Inferring Implicit Goals Across Differing Task Models

Silvia Tulli[1] , **Stylianos Loukas Vasileiou**[2] , **Mohamed Chetouani**[1] and **Sarath Sreedharan**[4]
[1]Institute of Intelligent Systems and Robotics (ISIR) - CNRS - INSERM - Sorbonne University
[2]McKelvey School of Engineering at Washington University in St. Louis
[3]Department of Computer Science at Colorado State University
silvia.tulli@sorbonne-universite.fr, v.stylianos@wustl.edu, mohamed.chetouani@sorbonne-universite.fr, sarath.sreedharan@colostate.edu

Abstract

One of the significant challenges to generating value-aligned behavior is to not only account for the specified user objectives but also any implicit or unspecified user requirements. The existence of such implicit requirements could be particularly common in settings where the user's understanding of the task model may differ from the agent's estimate of the model. Under this scenario, the user may incorrectly expect some agent behavior to be inevitable or guaranteed. This paper addresses such expectation mismatch in the presence of differing models by capturing the possibility of unspecified user subgoal in the context of a task captured as a Markov Decision Process (MDP) and querying for it as required. Our method identifies bottleneck states and uses them as candidates for potential implicit subgoals. We then introduce a querying strategy that will generate the minimal number of queries required to identify a policy guaranteed to achieve the underlying goal. Our empirical evaluations demonstrate the effectiveness of our approach in inferring and achieving unstated goals across various tasks.

1 Introduction

Humans often omit details they consider obvious, unavoidable, or not worth mentioning when providing instructions. In the context of human-AI interaction, such omissions could lead to implicit goals and unstated preferences that AI systems must navigate to achieve full alignment with user intent. One potential source of such unstated subgoals or preferences could be behaviors that the user may identify as inevitable. The user would never bother stating anything regarding such behaviors, since they believe that they cannot be avoided. Take the case of visiting *bottleneck states* in the context of goal-based Markov Decision Process (MDP). Here, bottleneck states refer to environment states that the agent must pass through to reach the stated goal. In many cases, the user may want the agent to pass through or visit some bottleneck states in addition to the goal, thus forming a set of intermediate subgoals. However, the user may never specify them since, as far as the user is concerned, every path that leads to the goal passes through all the bottleneck states. This should be all well and good, provided the human bottleneck states are also bottleneck states for the agent. Otherwise, the agent must make an effort to figure out what the user's underlying subgoals may be.

To see how such problems may arise, consider an agent tasked with guiding a tourist to a famous art museum. The tourist simply says, "Get me a plan to get to the art museum," unaware of the city's metro system and expecting an above-ground route passing certain landmarks. The agent, however, would use the metro system, as some of the city roads are under construction. For the metro route, bottlenecks might include various stations and the transfers. For the tourist's expected route, they might include crossing a river and passing through the city center, which they wanted to visit. An AI system that blindly generates a goal-reaching policy might end up skipping all of these intermediate implicit goals. This misalignment stems from differing world models: the agent's comprehensive transit data versus the tourist's limited knowledge of the city's layout and the current state of its roads. The challenge in AI alignment lies in bridging this gap, identifying and accounting for implicit aspects of the task that weren't mentioned.

This paper explores how an agent can learn and achieve the implicit subgoals of another agent, particularly when their understanding of the environment differs. We do so by developing a novel approach that introduces and formalizes the notion of implicit subgoals within the MDP framework. Our method will then compute policies that align with implicit subgoals even when the user's knowledge about the environment isn't completely known. Our planning approach will simultaneously use two distinct MDPs: the executing agent's, i.e., the robot's[1] model of the environment, and the user's, possibly unknown, beliefs about the environment. We will use the possible estimates for the human model as a basis for generating candidate implicit subgoals that the robot will try to achieve. When potential implicit subgoals cannot be achieved, we will also make use of minimal querying to refine the agent's hypotheses about the human subgoals. Figure 1 provides a visualization of our proposed method. To evaluate our approach, we performed empirical evaluations in MDP benchmark domains to determine the computational characteristics of our proposed method. We also present results from a

[1]Note, we use the term robot to denote an autonomous agent. Our approach does not require the agent to be a physically embodied one.

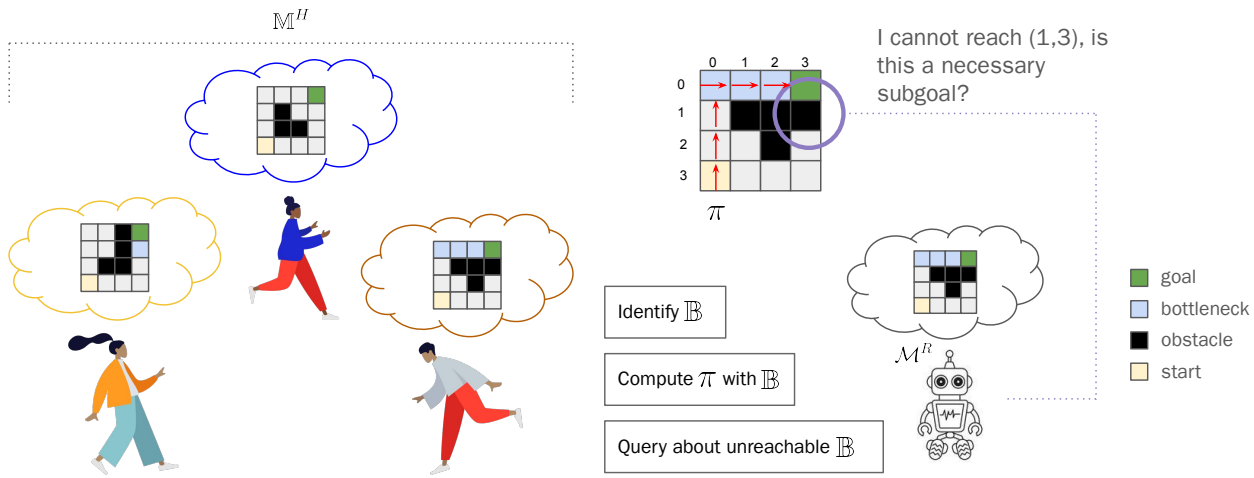


Figure 1: Bottleneck states are critical waypoints essential for reaching the goal in a given world model. Given a set of humans' world models $\mathbb{M}^H$, the robot has to compute a policy π accounting for humans' $\mathbb{B}$, as they might be candidates for human implicit subgoals. Whenever the robot cannot reach a human's bottleneck due to discrepancies in world models, it queries whether this bottleneck is in fact a human subgoal.

small-scale user study aimed at evaluating the usefulness of our approach to non-AI experts.

2 Background

The paper studies problems modeled as infinite horizon discounted Markov Decision Processes (MDPs), focusing on achieving specific goal states.

The algorithms used exploit traditional reward-based MDPs. An infinite horizon MDP is defined as a tuple $\mathcal{M} = \langle S, A, T, s_0, \gamma, R\rangle$, where S is the state space, A is the action space, $T : S \times A \times S \to [0, 1]$ is the transition function (e.g., $T(s_i, a_i, s')$ gives the probability of transitioning from state s to s' under action a), $R : S \to \mathbb{R}$ is the reward function, $s_0 \in S$ is the initial state, and $\gamma \in [0, 1)$ is the discount factor. We generally consider models where S and A are finite sets.

In this setting, the solution is a deterministic, stationary policy $\pi : S \to A$ mapping states to actions. The value of a policy π, denoted as $V^\pi : S \to \mathbb{R}$, gives the expected cumulative discounted reward from following the policy from a given state. A policy is optimal if no policy with a higher value exists. We focus on goal-directed problems, where the set of goal states is $S_G \subseteq S$. The reward function is sparse, returning a small positive value for states in S_G and 0 otherwise. States in S_G are absorbing, with all transitions out having zero probability. In such cases, we represent the MDP in goal terms as $\mathcal{M} = \langle S, A, T, s_0, \gamma, S_G\rangle$.

We leverage the concept of goal-reaching traces. A goal-reaching trace of a policy from a state s, denoted as $\tau \sim_{\mathcal{M}} \pi(s)$, where $\tau = \langle s, \pi(s), ..., s_g\rangle$, is a state-action sequence with non-zero probability that terminates at a goal state, $s_g \in S_G$. Since reward is only provided by the goal, the policy's value in a state is directly proportional to the probability of reaching the goal state under the given policy. We denote this as $P_G(s|\pi)$, given by

$$P_G(s|\pi) = \sum_{\tau \sim_{\mathcal{M}} \pi(s)} P(\tau|\pi),$$

where τ are possible traces ending in a goal state and $P(\tau|\pi)$ is their likelihood under the policy π.

3 Planning for Implicit Subgoals

Consider a scenario where the robot's model of the task $\mathcal{M}^R = \langle S, A, T^R, s_0, \gamma, S_G\rangle$ differs from the user's beliefs $\mathcal{M}^H = \langle S, A, T^H, s_0, \gamma, S_G\rangle$ in terms of the transition function. This difference leads the user to overlook specifying subgoals they believe are inevitable.

We leverage the notion of a **bottleneck state**, where a state is a bottleneck if it is reached in every path from the initial state to the goal. Formally,

Definition 1. *For a given MDP model $\mathcal{M} = \langle S, A, T, s_0, \gamma, S_G\rangle$, a **bottleneck state** is a state $s \in S$ that must be in every valid trace starting from s_0, for any policy, with non-zero probability of reaching a state in S_G. The set of all bottlenecks is denoted as $B \subseteq S$.*

Implicit subgoals ($\mathcal{I}_G$) are a subset of B that the user wants the robot to achieve en route to the goal. From the user's perspective, these subgoals need not be specified since they are bottleneck states in their model. However, due to model differences, what is a bottleneck in one model may not be in the other. Our goal is to find a policy that achieves these subgoals in the robot's model. Formally,

Definition 2. *For a given robot model $\mathcal{M}^R = \langle S, A, T^R, s_0, \gamma, S_G\rangle$, a policy π **achieves a set of implicit subgoal(s)** $\mathcal{I}_G$, i.e., $\pi \models_{\mathcal{M}^R} \mathcal{I}_G$, if every goal-reaching trace from s_0, $\tau \sim_{\mathcal{M}^R} \pi(s_0)$, passes through every state $s \in \mathcal{I}_G$.*

The main challenge is finding a policy that achieves all implicit subgoals without knowing them or the exact user model. To address this, we assume: (1) a set of potential user models $\mathbb{M}^H$ with different transition functions, and (2) the ability to query the user about potential implicit subgoals. This is represented by an oracle function $\mathcal{O}_{\mathcal{I}_G} : S \rightarrow \{0, 1\}$, where

$$\mathcal{O}_{\mathcal{I}_G}(s) = \begin{cases} 1 \text{ if } s \in \mathcal{I}_G \\ 0 \text{ otherwise} \end{cases}$$

Note that the first assumption is a rather weak one, as the set $\mathbb{M}^H$ could be infinite, and contain all possible models that can be expressed in the current set of states. The second assumption holds as long as the user is truthful, wants to achieve the implicit subgoals, and can recognize the ones they want to achieve when queried about them. We set the cost of querying the user extremely high, aiming to minimize the expected query cost.

Definition 3. *For a given robot model $\mathcal{M}^R$, a set of possible user models $\mathbb{M}^H$, and an oracle $\mathcal{O}_{\mathcal{I}_G}$ for an unknown implicit subgoal set $\mathcal{I}_G$, the problem of* ***query identification for implicit subgoals*** *is to choose states to query the oracle to identify a policy π that achieves the implicit subgoal $\mathcal{I}_G$, or determine that no such policy exists.*

We focus on minimizing the expected query cost, where each query has a cost, and querying ends once the agent can determine if a policy exists that satisfies all implicit subgoals and the original goal. The optimal query minimizes the expected value.

Definition 4. *For a given identification problem, with a set of remaining bottlenecks $\mathcal{B}$ and known implicit sub-goals $\mathcal{K}_\mathcal{I} \subset \mathcal{I}_G$, where the existence of the policy that achieves all implicit sub-goals cannot be determined, a state query s_q is said to be optimal if it minimizes the expected query cost for the bottleneck states and known sub-goals, i.e.,*

$$\mathcal{Q}((\mathcal{B}, \mathcal{K}_\mathcal{I}), s) = \mathcal{C}(s) + (P(s \in \mathcal{I}_G) * \mathcal{V}(\mathcal{B} \setminus \{s\}, \mathcal{K}_\mathcal{I} \cup \{s\})) + P(s \notin \mathcal{I}_G) * \mathcal{V}((\mathcal{B} \setminus \{s\}, \mathcal{K}_\mathcal{I}))),$$

where $\mathcal{Q}()$ is the total expected query cost, $P(s \in)$ and $P(s \notin)$ are the probabilities of the state being an implicit goal, $\mathcal{C}(s)$ is the specific cost of querying about s, $\mathcal{V}()$ is the minimal expected cost associated with a set of bottleneck states and known implicit subgoals. Here, $\mathcal{V}((\mathcal{B}', \kappa')) = 0$ if κ' is unachievable or if the set $\mathcal{B}' \cup \kappa'$ is achievable, else $\mathcal{V}((\mathcal{B}', \kappa')) = \min_s \mathcal{Q}((\mathcal{B}', \kappa'), s)$.

Note that here, we are making an implicit assumption that responses to all queries are deterministic, and the user is always able to answer correctly. We believe this is a reasonable assumption to make given that this is the first work to deal with this problem. Additionally, we expect a user to be capable of correctly identifying whether a given state is an implicit subgoal or not in most simple scenarios.

We map the problem into finding an optimal policy in an MDP, introducing a query MDP in Section 4.

4 Query Identification for Implicit Subgoals Set

We now explore algorithms to identify queries for implicit subgoals. Our approach involves finding a set of potential implicit subgoals and querying the user to narrow it down to an achievable set.

Implicit subgoals are a subset of bottleneck states in the human model. Although we may have an infinite set of potential human models, bottleneck states are preserved through determinization Keller and Eyerich [2011]. Determinization converts stochastic transitions into deterministic actions.

Definition 5. *For a given MDP model $\mathcal{M} = \langle S, A, s_0, T, S_G \rangle$, a* ***determinized model*** *$\delta(\mathcal{M})$ is given as $\delta(\mathcal{M}) = \langle S, A', s_0, T', S_G \rangle$, such that for every non-zero transition $T(s_i, a_i, s')$ in $\mathcal{M}$, there exists a new action $a'_i \in A'$, such that $T(s_i, a'_i, s') = 1$.*

Proposition 1. *Given a model $\mathcal{M}$ and its determinization $\delta(\mathcal{M})$, a state s is a bottleneck state for $\mathcal{M}$ if and only if it is a bottleneck state in $\delta(\mathcal{M})$.*

Note that the number of possible unique determinized models for finite state and action sets is finite. Thus, the set of all determinized models $\delta(\mathbb{M}^H)$ is finite and $|\delta(\mathbb{M}^H)| \leq |\mathbb{M}^H|$.

To identify bottleneck states in a determinized model, we create a modified MDP where passing through the queried state is penalized. If the state under test is not a bottleneck, the optimal policy avoids it, resulting in a positive value for s_0.

Proposition 2. *For a determinized model $\delta(M) = \langle S, A, T, s_0, \gamma, S_G \rangle$, and for a target state s_i, we create a new MDP $\mathcal{M}^{s_i} = \langle S, A, T, s_0, \gamma, R^{s_i}_{S_G} \rangle$, such that*

$$R(s) = \begin{cases} n \textit{ when } s = s_i \\ p \textit{ when } s \in S_G \\ 0 \textit{ otherwise} \end{cases}$$

where $n < 0$, $p > 0$ and $|n| > p$. Here s_i is not a bottleneck state if and only if $V^*(s_0) > 0$ under the optimal policy for $\mathcal{M}^{s_i}$.

The validity of the proposition follows from the fact that if state s_i is not a bottleneck state, then there should exist a path from s_0 to the goal that doesn't pass through s_i, which should result in a positive value for s_0. The requirement $|n| > p$ is required to ensure that any path that does pass through s_i doesn't result in a positive value.

We collect all bottleneck states for each determinized model in $\delta(\mathbb{M}^H)$ to form our initial hypotheses set for implicit goals ($\mathcal{H}^0_\mathcal{I}$). Our objective is to identify the set of maximal subsets of $\mathcal{H}^0_\mathcal{I}$ for which the robot can generate achievable policies. However, before we discuss the search procedure to find maximal subsets, we need a procedure that can identify policies that achieve subgoal when one exists. We will again convert it to that planning over MDPs. In this case, this will involve planning over two different planning problems. Firstly, one that will only count goal achievement if the trace passes through all the subgoals.

For an MDP $\mathcal{M} = \langle S, A, T, s_0, \gamma, S_G \rangle$ and a subgoal set $\hat{S}$,

we create a new MDP $\mathcal{M}^{\hat{S}}$ to identify policies that achieve the subgoals. The new MDP tracks subgoal visits and rewards only traces passing through all subgoals.

Proposition 3. *For an MDP $\mathcal{M} = \langle S, A, T, s_0, \gamma, S_G \rangle$, and a subgoal set $\hat{S}$, we create a new MDP $\mathcal{M}^{\hat{S}} = \langle S^{\hat{S}}, A, T^{\hat{S}}, s_0, \gamma, R^{\hat{S}} \rangle$, where $S^{\hat{S}}$ tracks subgoal visits, $T^{\hat{S}}$ updates features for visited subgoals, and $R^{\hat{S}}$ rewards only traces passing through all subgoals. If there exists a policy π for $\mathcal{M}$ that achieves the subgoal set, then there exists a policy $\hat{\pi}$ for $\mathcal{M}^{\hat{S}}$, such that all goal reaching trace for $\hat{\pi}$ exits at the goal state copies where all the features for subgoals are true.*

This proposition is true since any trace that ends in a goal state without passing through all the subgoal states will provide a negative reward. However, one might not be able to tell if the identified policy corresponds to such a policy. One can only do so by running a test over the determinized version of $\mathcal{M}^{\hat{S}}$, similar to the one described in Proposition 2. One difference is that the actions in each state are limited to the one listed by $\hat{\pi}$. Here the test is run for each potential state in $\hat{S}$.

We search for maximally achievable subsets over the union of all bottleneck states across potential human models using a recursive depth-first approach with pruning. The algorithm systematically explores combinations of bottleneck states, confirming maximality by checking if adding any remaining element makes the subset unachievable. Algorithm 1, gives a pseudo-code for the overall procedure. GenerateAndTestSubsets function represents the recursive procedure that goes over each possible subsets by dropping one bottleneck state at a time. The procedure stops when the current subset is achievable (i.e., there exists a policy π such that π) or if the bottleneck set passed is an empty one.

With the identification of $\mathbb{I}$, we convert the querying strategy problem into an MDP planning problem.

Definition 6. *For a set of potentially achievable subgoals $\mathbb{I}$, selected from a bottleneck set $\mathcal{B}$, the query MDP is defined as $\mathcal{M}^Q = \langle S^Q, A^Q, T^Q, s_0^Q, \gamma, R^Q \rangle$, where each component is defined as follows:*

- *S^Q: Each state consists of known implicit subgoals and those known not to be, i.e., $S^Q = 2^{\mathcal{B}} \times 2^{\mathcal{B}}$.*
- *A^Q: One action to query each element in $\mathcal{B}$.*
- *T^Q: Transition function replicating potential oracle outcomes and determining absorber states.*
- *s_0^Q: Start state where nothing is known.*
- *R^Q: Reward function returning a heavy penalty for queries and positive values for achievable states.*
- *γ: High discount factor to consider future query costs.*

Proposition 4. *The optimal policy identified for the MDP $\mathcal{M}^Q$, corresponds to an optimal query strategy described in Definition 4.*

This follows from the structure of the MDP. The cost and transition function, here, are selected so the Bellman equations for the MDP replicate the optimality equations referred to in Definition 4 (with min replaced with max to reflect the switch from costs to rewards). One point of departure here from the earlier definition is the allocation of positive rewards to absorber states where the reward is proportional to its value associated in the model $\mathcal{M}^{\hat{S}}$. Given the role played by discount factor, this means that a higher value is associated with bottleneck subsets where the goals and subgoals are achieved over shorter traces. This means that the problem of finding subgoal sets that are easier to achieve becomes a secondary objective for the MDP. However, please note that the larger penalty for the query cost means that this secondary objective will never be pursued at the cost of a potentially larger number of queries.

Algorithm 1 Find the set of maximally achievable subsets of the set of all possible human bottleneck states.

Input: $\mathcal{M}^R$, $\mathcal{B}$
Output: Set of maximal achievable subsets $\mathbb{I}$
function FINDMAXIMALACHIEVABLESUBSETS($\mathcal{M}^R$, $\mathcal{B}$)
    return GenerateAndTestSubsets($\mathcal{B}$, $\mathcal{M}^R$)
end function
function GENERATEANDTESTSUBSETS($\hat{\mathcal{B}}$, $\mathcal{M}^R$)
    if CheckAchievability($\hat{\mathcal{B}}$, $\mathcal{M}^R$) **then**
        return $\{\hat{\mathcal{B}}\}$
    end if
    if $|\hat{\mathcal{B}}| == 0$ **then**
        return $\emptyset$
    end if
    $\hat{\mathbb{I}} = \{\}$
    for $s_i \in \hat{\mathcal{B}}$ **do**
        current_subset $\leftarrow \hat{\mathcal{B}} \setminus \{s_i\}$
        max_subsets $\leftarrow$GenerateSubsets(current_subset, $\mathcal{M}^R$)
        for $\hat{S} \in$max_subsets **do**
            $\hat{\mathbb{I}} = \hat{\mathbb{I}} \cup \hat{S}$
        end for
    end for
    return $\hat{\mathbb{I}}$
end function

Now, even though there are efficient MDP solvers, solving the above MDP could be computationally expensive if there exists a large number of possible bottleneck states. However, it is possible to show that we can actually build a smaller MDP that first filters out all the bottleneck states that cannot be achieved in the robot model and create a new query MDP only containing the remaining states. We will refer to the resulting MDP as the pruned query MDP (and represent it as $\hat{\mathcal{M}}^Q$). Now, we will create a meta query policy that will first query about all non-achievable bottlenecks before switching over to the optimal policy for the pruned query MDP ($\hat{\pi}^Q$). We will refer to this new modified query as Π^Q, and it is defined as

$$\Pi^Q((K_{\mathcal{I}}, K_{\neg\mathcal{I}})) = \begin{cases} s_i \text{ if } |\mathcal{B} \setminus (K_{\mathcal{I}} \cup K_{\neg\mathcal{I}})| > 0, \\ \qquad \text{where } s_i \in \mathcal{B} \setminus (K_{\mathcal{I}} \cup K_{\neg\mathcal{I}}) \\ \hat{\pi}^Q((K_{\mathcal{I}} \setminus \mathcal{B}, K_{\neg\mathcal{I}} \setminus \mathcal{B})) \text{ Otherwise,} \end{cases}$$

where $\mathcal{B}$ is the set non-achievable bottleneck states. Even

though such a pruning method could result in an exponential reduction in the state space of the MDP problem to be solved, we can show that this new policy is, in fact, optimal for the original query model. Or more formally,

Proposition 5. *The meta policy Π^Q is an optimal policy for the query MDP $\mathcal{M}^Q$.*

Proof Sketch. The primary proof for the above statement relies on establishing the fact that in a non-absorbing state for $\mathcal{M}^Q$, the cost of querying a non-achievable bottleneck state is always going to cheaper than or equal to the cost of a query about a state that is part of some achievable subset of bottlenecks. This can be shown by the fact that any query not involving a non-achievable state must involve at least one outcome, with at least one future query guaranteed to be required. This guarantee follows from two facts. For such a query, at least one of the outcomes must contain a potentially achievable subset. Without such an outcome, the original query state would be unachievable and thus an absorbing state. As for the at least one guaranteed future query comes from the fact that since this query skipped over a non-achievable bottleneck, the achievability of the outcome can only be established after resolving whether or not the non-achievable bottleneck is part of the human implicit subgoal.

Now, on the other hand, there can, at most, be one outcome where further querying is possibly required. In this case, future querying is also not guaranteed because, after the removal of the unachievable state, it could just result in a query state that corresponds to an absorber state for an achievable subset. In other words, the query about a non-achievable bottleneck is always guaranteed to remove a future query from all outcomes. But for bottleneck states that can be achieved under some policy, we are guaranteed that we need to query about the non-achievable bottleneck at some point in the future. Finally, there order in which the non-achievable states need to be queried doesn't matter as its not part of any achievable subsets and thus the order doesn't affect how any of the potentially achievable subsets can be queried. Finally, the secondary objective doesn't really affect this order, as it relates to reducing the expected number of queries and the secondary objective is dominated by the cost of the number of queries. □

5 Related Work

Our work intersects with three primary areas: reward misspecification, planning with different world models, and query mechanisms in assistance.

5.1 Reward Misspecification

Aligning AI systems with human values and intentions is a growing concern, with reward misspecification being a critical issue. Inverse reward design [Hadfield-Menell *et al.*, 2017] was a method developed to infer true objectives by treating reward specifications as observations of the true reward. Building on this, previous work [Majumdar *et al.*, 2017] has proposed a risk-sensitive inverse reinforcement learning framework to address reward uncertainties. Authors have also explored implicit preferences [Shah *et al.*, 2019] in reward functions, and investigated reward hacking [Gleave *et al.*, 2021]. Works have also looked at formalizing reward specification in terms of discrepancies between agent and user expectations [Sreedharan and Mechergui, 2024], proposing a framework to identify misalignments and consider model differences. Our work extends these ideas by focusing on unspecified subgoals arising from differing task models.

5.2 Planning with Different World Models

Agents and humans often have different world models, a challenge explored in AI planning and human-robot interaction. Model reconciliation [Chakraborti *et al.*, 2017] was a framework that was proposed for explainable planning that focuses on bridging the knowledge gap between the user and the AI system. Works have also looked at how such notions could be extended to cases where use models could be captured as abstractions of the true agent model [Sreedharan *et al.*, 2018].

Related works have also looked at learning from corrections with differing feature spaces [Bobu *et al.*, 2018] and addressing representation misalignment [Peng *et al.*, 2024]. Theory of Mind research also provides insights into planning with different world models. Works have also highlighted the use of abstract causal models for effective intervention planning [Ho and Griffiths, 2021]. These insights complement model reconciliation approaches, suggesting that incorporating Theory of Mind representations could enhance AI planning effectiveness.

5.3 Handling Unspecified User Goals

Query mechanisms enable robots to seek clarification from humans when facing ambiguous reward signals. Hidden Goal Markov Decision Processes (HGMDPs) was a decision-theoretic framework for intelligent assistance, focusing on reasoning under uncertainty about user objectives [Fern *et al.*, 2007]. Assistive games use human reward inference as an alternative to the user specifying reward functions completely [Hadfield-Menell *et al.*, 2016]. Query mechanisms have also been used within hierarchical reinforcement learning. For example, a POMDP framework to query humans about potential subgoals in such settings [Nguyen *et al.*, 2021]. Natural language has also been employed as a means for feedback, optimizing language abstractions to minimize queries [Zheng *et al.*, 2023]. Our work leverages potential estimates of human understanding of the task model to identify more effective queries.

5.4 Preferences in Planning

Preference-based planning has evolved from simple goal satisfaction to frameworks that capture user intentions and quality criteria [Baier and McIlraith, 2008]. Early work established formal mechanisms for expressing temporal preferences over state trajectories, enabling planners to reason about not just what goals to achieve, but how and when to achieve them [Gerevini and Long, 2005]. These frameworks introduced the distinction between hard constraints that must be satisfied and soft preferences that guide plan quality, creating a foundation for handling incomplete or conflicting user specifications [Gerevini *et al.*, 2009]. The integration of hierarchical task networks with preference handling revealed how procedural knowledge can be combined with user preferences to generate

contextually appropriate plans [Sohrabi *et al.*, 2009]. This work highlighted that effective planning requires understanding not just explicit goals, but the implicit assumptions about how tasks should be decomposed and executed within specific domains. The challenge of preference elicitation emerged as a bottleneck, as users often cannot fully articulate their preferences upfront, leading to frameworks that learn preferences through limited interaction and generalize across similar planning contexts [Tabakhi *et al.*, 2022]. These developments directly connect to the problem of inferring implicit goals when operating with different world models. Just as preference-based planning recognizes that users have unstated quality criteria and contextual expectations [Pigozzi *et al.*, 2016], inferring implicit goals requires discovering the underlying objectives that drive observed behavior, even when those objectives are not explicitly communicated.

6 Evaluation

We evaluate our approach on standard Markov Decision Process (MDP) benchmarks using value iteration with convergence threshold $\epsilon = 0.001$ and maximum 1000 iterations. For larger state spaces, we employed sparse matrix implementation and robust vectorized versions handling edge cases Puterman [2014].

Experiments were conducted across multiple environment types with varying grid sizes and obstacle percentages. For reproducibility, each configuration (world type, grid size, human models, obstacle percentage) was run 3 times with random seeds sampled from [1,10000], controlling robot/human model generation and stochastic algorithm elements. Policy extraction used standard or robust methods depending on environment complexity and state space size[2].

6.1 Environments

The base environment is a Maze, a basic setting where agents navigate a grid with randomly placed obstacles, providing our baseline scenario with simple navigation challenges and clear bottleneck states at narrow passages. Building on this, Four-Rooms extends Maze by dividing it into four quadrants connected by doorways. The fixed room structure with randomized door placements creates natural bottlenecks, testing our method's ability to identify critical transition points. PuddleWorld introduces additional complexity by adding puddles that incur penalties when traversed. This environment forces trade-offs between path length and safety, creating interesting bottleneck scenarios where avoiding puddles competes with finding shortest paths. Finally, RockWorld features two types of rocks - valuable rocks that provide rewards when collected and dangerous rocks that incur penalties. This tests bottleneck identification in scenarios with resource management and risk-reward trade-offs.

6.2 Methodology

Our framework runs multiple independent trials with varying parameters including grid sizes, obstacle percentages, and different numbers of human models to test scalability with varying levels of preference diversity. Each model uses a unique obstacle seed to ensure statistically independent trials. We compared the condition in which the robot queries for all the bottleneck states (Query-All) with the condition in which it queries strategically, based on the achievable bottleneck states (Strategic Query). We set a query threshold of 1000[3].

6.3 Results

Our analysis reveals significant performance improvements across environments. For 4×4 grids, Strategic Query showed particularly strong results in Four Rooms ($p < 0.001$) and Rocks ($p = 0.012$), with query counts reduced from $3.7 - 4.8$ (Query-All) to $2.0 - 3.3$ queries, achieving reductions of $22 - 41\%$. Query times remained efficient at $2 - 3$ seconds. In 6×6 environments, while pruning times increased, efficiency gains persisted with query counts of $3.2 - 4.3$ versus $3.7 - 7.7$ for Query-All, yielding $13 - 40\%$ reductions, though with marginally significant differences (Puddle: $p = 0.053$, Rocks: $p = 0.073$). Analysis across 20 human models with 10% obstacle density demonstrates consistent performance, particularly in basic grid environments (35.6% reduction). More complex environments like Rocks and Puddle maintained substantial improvements ($26 - 33\%$ reduction). The stability of these improvements across varying model counts suggests robust scalability. Notably, Four Rooms showed the strongest statistical significance in 4×4 grids ($p < 0.001$) while maintaining performance benefits in larger environments, albeit with reduced statistical significance ($p = 0.411$ for 66) (Figure: 2). Query times remained efficient at $2 - 3$ seconds. In 6×6 environments, while pruning times increased, efficiency gains persisted with query counts of $3.2 - 4.3$ versus $3.7 - 7.7$ for Query-All.

Scaling to 8×8 grids further validates the effectiveness of our approach. Strategic Query maintained consistent query counts (3.2 ± 1.6) across environments while Query-All increased to 11.6 ± 2.9 (except Four Rooms: 3.2 ± 1.0), resulting in substantial reductions ($71.9 \pm 10.9\%$) for Maze, Puddle, and Rocks environments. While pruning times increased to $\sim 308 - 325$ seconds, the method achieved consistent speedup ($1.05\times$) across all environments. Human bottleneck values stabilized around 5.8 ± 1.5 (Four Rooms: 1.6 ± 0.5), demonstrating reliable preference modeling despite increased environmental complexity (Figure: 1).

Further analysis with varying parameters reveals interesting trends. With 10 human models and 0.1 obstacle density, bottleneck finding times remained consistent ($1.7 - 1.8s$) with total runtimes of $4.1 - 9.2s$. Reducing to 5 human models improved computational efficiency ($0.8 - 0.9s$ bottleneck finding, $2.1 - 4.0s$ total runtime). However, increasing obstacle density to 0.15 significantly impacted performance, particularly in environments like Maze and Rocks ($27.4s$ and $13.7s$ bottleneck finding respectively), with total runtimes increasing to $13.1 - 50.7s$ and higher human bottleneck values ($3.3 - 4.2$). This demonstrates the method's stability with increased human

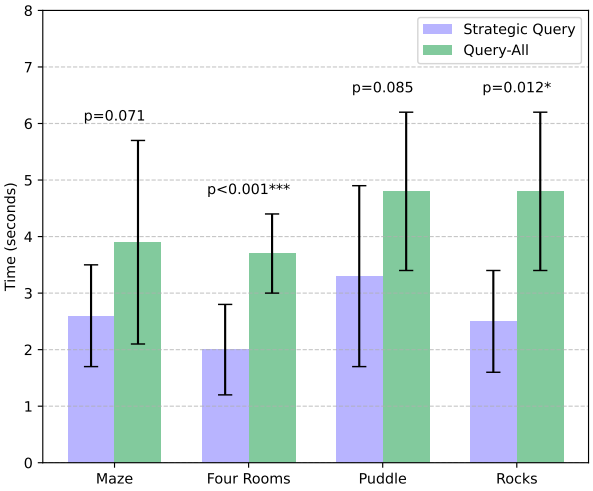
[2]Experiments were executed on a server with 64 cores (AMD EPYC Milan) and 2048 GB DDR4 memory, running 4 parallel workers processing batches of 10 configurations.

[3]For reproducibility, the code and implementation details will be available on GitHub. The repository includes instructions to replicate all experiments.

Domain	State Space Size Size	Pruning Time (s)	Pruning Speedup	Str. Query Count	Query All	Red. (%)	Human Bottlenecks
Maze	64	308.237±6.308	1.05x±0.03	3.2±1.6	11.6±2.9	71.9±10.9	5.8±1.5
Four R.	64	324.602±6.013	1.00x±0.01	3.2±1.0	3.2±1.0	0.0±0.0	1.6±0.5

Table 1: Performance comparison with 8x8 grid size with the same parameters as above. Maze, Puddle, and Rocks domains showed similar performance metrics.

models but sensitivity to environmental complexity at higher obstacle densities.



Domain	State Space Size Size	Str. Query	Query All	Red. (%)
Maze	16	2.6±0.9	3.9±1.8	22.0±30.1
	36	3.8±1.5	6.8±3.9	35.6±33.6
Four R.	16	2.0±0.8	3.7±0.7	0.0±0.0
	36	3.2±1.0	3.7±0.7	13.3±22.1
Puddle	16	3.3±1.6	4.8±1.4	26.7±29.7
	36	4.3±3.3	7.7±4.0	40.7±29.6
Rocks	16	2.5±0.9	4.8±1.4	41.1±27.1
	36	4.1±2.4	6.9±3.1	33.7±29.4

Figure 2: Performance comparison between Strategic Query and Query-All approaches across four environments with 4×4 grid size. Results show mean execution times ± standard deviation based on 20 human preference models, 10% obstacle density, and 3 runs per configuration with a query threshold of 1000. The table shows basic performance metrics including query counts and reduction percentages.

7 Discussion

The paper presents a way a planning system can identify hidden subgoals of users, even when the human model may not be exactly known. We present algorithms to both identify potential candidates and generate an optimal number of queries. We evaluate the effectiveness of the proposed method on a set of standard benchmark problems. In terms of future work, one of the immediate next steps would be to run user studies. We plan to do them in realistic and everyday scenarios, possibly a robotic one, with a significant population size. This would allow us to capture the effectiveness of our method in terms of the load placed on the humans and also test another related hypothesis. For example, one could test whether people would be open to more queries if it significantly improves the agent efficiency. We assume that the main difference between the robot's model and the human's model of the task is the transition function and rewards. In particular, the final goal state remains the same in both cases. However, in a different type of representation, e.g., in linear temporal logic, where trajectories of states could be queried, the problem might be able to be recast as one of goal difference. This paper also focuses on the exact method that identifies optimal solutions. It would be interesting to see if we could leverage approximate methods. It would also be interesting to see if we could use other knowledge sources like pre-trained large language models, to get more information about user knowledge and preferences [Zhou *et al.*, 2024]. In relation to the broader themes of assessment and system compliance, our approach represents a step toward providing guarantees about the types of plans the system generates. By actively querying users about hidden subgoals, the system essentially audits its understanding and ensures alignment with user intentions, offering a form of compliance verification that could contribute to more interpretable and trustworthy AI systems. This connection between interactive planning and system compliance deserves further exploration, particularly in how such query-based approaches can provide formal guarantees about plan quality and user alignment.

Ethical Statement

This work focuses on improving human-AI alignment by better understanding implicit user goals. The research aims to make AI systems more helpful and aligned with human intentions, which has positive ethical implications. There are no significant ethical concerns with this theoretical and empirical work.

Acknowledgments

Sarath Sreedharan's research is supported in part by grant NSF 2303019.

References

Jorge A. Baier and Sheila A. McIlraith. Planning with preferences. *AI Magazine*, 29(4):25–36, 2008.

Andreea Bobu, Andrea V. Bajcsy, Jaime Fernández Fisac, and Anca D. Dragan. Learning under misspecified objective spaces. In *Conference on Robot Learning*, 2018.

Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: moving beyond explanation as soliloquy. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 156–163. AAAI Press, 2017.

Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. A decision-theoretic model of assistance. In *International Joint Conference on Artificial Intelligence*, 2007.

Alfonso Gerevini and Derek Long. Plan constraints and preferences in pddl3: The language of the fifth international planning competition. Technical report, University of Brescia, 2005.

Alfonso Gerevini, Simone Palombi, Alessandro Saetti, Ivan Serina, and Derek Long. Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

Adam Gleave, Michael D Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions. In *International Conference on Learning Representations*, 2021.

Dylan Hadfield-Menell, Stuart Russell, Pieter Abbeel, and Anca D. Dragan. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 3909–3917. Advances in Neural Information Processing Systems, 2016.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart Russell, and Anca D. Dragan. Inverse reward design. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6768–6777, Red Hook, NY, USA, 2017. Curran Associates Inc.

Mark K. Ho and Thomas L. Griffiths. Cognitive science as a source of forward and inverse models of human decisions for robotics and control. *ArXiv*, abs/2109.00127, 2021.

Thomas Keller and Patrick Eyerich. A polynomial all outcome determinization for probabilistic planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 21(1):331–334, Mar. 2011.

Anirudha Majumdar, Sumeet Singh, Ajay Mandlekar, and Marco Pavone. Risk-sensitive inverse reinforcement learning via coherent risk models. In *Robotics: Science and Systems*, 2017.

Khanh Nguyen, Yonatan Bisk, and Hal Daum'e. Learning when and what to ask: a hierarchical reinforcement learning framework. *ArXiv*, abs/2110.08258, 2021.

Andi Peng, Belinda Z. Li, Ilia Sucholutsky, Nishanth Kumar, Julie Shah, Jacob Andreas, and Andreea Bobu. Adaptive language-guided abstraction from contrastive explanations. In *8th Annual Conference on Robot Learning*, 2024.

Gabriella Pigozzi, Alexis Tsoukiàs, and Paolo Viappiani. Preferences in artificial intelligence. *Annals of Mathematics and Artificial Intelligence*, 77(3-4):361–401, 2016.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Rohin Shah, Dmitrii Krasheninnikov, Jordan Alexander, Pieter Abbeel, and Anca Dragan. The implicit preference information in an initial state. In *International Conference on Learning Representations*, 2019.

Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. Htn planning with preferences. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1790–1797, Pasadena, California, USA, 2009.

Sarath Sreedharan and Malek Mechergui. Handling reward misspecification in the presence of expectation mismatch. *ArXiv*, abs/2404.08791, 2024.

Sarath Sreedharan, Siddharth Srivastava, and S. Kambhampati. Hierarchical expertise-level modeling for user specific robot-behavior explanations. *ArXiv*, abs/1802.06895, 2018.

Amir Masoud Tabakhi, Charles Gretton, Sylvie Thiébaux, and Patrik Haslum. A preference elicitation framework for automated planning. *Expert Systems with Applications*, 205:117596, 2022.

Ruijie Zheng, Khanh Nguyen, Hal Daum'e, Furong Huang, and Karthik Narasimhan. Progressively efficient learning. *ArXiv*, abs/2310.13004, 2023.

Sizhe Zhou, Sha Li, Yu Meng, Yizhu Jiao, Heng Ji, and Jiawei Han. Establishing knowledge preference in language models, 2024.