Inferring Implicit Goals Across Differing Task Models

Silvia Tulli[1], Stylianos Loukas Vasileiou[2], Mohamed Chetouani[1], Sarath Sreedharan[3]

[1] Institute of Intelligent Systems and Robotics (ISIR) - CNRS - INSERM - Sorbonne University
[2] Department of Computer Science, New Mexico State University
[3] Department of Computer Science, Colorado State University
silvia.tulli@sorbonne-universite.fr, stelios@nmsu.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.

Code — https://github.com/Silviatulli/implicitgoals

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 (Hadfield-Menell et al. 2017; Mechergui and Sreedharan 2024b). 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) (Puterman 2014). 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 (Sutton, Precup, and Singh 1999; Şimşek and Barto 2009). 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 (Chakraborti et al. 2017; **?**). 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 (Shah et al. 2019; Liu, Zhu, and Zhang 2022). 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 (Ho, Saxe, and Cushman 2022). 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

[1]We use the term robot to denote an autonomous agent. Our approach does not require the agent to be physically embodied.

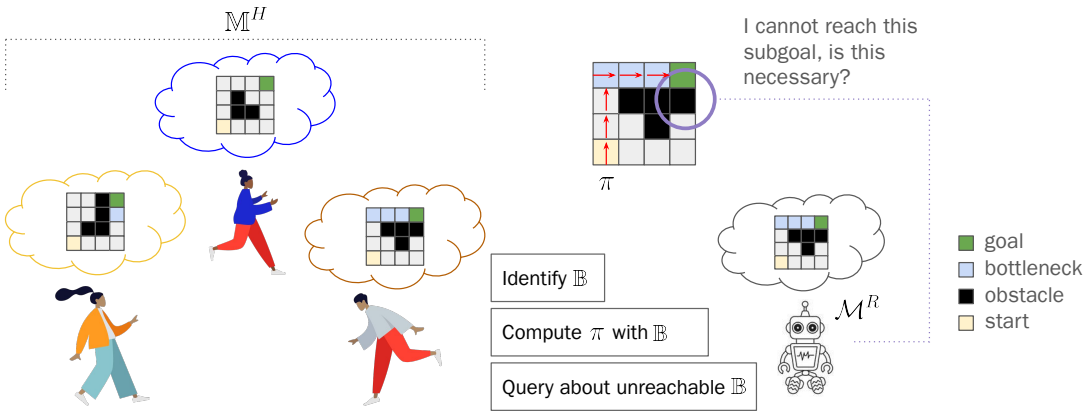

Figure 1: The robot computes policy π over the set of human models $\mathbb{M}^H$, treating bottlenecks $\mathbb{B}$ as candidate implicit subgoals. When model discrepancies prevent reaching a bottleneck, the robot queries whether it is a human subgoal.

use of minimal querying to refine the agent's hypotheses about the human subgoals (Biyik, Anari, and Sadigh 2024; Reddy et al. 2020). 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 small-scale user study aimed at evaluating the usefulness of our approach to non-AI experts.

Background

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

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 $B \subseteq S$.*

Bottleneck states are objective properties of the MDP structure, determined solely by the transition dynamics T and goal specification S_G, independent of any agent's beliefs or computational model. 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 model $\mathcal{M}^R = \langle S, A, T^R, s_0, \gamma, S_G \rangle$ and implicit subgoal set $\mathcal{I}_G$ (derived from the human's model $\mathcal{M}^H$), a policy π **is guaranteed to achieve** $\mathcal{I}_G$, denoted $\pi \models_{\mathcal{M}^R} \mathcal{I}_G$, 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. Here, we will assume: (1) access to a set of potential user models $\mathbb{M}^H$, 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.[2] 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, $P(s \in \mathcal{I}_G)$ and $P(s \notin \mathcal{I}_G)$ are the probabilities of the state being, and not being, an implicit goal, respectively, $\mathcal{C}(s)$ 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.

[2]In practice, a smaller set of possible human models can also be effectively learned (Sreedharan, Chakraborti, and Kambhampati 2018).

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

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. To start with, we need a way to identify potential implicit subgoals. We can do so by exploiting two salient facts about our setting. Firstly, all implicit subgoals are bottleneck states, and secondly, bottleneck states are preserved through determinization, i.e., converting stochastic transitions into deterministic ones (Keller and Eyerich 2011).

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 $|\delta(\mathbb{M}^H)| \leq |\mathbb{M}^H|$. This means that even if we initially had an infinite set of possible human models, we can calculate the possible implicit subgoal by operating over a finite set of models. 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*

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

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. We will denote the set of maximally achievable subsets of bottleneck states as $\mathbb{I}$. Every element $\mathcal{I}' \in \mathbb{I}$, must satisfy the following three conditions, namely, $\mathcal{I}' \subseteq \mathcal{H}^0_\mathcal{I}$, there exists a policy π in the robot model that satisfies $\mathcal{I}'$ and

finally, there exists no other set of bottleneck state $\hat{\mathcal{I}} \supseteq \mathcal{I}'$ that can be achieved in the robot model, such that $\hat{\mathcal{I}} \subseteq \mathcal{H}^0_{\mathcal{I}}$.

Before we discuss the search procedure to find maximal subsets $\mathbb{I}$, we need a procedure that can identify policies that achieve any given subgoal, if such a policy exists. Unfortunately, it's not easy to determine this from the original MDP, but one could test for its existence in 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 $S^{\hat{S}}$ includes information about visited subgoals, updated via $T^{\hat{S}}$, and $R^{\hat{S}}$ rewards only traces passing through all subgoals. For an optimal policy $\hat{\pi}$ for $\mathcal{M}^{\hat{S}}$, all traces must exit at the goal state copy after visiting all subgoal, if and only if, exists a policy π for $\mathcal{M}$ that achieves $\hat{S}$.*

This proposition is true since any trace that ends in a goal state without passing through all the subgoal states will not result in a positive reward. However, one might not be able to tell if the identified policy corresponds to such a policy from its value. One needs to run a test over the determinized version of $\mathcal{M}^{\hat{S}}$, similar to Proposition 2, to do so. However, here 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 move on to the problem of generating queries, which we again convert 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 $\mathcal{M}^Q = \langle S^Q, A^Q, T^Q, s^Q_0, \gamma, R^Q \rangle$, 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^Q_0: Start state where nothing is known.*
- *R^Q: Reward function returning a heavy penalty for queries and positive values for achievable states.*

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

```
Input: M^R, B
Output: Set of maximal achievable subsets 𝕀
function FindMaxAchievableSubsets(M^R, B)
    return GenerateAndTestSubsets(B, M^R)
end function
function GenerateAndTestSubsets(B̂, M^R)
    if CheckAchievability(B̂, M^R) then
        return {B̂}
    end if
    if |B̂| == 0 then
        return ∅
    end if
    𝕀̂ = {}
    for s_i ∈ B̂ do
        current_subset ← B̂ \ {s_i}
        max_subsets ←GenSubsets(current_subset, M^R)
        for Ŝ ∈max_subsets do
            𝕀̂ = 𝕀̂ ∪ Ŝ
        end for
    end for
    return 𝕀̂
end function
```

- *γ: High discount factor to consider future query costs.*

Proposition 4. *An optimal policy for the MDP $\mathcal{M}^Q$, corresponds to an optimal query strategy (as per 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.

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

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

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,

Theorem 1. *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, 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, the order in which the non-achievable states need to be queried does not matter as its not part of any achievable subsets and thus it 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. □

Empirical Evaluations

We present two forms of empirical evaluations: a computational experiment and a human-user study.

Computational Experiments

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[3].

Environments. We focused on popular domains from MDP planning literature where we could easily randomize the dynamics and the possible set of bottlenecks. This included basic navigation settings like Maze and Four-rooms. PuddleWorld domain introduces a penalty for navigating over puddles. And finally, we looked at RockWorld, which features two types of rocks: valuable rocks that provide rewards when collected and dangerous rocks that incur penalties. This domain helped us test our method in scenarios requiring resource management and risk-reward trade-offs.

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.

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

[3]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.

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 64 states. Maze, Puddle, and Rocks domains showed similar performance metrics.

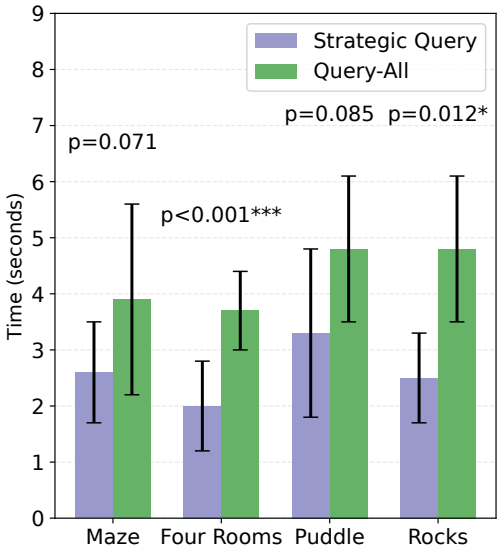

Figure 2: Performance comparison between Strategic Query and Query-All approaches across four environments with 16 states. Results show mean execution times ± standard deviation based on 20 human preference models, 10% obstacle density, and 3 runs per configuration.

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 (Table 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 models but sensitivity to higher environment complexity.

Domain	State Space 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	36	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

Table 2: Query counts and reduction percentages, same configuration of Figure 2.

User Study

We now compare our strategic querying method against an exhaustive baseline via a user study on Prolific (Palan and Schitter 2018).

Study Design and Hypothesis. We employed a within-subjects design where participants ($n = 54$) were asked to evaluate two robot models, XR35 (using exhaustive querying) and ST55 (using our strategic querying approach). Particularly, participants were asked to evaluate which of the two robots they preferred to purchase by being randomly assigned to one of two scenarios: a *Tour Guide* scenario ($n = 28$), in which participants wanted to visit a museum while passing by the port (a bottleneck in their mental model due to believing ferry transport was necessary); and an *Office Assistant* scenario ($n = 26$), where participants requested coffee with an implicit expectation that the robot would use dark roast beans from the shelf (believing the robot could not access the fridge or pantry where light roast was stored). These scenarios were designed to create natural discrepancies between user and robot world models, where users have implicit subgoals based on perceived bottlenecks.

For each robot, participants first observed the robot asking questions about their task preferences, then watched a video of the robot executing the task. Specifically, the XR35 robot asked exhaustive questions about every possible location or action, while the ST55 robot asked only one targeted question about a critical bottleneck state identified by our algorithm. After experiencing both robots, participants indicated which model they would purchase and provided a justification for it. Moreover, they rated each robot on six dimensions using 7-point Likert scales: willingness to use the robot, perceived competence, behavioral predictability, reliability, faith in future performance, and overall trust. More details about the study (including a detailed description, demographics, etc.) can be found in the supplement.

Our main hypothesis for this study is as follow:

H1: Users will significantly prefer agents performing strategic querying over ones that perform exhaustive querying, when asked to pick one over the other.

Results. Our analysis revealed that a clear majority of participants (68.5%) preferred the ST55 robot over the XR35 (31.5%) across both scenarios. Using a binomial test against the null hypothesis of no preference, this difference is statistically significant with $p = 0.009$ and a medium effect size (Cohen's $h = 0.38$). Examining each scenario individually, we observed consistent but varying levels of preference: in the Tour scenario, 71.4% of participants ($20/28$) preferred ST55, a statistically significant preference ($p = 0.03$), while in the Office scenario, 65.4% of participants ($17/26$) preferred ST55, which did not reach statistical significance ($p = 0.16$). Despite the difference in statistical significance between scenarios, a chi-square test of independence revealed no significant difference in preference patterns between the two scenarios ($p = 0.85$), suggesting that the preference for strategic querying is robust across different task contexts. The stronger preference in the Tour scenario may reflect the greater query burden in that context, where the exhaustive approach required 22 questions compared to only five in the Office scenario. Participants demonstrated moderate accuracy (68.1%) in selecting responses matching optimal execution, with a mean of 4.08 mistakes (median: 4.0, SD: 3.15). Error distribution varied: 18.6% made two or fewer mistakes, while 42.4% made exactly four. Navigation questions yielded 2.37 mistakes per participant versus 1.71 for Office scenarios, suggesting clearer decision space in the latter. Certain questions proved particularly challenging, coffee bean loading (96.4-100% error rates) and visiting station D4 (85.7% error), indicating pronounced model discrepancies. These patterns validate strategic querying at critical decision points, supporting H1: bottleneck-based querying provides better user experience than exhaustive approaches while confirming users hold implicit subgoal assumptions differing from optimal robot execution.

Related Work

Our work intersects with several areas of research.

Subgoal Discovery. Subgoal discovery improves RL sample efficiency through hierarchical decomposition via goal-space planning (Lo et al. 2024), slow dynamics (Li et al. 2021), sub-goal trees (Jurgenson et al. 2020), and high-probability occurrence on successful trajectories (Pateria et al. 2022). Other methods include option discovery (Sutton, Precup, and Singh 1999), successor representations (Kulkarni et al. 2016), and bottleneck-based approaches (Stolle and Precup 2002; Şimşek and Barto 2009). These methods assume a shared world model and optimize the agent's planning efficiency. In contrast, our method identifies states users may expect the agent to reach or avoid due to differing understanding of task feasibility.

Reward Misspecification. Inverse reward design (Hadfield-Menell et al. 2017) addresses reward misspecification by inferring true objectives from reward specifications. Extensions include risk-sensitive inverse RL (Majumdar et al. 2017), implicit preferences (Shah et al. 2019), reward hacking (Gleave et al. 2021), and formalizing misspecification through agent-user expectation discrepancies (Mechergui and Sreedharan 2024a). Our work extends these ideas by focusing on unspecified subgoals arising from differing task models.

Planning with Different World Models. Agents and humans often have different world models, explored in AI planning and human-robot interaction. Model reconciliation (Chakraborti et al. 2017; Sreedharan, Chakraborti, and Kambhampati 2021; Vasileiou et al. 2022, 2024) bridges knowledge gaps between users and AI systems for explainable planning. Related work addresses learning from corrections with differing feature spaces (Bobu et al. 2018) and representation misalignment (Peng et al. 2024). Theory of mind research suggests abstract causal models enhance intervention planning (Ho and Griffiths 2021), complementing model reconciliation approaches.

Handling Unspecified User Goals. Query mechanisms enable clarification under ambiguous rewards. Hidden Goal MDPs provide decision-theoretic frameworks for intelligent assistance under objective uncertainty (Fern et al. 2007), while assistive games use human reward inference (Hadfield-Menell et al. 2016). Within hierarchical RL, POMDP frameworks query humans about subgoals (Nguyen, Bisk, and III 2022), and natural language optimizes abstractions to minimize queries (Zheng et al. 2023). Our work leverages estimates of human task model understanding to identify more effective queries.

Preferences in Planning. Preference-based planning has evolved from simple goal satisfaction to frameworks capturing user intentions and quality criteria (Baier and McIlraith 2008), including temporal preferences (Gerevini and Long 2006), soft versus hard constraints (Gerevini et al. 2009), and hierarchical knowledge (Sohrabi, Baier, and McIlraith 2009). Preference elicitation remains a major bottleneck (Mantik, Li, and Porteous 2022). Our subgoal querying method contributes to preference elicitation approaches.

Conclusions

The paper presents a way a planning system can identify hidden subgoals of users, even when the human model may not be fully 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 and with a user study. As future work, we would like to investigate other forms of user preference, including trajectory preferences expressed through various temporal logical fragments. As a related effort, we will also consider other forms of querying strategy, including querying about possible preference between trajectories, similar to ones adopted by RLHF (Ouyang et al. 2022). We also plan to investigate various forms of approximations, including faster suboptimal algorithms and the use of pre-trained large language models as proxies for user models (Zhou et al. 2024). We also hope to extend and test our approach in continuous state-space domains such as robotics.

Acknowledgements

Silvia Tulli's research was partially funded by HumanE-AI-Net grant 952026. Sarath Sreedharan's research is partially funded by NSF through grant 2303019. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or governments.

References

Baier, J. A.; and McIlraith, S. A. 2008. Planning with Preferences. *AI Magazine*, 29(4): 25–36.

Biyik, E.; Anari, N.; and Sadigh, D. 2024. Batch Active Learning of Reward Functions from Human Preferences. *J. Hum.-Robot Interact.*, 13(2).

Bobu, A.; Bajcsy, A. V.; Fisac, J. F.; and Dragan, A. D. 2018. Learning under Misspecified Objective Spaces. In *Conference on Robot Learning*.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: moving beyond explanation as soliloquy. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 156–163. AAAI Press. ISBN 9780999241103.

Fern, A.; Natarajan, S.; Judah, K.; and Tadepalli, P. 2007. A Decision-Theoretic Model of Assistance. In *International Joint Conference on Artificial Intelligence*.

Gerevini, A.; and Long, D. 2006. Plan Constraints and Preferences in PDDL3. *ICAPS*, 7.

Gerevini, A.; Palombi, S.; Saetti, A.; Serina, I.; and Long, D. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6): 619–668.

Gleave, A.; Dennis, M. D.; Legg, S.; Russell, S.; and Leike, J. 2021. Quantifying Differences in Reward Functions. In *International Conference on Learning Representations*.

Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S.; and Dragan, A. D. 2017. Inverse reward design. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6768–6777. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964.

Hadfield-Menell, D.; Russell, S.; Abbeel, P.; and Dragan, A. D. 2016. Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, 3909–3917. Advances in Neural Information Processing Systems.

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

Ho, M. W. K.; Saxe, R.; and Cushman, F. A. 2022. Planning with Theory of Mind. *Trends in Cognitive Sciences*, 26: 959–971.

Jurgenson, T.; Avner, O.; Groshev, E.; and Tamar, A. 2020. Sub-goal trees: a framework for goal-based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Keller, T.; and Eyerich, P. 2011. A Polynomial All Outcome Determinization for Probabilistic Planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 21(1): 331–334.

Kulkarni, T. D.; Saeedi, A.; Gautam, S.; and Gershman, S. J. 2016. Deep Successor Reinforcement Learning. *arXiv preprint arXiv:1606.02396*.

Li, S.; Zheng, L.; Wang, J.; and Zhang, C. 2021. Learning Subgoal Representations with Slow Dynamics. In *International Conference on Learning Representations*.

Liu, M.; Zhu, M.; and Zhang, W. 2022. Goal-Conditioned Reinforcement Learning: Problems and Solutions. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 5502–5511. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Lo, C.; Roice, K.; Panahi, P. M.; Jordan, S. M.; White, A.; Mihucz, G.; Aminmansour, F.; and White, M. 2024. Goal-Space Planning with Subgoal Models. *Journal of Machine Learning Research*, 25(330): 1–57.

Majumdar, A.; Singh, S.; Mandlekar, A.; and Pavone, M. 2017. Risk-sensitive Inverse Reinforcement Learning via Coherent Risk Models. In *Robotics: Science and Systems*.

Mantik, S.; Li, M.; and Porteous, J. 2022. A preference elicitation framework for automated planning. *Expert Systems with Applications*, 208: 118014.

Mechergui, M.; and Sreedharan, S. 2024a. Expectation alignment: handling reward misspecification in the presence of expectation mismatch. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24. Red Hook, NY, USA: Curran Associates Inc. ISBN 9798331314385.

Mechergui, M.; and Sreedharan, S. 2024b. Goal Alignment: Re-analyzing Value Alignment Problems Using Human-Aware AI. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(9): 10110–10118.

Nguyen, K. X.; Bisk, Y.; and III, H. D. 2022. Learning When and What to Ask: a Hierarchical Reinforcement Learning Framework.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Palan, S.; and Schitter, C. 2018. Prolific. ac—A subject pool for online experiments. *Journal of Behavioral and Experimental Finance*, 17: 22–27.

Pateria, S.; Subagdja, B.; Tan, A.-H.; and Quek, C. 2022. End-to-End Hierarchical Reinforcement Learning With Integrated Subgoal Discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12): 7778–7790.

Peng, A.; Li, B. Z.; Sucholutsky, I.; Kumar, N.; Shah, J.; Andreas, J.; and Bobu, A. 2024. Adaptive Language-Guided Abstraction from Contrastive Explanations. In *8th Annual Conference on Robot Learning*.

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

Reddy, S.; Dragan, A. D.; Levine, S.; Legg, S.; and Leike, J. 2020. Learning human objectives by evaluating hypothetical behavior. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Shah, R.; Krasheninnikov, D.; Alexander, J.; Abbeel, P.; and Dragan, A. 2019. Preferences Implicit in the State of the World. In *International Conference on Learning Representations*.

Şimşek, Ö.; and Barto, A. G. 2009. Skill characterization based on betweenness. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 21*, 1497–1504.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2009. HTN planning with preferences. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 1790–1797. Pasadena, California, USA.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2018. Handling Model Uncertainty and Multiplicity in Explanations via Model Reconciliation. *Proceedings of the International Conference on Automated Planning and Scheduling*, 28(1): 518–526.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2021. Foundations of explanations as model reconciliation. *Artificial Intelligence*, 301: 103558.

Stolle, M.; and Precup, D. 2002. Learning Options in Reinforcement Learning. In *Abstraction, Reformulation, and Approximation*, volume 2371 of *Lecture Notes in Computer Science*, 212–223. Berlin, Heidelberg: Springer.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2): 181–211.

Vasileiou, S. L.; Kumar, A.; Yeoh, W.; Son, T. C.; and Toni, F. 2024. Dialectical Reconciliation via Structured Argumentative Dialogues. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 777–787.

Vasileiou, S. L.; Yeoh, W.; Son, T. C.; Kumar, A.; Cashmore, M.; and Magazzeni, D. 2022. A Logic-based Explanation Generation Framework for Classical and Hybrid Planning Problems. *Journal of Artificial Intelligence Research*, 73: 1473–1534.

Zheng, R.; Nguyen, K.; Daum'e, H.; Huang, F.; and Narasimhan, K. 2023. Progressively Efficient Learning. *ArXiv*, abs/2310.13004.

Zhou, S.; Li, S.; Meng, Y.; Jiao, Y.; Ji, H.; and Han, J. 2024. Establishing Knowledge Preference in Language Models.