Learning to locate from demonstrated searches

Paul Vernaza and Anthony Stentz
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Email: {pvernaza,axs}@nrec.ri.cmu.edu

***Abstract*—We consider the problem of learning to locate targets from demonstrated searches. In this concept, a human demonstrates tours of environments that are assumed to minimize the human's expected time to locate the target, given the person's latent prior over potential target locations. The latent prior is then learned as a function of environmental features, enabling a robot to search novel environments in a way that would be deemed efficient by the teacher. We present novel approaches to solve both the inference problem of planning an expected-time-optimal tour given a prior and the learning problem of deducing the prior from observed tours. Our learning algorithm is inspired by and similar to maximum margin planning (MMP), although it differs in key ways. On the inference side, we advance the state-of-the-art by proposing novel relaxations that are integrated into a heuristic-driven search algorithm. An application to a home assistant scenario is discussed, and experimental results are given validating our methods in this domain.**

I. Introduction

The subject of our work is teaching a robot to locate a target object (or objects) by demonstrating search tours that are expected to locate the target efficiently. Although several applications motivate our interest in this concept, the primary application discussed in this work is a home assistant and/or elderly care scenario. We envision that in order to perform a variety of meaningful tasks, a home assistant robot should be able to locate common household objects efficiently. In designing a system to perform this task, we can imagine a few primary concerns: we would like the robot to be able to efficiently search any environment (including ones not previously seen), and we would like to minimize the amount of user or programmer effort spent devising heuristics to accomplish this task.

Our learning approach is designed primarily to address these concerns. The general idea is as illustrated in Fig. 1. In this paradigm, an engineer or user can simply demonstrate the path that they would take to locate a target object most efficiently. By collecting many such examples across different environments, we can learn to generalize these efficient search plans to novel environments.

Our approach is inspired by inverse optimal control techniques such as Max Margin Planning (MMP [17]). As such, we assume that the demonstrations optimize an objective with some hidden parameters to be discovered; namely, we assume the demonstrations are tours of the environments that minimize the demonstrator's expected time to locate the target. The latent parameters are therefore the probabilities with which the demonstrator expects to find the target at each location in the environment. In order to achieve generalization, we learn these probabilities as functions of features of the environment. We can then apply these to guess the user's prior distribution in a novel situation and use the prior to plan a search path that is efficient under that prior.

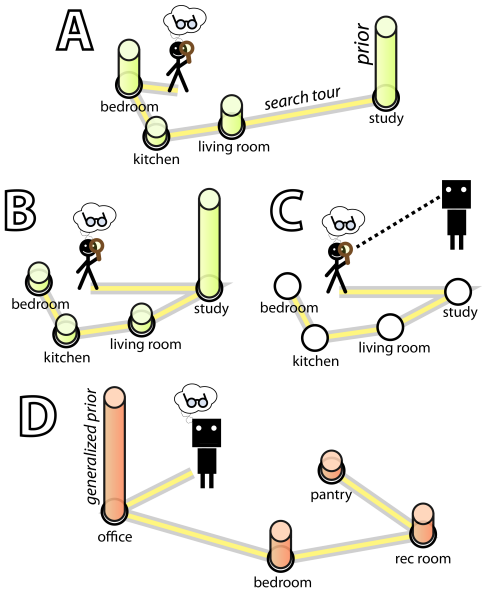


Fig. 1: Caricature of learning-to-locate concept. Fig. A depicts expected-time-optimal search for a target object: a person searches for the object based on an environment-dependent latent belief over potential object locations. The strength of the belief at each location is proportional to the height of the cylinder there. Fig. B demonstrates how the person's expected-time-optimal path might change depending on relative distances and belief strengths. In Fig. C, a robot observes a human demonstrating a search for a target object; notably, the robot is not able to observe the human's latent prior. Fig. D illustrates the result of learning: the robot is able to deduce the human's likely belief in a novel environment and uses this belief to plan an expected-time-optimal search path.

In this work, we present the details of carrying out such an approach. Although our method is inspired by MMP and related structured prediction methods, it differs in key practical and theoretical ways that we will discuss in more detail. Additionally, as in MMP, our learning algorithm consists of alternating between solving the inference problem (i.e.,

planning optimal search tours given the prior) and updating the parameters (i.e., the prior); a major issue is therefore how to solve the planning problem as efficiently as possible. To this end, we present advances in planning expected-time-optimal search tours by leveraging novel relaxations of the problem that are incorporated into a heuristic-driven search algorithm.

The next section provides a general overview of the method. Section III then delves into the details of our inference algorithm. Section IV describes technical details regarding the learning procedure, including differences between our method and comparable methods for structured prediction. We relate our method to previous work in Sec. V. Details of our experiments in the home assistant scenario are given in Sec. VI, followed by discussion.

II. Method overview

A. The inference objective

As stated earlier, our method assumes that we are able to obtain demonstrated search paths that minimize the demonstrator's expected time to locate the target. To make this notion more precise, we first introduce some notation. An *environment* is associated with a graph, consisting of a set of *locations* and a graph on those locations, which we assume to be complete wlog. To simplify notation, we assume that all environments consist of the N locations $\{1, \ldots, N\}$. Each arc in the graph is also associated with a transition time (or cost), such that $c(a, b)$ is the time taken in traversing the arc from a to b. A tour x of the environment is represented by a path visiting all the nodes; i.e., a permutation of $\{1, \ldots, N\}$.

We can associate with each tour x the random variable $T_d(x)$ representing the earliest time at which the tour intersects the target, denoting its expectation by $\mathbb{E}T_d(x)$. We are interested in the expectation taken with respect to the demonstrator's belief, which we characterize by the *prior* distribution Q: Q_i is defined as the probability with which the demonstrator expects to find the target at location i. The dependence of the expectation on Q is emphasized by the notation $\mathbb{E}[T_d(x) \mid Q]$. Finally, we can write the expected time to detection of the target under the belief Q as

$$\mathbb{E}[T_d(x) \mid Q] = \sum_{i=1}^{N} Q_{x_i} \sum_{j=1}^{i-1} c(x_j, x_{j+1}). \tag{1}$$

At run-time, our robot will need to deduce Q for its environment and then compute $\arg\min_x \mathbb{E}[T_d(x) \mid Q]$ in order to plan a search path. This constitutes a hard combinatorial optimization problem. We defer discussion of our solution to this problem until Sec. III, focusing for now on the problem of learning Q.

B. The learning objective

In order to achieve generalization across environments, we consider Q to be an unknown function of the environment to be deduced from our training data[1]. In order to do this, we will search over a set of Q functions parameterized by a parameter vector w. We use the notation $Q(e; w)$ to denote the prior Q evaluated on the environment e given the parameters w. Our training data $\mathcal{X}$ consists of pairs $(\hat{x}, \hat{e})$, where $\hat{x}$ is an example tour in environment $\hat{e}$.

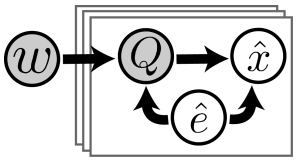


Fig. 2: Learning assumptions expressed as a plate model. Shaded circles represent latent variables. Arrows represent immediate, causal dependencies between quantities. See text for notation.

We define the *score* function $S(x, e, w)$ to be the objective value that the demonstrator assigns to the path x in the environment e, given parameters w to be determined. For now, we assume that the user optimizes just the expected time to find the target, which can be expressed as:

$$S(x, e, w) := \mathbb{E}[T_d(x) \mid Q(e; w)] \tag{2}$$

$$\hat{x} = \arg\min_x S(x, \hat{e}, w), \ \forall (\hat{x}, \hat{e}) \in \mathcal{X}. \tag{3}$$

The dependencies between and assumptions on w, Q, $\hat{x}$, and $\hat{e}$ are summarized in the graphical model illustrated in Fig. 2.

We would like to find w such that (3) holds as well as possible for all the training data. In order to do so, we consider selecting w such that the score of $\hat{x}$ is as low as possible compared to the score of any other tour in $\hat{e}$. This can be achieved by solving an optimization problem of the following form:

$$\min_w R(w) + C\Big(\sum_{(\hat{x},\hat{e})\in\mathcal{X}} S(\hat{x}, \hat{e}, w) - \min_x S(x, \hat{e}, w)\Big). \tag{4}$$

This is similar to objectives used in structured prediction methods such as MMP or structured perceptron, with notable differences to be explained in Sec. V-A. Here $R(w)$ is a regularization term meant to penalize overly complex solutions, and C is a fixed parameter that selects a trade-off between complexity and the degree to which the desired training data constraints are achieved; for each instance not satisfying the constraint, we pay a penalty proportional to the amount by which the expected search time of the training tour exceeds that of the best tour under the current prior.

We optimize this objective in a way similar to the subgradient method for max-margin structured prediction: given a current w, we solve the inner minimization over x, compute the gradient of the objective as if x were fixed to that value, and then take a step in the negative of that direction. More

[1]For simplicity, we assume the target object is fixed, and hence we do not generalize over target objects. However, there is no conceptual difficulty in using our method to generalize over target objects as well.

precisely, we alternate the following steps:

$$x^*_{\hat{e}} \leftarrow \arg\min_x \mathbb{E}[T_d(x) \mid Q(\hat{e}; w)],\ \forall(\cdot, \hat{e}) \in \mathcal{X} \quad (5)$$

$$w \leftarrow w - \alpha_k \nabla_w [R(w) + C(\sum_{(\hat{x},\hat{e})\in\mathcal{X}} S(\hat{x}, \hat{e}, w) - S(x^*_{\hat{e}}, \hat{e}, w))], \quad (6)$$

where α_k is a sequence of step sizes. Notably, this requires us to find an expected-time-optimal tour for each training example in order to compute the gradient step, the details of which we defer to Sec. III.

C. The form of the prior

A last design issue is the selection of the family of priors that we will consider in our optimization. We believe that the most natural choice is the family of maximum entropy (MaxEnt) priors expressed as

$$Q_i(e; w) = \frac{\exp \sum_j w_j \phi_{ij}(e)}{\sum_{k=1}^N \exp \sum_j w_j \phi_{kj}(e)}. \quad (7)$$

Here $\phi_{ij}(e)$ denotes the jth *feature* of the ith location in environment e; to give an example, we might define $\phi_{11}(e)$ to be the number of televisions at location 1, and we might define $\phi_{12}(e)$ to be the number of chairs at the same location.

The type of prior in (7) may be derived from first principles in the following way. Suppose that the demonstrator searches for and locates something many times and accumulates expectation statistics of the features—continuing the previous example, this would correspond to calculating the expected number of televisions and chairs at the target's location. Then the maximum entropy distribution consistent with these statistics is of the form (7). The MaxEnt distribution is in a sense the least biased distribution consistent with the data [7]. By assuming that the demonstrator's prior is of this form, we are assuming that they are choosing a rational belief based on the type of information that is likely available to them.

D. The latent prior assumption

A critical assumption of our method is that we observe optimal search paths and that the underlying priors are hidden. If the priors were not latent, we could simply fit models of the form (7) directly via regression. However, we prefer the latent prior model for several reasons. First, our intuition suggests that most people are not particularly adept at precisely quantifying uncertainty in their beliefs, since this kind of information is rarely required of them. Searching, on the other hand, is a familiar and intuitive task that people perform regularly, and we might expect people to have good heuristics to solve it. Furthermore, in a practical setting, we may wish to model other aspects of searching behaviors (see Sec. IV-B) that may only be discovered via demonstrations of search paths. If search paths are necessary anyway, then we might as well use these paths to infer the priors as well.

III. Planning optimal search tours

Solving the expected-time-optimal search problem (1) is a difficult combinatorial optimization problem. A dynamic programming solution was previously presented in [12]. That work computed the value function for a given start location and target probability distribution by writing the recursive Bellman equations for the value function and solving the equations iteratively via value iteration. We present an alternate approach here that leverages novel relaxations as heuristics in A* in order to solve the problem much more efficiently.

A. Optimization via graph search

The most straightforward way to approach the minimization of (1) as graph search is to explore the tree of all possible visitation orders of the locations, computing an additional term in the outer sum of (1) each time we append a location to the permutation. The worst-case complexity of this approach is proportional to $N!$, as it considers every possible permutation of the locations. We can improve on this by simply switching order of the sums in (1), obtaining the expression

$$\sum_{j=1}^{N-1} c(x_j, x_{j+1}) \sum_{i=j+1}^{N-1} Q_{x_i}. \quad (8)$$

The inner sum is equal to the probability that the target is not in the set $\{x_1, \ldots, x_i\}$, leading to the equivalent expression:

$$\sum_{j=1}^{N-1} c(x_j, x_{j+1})(1 - \sum_{i=1}^{j} Q_{x_i}). \quad (9)$$

We can convert this to graph search by defining our state as a minimal sufficient statistic necessary to compute an additional term in the outer sum. Specifically, this statistic consists of the last visited state and the *set* of all previously visited states (as opposed to the sequence). We therefore define a state as a pair $(y, \mathcal{V})$, where y is the last visited state and $\mathcal{V}$ is the set of previously visited states. The state successor function $\text{succ}(\cdot)$ and transition cost function $c(\cdot \to \cdot)$ are defined by

$$\text{succ}((y, \mathcal{V})) = \{(z, (\mathcal{V} \cup y)) \mid z \notin \mathcal{V} \cup y\} \quad (10)$$

$$c((y, \mathcal{V}) \to (z, (\mathcal{V} \cup y))) = c(y, z)(1 - \sum_{v \in \mathcal{V} \cup y} Q_v).$$

As the size of the state space is $N2^N$, and there are $O(N)$ successors per state, the worst-case complexity of searching this graph via Dijkstra's algorithm with a d-heap is $O(N^2 2^N)$ [1]. By contrast, the complexity of the value iteration (or Bellman-Ford [2]) approach of [12] is $O(N^3 4^N)$.

B. Incorporating relaxations as heuristics

We now relate how to employ the solutions of arbitrary relaxations of the problem as heuristics for planning in this graph-based framework. A *heuristic* is defined as a lower bound on the minimum cost to the goal state, as a function of an arbitrary state in the graph [6]. Note that having a way to obtain a lower bound on the optimum of (1) does not immediately translate into a lower bound *starting from an*

arbitrary state of the graph search. We would therefore like to convert the problem of obtaining a heuristic to that of bounding a problem of the form (1). Consider the graph representation of (10) and an arbitrary state $(y, \mathcal{V})$. The remaining cost to the goal is given by the expression

$$\sum_{j=|\mathcal{V}|+1}^{N-1} c(x_j, x_{j+1})(1 - \sum_{i=1}^{j} Q_{x_i}), \tag{11}$$

where $x_{|\mathcal{V}|+1}$ is identified with y. The probability on the right is equal to the probability that the target is not in the set $\{x_1, \ldots, x_j\}$, which is equal to the probability that the target is neither in the set $\{x_1, \ldots, x_{|\mathcal{V}|}\}$ nor in the set $\{x_{|\mathcal{V}|+1}\}$. The previous expression is therefore equivalent to

$$P(\text{target } \notin \{x_1, \ldots, x_{|\mathcal{V}|}\}) \sum_{j=|\mathcal{V}|+1}^{N-1} c(x_j, x_{j+1})(1 - \sum_{i=1}^{j} Q'_{x_i}),$$

where Q' denotes Q conditioned on the event that the target is not in the set $\{x_1, \ldots, x_{|\mathcal{V}|}\}$. Up to a constant, this expression is of the form (1) with a new prior Q', starting at location y.

C. Obtaining heuristics via relaxations

The previous discussion prompts us to consider bounds that may be obtained by optimizing relaxations of the expected-time-optimal search problem with efficient solutions. We consider two such relaxations, which are illustrated in Fig. 3. First, we may consider relaxing the constraint that in order to traverse an arc, the current location must be equal to the source of the arc. The resulting problem has the important property that the cost of inspecting a new location is independent of the current location. This variant of the expected-time-optimal search problem has been studied before in the search theory literature, and is known to have a simple solution [4]: greedily visit the location with the highest ratio of prior probability to inspection cost. Specifically, we iterate the following update until $Q_i = 0, \forall i$:

$$y \leftarrow \arg\max_{y'} \frac{Q_{y'}}{\min_z c(z, y')},\ Q_y \leftarrow 0 \tag{12}$$

This is illustrated in Fig. 3. We first teleport to location 1, travel to location 2, and then inspect location 2. After traveling back to and inspecting 1, we then teleport to location 3 in order to travel to and inspect the last location, which has a high probability, but is far from all other locations.

Another heuristic is obtained by relaxing the implied constraint that we have only one searcher with which to locate the target. Given N searchers, the optimal strategy is clearly to assign each location to a searcher and have the searchers travel to their assigned locations optimally and in parallel. Let $d(y)$ represent the time associated with the least-time path from the start location to y. The expected time to locate the target under this policy is simply

$$\sum_{i=1}^{N} Q_i d(y). \tag{13}$$

We evaluate these heuristics in Sec. VI.

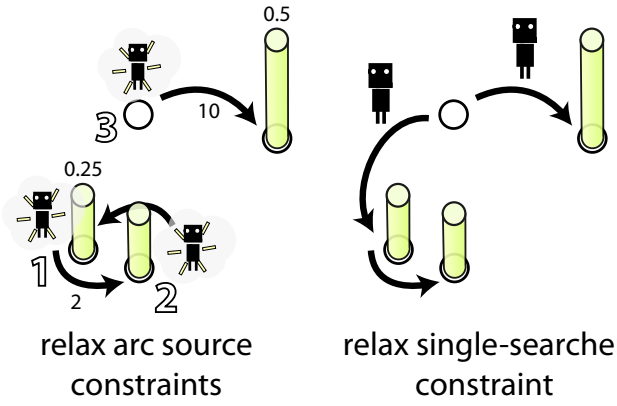


Fig. 3: Illustration of relaxations of expected-time-optimal search. See text for details.

IV. Learning details

A. Gradient-based optimization

A key difference between our method and max-margin methods is the nonconvexity of our score function. Although the mechanics of the alternating optimization method proposed in Sec. II-B method are identical to the subgradient optimization method for max-margin objectives, it is important to emphasize that our optimization can no longer be called a subgradient method [3]. This is due to the nonconvexity of our score function, which implies that there must be points at which subgradients of the objective do not exist. Unfortunately, this means that we cannot apply standard results regarding convergence of the subgradient method to our method. However, our results in Sec. VI demonstrate that convergence is observed in practice.

Another important sanity check in the absence of theoretical convergence results is to examine the precise form of the update rule (6) to see whether it seems reasonable. We obtain the following expression for the derivative of the score function in (6):

$$\frac{\partial S(x, e, w)}{\partial w_k} = -\mathbb{E}_z(\phi_{zk}(e) - \mathbb{E}_{z'}\phi_{z'k}(e)) \cdot \sum_{i=1}^{N} c(x_i, x_{i+1}) \mathbb{1}\{z \in \{x_1, \ldots, x_i\}\},$$

where $\mathbb{E}_z$ denotes expectation over locations z with respect to the prior $Q(e; w)$ and $\mathbb{1}\{\cdot\}$ denotes the indicator function. This expression is an expectation of a product of two terms: the first being a deviation of the feature from the mean, and the second being the total amount of time that the target is observed by the path. We can deduce from this that the features that matter to the objective are those that occur at locations early in the path and at locations where their values are far from the mean. This corresponds with intuition, as locations occurring late in the path correspond to locations that are probably unlikely, and features with low variance are probably not significant.

B. Incorporating additional preferences

It is important to note that it is straightforward to model additional preferences in the demonstrator's objective (2). For instance, in practice, we might expect that the demonstrator may prefer tours that are short, even if they take slightly longer to locate the target in expectation. This preference can be modeled by adding an additional term to the user's objective:

$$S(x, e, [w, \alpha]) = E[T_d(x) \mid Q(\hat{e}; w)] + \alpha \sum_{i=1}^{N-1} c(x_i, x_{i+1}). \tag{14}$$

We then modify our update rules and the inference routine accordingly in order to optimize over α as well as w.

V. Related work

A. Structured prediction and energy-based methods

The concept of an *energy-based method* (EBM [13]) is a very general one that describes most structured prediction approaches. In this view, the goal of learning is to find an *energy function* that accepts input-output pairs, returning a scalar value that is low when the pairs are likely to be observed, and is high otherwise. The key to understanding different interpretations of our work as an EBM is the observation that our method incorporates *two* kinds of energy (or score) functions: the path energy of Eq. (2) and a location energy, equal to the log of Eq. (7). Considering either as the *principal* energy leads to very different interpretations of our method.

1) Unnormalized EBMs: First, we can consider the path energy as primary and the location energy as an incidental feature of a particular parameterization of the path energy. In this case, our method is most naturally viewed as a *unnormalized* EBM distinguished by two features: first, the nonlinearity of our energy in the parameters; and second, the lack of a loss function[2]. Linear methods with regularization, such as MMP/M3N [17, 21] require a loss function in order to avoid the trivial solution of zero energy everywhere; furthermore, the MMP/M3N objective is a convex upper bound of the loss, which leads to loss-based bounds on generalization error. Although our method is similar to *structured perceptron* [5] in its lack of a loss function, structured perceptron is dissimilar in that it additionally lacks an explicit regularization term, necessitating additional tricks such as voting, averaging, or early stopping are necessary to yield good generalization [5].

Our decision to omit the loss and use regularization is motivated by several observations. First, although regularization drives the parameters towards zero, this generally *increases* energies due to our nonlinearity, preventing the global energy collapse that would happen in the linear case. Moreover, significant computational efficiency gains are realized by omitting the loss, since solving the loss-augmented inference problem in our case requires solving a more general optimization that likely does not admit lower bounds as tight as those derived in Sec. III-C. However, there is a deeper issue that motivates the omission of the loss, which is the fact that no particular choice of loss seems appropriate for this problem. The following alternative interpretation of our method may clarify the issue.

2) Normalized EBMs: If we consider the location energy as the "primary" energy, then our method can be interpreted as a *normalized* EBM, where energies are normalized across outputs and are hence interpretable as probabilities. When the energies are linear in the features, normalized EBMs coincide with MaxEnt models, which include logistic regression, Conditional Random Fields [11], and MaxEnt IOC [23, 24]. In particular, our method can be derived from a logistic regression view, since Eq. (7) is equivalent to a multiclass logistic regression model, where each location corresponds to a class. Unlike a typical logistic regression approach, though, we cannot fit the model from observations of (feature, class) pairs, since we are only able to observe optimal searcher paths; therefore, we introduce the concept of a path energy in order to constrain the location distribution such that the observed search paths appear optimal. In this view, adding loss to these constraints is not appropriate, as it is not observable: in other words, we observe that some paths must be optimal, but we do not observe that they are optimal by any margin.

Finally, we note that an alternative that we did not pursue consists of normalizing the path energy over paths, thus yielding a method similar to MaxEnt IOC [23, 24]. In this case, we would maximize the likelihood of the observed paths, assuming a log-likelihood proportional to Eq. (2). The main disadvantage of this approach is having to compute the normalization factor (or *partition function*). Although the partition function may be computed via value iteration in path-structured energies such as ours [11, 23, 24], the expense of doing so scales quadratically in the size of the state space—which in our case is already exponential in the number of locations. By contrast, our method generally requires less than a single pass over the state space to perform inference, since we can optimize the path energy via heuristic-guided search as described in Sec. III. Although we did not try to compute the partition function, we expect that the practical expense of doing so would be comparable to the value-iteration-based planning approach of [12], which was implemented and analyzed in Sec. VI-A.

B. Other learning approaches

Despite having a name similar to our work, the "learning to search" method proposed in [18] is an extension of MMP applied to the problem of learning a fast planner from a slow one—that work is not related in any way to the problem of locating things.

An example of work that is conceptually related to ours is that of Joho et. al. [8, 9]. That work is also concerned with efficiently locating objects. Two different learning-based approaches are considered: one that learns a reactive search policy from search demonstrations, and another that learns MaxEnt priors over target locations from example environ-

[2]Note that the literature is inconsistent in the precise usage of the term *loss function*. Here, we use it in the sense that it is used in the max-margin structured prediction literature [22].

ments. The first case is similar to ours in that we assume a similar type of training data; however, the approach taken is vastly different. While they directly learn a mapping from features to policy, we learn a mapping from features to priors, and then act optimally given the prior. The second approach is fundamentally different from our method because the training data consists of environments instead of paths. However, we coincidentally also use a MaxEnt-type distribution to represent our prior. Another example of work with similar motivation to ours is [10], which focuses on finding novel objects by building probabilistic models from co-occurrence data. Our work again differs mostly in that we learn from demonstrated searches as opposed to statistics about environments. We also focus on generalizing over environments instead of target objects, although it is straightforward to apply our method to the latter problem as well.

C. Planning optimal search tours

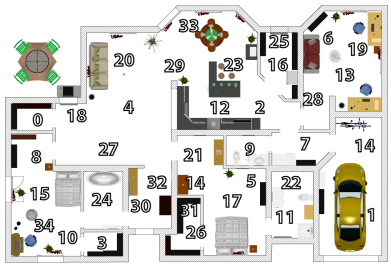
Another vein of related work is that concerned with the problem of planning expected-time-optimal search tours. The aforementioned work of [12] solves the same planning problem that we solve, but via a less efficient dynamic programming method than the one we employ—see the discussion in Sec. III and the experiments for more details. Similar but less related are a few examples of work concerned with finding locally optimal search paths in continuous spaces [14, 19]. Vice-versa, there is a body of earlier work focusing mostly on finding optimal search policies for purely discrete problems with no sense of continuity [4, 20]. We leverage one of the results from this field in order to develop one of our heuristics, as described in Sec. III-C.

VI. Experiments

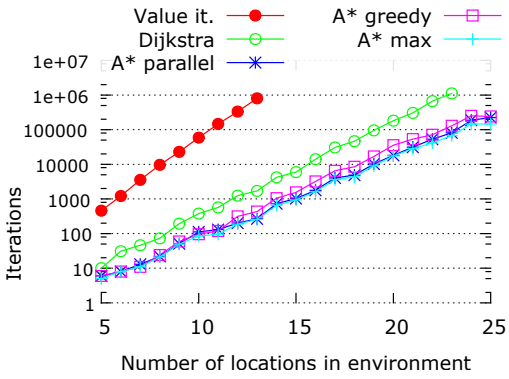
A. Planner evaluation

We first evaluated our proposed planning method empirically by examining its performance in a particular environment as we scaled the number of locations included in it. The environment used for this experiment is depicted in Fig. 4a. The prior Q was drawn from a Dirichlet distribution with all parameters set to 1 and then fixed for all trials. The results are shown in Fig. 4b. In the figure, an 'iteration' refers to a value update in value iteration or a node expansion in A*, both of which are associated with approximately equal work.

The first observation we make is that the value iteration method of [12] is significantly slower than our graph-based methods, as expected. Even without using heuristics, the graph-based method was orders of magnitude faster for all problem sizes. The difference between using heuristics and not was less pronounced but still substantial. None of the heuristics was able to counteract the exponential dependence of the number of nodes expanded on the number of locations. However, the number of nodes expanded decreased by a factor exceeding 10 when using the heuristics on large problems, compared to using no heuristic. Somewhat surprisingly, both of the heuristics performed very similarly, although the parallel heuristic performed slightly better than the greedy heuristic.



(a) The environment used to evaluate the planner



(b) Planner scaling results (note log scale of y axis)

Fig. 4: Results of planner performance evaluation experiment. Each line corresponds to a different planning method. *Value it.* refers to the value iteration method of [12]. *A* parallel* indicates A* with the parallel search heuristic, and *A* greedy* indicates A* with the heuristic obtained by relaxing arc source constraints, as described in Sec. III-C. *A* max* indicates the heuristic obtained by taking the max of the two heuristics. *Dijkstra* refers to Dijkstra's algorithm (A* with zero heuristic).

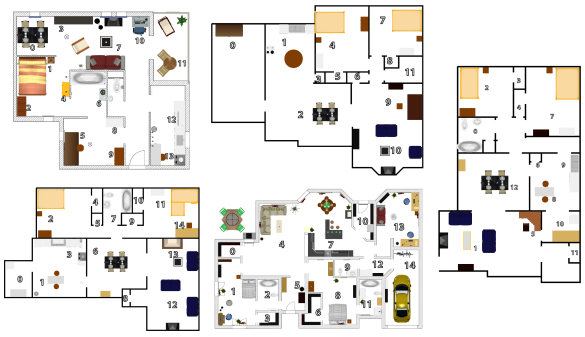


Fig. 5: The environments used in the experimental evaluation.

Taking the max of the two heuristics produced only a very modest improvement over using either one in isolation.

B. Adaptation to the home assistant scenario

We implemented the learning algorithm and applied it to the aforementioned home-assistant scenario. We collected the set of synthetic environments depicted in Fig. 5. Each environment was annotated with between 12 and 15 locations, and each location was tagged with between one and 10 tags describing it. These tags consisted mostly of room types and

objects imagined to be present at the location.

This information was used to generate a feature vector for each location in the following way. We created a set of eight semantic clusters consisting of words associated with prototypical locations. For instance, one cluster consisted of the words {office, desk, chair, laptop}, and another consisted of {garage, car, tools, shovel, ladder}. Each element in the feature vector for a location then consisted of a semantic overlap score between the tags assigned to the location and one of the semantic clusters. This overlap score was computed using a rudimentary semantic word similarity score, which we defined as $1/(1+d)$, where d denotes the minimum tree distance between senses of the words in the WordNet database [15]. Denoting this similarity score by S, the overlap score was computed as follows. We first defined a raw overlap score between word clusters C_0 and C_1:

$$\text{RawOverlap}(C_0, C_1) := \frac{1}{2}\left(\sum_{x_0 \in C_0} \max_{x_1 \in C_1} S(x_0, x_1) + \sum_{x_1 \in C_1} \max_{x_0 \in C_0} S(x_0, x_1)\right).$$

The final overlap score was then defined as the following normalized score:

$$\text{Overlap}(C_0, C_1) = \frac{\text{RawOverlap}(C_0, C_1)}{|C_0| + |C_1| - \text{RawOverlap}(C_0, C_1)}.$$

For this experiment, we also modeled a preference for short tours by assuming the demonstrator optimized the objective (14), as described in Sec. IV-B. We used the regularization term $R(w) := \|w\|^2$ in the learning objective.

C. Learning results

In order to evaluate the generalization ability of the learning algorithm, we first generated synthetic priors for each environment. Three types of priors were generated for each environment according to some simple rules. These are depicted in Fig. 6. The priors were then used to generate expected-time-optimal search tours, which were subsequently treated as training data for our method, yielding five training paths for each of the three scenarios.

It is important to emphasize the fact that we would not use our method in this way in an actual application, since our key assumption is that it is more difficult to obtain priors than search paths. However, since we cannot evaluate the actual efficiency of the search plans generated by our method without knowing the true prior, we resort to the method of generating plans from known priors for the purpose of evaluation only.

We evaluated our learning method on each scenario independently by running leave-one-out cross-validation on each training set. We first examined the convergence of the learning optimization. Results are shown in Fig. 7. In summary, we witnessed no convergence problems with appropriate step sizes. The objective did oscillate significantly across iterations, but this is typical and expected in nondifferentiable optimization.

The learning results were evaluated by computing the ratio of the true score of the learned path to the true score of the best

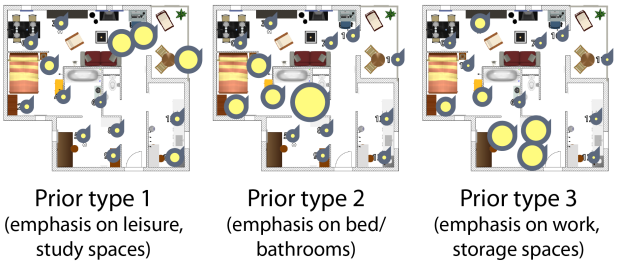


Fig. 6: A depiction of the three types of synthetic prior used in the evaluation. The radius of each circle is proportional to the strength of the prior at its location. The prior is only depicted for one of the five environments.

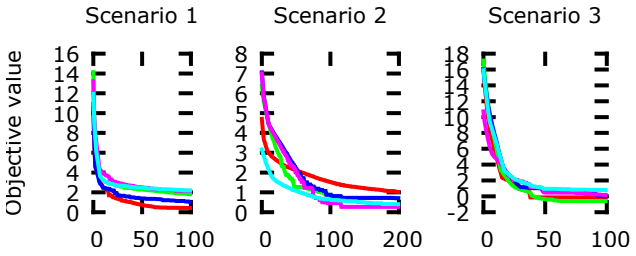


Fig. 7: Convergence of the learning objective for each of three tested scenarios. Each plot shows the convergence of the objective for all five cross-validation folds. For clarity, each line plots the best objective found up to the current iteration.

path given the true parameters. To be more precise, each held-out example consisted of an environment $\hat{e}$, a true prior $\hat{Q}$ and a true length coefficient $\hat{\alpha}$. Let w and α denote the parameters learned for the corresponding cross-validation fold. Defining the following quantities:

$$S(x, Q, \alpha) := E[T_d(x) \mid Q] + \alpha \sum_{i=1}^{N-1} c(x_i, x_{i+1}) \quad (15)$$

$$x_l := \arg\min_x S(x, Q(\hat{e}; w), \alpha) \quad (16)$$

our evaluation metric consisted of

$$\frac{S(x_l, \hat{Q}, \hat{\alpha})}{\min_x S(x, \hat{Q}, \hat{\alpha})}. \quad (17)$$

These results are shown in Fig. 8. The *learned* column indicates the result obtained by our learning method. Several other baselines are provided for reference. *greedy (oracle)* indicates the result obtained by greedily visiting the location maximizing the prior-to-distance ratio, given the true prior. *greedy (learned)* indicates the result using the same greedy policy, but under the learned prior. *blind* is the cost of an optimal tour of the environment, ignoring the prior. Finally, *reactive (cheating)* is the reactive learning approach of [9]: training an ID3 decision tree [16] to directly predict search actions. The tree was pruned using a combination of max-depth pruning and reduced error pruning, using the *test set* to optimally prune the tree, thus giving this method an unfair

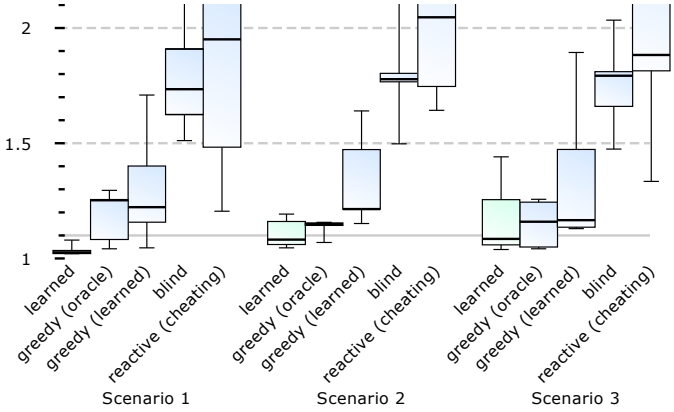


Fig. 8: Results of cross-validation experiment. Box plots show median, min, max, and quartile values across folds—as there were only five folds, each box plot line corresponds to an actual trial value. See text for details.

advantage. The features used were discretized versions of those used in our method.

The results show that in a majority of the trials, the true cost of the optimal plan under the learned parameters came within 10% of the cost of the optimal plan given the true parameters and was always within 50%. By comparison, the median performance of the greedy (oracle) policy, despite knowing the true parameters, varies to within 15% to 25% of the best possible cost. The other baselines show that both the learning and optimal planning components are critical: if we learn without planning optimally or vice-versa, performance is poor. The poor performance of reactive policy learning shows that learning suffers if it cannot take into account the future consequences of actions.

Qualitative results taken from scenario 2 are shown in Fig. 9. These specific examples were chosen because the search paths are simple enough to visualize clearly. We observe a strong correlation between the learned and held-out priors, and the search paths generated from the learned priors appear intuitive. In the top figure, for instance, the path begins in the kitchen and proceeds in a generally clockwise direction. Although the bathroom and bedroom collectively have the highest probability of containing the target, a few less-likely locations are visited opportunistically on the way.

VII. Conclusion

We have demonstrated a method for learning to efficiently locate targets from demonstrated search paths, and we have improved on existing methods to solve the associated inference problem of planning expected-time-optimal search tours by proposing novel heuristics for heuristic-driven search. Although our method is inspired by and similar to max-margin structured prediction methods in practice, it differs in crucial ways. We have elucidated some of these differences and proposed alternate motivations for our approach. Experimentally, we applied our method to a home-assistant robot scenario. We validated our performance gains in the planning problem, the

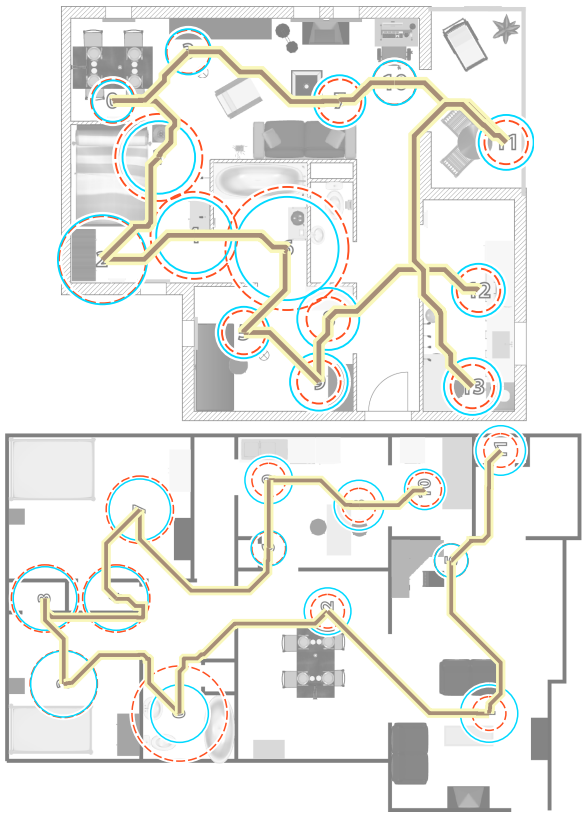


Fig. 9: Visualization of selected learning results evaluated on held-out data from scenario 2. Each location is circled with a dashed circle indicating the strength of the held-out prior there and a solid circle indicating the strength of the prior inferred by learning, with area proportional to the prior strength. The optimal search path given the learned prior is also shown.

convergence of our learning method, and the generalization performance of the learning results. The search paths generated by the learning method were observed to be nearly as efficient as optimal paths given the latent prior.

There are several other potential applications of this work that we would like to explore. We envision applications in large-scale, outdoor, and potentially hostile environments; scenarios such as rescue operations and finding explosives come to mind. On the theoretical side, we believe that this work raises a number of interesting questions for future research as well; particularly, the fact that the method combines aspects of both normalized and unnormalized energy-based methods is interesting and warrants further contemplation.

Acknowledgments

This work was conducted (in part) through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. This work was supported in part by ONR under MURI grant ‘Reasoning in Reduced Information Spaces’ (no. N00014-09-1-1052).

REFERENCES

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows: theory, algorithms, and applications*, chapter 4. Prentice Hall, 1993.

[2] Richard Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[3] Dimitri P Bertsekas. Nonlinear programming. 1999.

[4] Milton C Chew. A sequential search procedure. *The Annals of Mathematical Statistics*, 38(2):494–502, 1967.

[5] Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

[6] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[7] E.T. Jaynes. Information theory and statistical mechanics. *The Physical Review*, 106(4):620–630, 1957.

[8] Dominik Joho and Wolfram Burgard. Searching for objects: Combining multiple cues to object locations using a maximum entropy model. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 723–728. IEEE, 2010.

[9] Dominik Joho, Martin Senk, and Wolfram Burgard. Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems*, 59(5):319–328, 2011.

[10] Thomas Kollar and Nicholas Roy. Utilizing object-object and object-scene context when planning to find things. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2168–2173. IEEE, 2009.

[11] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

[12] Haye Lau, Shoudong Huang, and Gamini Dissanayake. Optimal search for multiple targets in a built environment. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3740–3745. IEEE, 2005.

[13] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. In *Predicting structured data*. MIT Press, 2006.

[14] Markku Lukka. On the optimal searching tracks for a moving target. *SIAM Journal on Applied Mathematics*, 32(1):126–132, 1977.

[15] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[16] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[17] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

[18] Nathan D Ratliff, David Silver, and J Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009.

[19] Alejandro Sarmiento, Rafael Murrieta-Cid, and Seth Hutchinson. An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments. *Advanced Robotics*, 23(12-13): 1533–1560, 2009.

[20] Lawrence D Stone. *Theory of optimal search*. Academic Press New York, 1975.

[21] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press, 2004. URL http://papers.nips.cc/paper/2397-max-margin-markov-networks.pdf.

[22] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005.

[23] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438, 2008.

[24] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Proc. of the International Conference on Intelligent Robotsi and Systems*, 2009.