse in Philosopher (see discussion below).

The results show that LaSO* is able to improve on the state-of-the-art heuristic LEN and that in the majority of our domains learning is beneficial compared to uniform weights. In general, the best performance for LaSO* was achieved for small beam widths close to those used for training.

Comparing LaSO* with Linear Regression. To compare with prior non-discriminative heuristic learning work we learned weights using linear regression as done in [Yoon *et al.*, 2006] utilizing the Weka linear regression tool. The results for the resulting learned linear-regression heuristics are shown in the columns labeled LR.

For Blocks world, LR solves fewer problems than LaSO*

with beam widths smaller than 100 but solves more problems than LaSO* with beam widths larger than 100. For Pipesworld and Pipesworld-with-tankage, LaSO* always solves more problems than LR. In PSR, LaSO* is better than LR with beam width 5, but becomes slightly worse as the beam width increases. In Philosopher, LR outperforms LaSO*, solving all problems with small beam widths.

The results indicate that LaSO* can significantly improve over non-discriminative learning (here regression) and that there appears to be utility in integrating learning directly in to search. The results also indicate that LaSO* can fail to converge to a good solution in some domains where regression happens to work well, particularly in Philosopher. In this domain, since action sequences can be almost arbitrarily permuted, there is a huge set of inherently identical optimal/good solutions. LaSO* tries to make the single training solution look better than all others, which appears problematic here. More technically, the large set of inherently identical solutions means that the beam-width threshold required by Theorem 1, i.e. $(1 + \frac{\delta_2}{\delta_1})b'$, is extremely large, suggesting poor convergence properties for reasonably beam widths.

Plan Length. LaSO* can significantly improve success rate at small beam widths, which is one of our main goals. However, the plan lengths at small widths are quite suboptimal, which is typical behavior of beam search. Ideally we would like to obtain these success rates without paying a price in plan length. We are currently investigating ways to improve LaSO* in this direction. Also we note that typically one of the primary difficulties of AI planning is to simply find a path to the goal. After finding such a path, if it is significantly sub-optimal, incomplete plan analysis or plan rewriting rules can be used to significantly prune the plan, e.g. see [Ambite *et al.*, 2000]. Thus, we can use the current LaSO* to quickly find goals followed by fast plan length optimization.

7 Summary and Future Work

We discussed the potential difficulties of applying LaSO to AI planning given the qualitative differences between search problems in AI planning and those in structured classification. Nevertheless, our preliminary investigation shows that in several planning domains our LaSO variant is able to significantly improve over the heuristic of FF plan and over regression-based learning [Yoon *et al.*, 2006]. We conclude that the approach has good promise as a way of learning heuristics to control forward state-space search planners. Our results also demonstrated failures of the discriminative approach, where it performed significantly worse than linear regression, which suggest future directions for improvement.

In future work we plan to extend our approach to automatically induce new features. Another important direction is to investigate the sensitivity of the LaSO approach to the particular solutions provided in the training data. In addition, understanding more general conditions under which the approach is guaranteed to converge is of interest. Currently, we have shown a sufficient condition for convergence but not necessary. We are also interested in determining the computational complexity of learning linear heuristics for controlling beam search. Also of interest is to investigate the use of plan analysis in LaSO to convert the totally ordered training plans to partially-order plans, which would help deal with the problem of "many inherently identical solutions" experienced in domains such as Philosopher. Finally, we plan to consider other search spaces and settings such as partial-order planning, temporal-metric planning, and probabilistic planning.

Acknowledgments

This work was supported by NSF grant IIS-0307592 and DARPA contract FA8750-05-2-0249.

References

[Ambite *et al.*, 2000] Jose Luis Ambite, Craig A. Knoblock, and Steven Minton. Learning plan rewriting rules. In *ICAPS*, 2000.

[Bonet and Geffner, 1999] Blai Bonet and Hector Geffner. Planning as heuristic search: New results. In *ECP*, 1999.

[Boyan and Moore, 2000] J. Boyan and A. Moore. Learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research*, 1:77–112, 2000.

[Buro, 1998] Michael Buro. From simple features to sophiscated evaluation functions. In *International Conference on Computers and Games*, 1998.

[Collins, 2002] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with the perceptron algorithm. In *Conf. on Empirical Methods in NLP*, 2002.

[Daume III and Marcu, 2005] H. Daume III and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*, 2005.

[Fern *et al.*, 2003] Alan Fern, Sungwook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias. In *NIPS*, 2003.

[Hoffmann and Nebel, 2001] Jorg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:263–302, 2001.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[Nguyen *et al.*, 2002] XuanLong Nguyen, Subbarao Kambhampati, and Romeo Sanchez Nigenda. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *Artificial Intelligence*, 135(1-2):73–123, 2002.

[Slaney and Thiébaux, 2001] J. Slaney and S. Thiébaux. Blocks world revisited. *Artificial Intelligence*, 125:119–153, 2001.

[Taskar *et al.*, 2003] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2003.

[Yoon *et al.*, 2005] Sungwook Yoon, Alan Fern, and Robert Givan. Learning measures of progress for planning domains. In *AAAI*, 2005.

[Yoon *et al.*, 2006] Sungwook Yoon, Alan Fern, and Robert Givan. Learning heuristic functions from relaxed plans. In *ICAPS*, 2006.

[Yoon, 2006] Sungwook Yoon. Discrepancy search with reactive policies for planning. In *AAAI-06 Workshop on Learning for Search*, 2006.

[Zhang and Dietterich, 1995] W. Zhang and T. G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *IJCAI*, 1995.