Planning with Actively Eliciting Preferences

Mayukh Das[a,*], Phillip Odom[b], Md. Rakibul Islam[c], Janardhan Rao (Jana) Doppa[c], Dan Roth[d], Sriraam Natarajan[a,b]

[a] *University of Texas at Dallas, Richardson, Texas, USA*
[b] *Indiana University, Bloomington, Indiana, USA*
[c] *Washington State University, Pullman, Washington, USA*
[d] *University of Pennsylvania, Philadelphia, PA, USA*

Abstract

Planning with preferences has been employed extensively to quickly generate high-quality plans. However, it may be difficult for the human expert to supply this information without knowledge of the reasoning employed by the planner. We consider the problem of actively eliciting preferences from a human expert during the planning process. Specifically, we study this problem in the context of the Hierarchical Task Network (HTN) planning framework as it allows easy interaction with the human. We propose an approach where the planner identifies when and where expert guidance will be most useful and seeks expert's preferences accordingly to make better decisions. Our experimental results on several diverse planning domains show that the preferences gathered using the proposed approach improve the quality and speed of the planner, while reducing the burden on the human expert.

Keywords: Active Preference Elicitation, Human-in-the-loop, Planning, HTN, Human-Agent interaction

1. Introduction

Planning under uncertainty has exploited human (domain) expertise in several different directions [1, 2, 3, 4, 5, 6]. This contrasts with traditional learning techniques that require large amount of labeled data and treat the human as a "mere labeler". One key research thrust in this direction is that of specifying preferences as advice to the

*Corresponding author

planner in order to constrain the search over the space of plans. While successful, most of the preference specification approaches require that the human input be provided in advance before planning commences. There are at least two main issues with this approach: (1) the human sometimes provides the most "obvious" advice that can be potentially inferred by calculating the uncertainty in the plan space, and (2) the planner may not reach the part of the plan space where the preferences apply.

We propose a framework in which the planner actively solicits preferences as needed. More specifically, our proposed planning approach computes the uncertainty in the plan explicitly and then queries the human expert for advice as needed. This approach not only alleviates the burden, of specifying all the advice upfront, on the human expert but also allows the learning algorithm to focus on the most uncertain regions of the plan space and query accordingly. Thus, it avoids human effort on trivial regions and improves the relevance of the preferences.

We present an algorithm for active preference elicitation in planning called the *Preference-Guided Planner* (PGPLANNER) where the agent treats the human advice as *soft preferences* and solicits these preferences as needed. Such preferences guide the search process towards, potentially, better quality plans.We consider a Hierarchical Task Network (HTN) planner for this task as it allows for seamless natural interaction with humans who solve problems by decomposing them into smaller problems. HTN planners can facilitate humans in providing knowledge at varying levels of generality. We evaluate our algorithm on several standard domains and a novel blocksworld domain against several baselines. Our results show that this collaborative approach allows for more efficient and effective problem solving compared to both standard planning techniques as well as preference-based counterparts with all preferences provided in advance.

Contributions: Our key contributions include: (1) We introduce active preference elicitation for HTN planning; (2) Our framework treats the human input as soft preferences and allows for a trade-off between potentially a sub-optimal expert and a complex plan space; and (3) We evaluate our algorithm on several tasks and demonstrate its efficacy against other baselines.

2. Background and related work

Given an initial state and a goal specification as an instance of a planning task, automated planners produce a sequence of actions (aka. plan) to satisfy the goal specification. The most basic form of automated planning is computationally hard (specifically, PSPACE-complete [7]). However, real-world applications require fast planners to satisfy time constrains. Consequently, there is a large body of work to address this challenge [8]. Some representative approaches include reduction to SAT solving [9, 10, 11], forward state space search with human-designed heuristics [12, 13], learned heuristics from solved planning problems [14, 15, 16]; planning with human-written control knowledge [17, 18] and solving probabilistic planning via reduction to deterministic planning [19]. Learning-based and knowledge-based approaches have seen great success, but they assume a fixed knowledge representation and can be brittle.

Our PGPLANNER relates closely to mixed-initiative planning [20, 21, 22], which interleaves planning by expert with automated planning. However, they are conceptually different in the expert's role and intervention in the planning process. Mixed-initiative planning is a negotiation between an agent and human on mutable goals/subgoals, and partial plans. Such an intervention can be initiated by either party. On the other hand, PGPLANNER is responsible for only acquiring expert knowledge wherever it is expected to be most useful. Intuitively, in PGPLANNER, the responsibility to seek human intervention lies with the planner itself. PGPLANNER builds on the HTN planning framework since it allows the human to encode coarse knowledge and subsequently learn fine-grained knowledge via interaction.

2.1. HTN Planning

An HTN planner [8], one of the well-known neo-classical planners, searches for valid plans in the space of task decompositions. It recursively decomposes the current task into sub-tasks based on pre-defined control knowledge, called *methods*, and adds the new sub-tasks into the current set. If a primitive task is solvable by an atomic action, it is removed from the set of tasks; the corresponding action is then added to the plan, and the state is updated. Remaining *non-primitive* tasks are then decomposed further. The resultant network of decompositions is a task network [8]. Formally,

Definition 1. *A task network is directed acyclic graph $\mathbb{W} = (N, E)$. A directed edge $e = \langle \tau, \tau^{(sub)} \rangle$, where $\tau, \tau^{(sub)} \in N$ and $e \in E$, is always from a task τ to one of its sub-tasks $\tau^{(sub)}$ (i.e. $\tau^{(sub)} \in subtasks(\tau)$).*

Definition 2. *A method is a tuple $m_\tau = (\tau, \mathcal{F}(a, s), \{\tau_j\}_{j=1}^{k})$ where τ is the task for which m_τ is applicable, $\mathcal{F}(a, s)$ ensures m_τ is admissible (when s satisfies a) in current state s (a is some admissibility criteria) and $\{\tau_j\}_{j=1}^{k}$ is the set of sub-tasks to which the task τ will be decomposed on application of m_τ.*

Note that, the above definition formalizes admissibility[1] of methods in a generalized fashion, independent of representational syntax. In HTN domain descriptions, however, admissibility of methods are represented as preconditions or conjunctive formulas in predicate logic that the current state 's' must satisfy.

Definition 3. *A hierarchical task network problem is defined as $P = (s_0, w_0, O, M)$ where s_0 is the initial state, w_0 is the initial task network, O is the set of operators (atomic actions) and M is the set of decomposition methods.*

As an intuitive example of how HTNs facilitate human experts in specifying knowledge/feedback, consider the problem of building a house. The primary task is "Build House" which can be decomposed into subtasks: *"Build House"* $\rightarrow$ *["Build Foundation", "Build Walls", "Build Roof"]*$_1$. Again the subtask "Build Foundation" can be decomposed further: *"Build Foundation"* $\rightarrow$ *["Dig x feet", "Reinforcement Bars", "Pour Concrete"]*$_2$ (subscripts indicate decomposition level). Methods guide how such tasks need to be decomposed. Clearly, the varying levels of task abstraction allow humans to provide feedback at different levels of generality. For instance, a human could say "Base needs to be deeper than 10 feet" at the *"Dig x ft"* level (level 2) or the human could also say "Foundation must be $1/3^{rd}$ the height of the house" at the higher *"Build Foundation"* task level (level 1). As we explain the formalism and evaluation of our approach through the following sections it will be clearer that the strength of PGPLANNER essentially lies in its ability to identify the most appropriate level of generality for preference elicitation.

[1] Not to be confused with admissible heuristics.

2.2. Preference Elicitation

There has been a surge in interest towards using human experts to improve decision-making [6, 23, 24, 25, 26]. While distinct methods differ in the form of knowledge that can be specified, preference elicitation has been explored inside automated planning [2, 27, 3, 5, 6], reinforcement learning (RL) [28, 29, 30] and inverse reinforcement learning (IRL) [24, 26].

Preferences used in our approach correspond to IF-THEN statements where the IF defines the conditions (without negation) under which the preferences should apply and the THEN represents the preference. In RL or IRL, preferences could represent sets of preferred/non-preferred actions in a given set of states [31]. In HTN planning, preferences could correspond to preferred/non-preferred methods for decomposing certain tasks [23]. Across these approaches, preferences have shown to be a good choice for specifying expert knowledge.

However, many of these approaches require all of the preferences/advice upfront. This requires the domain expert (who may not have machine learning expertise) to provide the knowledge that would be useful for the learner. *Alternatively, our approach solicits preferences during planning only as needed. This allows for effective interaction by reducing the number of uninformative queries*.

Recent work on active advice-seeking [26] introduces a framework to allow the learning algorithm to request preferences over interesting areas of the feature or state space. To the best of our knowledge, our approach is the *first to actively solicit preferences in the planning setting*. While our proposed solution shares conceptual connections with Sarne et. al.'s work on multi-agent planning & scheduling with mixture of human and autonomous agents [32], the difference however lies in the problem setup. Their approach explicitly estimates the value of (future) additional information from humans about the feasibility of given options in order to generate queries. Our formulation does not assume existence of a "coordination" module making estimation of the value of information impossible. As a surrogate for such an estimate, PGPLANNER instead identifies regions in task space where it has inadequate information. We show that our formulation facilitates human expert in providing more useful information, not limited to feasibility of given options.

Key contributions: Unlike active advice-seeking [26], planning does not have training examples from which to generalize. Instead, we select points to query during the planning process based on estimated quality of the available options. We show that even when learning without examples, an active learning framework can guide the learner towards effective and efficient communication with domain experts.

3. Active Preference-Guided Planning

Preference-guided planning employs preferences to guide the search through the space of possible plans, sequence of primitive actions. We make use of HTNs to search through these plans by recursively breaking a higher-level task into sub-tasks until every task can be solved using primitive actions. Our work differs from prior research on preference-based planning [6, 23] in two distinct ways: 1) Our preferences are not used to define the best plan. Instead, they guide the decomposition of the network to efficiently find high-quality plans; and 2) We aim to actively acquire preferences as needed during the search process as opposed to requiring the preferences upfront. Our preferences are formally defined as:

Definition 4. *A user-defined preference is a tuple* $\mathfrak{P} = (\wedge f_i, \tau_j, M^+_{\tau_j}, M^-_{\tau_j})$*, where* $\wedge f_i$ *corresponds to conditions of the current state under which the preference should be applied*[2]*,* τ_j *is the relevant task,* $M^+_{\tau_j}$ *is the set of methods which are in the user's preferred set and* $M^-_{\tau_j}$ *is the set of methods which are in the user's non-preferred set.*

A preference can be considered an IF-THEN rule where $\wedge f_i$ corresponds to the conditions which the current state should satisfy for preference to apply to a particular task, τ_j, and $M^+_{\tau_j}/M^-_{\tau_j}$ represents the method(s) preferred/non-preferred by the user. It is important to note that a preference may be defined for (1) all instances of a particular task ($\wedge f_i = true$), (2) only a single instance of a task or, (3) any level of abstraction in-between.

Our approach uses these preferences to guide the search through the space of possible decompositions in the HTN. Consider the network shown in Figure 1. Each node

[2]We use $\wedge f_i$ to denote that this could be a set of multiple conditions

in the network is labeled by the current state - the current configuration of the blocks - and task. For example, the root node represents the task τ_1 of clearing block B in the state where block F is on A, A is on B, etc. Note that we use predicate notation internally to represent the states. The edges in the network represent decompositions and are labeled with the method name. Method m_1 breaks task τ_1 into the operator $PutOnTable(F)$ and, recursively, the task $clear(B)$.

Example 1. *Figure 1 represents a preference in Blocks World.*

$$\mathfrak{P} = (Space(Table), Clear(B), \{PutOnTable\}, \{StackonE\})$$

Shaded green areas represent preferred decomposition while shaded red areas represent non-preferred decompositions.

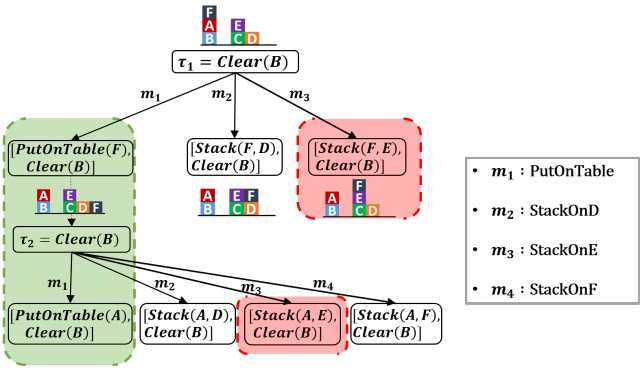


Figure 1: Preference guided search in a Blocks world problem. Rectangular nodes signify a set of task(s) to be solved. Admissible methods for decomposing a task τ_1 are m_1, m_2 and m_3. Note, in the lower sub-tree we have an additional admissible method m_4. Block configuration pictures signify current state. The green and red shaded areas denote preferred and non-preferred decompositions (best viewed in color).

The preference represents the intuition that it is easier to build arbitrary towers when all the blocks are on the table as they can be positioned quickly. As specified, this single preference can apply at multiple points during the search. The effect of the preferences is to update the distribution that increases the probability of the methods M^+ and decreases the probability of the methods M^-. The subtree for M^+ is

highlighted in green while the subtree for M^- is shown in red. While a similar preference can be given upfront, our active approach evaluates whether this preference is necessary by estimating the quality of each method. Therefore, our approach increases the value of each preferences by reducing redundancy that may be present in upfront preferences.

In the context of acquiring preferences from the expert, although we assume availability of the expert throughout the planning process, we aim to rely on him/her only when necessary. This setting is similar to stream-based active learning [33] where examples are shown online and the algorithm must decide whether to query for a label for the example or ignore it. In contrast to an acquired label, our preferences are more general allowing the expert to prefer/non-prefer methods for decomposition. A query is solicited over the current state s_n and the current task t_n,

Definition 5. *A query is defined over an HTN node n as a tuple $q_n = (s_n, \tau_n)$.*

An expert's response to a query is a preference. As HTNs are hierarchical, the expert is not restricted to providing preferences only over the current state/task. It could also be defined over any subset of the state space that contains s_n. The expert selects the proper generality of the preference. When the space is factored, this involves removing features or introducing variables in description of s_n.

Example 2. *In Figure 1, $Clear(B)$ at the root has 3 different decomposition choices all of which may seem equally valuable to* PGPLANNER *and generate the query: $q = (< Space(Table), On(A, B), On(F, A), ... >, \tau = Clear(B))$. The expert may provide the preference specified in Example 1. Note that in that case, the expert gives a general advice that applies to any state in which there is space on the table.*

Figure 2 illustrates the overall architecture of the PGPLANNER framework. Briefly, at every step (when a task τ needs to be decomposed), PGPLANNER performs a bounded-depth roll-out (simulation of the subsequent decompositions) to evaluate the quality of each method. This allows for estimating a distribution over all methods that can decompose τ ($\mathcal{M}_\tau$). If the uncertainty in this distribution is too high (above ϵ) PGPLANNER decides to seek human guidance via the interface. Human input (preference rule $\mathfrak{P}$) is used to update the distribution. The best method is then selected for

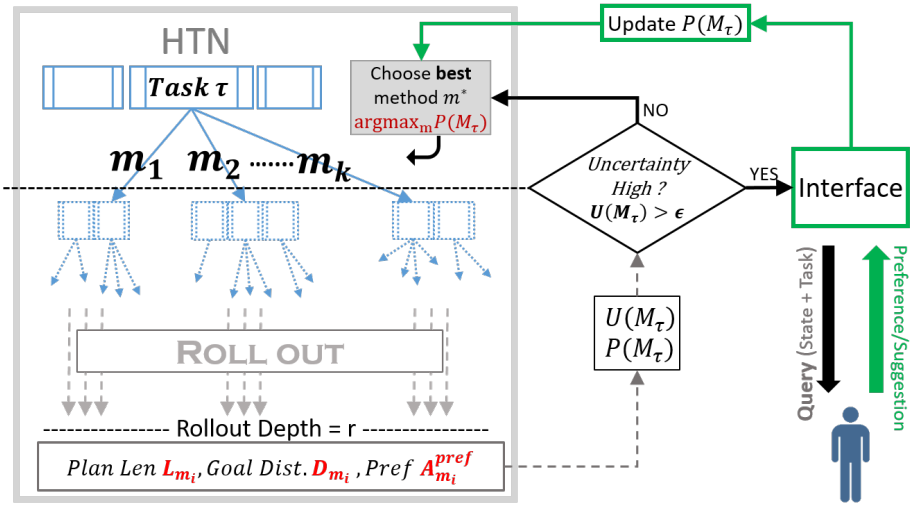


Figure 2: Overall architecture of PGPLANNER. At an HTN node, τ represents the task, $M_\tau = \{m_1, m_2, \ldots, m_k\}$ are the methods that can decompose τ. Roll-out allows for estimating distribution over methods and uncertainty $(P(M_{tau}), U(M_\tau))$. Acceptable uncertainty threshold is ϵ. Preference from human expert is used to update the distribution. Method distribution is used to choose the best method.

proceeding with the decomposition, based on the distribution. The illustration, essentially depicts a snapshot of the decision making pipeline of PGPLANNER. This process occurs at every HTN node that is processed. Subsequent sections elaborate the theory and the functionality of the components of this framework.

3.1. Problem Overview

PGPLANNER finds a plan given an HTN problem, defining the initial state/goal task(s), and access to an expert. The goal of PGPLANNER is to find the policy π, a distribution over the methods for each HTN node n, such that the best plan is reached:

$$\arg\min_{\pi} \ (J(\pi) = T\mathbb{E}_{n\sim d_\pi} C_\pi(n)) \tag{1}$$

$J(\pi)$ is the total expected cost of finding a plan. If π is used to select decompositions, d_π represents the distribution of HTN nodes reached. T is the depth of the decomposition. $C_\pi(n)$ is the expected cost at node n. $C_\pi(n) = \mathbb{E}_{m\sim\pi_n} C(n, m)$ where $C(n, m)$

Algorithm 1 Preference-Guided Planning

procedure PGPLANNER(s_0, w_0, O, M)
  $Frontier \leftarrow$ all nodes in w_0, $Plans = \emptyset, \mathfrak{P} = \emptyset$ ▷ $\mathfrak{P}$ denotes preference set
  while $Plans == \emptyset$ and $Frontier \neq \emptyset$ **do**
    $currPlan \leftarrow$ RECURSEARCH($\emptyset, Frontier$)
    if $currPlan \neq \emptyset$ and $currPlan \neq NULL$ **then**
      $Plans \leftarrow Plans \cup currPlan$
    end if
  end while
  return $Plans$
end procedure

is the immediate cost of selecting m at node n. If the planner aims to find the shortest plan, then the immediate cost C is the number of actions added to the current plan. PGPLANNER's objective is find the best method distribution (with expert guidance) for any task such that the total expected cost is optimized.

PGPLANNER, Algorithm 1 (along with the called procedures 2 & 3), recursively searches through the space of possible HTN decompositions to reach a valid plan. Each node n in the HTN with task τ_n could potentially decompose in several ways according to the available methods (M_{τ_n}). The cost of selecting a method $m \in M_{\tau_n}$ ($C_\pi(n)$) is estimated by rolling out the current plan and then approximating the distance to the goal. The methods are also scored according to the current set of preferences. The overall cost estimate of a method m ($\hat{C}(m)$) is a combination of this preference score and the estimated cost function. Finally, this is converted into a probability distribution (π) over the methods M_{τ_n}. If this distribution has a high-level of uncertainty (entropy in our case), the expert is queried about the current set of possible methods.

3.2. The PGPLANNER *Algorithm*

The PGPLANNER maintains a $Frontier$, the set of all HTN nodes that have to be explored. It is initialized with 1 or more nodes containing the goal task(s). PGPLANNER proceeds by recursively decomposing the task τ_n of the node n at the head of the frontier and inserting new nodes for the sub-tasks of τ_n. The methods of non-primitive

Algorithm 2 Recursive Search

procedure RECURSEARCH($currPlan, Frontier$)
  if $Frontier \neq \emptyset$ **then**
    n = POP($Frontier$)
    if τ_n is *non-primitive* **then**
      $\pi(M_{\tau_n}), U(M_{\tau_n}) \leftarrow$ EVALNODE($n, \mathfrak{P}$)
      if ***Not*** ACCEPTABLEUNCERTAINTY(U,n) **then**
        $\mathfrak{P} \leftarrow \mathfrak{P} \cup$ QUERYEXPERT(m, s) ▷ Generate query and acquire preference
        $\pi(M_{\tau_n}), U(M_{\tau_n}) \leftarrow$ EVALNODE($n, \mathfrak{P}$)
      end if
      $M_{cur} = M_{\tau_n}$
      while $M_{cur} \neq \emptyset$ **do**
        $M^* \leftarrow \arg\max_m \pi(M_{cur})$ ▷ $m \in M_{curr}$
        $\{n'\} \leftarrow$ DECOMPOSE(τ_n, M^*)
        $NewFrontier \leftarrow$PUSH($Frontier, \{n'\}$) ▷ Temporary frontier stack
        $retVal \leftarrow$ RECURSEARCH($currPlan$, $NewFrontier$) ▷ Recursive call
        if $retVal \neq NULL$ **then**
          return retVal ▷ Backtracking
        end if
        $M_{cur} \leftarrow M_{cur} - M^*$
      end while
    else
      if $Success\,(apply(currPlan, s_n, a_{\tau_n}))$ **then** ▷ Applying primitive action
        return $currPlan$
      end if
    end if
  end if
  return NULL ▷ Backtracking
end procedure

Algorithm 3 Evaluate Node

1: **procedure** EVALNODE(HTN node n, Preference $\mathfrak{P}$)
2: $\hat{C} = \emptyset$ ▷ Set of scores $\forall m : m \in M_{\tau_n}$
3: **for *each*** $m \in M_{\tau_n}$ **do**
4: Node $r_m \leftarrow$ ROLLOUT(m, n, d) ▷ rollout depth d is set to a constant
5: $L_m \leftarrow cost(pl_{r_m})$, $D_m \leftarrow \delta(s_{r_m}, goal)$,
6: $\mathfrak{P}(s_n) \leftarrow \forall_{p \in \mathfrak{P}} \ (s_n \vDash \wedge f_i^j) \wedge (\tau_p = \tau_n)$ ▷ applicable & state satisfies conditions
7: $A_m \leftarrow N_m^+(\mathfrak{P}(s_n)) - N_m^-(\mathfrak{P}(s_n))$
8: $\hat{C} \leftarrow \hat{C} \cup \langle m, ((D)^{-1} + (L)^{-1} + A) \rangle$ ▷ Maximize adherence & minimize cost
9: **end for**
10: Compute $\pi(M_{\tau_n})$ then $U(M_{\tau_n}) \leftarrow \sum_{m \in M_{\tau_n}} p(m).\log(1/p(m))$ ▷ Entropy from π
11: **return** $\pi(M_{\tau_n}), U(M_{\tau_n})$
12: **end procedure**

tasks are recursively selected based on $\hat{C}(m)$ (RECURSEARCH, Alg 2, lines **11-20**). Primitive tasks are solved by adding the operator to the current plan (Alg 2, line **22**). This apply step updates the plan for all ancestors of the current node.

When evaluating a node n of the HTN (EVALNODE), the methods $m \in M_{\tau_n}$ represent the set of possible choices. We estimate the cost $\hat{C}(m)$ for each method by rolling out for d steps. L_m represents the estimated cost of the roll-out on method m. In our case it corresponds to the plan length. D_m approximates cost to reach the goal state from the state after the roll-out is completed (EVALNODE, Alg 3, line **4**). This distance, denoted as δ, is the number of unsatisfied goal atoms in the current state. Along with the estimated cost, methods are also evaluated with respect to the set of preferences by the adherence score (A_m). This score (Alg 3, line **7**) is determined by the number of preferences applying to the current node n ($\mathfrak{P}(s_n)$). The number of preferences which prefer method m is represented by N_m^+ while N_m^- represents the number of preferences which non-prefer it. Notice that this formulation could allow conflicting preferences on a single method and task. The final score ($\hat{C}(m)$) is a combination of the estimated cost (L_m, D_m) and the adherence score (line **8**). We convert this score into a distribution ($\pi(n)$) over the methods using a Boltzmann softmax, $p(m) = e^{\hat{C}(m)} / \sum_{x \in M_{\tau_n}} e^{\hat{C}(x)}$ where $m \in M_{\tau_n}$ and $p(m) = \pi(n, m)$. This distribution is used in 2 ways: (1) to select

the exploration order of the methods and (2) to decide whether a query is necessary.

The query decision is based on the uncertainty over the set of possible methods (RECURSEARCH, Alg 2, lines **6-9**). Inspired by the success of Active Learning[34], we use the uncertainty measure to query the expert. However, one could replace this with any function that needs to be optimized - cost to goal, depth from the start state or a domain specific utility function etc. to name a few. PGPLANNER uses entropy computed from $\pi(n)$ as the measure of uncertainty. Our framework uses a threshold on the entropy, the ***Not*** ACCEPTABLEUNCERTAINTY ($> \epsilon$), to initiate a query. The expert can either provide decomposition preferences over all tasks of a given type, or specify preferences over a single instance of a task, or even provide a partial plan to solve a particular subtask. This establishes an expressive framework for the expert to interact with the planner. The interaction is driven by the planner, allowing it to only ask as and when needed.

Overall, PGPLANNER interacts with the expert to guide the search through the space of possible plans. In an HTN setting, our approach transforms the problem of search of possible plans to sequential decision making problem in the space of possible task decompositions . This facilitates the expert to specify preference in varying levels of generality. The active elicitation formulation ensures that expert knowledge is obtained at the correct and most relevant level of generality allowing our algorithm to learn potentially better plans in a more efficient manner. We now briefly analyze some of the properties of our formulation.

3.3. Properties of PGPLANNER

3.3.1. Difference in obtaining preference at various steps

First, we aim to quantify getting preference in earlier steps when compared to getting preference at later steps. Let us denote the probability of choosing a method according to the optimal plan as $p^o(m)$ given by the Boltzmann distribution. Recall that the cost of selecting method m at HTN node n as $C(n, m)$ and the cost of a policy π is $C_\pi(n) = \mathbb{E}_{m\sim\pi_n} C(n, m)$. Now if we use a boolean error function that is set when the method at node n is not chosen according to the policy ($e(n, m) = I(m \neq \pi^*(n))$, then the error of the policy is $e_\pi(n) = \mathbb{E}_{m\sim\pi_n} e(n, m)$.

Our goal is to minimize the total expected cost of a policy π, $J(\pi) = T\mathbb{E}_{n\sim d_\pi} C_\pi(n)$. Ross and Bagnell [35] have shown for any policy π, $J(\pi) \leq J(\pi^*) + kT\bar{\epsilon}$, where π^* is the optimal policy, T is the task horizon, k is the number of steps to the goal, and $\bar{\epsilon} = \frac{1}{T}\sum_i \epsilon_i$, where $\epsilon_i = \mathbb{E}_{n\sim d_{\pi^*}} e_\pi(n_i)$ is the expected error at node i.

A natural question is, does it benefit to ask the query early or should the planning algorithm wait to query the expert. This can be analyzed using the regret framework. Let π_i and π_j denote the policy π when asking the query at steps i and j respectively ($j > i$). Now, rearranging terms, we can show that $J_{\pi_i} - J_{\pi_j} = \Delta(j - i)$, where Δ denotes the expected change in error, if assumed to be the same in both the steps i and j. Thus, the difference between the two choices to solicit preference is linear in the time difference between the two steps, and linear in the change in error in the two steps. Here it is clear that soliciting advice early can reduce the expected total cost.

3.3.2. Benefit of Preference over rollout

We now consider briefly analyzing the value of PGPlanner vs a simple rollout based planning. Let us denote the distribution over methods of optimal policy, PGPlanner, and rollout as π^o, π^A and π^R respectively, where $p^i(m) = \pi^i(n, m)$ where $m \in M_{\tau_n}$ is the posterior of choosing a particular method according to the policy. Suppose we compute the KL divergence of the probability distribution of methods of PGPlanner and rollout with the optimal distribution,

$$D_R = D_{KL}(\pi^o||\pi^R) = \sum_{m\in M(T)} p^o(m).\ln\left(\frac{p^o(m)}{p^R(m)}\right)$$

$$D_A = D_{KL}(\pi^o||\pi^A) = \sum_{m\in M(T)} p^o(m).\ln\left(\frac{p^o(m)}{p^A(m)}\right)$$

obtaining the difference between D_R & D_A as $D_R - D_A = \sum_{m\in M(T)} p^o(m).log\frac{p^A(m)}{p^R(m)}$, which is simply a weighted sum of the log odds of the probability of choosing a method. It is possible to find the best π^A that maximizes this difference by setting $\sum_m p^A(m) = 1$ as a constraint, but this requires having access to the optimal distribution. Since that is unknown in many cases, one can simply observe that when the preferences drive the distribution over methods towards the optimal one, i.e., choose a

method that is close to the optimal, the difference is ≥ 0 indicating that the preference is more useful than the simple rollout. We next show empirically, this is indeed the case in many planning problems.

3.3.3. Performance analysis

While PGPLANNER may have an overhead with respect to space and time, owing to the *roll-out* for evaluating the quality of every method, the incurred cost is *not unbounded* and is practically much lower. The roll-out depth is limited to a pre-defined constant d (which is typically reasonably low), and if the worst-case search depth for the actually planning process is Δ then the roll-out overhead (wrt. time) is bounded at $O(\Delta.d)$. The space overhead is also nominal since all data-structures are re-initialized after each roll-out. *The worst-case space complexity of the recursive plan search in* PGPLANNER *is no worse than the original HTN planning system, SHOP2 [36], used as the core planner.*

Most importantly, the overhead cost becomes inconsequential on large problems where the combination of search depth and branching factor is very high. In such cases, actively acquired preferences are highly likely to guide the PGPLANNER away from deeper subtrees, whenever possible. Naturally, it will depend on the quality of human inputs which we assume to be reasonable. Note that the human is treated as an imperfect teacher and not as an adversary. In large problems, guidance provided by preference results in substantial pruning of the search space and, hence, gain in efficiency and performance. Additionally, as upfront preferences are provided by the expert without visibility into the current state, they may not be as effective as actively acquired ones in pruning out low quality decompositions.

4. Experiments

Our PGPLANNER is built on top of the SHOP2 [36] architecture (called JSHOP), an HTN planner. We have extended the base planner to: (1) perform a roll-out to evaluate all admissible methods for decomposing the task, (2) elicit human feedback/preferences based on our evaluation, and (3) utilize the preferences to guide the search.

Evaluation Questionnaire. Our experiments aim to answer the following questions:

Q1: Does PGPLANNER generate plans efficiently?

Q2: How effective are the generated plans?

Q3: Does active preference elicitation improve the interaction with the expert?

We compared PGPLANNER against several alternate approaches for preference elicitation including (1) *Upfront Preferences* - where all the preferences are specified before planning, similar to [6, 23], (2) *Random Query* - selecting whether to query randomly (for each step), and (3) *No Preferences* - planning without preferences. In all of the experiments, we perform the role of the experts in providing preferences. Performing user studies is an interesting future direction.

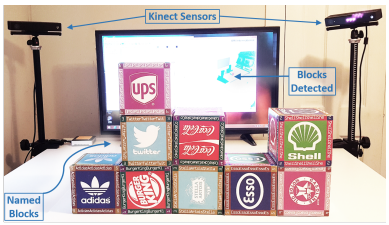


Figure 3: Blocks World Apparatus

Domains	#[Relations]	#[Objects]	#[Problems]
Freecell	5	52	20
Rovers	27	50	20
Trucks	10	32	20
Depots	6	45	20
Satellite	8	69	20
TidyBot	24	100	10
Towers of Hanoi	10	9	20
Barman	8	50	10
Mystery	12	35	10
Assembly	10	15	10
Rockets	6	15	10
Blocks World	3	40	20

Table 1: Experimental domains

We evaluate PGPLANNER on several standard planning domains as well as a novel Blocks World domain, listed in Table 1 (Number of relations, maximum number of objects and number of problems considered in each domain). There is no straightforward way to succinctly describe the complexity of the domains. However, the maximum number of objects and relations in each of them should provide a fair idea of its complexity. Experiments for the Blocks World were performed using a surrogate real-world environment, an apparatus (Figure 3) that can detect block configurations of actual named blocks via sensors. State encoding is then generated by processing the sensor data.

For all experiments, we set ACCEPTABLEUNCERTAINTY to $entropy \leq \epsilon \ (= 0.5)$.

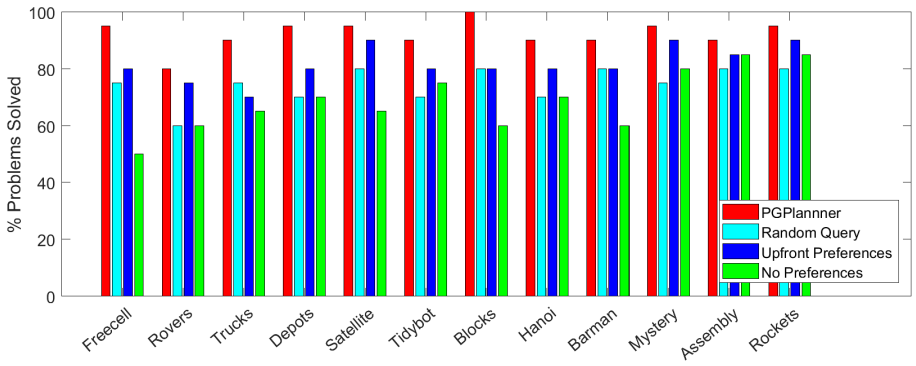


Figure 4: Efficiency comparison of all approaches across 12 domains. Percent problems solved in 10 minutes, higher is better. (best viewed in color)

We also performed line search on the value space, however, a threshold of 0.5 worked well throughout and we report results with that. Preferences were provided by the person designing and conducting the experiments. We avoid experimental bias in the quality of preferences given across all the preference-based approaches. We verify this experimentally by storing all the actively elicited preference statements in a log file and using those as the set of input preferences in the Upfront approach and observing how they affect the decision making process in both cases (see Discussion).

We have developed an interface that facilitates the interaction between the expert and the planner. The interface has three main components: the state module, the partial plan module and the interaction module to visually render the current state, to expose the presently selected set of primitives and to provide a console for the human-agent interaction respectively. The interaction module allows the planner to query the user and the user to respond to the query with a preference.

4.1. Experimental Results

In each domain, the planners were executed for 10 minutes. This allows for validating the ability of the expert to guide the algorithm to efficient solutions (as well as accommodating the limited time and attention of the expert). Figure 4 shows the percentage of problems where a plan was found. We evaluate the quality of the learned plans separately.

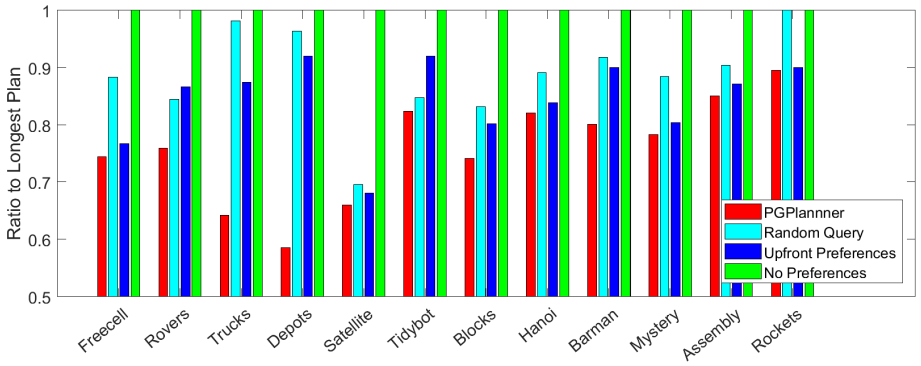
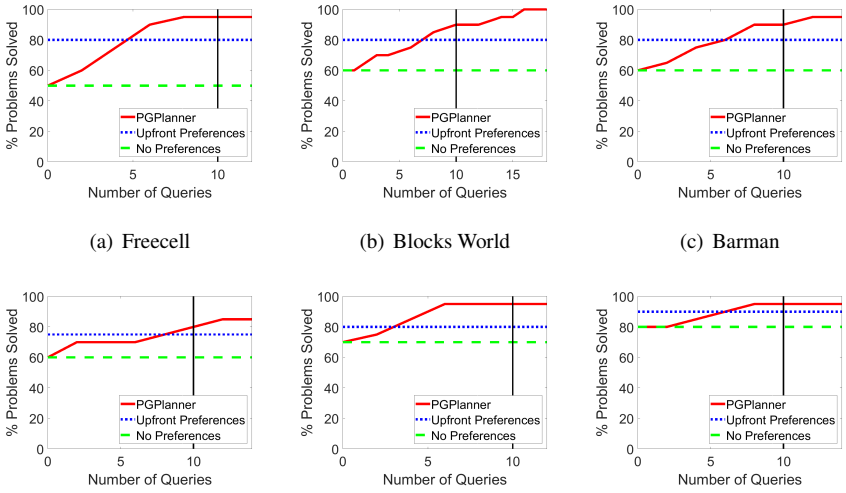


Figure 5: Performance comparison of all approaches across 12 domains. Compares the ratio of average plan lengths for every approach to the longest average plan length, lower implies better (best viewed in color)

(Efficiency). In most domains, every planner using preferences is able to outperform standard planning (no preferences). This indicates that preferences have a positive impact during planning. Planning with upfront preference outperforms randomly querying for preferences in 9 out of 12 domains. A disadvantage of random querying is that it may not query at points in planning where the preferences could have the most impact. Specifying preference upfront can take advantage of preferences at these crucial decisions at the cost of placing additional responsibility on the expert to give useful preferences. Across all domains, PGPLANNER outperforms all of the baselines. This answers **Q1** affirmatively in that actively eliciting preferences guides the planner to solutions more efficiently

(Effectiveness). Next, we investigate the generated plans. In every domain, we only compare problems where all planning methods are able to generate a plan (in the given time constraint). Figure 5 illustrates the ratio of the average plan length of each planning method compared to the average of plans generated without preferences[3]. Planning with preferences generates shorter plans in all the domains. Since, as evident, no preference always results in higher average plan lengths. The ratios for all methods

[3]The value for No Preference case is always 1, since the ratio is taken with itself.



(a) Freecell (b) Blocks World (c) Barman

(d) Rovers (e) Depots (f) Mystery

Figure 6: Learning Curves in 3 construction (top) and 3 route-finding/ordered-assignment (bottom) domains. Performance: % of problems solved, vs. the # queries. (best viewed in color).

with preferences are less than 1. However, PGPLANNER has the lowest ratio across all domains. Thus, PGPLANNER is able to produce shorter plans on an average than all of the baselines, thus answering **Q2** affirmatively as well.

(Effect of query budget). Finally, we investigate how our method performs relative to the number of queries solicited. Figure 6 shows learning curves, in 3 construction oriented domains (Freecell, Blocks World and Barman) and 3 route-finding and ordered-assignment problem solving type domains (Rovers, Depots & Mystery), respectively. Note that learning with no preferences and learning with upfront preferences are constant as the number of preferences never changes. In each domain, the x-axis represents the number of preferences given by the expert. *The vertical line denotes the point in the curve where the number of preferences given upfront equals the number of queries*. Our method outperforms learning with upfront preferences using the same number of preferences in all domains. This suggests that actively eliciting preferences succeeds in generating queries at important stages during the search process, improving the interaction with the expert (affirmatively answering **Q3**). Note that, the curves clearly

illustrate how the performance varies with the query budget. Interestingly, the rise is gradual. A sharp rise would have indicated that most of the performance gain is achieved via first few preferences towards the early stages of planning (favoring the upfront case). Instead, we observe that PGPLANNER identifies the correct junctures throughout the entire planning process at which expert guidance is most useful, allowing it to pose the optimal set of queries to the human.

4.2. Discussion

PGPLANNER is effective, since the framework elicits preferences when and where they are necessary as well as relevant. The expert can provide informative preferences to steer the search process towards more useful parts of the search space. On the other hand, upfront preferences may not always be relevant, because the preferences might be about unreachable regions, or may not alter the decision that the planner would have taken. We verify this by measuring the number of times the preferences were used during the search and how many of those alter the decision of the planner.

Applicability of preferences. We observe that on an average, across all domains, upfront preferences are used/applied at least 20% fewer times (corresponding to 2.04 fewer uses) than preferences elicited by PGPLANNER. We evaluated how PGPLANNER queries for (uses) preference across different stages/depths of plan search. Figure 7 shows how total average preferences used varies with relative depth. Since planning depth is different for every problem in every domain, 'depth ratio' denotes a standardized scale computed by considering equidistant fractions of the total planing depth. Similarly, as total number of preferences used varies across domains, they were normalized and averaged over all domains, and their cumulative values were plotted. We observe how PGPLANNER acquires 80% of the preferences by 60% of the planning depth as compared to the upfront case which uses preferences uniformly till completion. This empirical result corroborates the earlier discussion on theoretical properties that showed the relationship between impact of the preference and the relative depth. Note, however, that PGPLANNER does elicit/use some preferences at higher depths. So it cannot be claimed acquiring early would be a valid alternative. It is necessary to identify the right situations where guidance is needed, as done by our approach.

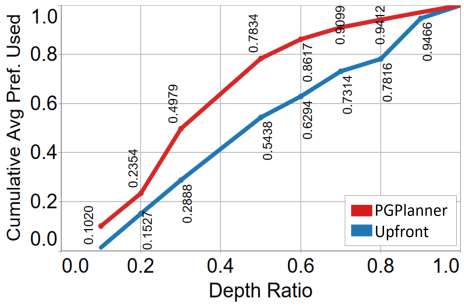


Figure 7: Preference used (cumulative) against relative depth (best viewed in color)

Domains	PGPLANNER	Upfront Preference
Freecell	91.875	70.625
Rovers	88.125	72.5
Trucks	89	73
Depots	86.25	75.625
Satellite	95.5	84
TidyBot	89	80
Blocks World	89.1	74.28
Towers of Hanoi	84.74	74
Barman	84	80
Mystery	87	86
Assembly	84	83
Rockets	74	73
Average	86.88	77.17

Table 2: Average % of applicable prefs. that influence decisions.

Influence on decision making. Furthermore, actively acquired preferences influence the decisions in 86.88% of the cases where the preference applies, compared to 77.17% for upfront preferences. Table 2 shows the the statistics of every domain. Notice that, while the measures for planning with upfront preferences are almost always less than PGPLANNER across all domains, the difference is negligible in the 'Mystery' and 'Rockets' domains which aligns with what we observe in terms of performance and efficiency. In the 'Depots' domain, however, the difference in percentage of decision impacting preference is around 11% (row 4) but the difference in average plan length is substantially high (Figure 5). On closer inspection we observed that in 2 particular problems in the Depots domain, upfront preferences, though applicable, lead to sub-optimal plans, particularly with substantially high plan length. While it is difficult to discover the exact reason for this behavior, we realized that the 2 particular problems had less 'crates' than 'packages' unlike all other problems which had greater or equal creates as packages. Since this domain essentially solves ordered-assignment problems, suggestion/preference at the right point is crucial especially in problems where assignment slots are less than the assigned objects, emphasizing the value of our active approach.

We observe exactly the opposite scenario in the 'Towers of Hanoi' domain. Here the difference is around 10%, but we observe that difference in performance (average

plan length) is not significantly large. This suggests that *we have not compromised on the quality of preferences provided upfront*. Particularly the upfront preference, *"If possible then* ***avoid empty pegs****"*, seemed to be effective in influencing the planner towards that part of the search space which mostly resulted in better (shorter) plans. A similar, more prominent case, is the 'Freecell' card game domain. The upfront preference *"If possible Then prefer decompositions that allow for immediately finishing a card compared to other choices"* led to shorter plans, even though this preference altered the decisions at only a few points. Clearly, both *Towers of Hanoi* and *Freecell* are intuitive domains and a little practice allows us to formulate high quality upfront preferences. But other domains are more challenging and it is difficult for an expert to imagine all possible scenarios and formulate useful preferences without knowing the current stage and task. Also, planning with upfront preferences performs better than random querying in most domains, indicating that the preferences provided were reasonable. However, PGPLANNER is able to elicit more relevant preferences and use them to find more effective plans efficiently.

Challenges. One natural question that arises is that the planner assumes that the human preferences are close to optimal (or at least non sub-optimal). This is indeed a correct observation that is true in many human-in-the-loop systems. In inverse RL [26] and probabilistic learning [37], the expert's preferences can be explicitly traded-off with trajectories or labeled data respectively. In such cases, the expert's preferences serve to reduce the effect of targeted noise. However, in planning we do not assume access to such trajectories and instead rely on rollout. Thus, an explicit trade-off is not quite sufficient and warrants a deeper investigation which is beyond the scope of the current work. We note that querying a set of different experts based on their expertise level on certain sub-tasks remains an interesting and challenging future direction.

5. Conclusion

We present a novel method for preference-guided planning where preferences are actively elicited from human experts. PGPLANNER allows for the planner to query

only as needed and reduces the burden on the expert to understand the planning process to suggest useful advice. We empirically validate the efficiency and effectiveness of PGPLANNER across several domains and demonstrate that it outperforms the baselines even with fewer preferences. Our formulation transforms plan search into a sequential decision making process in the space of task decompositions, based on estimated distribution and uncertainty. In essence, PGPLANNER is able to infer *what-it-knows* (or does not know) and pose the optimal set of queries to the expert. This not only effectively prunes the sub-optimal regions of the search space but also generates better plans.

Currently, our planner does not validate the preferences, but rather assumes the user is an expert. We will extend the planner to recommend improvements to the set of preferences. We will investigate other avenues to obtain preferences including crowd of experts, transfer across subtasks as well as adapting from other domains.

Acknowledgment

We gratefully acknowledge the support of CwC Program Contract W911NF-15-1-0461 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). Any opinions, findings and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, ARO or the US government.

References

[1] S.-W. Tan, J. Pearl, Specification and evaluation of preferences for planning under uncertainty, in: KR, 1994.

[2] K. L. Myers, Advisable planning systems, Advanced Planning Technology (1996) 206–209.

[3] Y.-C. Huang, B. Selman, H. Kautz, et al., Control knowledge in planning: benefits and tradeoffs, in: AAAI/IAAI, 1999, pp. 511–517.

[4] J. Allen, G. Ferguson, Human-machine collaborative planning, in: International NASA Workshop on Planning and Scheduling for Space, 2002.

[5] R. I. Brafman, Y. Chernyavsky, Planning with goal preferences and constraints., in: ICAPS, 2005, pp. 182–191.

[6] S. Sohrabi, S. A. McIlraith, On planning with preferences in htn, in: (NMR), 2008.

[7] T. Bylander, Complexity results for planning., in: IJCAI, 1991.

[8] M. Ghallab, D. Nau, P. Traverso, Automated planning: theory & practice, Elsevier, 2004.

[9] H. A. Kautz, B. Selman, Planning as satisfiability., in: ECAI, 1992.

[10] H. Kautz, B. Selman, Pushing the envelope: Planning, propositional logic, and stochastic search, in: AAAI, 1996.

[11] A. L. Blum, M. L. Furst, Fast planning through planning graph analysis, Artificial intelligence 90 (1) (1997) 281–300.

[12] J. Hoffmann, B. Nebel, The FF planning system: Fast plan generation through heuristic search, JAIR 14 (2001) 253–302.

[13] S. W. Yoon, A. Fern, R. Givan, Ff-replan: A baseline for probabilistic planning., in: ICAPS, 2007.

[14] S. Yoon, A. Fern, R. Givan, Learning control knowledge for forward search planning, JMLR 9 (Apr) (2008) 683–718.

[15] Y. Xu, A. Fern, S. Yoon, Learning linear ranking functions for beam search with application to planning, JMLR (2009) 1571–1610.

[16] Y. Xu, A. Fern, S. W. Yoon, Iterative learning of weighted rule sets for greedy search., in: ICAPS, 2010.

[17] K. Erol, J. Hendler, D. S. Nau, HTN planning: Complexity and expressivity, in: AAAI, 1994.

[18] F. Bacchus, F. Kabanza, Using temporal logics to express search control knowledge for planning, Artificial Intelligence 116.

[19] S. W. Yoon, A. Fern, R. Givan, S. Kambhampati, Probabilistic planning via determinization in hindsight., in: AAAI, 2008.

[20] G. Ferguson, J. F. Allen, B. W. Miller, Trains-95: Towards a mixed-initiative planning assistant., in: AIPS, 1996.

[21] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, et al., Mapgen: Mixed-initiative planning and scheduling for the mars exploration rover mission, IEEE Intelligent Systems 19 (1) (2004) 8–12.

[22] K. Talamadupula, G. Briggs, M. Scheutz, S. Kambhampati, Architectural mechanisms for handling human instructions in open-world mixed-initiative team tasks, ACS 6.

[23] S. Sohrabi, J. A. Baier, S. A. McIlraith, HTN planning with preferences, in: IJCAI, 2009.

[24] G. Kunapuli, P. Odom, J. Shavlik, S. Natarajan, Guiding autonomous agents to better behaviors through human advice, in: ICDM, 2013.

[25] K. Judah, A. Fern, P. Tadepalli, R. Goetschalckx, Imitation learning with demonstrations and shaping rewards, in: AAAI, 2014.

[26] P. Odom, S. Natarajan, Active advice seeking for inverse reinforcement learning, in: AAMAS, 2016.

[27] C. Boutilier, R. Brafman, C. Geib, D. Poole, A constraint-based approach to preference elicitation and decision making, in: AAAI Spring Symposium on qualitative decision theory, 1997.

[28] R. Maclin, J. W. Shavlik, Incorporating advice into agents that learn from reinforcements, in: AAAI, 1994.

[29] R. Maclin, J. W. Shavlik, Creating advice-taking reinforcement learners, Machine Learning 22 (1) (1996) 251–281.

[30] S. Natarajan, P. Tadepalli, Dynamic Preferences in Multi-Criteria Reinforcement Learning, in: ICML, 2005.

[31] L. Torrey, T. Walker, J. Shavlik, R. Maclin, Using advice to transfer knowledge acquired in one reinforcement learning task to another, in: ECML, 2005, pp. 412–424.

[32] D. Sarne, B. J. Grosz, Estimating information value in collaborative multi-agent planning systems, in: AAMAS, 2007.

[33] Y. Freund, H. Seung, E. Shamir, N. Tishby, Selective sampling using the query by committee, Machine Learning 28 (1997) 133–168.

[34] B. Settles, Active learning literature survey, Tech. rep., University of Wisconsin (2010).

[35] S. Ross, D. Bagnell, Efficient reductions for imitation learning., in: AISTATS, 2010.

[36] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, F. Yaman, Shop2: An htn planning system, J. Artif. Intell. Res. (JAIR) 20 (2003) 379–404.

[37] P. Odom, S. Natarajan, Actively interacting with experts: A probabilistic logic approach, in: ECML, 2016.