Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

Proactive Assistance Agent with Online Goal Recognition

Qihao Shen[1], **Guang Hu**[1], **Chenyuan Zhang**[2]
[1]The University of Melbourne
[2]Monash University
qihaos@student.unimelb.edu.au, guang.hu@unimelb.edu.au, chenyuan.zhang@monash.edu

Abstract

Effective human–AI collaboration requires agents to do more than simply follow instructions; they must also infer and anticipate human intentions. This ability to understand and predict others' goals is also important in decentralized multi-agent settings, where communication is limited. Although goal recognition has been widely studied in the planning community, its integration with downstream collaborative behaviors, such as determining when and how an agent should assist, remains underexplored. This paper presents a unified model-based framework that combines online goal recognition with a proactive assistance module, enabling an agent to provide efficient support without prior knowledge of another agent's true objective. Experiments in a grid domain show our proposed approach significantly improves collaborative efficiency over existing algorithms and demonstrates strong robustness across diverse configurations. By integrating goal recognition with other decision-making components, this work establishes a solid algorithmic foundation and an extensible framework for the capabilities required in next-generation human–AI collaboration.

Code — https://github.com/qihaoshen0907/Proactive_Assistance_Agent

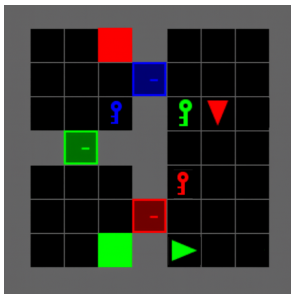
Introduction

Centralized coordination has been studied extensively in multi-agent reinforcement learning (Hu et al. 2024; Ning and Xie 2024) and multi-agent path finding (Chung et al. 2024; Gao et al. 2024). Yet fully centralized architectures represent only a narrow portion of real deployments. In most practical settings, agents, whether human or artificial, operate with decentralized control and local information (Emmanuel 2025; Kaur and Sharma 2025). Collaborative robots (cobots) exemplify this situation because they are designed to work alongside people under partial observability and independent decision-making (El Zaatari et al. 2019). Even in highly centralized environments such as Amazon warehouses, supporting decentralized behavior remains essential for maintaining efficiency (Emmanuel 2025). In current systems, routine human interventions often halt robot operations entirely in order to ensure safety. A decentralized framework that models both the central controller and the human as interacting agents could enable safe and continuous operation without full interruption.

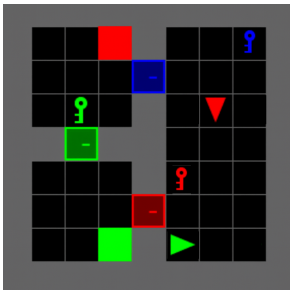
Decentralization also increases uncertainty about other agents' intentions, since explicit communication can be costly or infeasible due to bandwidth limitations, attention demands, or environmental constraints. For example, a mobile robot cannot repeatedly query a nearby worker who is focused on a task; instead, it must infer the worker's goal from observable behavior, such as whether the person is moving toward a loading area or preparing items for transfer. Goal recognition, which infers latent objectives from observed actions, provides a principled basis for effective coordination in decentralized multi-agent settings.

Goal recognition has been extensively studied within the automated planning community (Zhang, Kemp, and Lipovetzky 2023; Masters and Vered 2021; Vered, Kaminka, and Biham 2016; Ramırez and Geffner 2009). However, most existing assisting agents provide help only when users explicitly request it (Bercher et al. 2021; Roberts, Baker, and Andrew 2024; Zhang et al. 2023; Oh, Meneguzzi, and Sycara 2014). Bringing goal recognition into practical settings that require downstream collaborative behavior, such as proactive assistance, raises several important challenges (Masters and Vered 2021). First, many existing approaches assume a passive observer, while real-world assisting agents must often act to obtain informative observations (Bercher et al. 2021; Zhang et al. 2023). Second, effective assistance requires tighter coupling between goal inference and decision-making than simply attaching a goal-recognition component to a separate planner, which is the structure used in much prior work (Pozanco et al. 2018; Shvo et al. 2022). Third, the objectives of goal inference and assistance may diverge or even conflict. For instance, maintaining proximity to a human may improve predictive accuracy, while moving elsewhere may yield more effective help for task completion (Gall 2021). The algorithm should be able to achieve a good trade-off between these two.

To build intuition, consider a scenario in which an actor navigates an environment to reach one of two candidate goals. The helper aims to support the actor in achieving the true goal as efficiently as possible. As shown in Figure 1a, suppose the actor (green triangle) intends to reach the red goal. To do so, it must collect the red key, open the red door,



(a) To reach the green goal, the actor must open the red door. To reach the red goal, the actor must open the red door and then the green door.



(b) To reach the green goal, the actor must open the red door. To reach the red goal, the actor only needs to open the blue door.

Figure 1: Illustrated examples for proactive assistance. The green triangle represents the actor, and the red triangle represents the helper. The two potential goals are marked in green and red. Keys of different colors are placed in distinct locations.

then collect the green key and open the green door before arriving at the final destination. If the helper (red triangle) can predict the actor's true goal, then it can open the blue door in advance. If the helper does not perform the prediction, it may instead pick up the green key simply because it is nearby and open the green door before the actor. This behavior is not the most effective way to support the actor's true goal. This simple example illustrates the importance of performing goal recognition during assistance tasks.

This paper introduces proactive agent assistance via online goal recognition, a unified framework that integrates real-time goal inference with cooperative decision making, enabling the assisting agent to reason and act under uncertainty. It consists of two core components: an online goal recognition process based on subtask transitions, and an action selection strategy driven by expected utility. These components allow the assisting agent to interpret the actor's intentions, anticipate future task progress, and choose behavior that improve overall efficiency.

Related Work

This section reviews the key developments in related areas and clarifies how our work differs from existing studies.

Model-based Goal Recognition

Goal recognition is the task of inferring an actor's unknown goal from a sequence of observed actions. Ramırez and Geffner (2009) compile this problem as a planning task, known as Plan Recognition as Planning (PRP). In subsequent work, they evaluate each candidate goal by computing the cost of an optimal plan that achieves it, and then compare this cost with the cost of a plan that is constrained to be consistent with the actor's actions. Goals whose optimal plans align more closely with the observed actions receive higher support (Ramírez and Geffner 2010).

Building on these foundations, Ramırez and Geffner (2011) extend model-based goal recognition to agents acting in partially observable, stochastic environments. In their formulation, the observed agent is modeled as approximately optimal with respect to a shared POMDP model whose dynamics and costs are known, while the true goal remains hidden. A POMDP planner is used to compute a value function for each candidate goal, and execution trajectories are then sampled by treating the agent's action choices as Boltzmann-rational. The likelihood of an observation sequence under a goal is approximated by the fraction of sampled trajectories that are consistent with the observations, and Bayes' rule is used to obtain a posterior over goals. This POMDP-based formulation illustrates how goal recognition can systematically handle missing and noisy observations, and it motivates our use of belief-based goal inference in decentralized assistance settings.

Although goal recognition is well developed in the planning community, its integration into downstream decision making remains relatively limited. For example, Pozanco et al. (2018) formulate counter-planning as a combination of goal recognition and planning, in which an agent predicts an opponent's probable goals and critical landmarks in order to devise strategies that block or delay goal achievement. Similarly, Masters and Sardina (2017) study how an agent can deliberately generate misleading action sequences to obscure its true intentions and shape an observer's inferences. Both lines of work highlight the value of coupling goal inference with decision making, but they focus primarily on adversarial or deceptive interactions rather than cooperative assistance.

We next turn to approaches that embed goal inference within an assisting agent's policy in cooperative MDP and POMDP environments.

Implicit Goal Recognition in MDP Environments

A complementary line of work investigates proactive collaboration in which an assisting agent reasons about another agent's goals while acting in an MDP or POMDP environment. Oh, Meneguzzi, and Sycara (2014) introduce the POMCoP framework, which combines action observation with selective querying to infer a collaborator's intentions. Their findings demonstrate that belief-space planning can significantly reduce completion time in pursuit–evasion tasks. Shvo et al. (2020) show that improving goal recognition accuracy can directly enhance an agent's downstream performance, providing motivation for approaches that embed goal inference within the decision-making process itself. Building on this idea, Shvo et al. (2022) propose the Thinking of Mind framework, which enables an agent to infer human intentions without explicit communication and to provide assistance once its belief exceeds a threshold.

Although developed for different domains, these approaches share a common computational structure with the counter-planning model of Pozanco et al. (2018). Each system first performs goal recognition and then invokes a planner to compute an action. The two stages are separated, and the resulting belief distribution does not continuously shape action selection.

Fern et al. (2014) propose the Helper Action MDP for assistance in navigation tasks. Their approach uses Bayesian inference for goal estimation and applies a greedy helper policy. They show that even without knowing the true goal, this myopic strategy achieves performance close to that of an oracle assistant, and experiments indicate reduced user effort in path-following and desktop-navigation domains. However, the helper's contribution is limited to providing simple additive bonuses that leave the underlying task structure unchanged, restricting the method's applicability to more complex domains with richer interaction patterns.

More recently, Wu et al. (2021) introduce the Bayesian Delegation model, in which agents infer their current subtasks and coordinate accordingly. Their framework operates in the Overcooked domain and assumes predefined recipe flows and domain-specific reward functions. While the method performs well in this setting, its reliance on predefined task pipelines and handcrafted domain knowledge still limits its generality. Adapting the approach to new environments requires redesigning both the reward specification and the workflow.

Building on these studies, our model advances proactive assistance in two important ways. First, it does not rely on domain-specific reward shaping or predefined task pipelines that specify how the actor must complete the task. Second, we explicitly allow the helper's actions to interfere with the actor by changing the shared environment. This design enables proactive assistance to operate effectively in a wider range of complex, less constrained environments.

Problem Formulation

This section formalizes the Proactive Assistance Problem. The setting consists of two agents in a shared environment: an actor executing actions to achieve a goal, and a helper capable of providing assistance. The aim is to compute a helper policy that selects appropriate assistive actions to improve the actor's task performance without knowledge of the actor's goal.

We model the Proactive Assistance Problem as a factored DEC-POMDP with full state observability, where uncertainty arises from the observer's lack of knowledge of the actor's true goal and policy. Formally, the model is

$$\mathcal{M} \;=\; \langle S, s^0, G, g^*, \mathcal{A}, \Gamma, \pi_a \rangle,$$

where $G = \{g_1, \ldots, g_M\}$ is the finite set of candidate actor goals and $g^* \in G$ is the (fixed) true goal pursued by the actor. The helper does not know g^* and must act without access to it.

We factor the state space as $S = S_a \times S_h \times S_{\text{env}}$, with S_a and S_h the actor and helper local states (e.g., position, orientation) and S_{env} contains other environment variables. Each state $s \in S$ is $s = (s_a, s_h, s_{\text{env}})$, with initial state $s^0 = (s_a^0, s_h^0, s_{\text{env}}^0)$. Each candidate goal $g \in G$ is a partial state over S_a. We say the actor achieves goal g iff $g \subseteq s_a$. The actor pursues a true goal $g^* \in G$, which is unknown to the helper.

Let $\mathcal{A} = \mathcal{A}_a \times \mathcal{A}_h$ denote the joint action space, where $\mathcal{A}_a$ and $\mathcal{A}_h$ are the actor's and helper's action spaces, respectively. Let $\Gamma : S \times \mathcal{A}_h \times \mathcal{A}_a \times S \to [0, 1]$ denote the transition function, where within each time step actions the helper acts before the actor. Γ therefore can be factored into local transition models $\Gamma_i : S \times \mathcal{A}_i \times S \to [0, 1]$ for $i \in \{h, a\}$, so

$$\Gamma(s^t, \mathbf{a}_h^t, \mathbf{a}_a^t, s^{t+1}) = \sum_{s' \in S} \Gamma_h(s^t, \mathbf{a}_h^t, s')\, \Gamma_a(s', \mathbf{a}_a^t, s^{t+1}).$$

The actor selects actions via a policy function $\pi_a : S \times G \to \mathcal{A}_a$ such that $\mathbf{a}_a^t = \pi_a(s^t, g^*)$. This policy may be reactive or derived from classical planning, and it is not accessible to the helper.

We assume that the actor seeks to minimize its task cost (e.g., path length). When a helper is present, its role is to reduce this cost. Let π_h be the helper policy function $\pi_h : S \times G \to \mathcal{A}_h$ such that $\mathbf{a}_h^t = \pi_h(s^t, g^*)$. For the true goal g^* and initial state s_0, we define the *assistance improvement* $\Delta(\pi_h; s_0, g^*)$ as the difference between the actor's solo cost and the cost incurred when it is assisted by the helper policy π_h. The objective of Proactive Assistance Problem is to find the optimal policy π_h^* to maximize the assistance improvement:

$$\pi_h^* = \arg\max_{\pi_h} \Delta(\pi_h; s_0, g^*)$$

.

Methodology

The formulation in the last section provides the objective of the Proactive Assistance Problem. However, as g^* is unknown to the helper, choose π_h to optimize $\Delta(\pi_h; g^*)$ directly is infeasible. We therefore adopt a two-stage online approach: (i) a goal recognition module that maintains a belief $b^t(g) = \Pr(g^* = g \mid \text{history}^t)$ over G based on observed actor behavior and we assume the belief space is B; and (ii) an action selection module then selects actions to maximize the expected improvement given the current state and belief: $\pi_h : S \times B \to \mathcal{A}_h$. As in POMDPs, the belief b^t is a *sufficient statistic* for decision making, compactly summarizing the observation history relevant to the helper's next action. Concretely, at time t the helper select $\pi_h^*(s^t, b^t) = \arg\max_{a \in \mathcal{A}_h} \mathbb{E}_{g \sim b^t}\big[\Delta(\pi_h; s, g)\big]$.

The online algorithm proceeds as follows. At each timestep t, the helper first updates b_t through the goal recognition module using the current observations. The action selection module then computes a_t^h by maximizing the belief-weighted improvement. The selected action is executed, new observations are obtained, and the process repeats. The resulting policy tightly couples inference with control in the Proactive Assistance Problem: as the belief distribution becomes more concentrated, the helper naturally transitions from information-gathering actions to task-advancing assistance.

We first present a running example together with the of our proposed approach, and then describe each component in detail.

Running Example

We use Figure 1b as a running example to illustrate our proposed approach. In this scenario, the actor's true goal is $g^* = g_{\text{red}}$.

The initial state is $s^0 = (s_a^0, s_h^0, s_{\text{env}}^0)$. In this example, $s_a^0 = ((5,7), \text{right}, \text{none})$, $s_h^0 = ((6,3), \text{down}, \text{none})$. The environment contains three keys (green at $(2,3)$, red at $(5,5)$, blue at $(7,1)$), three corresponding doors (blue at $(6,2)$, green at $(4,2)$, red at $(4,6)$, all initially closed), and two candidate goals $G = \{g_{\text{red}} = (3,1),\ g_{\text{green}} = (3,7)\}$. The actor follows a goal-directed policy toward its private goal g_{red}. The helper maintains a belief over the two candidate goals, initialized as $P(g_{\text{red}}) = P(g_{\text{green}}) = 0.5$, and updates this belief online based on the actor's observed motion.

If no helper were present, the shortest path for the actor to reach g_{red} from s^0 would require 17 steps: it must first move toward the blue key in the upper-right region, pick it up, unlock the blue door, and then continue toward g_{red}. As the actor starts moving upward and right, it moves farther from the red and green keys and becomes increasingly aligned with the blue key. After just a few steps, the helper's belief may shift, for example, from $P(g_{\text{red}}) = 0.5$ to $P(g_{\text{red}}) \approx 0.8$, indicating that the observed behavior is more consistent with g_{red} than with g_{green}. This corresponds to the proactive goal-recognition phase (Lines 1–9 in Algorithm 1), during which the helper refines its belief at every timestep.

As additional evidence continues to favor g_{red}, the helper commits more strongly to this hypothesis. It moves toward the blue key, picks it up, and carries it to the blue door. After performing the toggle action, the door unlocks and the environment state changes for the actor. Because belief updating and assistance occur in an interleaved fashion, the helper may begin moving toward the blue key even while some uncertainty remains; it chooses actions that support the most likely goal while remaining compatible with alternative hypotheses. This behavior reflects the assistive action-selection phase in Lines 10–21 of Algorithm 1.

On the next timestep, the actor observes that the blue door is now open, replans accordingly, and follows the shorter path enabled by the helper's intervention. In this configuration, the assisted trajectory requires only 11 steps to reach g_{red}, yielding an assistance improvement of $\Delta(\pi_h; s^0, g_{\text{red}}) = 17 - 11 = 6$. This running example illustrates how integrating online goal recognition with proactive assistance can produce substantially more efficient behavior than the actor's self-sufficient policy.

Online Goal Recognition

Goal prediction in structured environments can be made substantially more efficient by exploiting hierarchical dependencies among environmental variables (Mirsky et al. 2017; Höller et al. 2018; Massardi, Gravel, and Beaudry 2020). Building on this insight, our framework introduces an explicit hierarchy of subtasks that captures key aspects of environment dynamics. We first formalize the notion of a subtask and describe how subtasks can be extracted and utilized within the reasoning process. These subtasks form the foundation of our Bayesian belief-update mechanism. It is important to emphasize that our subtask definition is derived directly from the environment representation, allowing subtasks to be identified automatically without any domain-specific engineering.

Algorithm 1: Proactive Assistance via Online Goal Recognition

Input: Timestep t; environment model $\mathcal{E}$; candidate goal set G; current state s^t
Output: Next helper action a_h^t

1: **if** $t = 0$ or DETECTEDSUBTASKTRANSITION(s^t) **then**
2: Update $P_t(g_i)$, $P_t(g_i \mid s^t)$, $\mathcal{T}$
3: **end if**
4: Compute $\mathcal{L}(s^t \mid c_j)$ and $P_t(c_i \mid g_j)$ based on $\mathcal{T}$
5: **if** $t = 0$ or DETECTEDSUBTASKTRANSITION(s^t) **then**
6: Compute $P_t(c_i)$
7: **else**
8: Update the goal posterior distribution $P_t(G \mid s^t)$
9: **end if**
10: $\mathcal{C} \leftarrow$ GENERATESUBTASKS$(\mathcal{E})$
11: **for** each subtask $c_i \in \mathcal{C}$ **do**
12: $E_i \leftarrow 0$
13: **for** each candidate goal $g_j \in G$ **do**
14: Update E_i based on g_j
15: **end for**
16: $\mathcal{V}[c_i] \leftarrow E_i$
17: **end for**
18: $c^* = \arg\max_{c_i \in \mathcal{C}} \mathcal{V}[c_i]$
19: $p_h^{c^*} =$ PLANPATH$(s^t, c^*, \mathcal{E})$
20: Set helper action a_h^t based on $p_h^{c^*}$
21: **return** a_h^t

Definition 1 (subtask and subtask transition) We factor the environment state into a set of components, let

$$s_{env} = (c_1, c_2, ..., c_k), s_{env} \in S_{env}$$

where each c_i is an environment variable. We refer to each such component *subtask* as a subtask. A *subtask transition* is said to occur during a state transition $\Gamma(s, \mathbf{a}_a^t, \mathbf{a}_h^t, s')$ if and only if there exists some subtask whose value changes during the state transition. When an agent's action induces such a change, we say that the agent has *completed* subtask c_i. In Figure 1b, picking up a key or unlocking a door each constitutes a subtask. When a locked door becomes unlocked, the corresponding door-state subtask undergoes a subtask transition.

Based on the above definitions, we now introduce the online goal recognition component of our algorithm.

Online Goal Recognition Initialization Our goal recognition module estimates the likelihood of each candidate goal by computing the actor's expected plan. Consequently, at the initial time step $t = 0$, or whenever a subtask transition is detected, the actor may react to the updated environment and generate a new plan. When such a change occurs, the helper must restart the goal recognition process by using the previous posterior distribution as the new prior, ensuring that the inference remains consistent with the actor's most recent behavior.

After updating the priors, the helper constructs a task network $\mathcal{T}(g_j) = c_1 c_2 \dots c_n$, which represents the ordered se-

quence of subtasks that the actor must complete in order to reach each candidate goal g_j. The network is generated using a breadth first search procedure. For every subtask c_i that the actor may complete, the algorithm precomputes two quantities. The first quantity is the optimal task cost $d^*_{\text{opt}}(c_i)$, which specifies the minimal cost of completing c_i from the current state. The second quantity is the minimal task count $L^*_{\text{opt}}(g_j) = |\mathcal{T}(g_j)|$ for each candidate goal g_j, which indicates the smallest number of subtasks required to achieve that goal. These operations correspond to Lines 1-3 of the Algorithm 1.

Likelihood Updates For Subtasks And Goals In our hierarchical formulation, goal recognition is performed at two levels: at the high level over the subtask network, and at the low level over the actor's executed actions that lead to the next subtask.

In Line 3, the helper evaluates how closely the actor is following an optimal sequence of subtasks. Following the PRP formulation of Ramırez and Geffner (2009), the difference between the current cost and the optimal cost for each subtask c_i is computed as $\Delta d_i = d_{\text{cur}}(c_i; s^t) - d^*_{\text{opt}}(c_i)$, where $d_{\text{cur}}(c_i; s^t)$ is the cost implied by the actor's observed behavior at state s^t. A smaller difference indicates that the actor is behaving in a way that more closely resembles an optimal plan for subtask c_i.

The helper then updates the likelihood of each subtask using the Softmax transformation introduced in the probabilistic PRP model (Ramírez and Geffner 2010):

$$\mathcal{L}(s_t \mid c_i) = \frac{\exp(-\Delta d_i)}{\sum_k \exp(-\Delta d_k)}.$$

Subtasks that deviate substantially from the optimal path receive lower likelihoods, allowing the helper to focus on subtasks that are more plausible given the actor's current behavior.

In parallel, the helper evaluates the actor's high-level progress toward each candidate goal. For every candidate goal g_j, the helper computes the remaining number of subtasks $L(g_j, c_i)$ that must still be completed after the current subtask c_i. This value is compared with the precomputed minimal task count $L^*_{\text{opt}}(g_j)$ to obtain a task progress difference, computed analogously to the cost based measure above. Using the same transformation as before, the helper converts this deviation into a likelihood $\mathcal{L}_t(c_i \mid g_j)$ for each candidate goal, reflecting how well the observed task progression aligns with pursuing g_j.

Based on the preceding steps, we now derive the complete Bayesian update used by the helper. For each candidate goal g_j, the belief $b^t(g_j)$ is computed as

$$b^t(g_j) = P_t(g_j \mid s^t) \tag{1}$$

$$= \sum_i P_t(g_j \mid c_i) P_t(c_i \mid s^t) \tag{2}$$

$$where \quad P_t(g_j \mid c_i) = \frac{\mathcal{L}_t(c_i \mid g_j) P_t(g_j)}{\sum_k \mathcal{L}_t(c_i \mid g_k) P_t(g_k)} \tag{3}$$

$$P_t(c_i \mid s^t) = \frac{\mathcal{L}_t(s^t \mid c_i) P_t(c_i)}{\sum_m \mathcal{L}_t(s^t \mid c_m) P_t(c_m)} \tag{4}$$

Where $P_t(g_j \mid s^t)$ denotes the posterior probability that the actor is pursuing candidate goal g_j at time t. $P_t(c_i)$ is the prior probability of subtask c_i before incorporating the current observation and $P_t(g_j)$ similarly denotes the prior probability of candidate goal g_j before the Bayesian update.

When $t \neq 0$ and no subtask transition is detected, the helper updates the likelihood values $\mathcal{L}_t(s^t \mid c_i)$ based on the observations at time t, and recomputes the goal posterior $P_t(g_j \mid s^t)$ accordingly. Intuitively, subtasks whose expected costs are more consistent with the observed behavior are assigned higher likelihood values $\mathcal{L}_t(s^t \mid c_i)$, and thus receive greater posterior probability mass after the update. This belief update allows the helper to progressively down-weight inconsistent goals while reinforcing the most likely ones as more observations are accumulated. The procedure is implemented in Line 7 of Algorithm 1.

Subtask Prior Computation At time $t = 0$, or whenever a subtask transition is detected, the inference enters a transition step in which the helper reinitializes the subtask distribution for the newly introduced subtasks. This step maintains consistency between hierarchical levels while preventing the reuse of previous observations. As a result, the subtask priors $P_t(c_i)$ are derived from the updated goal posterior, corresponding to Lines 4-6 of Algorithm 1.

In our framework, for each subtask c_i, we construct a goal distribution $P_t(g_j \mid c_i)$ based on the cost deviation between the observed behavior and the optimal plan, as defined in Eq. (3). We then estimate a set of subtask weights $\mathbf{w}$ such that the linear combination of these subtask-conditioned goal distributions best approximates the current goal belief $P_t(g_j)$. Concretely, we solve the following least-squares problem:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \left\| \sum_i w_i \, P_t(g_j \mid c_i) - P_t(g_j) \right\|_2^2 .$$

The resulting weights are transformed via exponentiation and normalization to ensure a valid probability distribution:

$$P_t(c_i) = \frac{\exp(w_i)}{\sum_k \exp(w_k)}.$$

This procedure reconstructs the current goal belief using subtask-level predictions, and derives a consistent subtask prior without requiring an explicit mapping from goals to subtasks (Nguyen, Ho, and Rinaldo 2024).

Action Selection

In this section, we explain how the goal posteriors are used by the action selection module.

Generating Candidate Subtasks After computing the posterior goal distribution $P(G \mid s^t)$, the helper must determine how to intervene effectively. It begins by enumerating all subtasks that could provide beneficial assistance (Line 8 of Algorithm 1). Completing a subtask may alter the environment dynamics and thereby change the actor's subsequent optimal plan. To identify such subtasks, we provide the helper with an abstract environment graph $\mathcal{E}$ that

captures structural dependencies among rooms, keys, doors, and other interactive components. For example, in Figure 1b, each room is represented as a node annotated with the keys and doors it contains, and edges encode feasible transitions between rooms. Based on this $\mathcal{E}$, the helper performs a breadth-first search to enumerate all reachable environment components whose state changes could influence the actor's future behavior. This process yields a candidate set of subtasks, $C = \{c_1, c_2 ... c_k\}$ each corresponding to a specific environment feature whose completion (e.g., unlocking a door) can potentially provide cooperative benefit.

Each c_i must satisfy the requirement that, under the current environment configuration, the helper can complete this subtask. In Figure 1b, if the helper is not holding any key, it can only perform four subtasks which are picking up each of the four keys. Because without the corresponding colored key, it cannot unlock any door. If the helper is carrying the red key, it additionally possesses the subtask of unlocking the red door.

Cost-Difference Evaluation We use the payoff value to quantify how beneficial it is for the helper to complete a given subtask on behalf of the actor. To be specific, in Lines 11–17 of Algorithm 1, the helper evaluates each subtask c_i by estimating how much its completion would reduce the actor's cost. For every candidate goal $g_j \in G$, the system computes the optimal cost C_{g_j} for the actor to achieve g_j, as well as the hypothetical cost C'_{g_j} assuming that c_i has already been completed. The improvement change is defined as $\Delta C_{g_j}(c_i) = C_{g_j} - C'_{g_j}$. A positive value indicates that completing c_i facilitates progress toward g_j, whereas a non-positive value suggests no immediate cooperative benefit. When $\Delta C_{g_j}(c_i) \leq 0$, the algorithm treats c_i as a provisional starting point and assumes that its associated subtask transition has already occurred. It performs a forward search over downstream subtasks to identify the first c_k such that $\Delta C_{g_j}(c_k) > 0$. If such a subtask exists, its improvement value is assigned as the payoff of c_i; otherwise, c_i is discarded.

In Figure 1b, picking up a key may not directly help the actor. Thus, when the key-pickup subtask $c_{K_{\text{blue}}}$ yields no improvement, the helper temporarily assumes that the key has already been collected and continues the BFS over successor subtasks. In this example, unlocking the corresponding door $c_{D_{\text{blue}}}$ becomes feasible and yields a positive improvement score, which is then propagated back as the payoff for $c_{K_{\text{blue}}}$.

Expected Payoff-Based Action Selection Since the actor's true goal is uncertain, the helper integrates belief probabilities across all goals to compute the expected payoff: $\text{ExpectedPayoff}(c_i) = \sum_j P(g_j \mid s^t) \cdot \Delta C_{g_j}(c_i)$. This value quantifies the average improvement in efficiency that completing c_i is expected to produce under the current belief state, providing a rational basis for cooperative decision-making.

In Line 16-19 of the Algorithm 1, after computing expected payoffs for all subtasks, the helper selects the one with the maximum expected benefit:

$$c^* = \arg\max_{c_i \in \mathcal{C}} \text{ExpectedPayoff}(c_i) \tag{5}$$

$$= \arg\max_{c_i \in \mathcal{C}} \sum_j P(g_j \mid s^t) \cdot \Delta C_{g_j}(c_i) \tag{6}$$

The chosen c^* represents the cooperative intervention expected to yield the greatest improvement in the actor's efficiency. The helper then generates a path $p_h^{c^*} = \textsc{PlanPath}(s_h, c^*, \mathcal{E})$ and executes the first action.

This strategy defines a greedy cooperative policy, where the helper selects at each step the subtask with the highest immediate expected payoff. Since the payoff computation considers only near-term task reduction and does not account for the helper's own cost or long-term belief evolution, the resulting behavior is myopic. Our experiments show that this greedy approach is nevertheless effective in practice. Future work can extend this framework by treating subtask selection as a planning problem over sequences of cooperative interventions, enabling the helper to optimize cumulative collaboration benefits rather than relying solely on one-step payoff estimates.

Experiment

We evaluate the proposed Proactive Assistance framework (Pro.Ass) through a series of controlled experiments conducted in a structured grid-based domain. The environment simulates a multi-room navigation task. Such environments are widely used to test agent decision-making strategies (Jin et al. 2023). The experiments are implemented using the MultiGrid Python library, which supports multi-agent simulation and discrete planning interfaces.

Environment Setup

The environment is composed of multiple interconnected rooms and a central hallway. Each room is obstructed by colored doors, with matching keys randomly distributed across rooms and the hallway. To increase task complexity, all goals are located within rooms rather than in the hallway. The actor needs to find the right key to open the matching door before reaching the goal room.

Both the actor and the helper move on a grid made of discrete cells. They share the same set of actions: $\mathcal{A}_a = \mathcal{A}_h = \{$ `MoveForward`, `TurnLeft`, `TurnRight`, `Stay`, `Pickup`, `Dropoff`, `Toggle`$\}$. Each agent's state includes its position, direction, and carried item. The environment contains doors and walls that the actor cannot traverse. The actor starts from an initial state s_a^0 and aims to reach a fixed goal location g^*. The helper also starts from a random state s^t, but unlike the actor, it is allowed to pass through the walls. The helper maintains a candidate goal set G that includes the true goal g^* and updates its belief $b^t(G)$ over time based on the actor's observed behavior.

The actor always follows an hierarchical optimal planning policy toward its true goal g^*. In our implementation, we use BFS to find the optimal hierarchical task sequence. Then, the A* algorithm (Hart, Nilsson, and Raphael 1968) is used to

Method	Separate Reco. & Assi.	GR Strategy	Assistance Strategy
Greedy	Yes	None	Greedy (nearest key & door)
Uniform	Yes	None	Payoff-based
CP-Step	Yes	Fixed-step belief update $m = 6$	Payoff-based
Thr.Base	Yes	Threshold commitment	Payoff-based
Pro.Ass	No	subtask transition Bayesian update	Excepted Payoff

Table 1: Comparison of Pro.Ass and other algorithms: Separate Reco. & Assi. means separate goal recognition and assistance.

generate the grid path for completing the first subtask in the hierarchical task sequence. When the actor notices changes in the environment, it replans the best path to stay optimal.

To control task difficulty, we set two factors: the number of rooms and the number of goals. We design three environment sizes, 3×3, 4×4, and 5×5 room setups, with 2 to 6 possible goals. Each episode ends when the actor reaches its true goal. This paper primarily examines the impact of the number of candidate goals on the algorithm's performance.

We evaluate five helper strategies across all experimental settings. In the Greedy strategy, the helper simply selects the nearest key to open the nearest door. Under the Uniform Belief strategy, the helper use our action-selection policy while maintaining a uniform prior over all candidate goals, without performing any goal recognition. The remaining two strategies use different belief-update mechanisms: one based on the Counter-Planning approach (Pozanco et al. 2018), and the other inspired by the Thinking of Mind framework (Shvo et al. 2022).

The first method (CP-Step) sets the actor's most likely candidate goal as the true goal after observing a fixed number m of actions (In our setting, $m = 6$) The second uses a threshold-based rule (Thr.Base), where the helper commits to a goal once its belief is higher than a dynamic threshold and suppresses all others. In our setting, the threshold is set 0.05 higher than the uniform probability.

A common feature of both methods is that once the helper commits to a goal, it never revises that decision. Before committing, the helper assumes a uniform prior over all candidate goals. This separation between recognition and assistance is used as a baseline in our study to examine how different belief-update rules influence assistance performance. The five baseline strategies used in our experiments are listed in Table 1. For each goal count, we generate 100 distinct instances, resulting in a total of 500 instances for experimentation to ensure statistical reliability.

Quantitative Evaluation of Assistance Effectiveness

Given our goal-directed configuration, we adopt the evaluation procedure used in Shvo et al. (2022). Let T_0 denote the number of steps required for the actor a to reach its true goal g^* when acting alone, and let T_1 denote the number of steps achieve the true goal when the actor and the helper h act together in the same environment. We define the *assistance improvement* as: $\Delta(\pi_h; s_0, g^*) = T_0 - T_1$, where a positive value indicates that the helper accelerates task completion. We further define the *saving ratio* as $\mathrm{R}_i = \frac{\Delta(\pi_h; s_0, g^*)_i}{T_0}$ to measure the relative improvement contributed by assistance on each instance. This metric complements the absolute step saving by reflecting proportional efficiency gains.

For the set of task instances $\mathcal{I}$, we report the average step saving $\overline{\Delta}(\pi_h; s_0, g^*)$ and the average saving ratio $\overline{R}$ as the primary evaluation metrics.

We also define the success rate

$$\mathrm{SuccRate} = \frac{|\{\, i \in \mathcal{I} \mid \Delta(\pi_h; s_0, g^*)_i > 0 \,\}|}{|\mathcal{I}|}$$

as the proportion of task instances in which the assistance yields a positive step saving. A higher improvement rate indicates that proactive assistance remains effective across diverse environments. Following Shvo et al. (2022), this metric is used as an auxiliary metrics of the robustness of proactive assistance.

Algo.	**# Goals**	**2**	**3**	**4**	**5**	**6**	**All**
No Help	Steps	34.00	37.04	44.90	43.27	41.24	40.04
Greedy	Δsteps	1.86	2.22	5.21	5.63	5.90	4.13
	Avg. ratio	1.69	3.25	4.18	5.44	6.30	4.16
	Success	34.00	37.04	44.90	43.27	41.24	40.04
Uniform	Δsteps	13.29	6.34	12.06	11.06	8.20	10.14
	Avg. ratio	18.52	11.37	14.35	12.29	9.12	13.12
	Success	61.00	48.15	43.88	43.27	28.87	45.17
Pro.Ass	Δsteps	14.42	9.89	17.01	15.88	11.57	13.71
	Avg. ratio	20.19	17.02	21.06	19.70	14.54	18.50
	Success	70.00	63.89	67.35	67.31	54.64	64.69
CP-Step	Δsteps	11.44	6.84	13.78	9.81	8.08	9.93
	Avg. ratio	14.91	10.94	17.79	11.23	10.39	13.00
	Success	67.00	57.41	56.12	62.50	48.45	58.38
Thr.Base	Δsteps	14.53	8.73	13.84	12.75	10.53	12.03
	Avg. ratio	20.30	14.89	18.10	15.53	12.62	16.27
	Success	77.00	61.11	59.18	65.38	48.45	62.33

Table 2: Performance comparison across goal numbers.

Results

Table 2 summarizes the overall performance across different numbers of goals. Compared with strategies that do not perform goal recognition, those equipped with goal recognition consistently achieve better performance across environments with varying numbers of goals. This further sup-

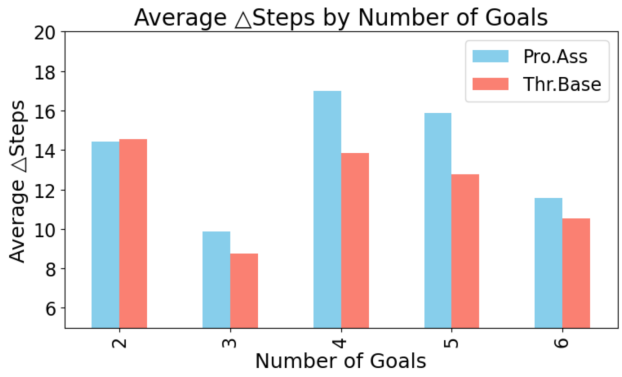


Figure 2: Average step improvement achieved by the Proactive Assistant (Pro.Ass) and the threshold-based baseline (Thr.Base) for environments with 2–6 candidate goals.

ports the conclusion of Shvo et al. (2022): when an assistant can accurately reason about the actor's goals, it can provide substantially more effective help. Compared with the goal-recognition baselines (CP-Step and Thr.Base), the proposed Pro.Ass framework performs better in most settings.

Figure 2 shows that Pro.Ass achieves larger average ΔSteps than the threshold-based baseline across most goal counts. The advantage becomes particularly pronounced when three or more goals are present, indicating that Pro.Ass scales more effectively as goal ambiguity increases. We used a two-way ANOVA to validate the results. The findings show that the performance differences between methods are statistically significant ($p < 0.05$), further confirming the effectiveness of the proposed approach. The running time remains comparable across all methods, showing no substantial differences. Detailed experimental results are available on our GitHub page.

The Thr.Base helper performs competitively when the goal set is small, where early commitment can reduce search effort and yield performance close to the theoretical upper bound. However, as the number of candidate goals grows, premature commitment often results in incorrect inference of the actor's intention. For example, in Figure 1a, suppose the actor's true goal is g_{red} and the optimal high-level sequence is: pick up the red key, unlock the red door, pick up the green key, unlock the green door, and proceed to g_{red}. Along this path, the actor inevitably passes near g_{green}, causing the Thr.Base helper to assign a higher belief to g_{green} than to the true goal g_{red}. While this behavior aligns with human intuition, it becomes increasingly problematic as the number of alternative goals grows.

Robustness Result

One potential limitation of our approach is the strong assumption regarding the actor's behavior. In our earlier experiments, the actor policy used during execution was identical to the policy assumed by the helper. In realistic settings, however, the actor model is often inaccurate, suboptimal, or unavailable (Zhang, Kemp, and Lipovetzky 2023). For instance, human users rarely follow optimal plans and may occasionally select actions that appear random.

ϵ	No Help	Pro.Ass	Uniform	Greedy	CP-Step	Thr.Base
0.0	54.55	40.84	44.41	50.42	44.62	42.52
0.1	63.59	47.29	49.65	56.56	53.23	51.34
0.2	75.25	50.55	51.75	64.34	59.62	59.62
0.3	90.58	57.69	60.09	73.85	70.12	69.32
0.4	112.52	65.35	69.11	103.40	84.29	64.11
0.5	143.84	76.18	78.26	104.83	93.06	87.07

Table 3: Steps performance comparison under different noise levels ϵ.

To evaluate the robustness of our proposed framework, we introduce stochasticity into the actor's decision process. We define a noise parameter ϵ such that, at each decision step, the actor selects a random legal action with probability ϵ instead of following its nominal policy. We assess the performance of all algorithms under noise levels ranging from 0.1 to 0.5.

Table 3 presents the comparison of assistance performance under different noise levels ϵ. As the noise increases, the performance of all methods degrades, since the actor's behavior becomes more random and less predictable. Notably, the two goal-recognition-based baselines perform worse than the uniform strategy under noisy conditions, indicating their limited ability to cope with stochastic behavior. In contrast, Pro.Ass consistently achieves the best performance across all noise levels, and its degradation is significantly smaller than that of the baseline methods. This demonstrates that Pro.Ass's Bayesian belief update based on subtask transitions, together with its cost-difference-based cooperation strategy, effectively handles behavioral uncertainty and yields strong robustness.

Conclusion

This paper presents a model-based framework for proactive assistance that integrates online goal recognition with action selection via expected payoff. The method supports real-time collaboration at subtask level, where subtasks are derived automatically from the environment representation. Our results across different settings show that the framework consistently outperforms existing baselines. Even in noisy environments, the proposed approach demonstrates the strongest robustness among all evaluated methods.

There are several promising directions for future research. One direction is to incorporate the helper's own cost, such as path length, effort, or energy consumption, in order to balance task efficiency with the helper's resource expenditure. Another direction is to formulate subtask selection as a planning problem within a hierarchical task model, allowing the helper to reason over sequences of cooperative subtasks rather than relying on the greedy strategy adopted in this work. A final direction is to extend the framework to partially observable environments, thereby improving its applicability in real world scenarios.

References

Bercher, P.; Behnke, G.; Kraus, M.; Schiller, M.; Manstetten, D.; Dambier, M.; Dorna, M.; Minker, W.; Glimm, B.; and Biundo, S. 2021. Do it yourself, but not alone: companion-technology for home improvement—bringing a planning-based interactive DIY assistant to life. *KI-Künstliche Intelligenz*, 35(3): 367–375.

Chung, J.; Fayyad, J.; Younes, Y. A.; and Najjaran, H. 2024. Learning team-based navigation: a review of deep reinforcement learning techniques for multi-agent pathfinding. *Artificial Intelligence Review*, 57(2): 41.

El Zaatari, S.; Marei, M.; Li, W.; and Usman, Z. 2019. Cobot programming for collaborative industrial tasks: An overview. *Robotics and Autonomous Systems*, 116: 162–180.

Emmanuel, M. 2025. Safety by Design: Human-Centered Engineering in Amazon Warehouses.

Fern, A.; Natarajan, S.; Judah, K.; and Tadepalli, P. 2014. A decision-theoretic model of assistance. *Journal of Artificial Intelligence Research*, 50: 71–104.

Gall, K. C. 2021. *Active goal recognition design*. University of New Hampshire.

Gao, J.; Li, Y.; Li, X.; Yan, K.; Lin, K.; and Wu, X. 2024. A review of graph-based multi-agent pathfinding solvers: From classical to beyond classical. *Knowledge-Based Systems*, 283: 111121.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2018. Plan and goal recognition as HTN planning. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 466–473. IEEE.

Hu, K.; Li, M.; Song, Z.; Xu, K.; Xia, Q.; Sun, N.; Zhou, P.; and Xia, M. 2024. A review of research on reinforcement learning algorithms for multi-agents. *Neurocomputing*, 599: 128068.

Jin, E.; Hu, J.; Huang, Z.; Zhang, R.; Wu, J.; Fei-Fei, L.; and Martín-Martín, R. 2023. Mini-behavior: A procedurally generated benchmark for long-horizon decision-making in embodied ai. *arXiv preprint arXiv:2310.01824*.

Kaur, N.; and Sharma, A. 2025. Robotics and automation in manufacturing processes. In *Intelligent Manufacturing*, 97–109. CRC Press.

Massardi, J.; Gravel, M.; and Beaudry, É. 2020. Parc: A plan and activity recognition component for assistive robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 3025–3031. IEEE.

Masters, P.; and Sardina, S. 2017. Deceptive Path-Planning. In *IJCAI*, 4368–4375.

Masters, P.; and Vered, M. 2021. What's the context? implicit and explicit assumptions in model-based goal recognition. In *International Joint Conference on Artificial Intelligence 2021*, 4516–4523. Association for the Advancement of Artificial Intelligence (AAAI).

Mirsky, R.; et al. 2017. SLIM: Semi-lazy inference mechanism for plan recognition. *arXiv preprint arXiv:1703.00838*.

Nguyen, H.; Ho, N.; and Rinaldo, A. 2024. On least square estimation in softmax gating mixture of experts. *arXiv preprint arXiv:2402.02952*.

Ning, Z.; and Xie, L. 2024. A survey on multi-agent reinforcement learning and its application. *Journal of Automation and Intelligence*, 3(2): 73–91.

Oh, J.; Meneguzzi, F.; and Sycara, K. 2014. Probabilistic plan recognition for proactive assistant agents. *Plan, activity, and intent recognition*, 10: 23.

Pozanco, A.; Yolanda, E.; Fernández, S.; Borrajo, D.; et al. 2018. Counterplanning using Goal Recognition and Landmarks. In *IJCAI*, 4808–4814.

Ramırez, M.; and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc*, 1778–1783.

Ramírez, M.; and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, 1121–1126.

Ramırez, M.; and Geffner, H. 2011. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *IJCAI*, 2009–2014. IJCAI/AAAI.

Roberts, J.; Baker, M.; and Andrew, J. 2024. Artificial intelligence and qualitative research: The promise and perils of large language model (LLM)'assistance'. *Critical Perspectives on Accounting*, 99: 102722.

Shvo, M.; Hari, R.; O'Reilly, Z.; Abolore, S.; Wang, S.-Y. N.; and McIlraith, S. A. 2022. Proactive robotic assistance via theory of mind. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9148–9155. IEEE.

Shvo, M.; Klassen, T. Q.; Sohrabi, S.; and McIlraith, S. A. 2020. Epistemic plan recognition. In *Proceedings of the 19th international conference on autonomous agents and multiagent systems*, 1251–1259.

Vered, M.; Kaminka, G. A.; and Biham, S. 2016. Online goal recognition through mirroring: Humans and agents. In *Annual Conference on Advances in Cognitive Systems 2016*. Cognitive Systems Foundation.

Wu, S. A.; Wang, R. E.; Evans, J. A.; Tenenbaum, J. B.; Parkes, D. C.; and Kleiman-Weiner, M. 2021. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2): 414–432.

Zhang, C.; Kemp, C.; and Lipovetzky, N. 2023. Goal recognition with timing information. In *Proceedings of the international conference on automated planning and scheduling*, volume 33, 443–451.

Zhang, J.; Krishna, R.; Awadallah, A. H.; and Wang, C. 2023. Ecoassistant: Using llm assistant more affordably and accurately. *arXiv preprint arXiv:2310.03046*.