General Dynamic Goal Recognition using Goal-Conditioned and Meta Reinforcement Learning

Osher Elhadad
Department of Computer Science, Bar Ilan University
Ramat Gan, Israel
oshereli2@gmail.com

Owen Morrissey
Department of Computer Science, Tufts University
Medford, MA, USA
morrisseyowen3@gmail.com

Reuth Mirsky
Department of Computer Science, Tufts University
Medford, MA, USA
reuth.mirsky@tufts.edu

ABSTRACT

Understanding an agent's goal through its behavior is a common AI problem called Goal Recognition (GR). This task becomes particularly challenging in dynamic environments where goals are numerous and ever-changing. We introduce the **General Dynamic Goal Recognition (GDGR)** problem, a broader definition of GR aimed at real-time adaptation of GR systems. This paper presents two novel approaches to tackle GDGR: **(1)** GC-AURA, generalizing to new goals using Goal-Conditioned Reinforcement Learning, and **(2)** Meta-AURA, adapting to novel environments with Meta-Reinforcement Learning. We evaluate these methods across diverse environments, demonstrating their ability to achieve rapid adaptation and high GR accuracy under dynamic and noisy conditions. This work is a significant step forward in enabling GR in dynamic and unpredictable real-world environments.

KEYWORDS

Goal Recognition; Agent Modeling; Meta-Reinforcement Learning

ACM Reference Format:
Osher Elhadad, Owen Morrissey, and Reuth Mirsky. 2026. General Dynamic Goal Recognition using Goal-Conditioned and Meta Reinforcement Learning. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/TTUH1381

1 INTRODUCTION

Goal Recognition (GR) is a subfield of artificial intelligence (AI) that focuses on inferring agents' goals based on observed actions. This task is essential in a variety of fields, particularly for applications in human-robot interaction [19, 26, 35] and multi-agent systems [5, 16, 23, 31], where predicting the goals and actions of other agents enables a system to respond effectively and appropriately.

GR is an online inference problem, yet recent approaches rely on training models to learn patterns associated with a pre-defined set of candidate goals in a single-environment GR problem. While effective in limited settings, such approaches struggle to adapt to new sets of goals within the same or entirely novel environments, requiring them to restart their process from scratch (e.g., reapplying planners or Reinforcement Learning (RL) for each new goal). This restart adds significant overhead time, rendering these approaches

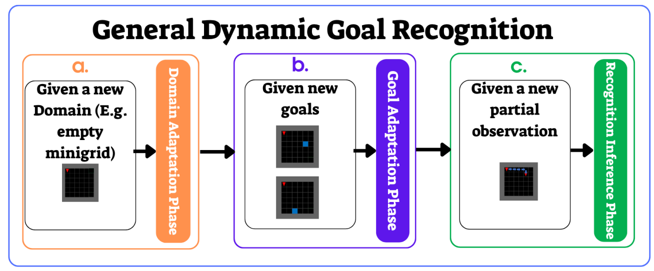


Figure 1: The General Dynamic Goal Recognition Problem

impractical in real-time scenarios where the goal space is continuous or there is a need for rapid adaptation to dynamically changing GR problems with diverse goals and environments. For example, in assistive technologies for the elderly or those with motor disabilities [14, 19, 36], robotic assistants must adapt to changing goals across domains: inferring needed baking ingredients from gestures and context in one task, and identifying a preferred book using similar cues and prior knowledge in another. These scenarios highlight the need for an adaptive GR system, one that can generalize knowledge across tasks and transfer it to new contexts. In real-time applications, such a system cannot afford to relearn from scratch each time.

This paper takes a first step towards addressing these limitations by **(1)** providing a definition for a new problem: **General Dynamic Goal Recognition (GDGR)**, a generalization of GR to account for changing goals and domains **(2)** outlining a solution paradigm for GDGR, named **Adaptive Universal Recognition Algorithm (AURA)**, and **(3)** present two implementations of AURA using Goal-Conditioned RL (**GC-AURA**) and Meta-RL (**Meta-AURA**) to enable real-time GR across multiple dynamically changing tasks within multiple domains.

Figure 1 highlights the three incremental phases that distinguish GDGR from classical GR: (a) *Domain Adaptation*, where the recognizer adapts to a newly introduced domain (e.g., an empty MiniGrid); (b) *Goal Adaptation*, where new candidate goals are introduced within the domain; and (c) *Recognition Inference*, where the system is required to infer the most likely goal based on partial observations. This decomposition emphasizes the dynamic nature of GDGR: unlike traditional GR, which assumes a fixed domain and goal set, GDGR requires continual adaptation across domains, goals, and observation sequences.

We present results in three environments, varying in their properties: Minigrid [6], Point Maze [8], and Panda-Gym [12]. The results

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/TTUH1381

demonstrate that AURA significantly reduces adaptation times for new goals and environmental changes compared to existing methods, showcasing the potential of AURA to advance GR in dynamic real-world scenarios. The full version appears in [9].

2 THEORETICAL BACKGROUND

Markov Decision Process (MDP) is a mathematical framework to model an agent's decision-making under uncertainty. It is defined as a tuple $\mathcal{M} = (S, A, \tau, R, \gamma)$, where: S is the set of states; A is the set of actions; $\tau : S \times A \times S \rightarrow [0, 1]$ is the state transition probability function; $R : S \times A \rightarrow \mathbb{R}$ is the reward function; $\gamma \in [0, 1]$ is the discount factor. Reinforcement Learning (RL) methods aim to find a policy $\pi : S \rightarrow A$ that maximizes the expected sum of discounted rewards [15, 32].

Goal-conditioned Reinforcement Learning (GCRL) [18] adapts RL to environments where agents must achieve specific goals. A GA-MDP extends the standard MDP with an additional tuple $\langle \mathcal{G}, p_g, \phi \rangle$, where: $\mathcal{G}$ is the goal space; p_g is the distribution of goals; $\phi : S \rightarrow \mathcal{G}$ maps states to goals. The reward function depends on the state and the goal, $R : S \times \mathcal{G} \times A \rightarrow \mathbb{R}$, and policy π aims to maximize the expected return for goal-directed actions:

$$J(\pi) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t, g), g \sim p_g, s_{t+1} \sim \tau(\cdot|s_t, a_t)} \left[\sum_t \gamma^t R(s_t, a_t, g) \right] \quad (1)$$

Meta-Reinforcement Learning (Meta-RL) trains agents to adapt quickly to new tasks using experience from related tasks. Given a distribution $p(\mathcal{T})$ of MDPs sharing state/action spaces but differing in dynamics or rewards, Meta-RL learns parameters θ that perform well on new tasks after few gradient steps [11].

Goal Recognition (GR) is the task of inferring the likely goal of an observed agent according to a series of observations [20, 31].

Definition 1. *A GR problem is defined as a tuple $T = (D, G, O)$, where D is the domain theory, G is the set of potential goals, and O is a sequence of observations. The objective of GR is to identify a goal $g \in G$ that best explains the observation sequence O.*

Different approaches to GR primarily differ in how they formulate the domain theory D, including planning domains [24], grammars [13], process graphs [17], or GA-MDPs [1, 25].

While GR focuses on identifying the specific goal an agent is pursuing based on a partial execution of its behavior, Inverse Reinforcement Learning (IRL) [4] seeks to recover the underlying reward function that explains the agent's behavior across multiple trajectories. GR is typically formulated as an online, one-shot inference task, often under partial observability, where the aim is to determine the most likely goal from a limited observation sequence. In contrast, IRL assumes access to multiple trajectories and aims to generalize the agent's preferences.

3 THE GENERAL DYNAMIC GOAL RECOGNITION (GDGR) PROBLEM

We introduce a transfer-learning formulation of the GR problem - General Dynamic Goal Recognition (GDGR) - which handles sequences of GR problems with changing goals and domains. Intuitively, GDGR consists of a series of GR problems, where each problem may involve a new observation sequence and potentially introduce previously unseen goals or changes in the underlying domain. The colors throughout the paper match the colors in Figure 2 to provide visual assistance in decomposing GDGR into single recognition tasks.

Definition 2. ***General Dynamic Goal Recognition (GDGR)*** *is a tuple representing the prior distribution of domain theories p_D and a sequence of GR problems:*

$$\langle p_D, (T_1 = \langle D_1, DG_1, O_1 \rangle, T_2, \ldots, T_n) \rangle,$$

where each input is provided at an increasing time step, starting with p_D at time step $t = 0$, and each GR problem corresponds to a distinct time step $t \in \{1, \ldots, n\}$.

For a given series of time steps $t \in \{1, \ldots, n\}$, each time step consists of three stages of inputs, which can be given incrementally: The first input is the domain theory D_t; The second input is the set of dynamic goals used for recognition DG_t; and the last input is an observation sequence O_t.

This work focuses on GA-MDPs as the domain theory, specifically $D = (S, A, \tau)$, where S is the set of states, A is the set of actions, and τ is the state transition probability. In GA-MDP-based formulations of GR, the observation sequence O is typically defined as $O = (\langle s_1, a_1 \rangle, \langle s_2, a_2 \rangle, \ldots)$, a sequence of state-action pairs, which may be consecutive or non-consecutive, where each pair denotes a state-action transition observed over time. The goal set $DG \subseteq \mathcal{G}$ is assumed to be a subset of the broader goal space $\mathcal{G}$ defined by the GA-MDP.

The solution of a GDGR problem $\langle p_D, (T_1, \ldots, T_n) \rangle$ is a sequence of goals $(g_1^*, \ldots, g_n^*)$ for each of the single GR problems $t \in \{1, \ldots, n\}$, such that $g_t^* = \arg\max_{g \in DG_t} P(g | O_t, M_{t-1})$. g_t^* is the recognized goal within the set of dynamic goals DG_t based on the given observations O_t and the *Memory* M_{t-1} carried over from the previous time step. Memory refers to the structured system or set of mechanisms responsible for storing, maintaining, and retrieving information that helps to transfer knowledge between iterations.

GDGR generalizes Online Dynamic Goal Recognition (ODGR) [28]. While ODGR handles changing goals in a single domain with fixed dynamics, GDGR extends this to distributions of domains, enabling adaptation across varying dynamics, goals, and observation sequences. ODGR is thus a special case of GDGR with a single domain.

4 ADAPTIVE UNIVERSAL RECOGNITION ALGORITHM (AURA)

To solve GDGR problems, we introduce the *Adaptive Universal Recognition Algorithm (AURA)* – a generic algorithm that leverages the fact that domain theories, goal specifications, and online observations may arrive at different times. AURA operates in three main phases. First, it initializes a global *Memory*, which can later reduce the computational cost of adapting to new problems. Then, for each *GR problem*, AURA performs task-specific *domain and goal adaptation* to tailor its knowledge to the new setting. Finally, once an observation sequence is received, AURA infers the most likely goal and updates the memory. Figure 2 and Algorithm 1 outline AURA's core components:

1. Initial Memory Phase (Algorithm 1, line 2 – `InitMemoryPhase`; Figure 2, component (a)) Given the prior distribution P_D, this phase

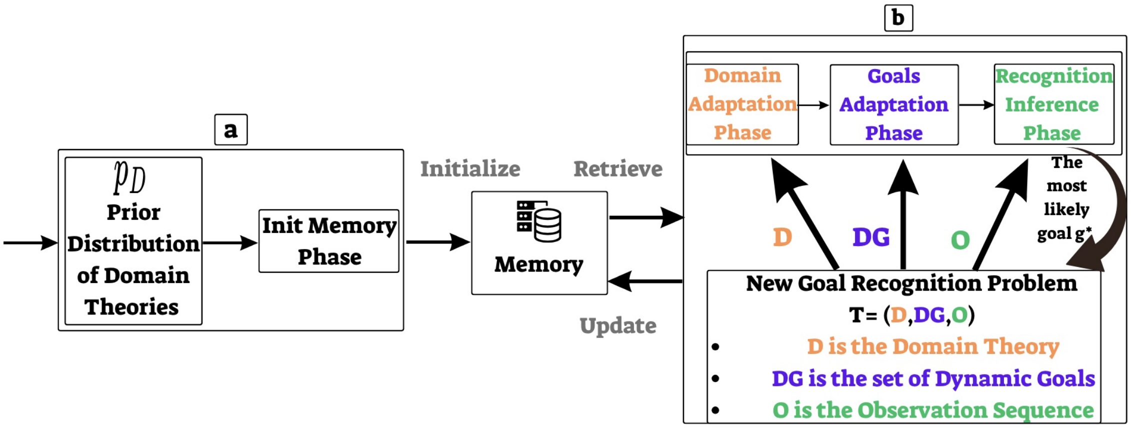


Figure 2: Adaptive Universal Recognition Algorithm (AURA)

Algorithm 1 Adaptive Universal Recognition Algorithm (AURA)

Require: p_D - prior distribution of domain theories

1: Init Memory M
2: $M \leftarrow \text{InitMemoryPhase}(p_D, M)$ ▷ Initialized M after adaptation to p_D
3: **for** all T_i in GetGoalRecognitionProblem() **do**
4: Get Domain theory D_i from T_i
5: $M_{D_i} \leftarrow \text{DomainAdaptationPhase}(D_i, M)$ ▷ Domain Memory M_{D_i} after domain adaptation
6: Get the set of new dynamic goals DG_i from T_i
7: $\{M_g\}_{g \in DG_i} \leftarrow \text{GoalsAdaptationPhase}(DG_i, M_{D_i})$ ▷ Goals Memory $\{M_g\}_{g \in DG_i}$ after goal adaptation
8: Get the Observation sequence O_i from T_i ▷ In the online GR setting, each state and action tuple is provided in a different time step and has its own recognition inference phase
9: $g^* \leftarrow \text{RecognitionInferencePhase}(\{M_g\}_{g \in DG_i}, O_i)$ ▷ $g^* \leftarrow \arg\max_{g \in DG_i} [\text{DISTANCE}(O_i, M_g)]$, where DISTANCE calculates the similarity between the observation sequence and the Goals Memory
10: Save and return g^*
11: $M \leftarrow \text{UpdateMemoryPhase}(M, M_{D_i}, \{M_g\}_{g \in DG_i}, T_i, g^*)$ ▷ Updated M using the previous M, and the current GR problem, adaptations and inference

initializes the memory before any GR problem is received. Its objective is to pre-process reusable domain or goal-related knowledge to reduce adaptation and inference costs in later phases.

2. GR Problem Processing (Algorithm 1, lines 3–10; Figure 2, component (b)) For each GR problem, AURA processes the domain theory, dynamic goals, and observation sequence, which may arrive at different times.

- **Domain Adaptation Phase:** Upon receiving the domain theory D, AURA applies the `DomainAdaptationPhase` to retrieve or refine domain-specific knowledge from memory. This adaptation supports subsequent goal reasoning and recognition.
- **Goal Adaptation Phase:** When the dynamic goals DG become available, AURA invokes the `GoalsAdaptationPhase` to tailor internal representations to the specific goal set. This enables efficient and accurate recognition later.
- **Recognition Inference Phase:** Once the sequence O is available, AURA performs inference using the `RecognitionInferencePhase` to identify the most likely goal.

Note: The components of a GR problem – domain, goals, and observations – are not assumed to arrive simultaneously. AURA is designed to conceptually take advantage of any time gaps between their arrivals. This enables early adaptation steps, such as beginning domain or goal learning before observations are received.

3. Memory Update Phase (Algorithm 1, line 11 – `UpdateMemoryPhase`) After solving each GR problem, AURA updates its memory with the new results and learned patterns. This formulation supports continual learning and improves future performance by transferring accumulated knowledge across problems.

AURA Abstraction Levels Algorithm 1 is a generic algorithm. It enables generalization at various levels of the problem, from basic GR problems with new observations, through changing goals, or new domain dynamics. AURA can be implemented in various ways to address each type of these *abstractions* that can be classified into three levels: **(1)** GR problems with a fixed set of goals within a single domain (O is changing between time steps, but D and DG are fixed). **(2)** GR problems with many possible sets of goals within a single domain (O and DG are changing between time steps, while D is fixed). **(3)** GR problems with many possible goals across multiple domains (changing O, DG and D).

Next, we introduce two RL-based implementations of AURA, each targeting a different level of abstraction within the framework. The first, GC-AURA, builds on GCRL to enable fast adaptation to a new set of goals, addressing AURA's second level of abstraction. The second, Meta-AURA, is based on Meta-RL and targets AURA's third level of abstraction by enabling adaptation to new domains that share the same state and action spaces and can vary in their transition and reward functions.

4.1 GC-AURA

This approach targets GDGR problems in which all GR problems share the same domain theory D. In this setting, the memory M consists of a GCRL policy that can be trained over the entire goal space of D or on a subset of the goal space. This enables AURA to reuse prior domain knowledge for new recognition problems without retraining from scratch.

The primary challenge here is to adapt this policy to newly introduced goals DG with minimal additional training. This requires a mechanism for lifelong goal adaptation, where the system continuously improves its ability to generalize across goals within the same domain.

During the `DomainAdaptationPhase`, a GCRL policy is trained on the domain goal space, or a goal subspace. This shared policy serves as a reusable basis for all subsequent recognition problems in that domain. Once a new set of goals is introduced, the `GoalsAdaptationPhase` uses transfer learning, for example, can use the following strategies:

1. **Zero-shot transfer:** Directly use the existing GCRL policy without any additional training.
2. **Few-shot adaptation:** Fine-tune the policy on a specific goal using a small number of gradient steps or episodes. Useful when the goal is substantially different from those seen in earlier steps.
3. **Goal recall:** Retrieve a previously stored goal-specific policy if the exact goal has been encountered in a past GR problem.

Zero-shot and recall are generally preferred for their efficiency. However, some goals may require minor updates of the policy to ensure accurate recognition. Few-shot adaptation is motivated by the need to handle goals that lie in underrepresented regions of the goal space or whose reward dynamics significantly differ from previously seen goals.

In Section 6.1, we demonstrate GC-AURA using TRPO [27] to train a GCRL policy over a continuous goal space, where zero-shot transfer was found to be effective in the Panda-Gym domain.

4.2 Meta-AURA

In RL-based GR approaches, new domains can differ in all MDP characteristics. For our Meta-AURA implementation using MAML-TRPO, the state and action spaces are assumed to be the same, and the transitions and rewards can change between different domains. This approach focuses on cases where GR problems span across multiple domain theories D. The Memory M is represented by a meta-policy trained to generalize over diverse domains with varying transitions and rewards. This meta-policy improves adaptation times to novel goals and domains by leveraging prior experience.

The primary challenge is to enable the meta-policy to efficiently adapt to a new domain D and its associated dynamic goals DG. This process demands a robust initialization strategy and domain adaptation mechanism to minimize adaptation time while maintaining high recognition accuracy.

During the `InitMemoryPhase` we train a meta-policy for all domains from the prior distributions p_D. The `GoalsAdaptationPhase` evaluates each goal in the dynamic goal set DG and decides between: (1) Few-shot adaptation of the meta-policy to specific goals, (2) Few-shot adaptation of the meta-policy to GCRL policy over all goals in DG, or (3) Retrieval of previously adapted goal-specific or GC policies for recurrent goals from earlier GR problems.

In Sections 6.2 and 6.3, we leveraged the MAML-TRPO algorithm to train a meta-policy capable of adapting to a diverse set of domains and goals. This meta-policy was later fine-tuned using TRPO to specific goals and domains.

5 EXPERIMENTAL SETUP

We start by detailing the general setup configurations used across all experiments.

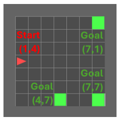
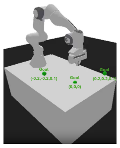
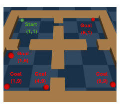
5.1 Domains

AURA supports both discrete and continuous environments, making it suitable for evaluation under a variety of conditions. We evaluate its variants across three benchmark domains from OpenAI Gymnasium [34]: MiniGrid, Point-Maze, and Panda-Gym. Table 2 summarizes the key differences between these domains.

MiniGrid. The MiniGrid domain was chosen due to its flexibility in representing environments in multiple formats, which makes it well-suited for modeling complex GR problems. This capability allows us to compare diverse approaches under consistent conditions, thereby offering insights into algorithmic performance across different representations. Our study focuses MiniGrid Empty 9x9 environment (shown in Table 2). Specifically, we compared AURA to three baseline algorithms: PDDL for reasoning and GR (R&G) using a complete environment model, GRAQL with symbolic state representations (coordinates and angle), and DRACO with a visual image-based representation (same as AURA visual image-based representation). Each representation provides the algorithms with varying degrees of information, enabling a comprehensive comparison of their strengths and limitations when tackling the same GR problem. We used Minigrid Empty 9x9 with different goal locations and lava locations as the training environments for MAML training. The adaptation rewards for MAML in this domain are shown in the Appendix [9].

Point-Maze. The Point-Maze domain presents a continuous environment where the agent navigates through a maze, starting from a fixed position and aiming to reach one of three predefined goal locations. This domain is particularly valuable for testing algorithms in continuous state, goal, and action spaces, as it introduces challenges such as path complexity, exploration efficiency, and obstacle avoidance. The environment's structure and continuous nature make it a suitable testbed for evaluating Meta-AURA. By assessing performance in this domain, we gain insights into the adaptability and effectiveness of different approaches in navigating and achieving

Table 2: Comparison of domains, visualizations, and their specific characteristics. Each domain is chosen to highlight different capabilities of the AURA framework.

Figure	Domain	Details
	**MiniGrid**	*Environments*: Empty-MiniGrid-9x9 environment. *Motivation*: Discrete navigation; comparison with GR baselines. *States*: Image + direction of agent (Discrete). *Actions*: Turn, move forward, stay in place (Discrete). *Reward*: $1 - 0.9 * (step_{count}/max_{steps})$ for success, else 0 (Sparse).
	**Panda-Gym**	*Environments*: PandaReach environment. *Motivation*: Realistic 3D robotic control scenario. *States*: Robotic arm positions, velocities (Continuous). *Actions*: Joint torques for robotic arm (Continuous). *Reward*: Gradual reward based on distance to goal (Dense).
	**PointMaze**	*Environments*: 4-Rooms-11x11 environment. *Motivation*: Continuous navigation; path complexity. *States*: Agent positions, velocities (Continuous). *Actions*: Force applied to agent (Continuous). *Reward*: -1 if not reaching goal; 0 when near goal (< 0.5m) (Sparse).

goals within complex spatial constraints. We used Mazes with different sizes: 6x6 - 9x9, and different start goal states, and obstacles locations as the training environments for MAML training. The adaptation rewards for MAML are shown in in the Appendix [9].

Panda-Gym. The Panda-Gym domain features a robotic arm tasked with manipulating objects to reach predefined goal positions within a continuous 3D space. This environment poses unique challenges by combining high-dimensional state and action spaces. The robotic arm begins at a default position and must plan and execute precise movements to achieve one of three target goals. Panda-Gym provides a realistic simulation for evaluating reinforcement learning algorithms, especially in robotics applications, as it tests spatial reasoning, motion control, and goal-oriented behavior. This domain highlights the robustness of algorithms when applied to real-world-inspired scenarios involving intricate control dynamics and the need for precision.

5.2 Baselines.

(1) *Plan Recognition as Planning (R&G):* This classic symbolic GR baseline, proposed by Ramírez and Geffner [24], leverages planner executions to infer the likelihood of goals. It relies on an underlying PDDL domain representation, meaning without significant and non-trivial modifications, it is limited to discrete domains, making it unsuitable for continuous setups. Consequently, we use it exclusively for evaluation in MiniGrid (Section 6.2).

(2) *GRAQL:* A Q-learning-based approach that learns a tabular policy for each goal and infers the most likely goal based on these [1]. As GRAQL is inherently tabular, it is also restricted to discrete domains. Thus, as with R&G, we compare it to AURA only on the MiniGrid domain (Section 6.2).

(3) *DRACO:* GR method that supports discrete and continuous states, goals, and action spaces using deep RL [22]. In the continuous domains: Panda-Gym and Point-Maze (Sections 6.1 and 6.3 respectively), our Goal-Conditioned and Meta-RL algorithms are based on TRPO, while DRACO's original implementation uses PPO. To better isolate recognition performance and ensure a fair comparison with our approach, we modified DRACO to use TRPO as its underlying RL algorithm. This change avoids confounding effects from differences in RL strategy.

5.3 Policy Learning Algorithms.

We implemented TRPO and MAML-TRPO using the Learn2Learn Library [3].

All information on computational resources and efficiency is provided in Appendix A [9], information on hyperparameters and training details are presented in Appendix B [9], and details about licenses and external packages can be found in Appendix C [9].

5.4 GR Methods.

We evaluate recognition performance, framed as a classification task, using standard metrics: Accuracy, Recall, Precision, and F-score. In addition, we employ two distinct distance metrics between observed trajectories and RL policies that are used by the GR algorithm for recognition inference (`DISTANCE` in Algorithm 1). In discrete state spaces (Section 6.2, Algorithms: Meta-AURA, DRACO, and GRAQL), we implement the recognition inference using Kullback–Leibler (KL) divergence as the recognition metric:

$$D_{\mathrm{KL}}(\pi_g \| \pi_O) = \sum_{i \in |O|} \pi_g(a_i|s_i) \log \frac{\pi_g(a_i|s_i)}{\pi_O(a_i|s_i)} \tag{2}$$

where π_g is a goal-dependent softmax policy and π_O represents a pseudo-policy derived from observations O. For continuous goal spaces (Sections: 6.1 and 6.3, Algorithms: GC-AURA, Meta-AURA, and DRACO), we utilize the Wasserstein distance metric, as introduced in Nageris et al. for recognition purposes [22]:

$$W(O, \pi) = \mathbb{E}_{(s,a) \in O}[\|a - \tilde{a}\|_{L_1}], \quad \tilde{a} \sim \pi(s) \tag{3}$$

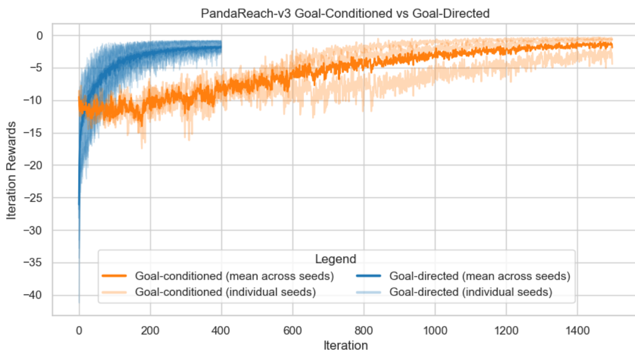


Figure 3: Training curves for the PandaReach-v3 domain. The orange curve represents the Goal-Conditioned TRPO policy (for GC-AURA) trained across all goals in the continuous goal space. The blue curve shows the goal-directed TRPO policy (for DRACO) trained for a specific goal. The shaded regions indicate variations across changing seeds and goals.

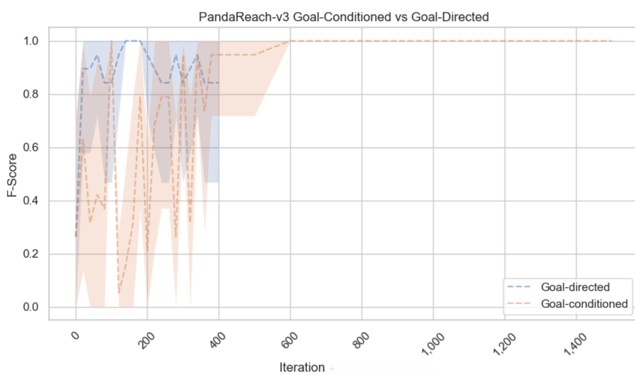


Figure 4: F-Score for PandaReach environment with 10% observability rate, evaluated across 20 different GR problems. The orange curve represents the F-Score of the GC-TRPO policy (for GC-AURA) across different iterations. The blue curve shows the F-Score of the goal-directed TRPO policy (for DRACO) across different iterations. The shaded regions indicate the standard deviations across the 20 GR problems.

where $W(O, \pi)$ is the Wasserstein distance between observation sequence O and policy π; $\mathbb{E}(s, a) \in O$ is the expected value over state-action pairs in observation sequence; $|a - \tilde{a}|_{L_1}$ is the L1 norm (absolute difference) between observed action a and sampled action $\tilde{a}$; $\tilde{a} \sim \pi(s)$ is an action $\tilde{a}$ sampled from policy π given state s.

6 RESULTS

We evaluate AURA across two core generalization constructs: adaptation to new goals, and adaptation to new domains. The latter is further split into adaptation in discrete vs. in continuous domains. In Section 6.1, we test goal generalization in a fixed domain, comparing GC-AURA to DRACO in continuous goal spaces without additional fine-tuning. Section 6.2 examines domain generalization in discrete environments, using Meta-AURA to adapt across changing transitions and goals in MiniGrid. Section 6.3 extends this evaluation to continuous domains, assessing Meta-AURA's ability to generalize across structurally varied PointMaze environments. Together, these experiments test AURA's capacity to learn transferable policies, recognize goals under uncertainty, and adapt across diverse settings.

6.1 Adaptation to New Goals - DG (Fixed Domains)

To evaluate AURA's generalizability to new goals, we compare GC-AURA to DRACO in the PandaReach environment with dense rewards. GC-AURA policies, did not require any additional fine-tuning after the initial training, while DRACO is trained from scratch for each new goal introduced. For Goal-Directed training (DRACO), we selected three settings with specific goals. During the Goal-Conditioned training process, the goal locations were chosen randomly from the continuous state space in each episode. TRPO was used to train all policies. Each instance consisted of 4 possible goals, and was tested with 3 different seeds. For recognition performance, 20 different GR problems were evaluated. In addition, we used observation sequences with 10% observability and various noise levels (0-90%). x% noise implies that for each action of the agent that generated this observation sequence, with a probability of $\frac{x}{100}$, the action was chosen based on the trained policy, and with a probability of $1 - \frac{x}{100}$, the action was chosen randomly.

In terms of **learning**, as shown in Figure 3, DRACO reached an approximate optimal policy within ≈ 400 iterations, while GC-AURA required ≈ 1400 iterations (approximately 3.5 times slower). However, GC-AURA trained a policy applicable across all goals in a continuous goal space, whereas the DRACO was limited to individual goals.

In terms of **recognition quality**, Figure 4 highlights that the GC policy converged to a perfect F-Score during recognition after 600 iterations, demonstrating robust performance even with 10% partial observability and high noise. In contrast, despite training for 400 iterations per new goal, DRACO policies struggled to achieve such performance under the same conditions. Thus, while GC-AURA requires longer initial training, it has two key advantages: (1) it can adapt to new goals in a continuous space without requiring additional fine-tuning, and (2) it exhibits less noise-susceptibility than DRACO's goal-directed policies.

6.2 Adaptation to New Discrete Domains - D (Changing Transitions and Goals)

To evaluate AURA's ability to generalize across new discrete settings, we compare Meta-AURA, GRAQL, DRACO, and the Ramirez & Geffner planner-based method (R&G). We tested each approach in three different goal configurations (Table 2) on the MiniGrid Empty-9×9 domain. Note that each method relies on a distinct input format: R&G operates over a hand-crafted PDDL domain and problem file; GRAQL uses a symbolic MDP representation of the grid (discrete states and actions); DRACO and Meta-AURA consume raw image observations (pixel vectors) as input, without any additional symbolic context.

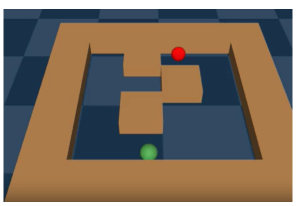
For Meta-AURA, the MAML-TRPO training involved randomly changing the goal location after each episode to vary the rewards, and placing 0–4 lava tiles in different locations within the domain

Table 3: Performance in MiniGrid over 20 GR settings under 10% and 30% partial observability (using KL-divergence as the GR distance metric). The table reports the average F-Score (± standard deviation) of the algorithms across different GR problems.

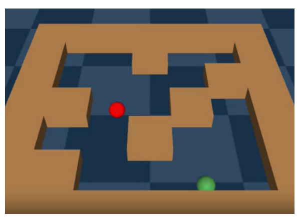
OBS	Problem	Meta-AURA (100 iters)	DRACO (350 iters)	GRAQL (700 iters)	Ramirez & Geffner
10%	2-Goals	0.90 ± 0.31	**1.00 ± 0.00**	0.75 ± 0.32	0.67 ± 0.00
	3-Goals	**0.90 ± 0.31**	**0.90 ± 0.31**	0.64 ± 0.35	0.52 ± 0.00
30%	2-Goals	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.83 ± 0.48
	3-Goals	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.77 ± 0.48

Table 4: Performance in PointMaze over 20 GR settings under 1% and 3% partial observability (OBS) using Wasserstein GR distance metric in domains with 2 or 3 goals (G). The table reports the average performance (± standard deviation) for Meta-AURA and DRACO across different GR problems.

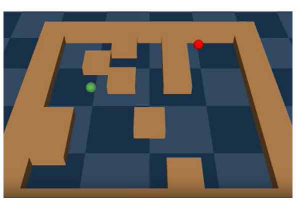
OBS	G	Meta-AURA (500 iterations)				DRACO (1000 iterations)			
		Accuracy	Precision	Recall	F-Score	Accuracy	Precision	Recall	F-Score
1%	2	**0.90 ± 0.32**	**0.90 ± 0.32**	**0.90 ± 0.32**	**0.90 ± 0.32**	0.40 ± 0.52	0.40 ± 0.52	0.40 ± 0.52	0.40 ± 0.52
	3	**0.99 ± 0.21**	**0.90 ± 0.32**	**0.90 ± 0.32**	**0.90 ± 0.32**	0.60 ± 0.34	0.40 ± 0.52	0.40 ± 0.52	0.40 ± 0.52
3%	2	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.90 ± 0.32	0.90 ± 0.32	0.90 ± 0.32	0.90 ± 0.32
	3	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.93 ± 0.21	0.90 ± 0.32	0.90 ± 0.32	0.90 ± 0.32



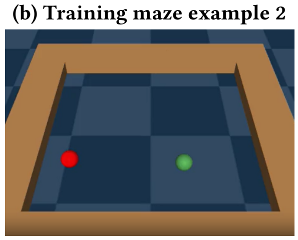
(a) Training maze example 1



(b) Training maze example 2



(c) Training maze example 3



(d) Training maze example 4

Figure 5: Examples from the train set of Point-Maze environments for MAML-RL training

to vary the transition dynamics. The meta-policy was trained using MAML-TRPO with 3 different seeds. The trained meta-policy (iteration 150 of MAML-TRPO)

was then fine-tuned using TRPO for every goal, while GRAQL and DRACO policies were trained from scratch for every goal.

GR performance was evaluated under low observability rates, where only 10% and 30% of the sequence was observed. Ten GR problems were evaluated for two-goal and ten for three-goal scenarios across DRACO policies (trained for ≈ 350 iterations), Meta-AURA policies (trained for ≈ 100 iterations), GRAQL policies (trained for ≈ 700 iterations), and R&G.

Figure 6 shows a comparison of Meta-AURA and DRACO in terms of goal adaptation (training) time for a novel goal in a new domain. Meta-AURA achieved high rewards after ≈ 10 fine-tuning TRPO iterations, while DRACO required ≈ 100 TRPO iterations to reach similar rewards for that goal. Table 3 indicates that Meta-AURA outperformed GRAQL and R&G under low observability (10% and 30%), even with a more challenging visual representation. Regarding runtime, the premise of using AURA is reducing goal adaptation and inference times. Meta-AURA's Init Memory Phase required 150 iterations of MAML-TRPO training, and the goal adaptation time was ≈ 100 fine-tuning TRPO iterations. DRACO's goal adaptation time was approximately ≈ 350 TRPO iterations, meaning that DRACO required ≈ 3.5 times more iterations to achieve similar or higher recognition rates. Domain adaptation times are constant, as they are not used in Meta-AURA and DRACO. R&G did not require training time but had significantly higher inference times (≈ 2 seconds) compared to Meta-AURA and DRACO (both with ≈ 0.13 seconds) and GRAQL (approximately ≈ 0.1 seconds).

6.3 Adaptation to New Continuous Domains (D) (Changing Transitions and Goals)

To evaluate AURA's generalization across new continuous settings, we used the PointMaze Four-Rooms 11x11 environment with sparse rewards (goals at 1x9, 1x6 and 4x9 locations). For DRACO, we followed a similar protocol to the previous experiment. We selected three domains with specific goals and trained a policy for each goal using TRPO with multiple seeds for ≈ 1000 iterations. For Meta-AURA, the MAML-TRPO training used domains with varying goal locations, rewards, sizes (6x6 to 9x9) and obstacles (see figure 5 and more details on MAML's performance in Appendix A [9]). The trained meta-policy (iteration 50) was then fine-tuned using TRPO for the same three goals used in DRACO for ≈ 500 iterations. GR

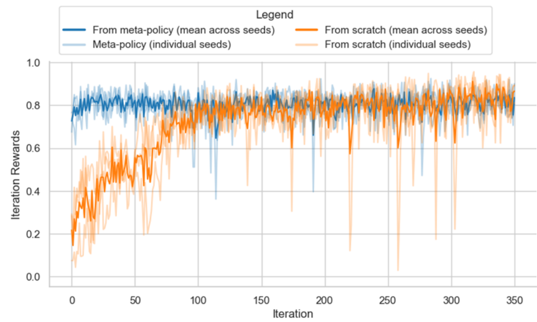


Figure 6: Adaptation to new goals (training) from a Meta-Policy and from scratch in Minigrid Empty 9x9 environment for goal 7x7.

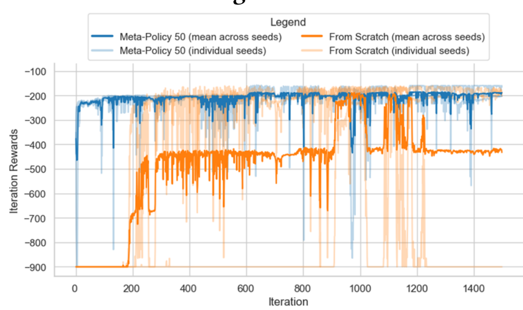


Figure 7: Adaptation to new goals (training) from a Meta-Policy and from scratch in Point-Maze 4 Rooms 11x11 with goal 1x9.

performance was evaluated under 1% and 3% partial observability levels. Ten GR problems were evaluated for two-goal and ten for three-goal scenarios using DRACO (trained for $\approx$ 1000 iterations) and Meta-AURA policies (trained for $\approx$ 500 iterations).

Figure 7 shows that in the goal adaptation phase, Meta-AURA policies achieved high rewards within fewer iterations ($\approx$ 10), while DRACO's policies that were trained from scratch required approximately $\approx$ 200 or more iterations until reaching similar rewards. Table 4 shows that the fine-tuned Meta-AURA policies (after $\approx$ 500 iterations) outperformed DRACO (after $\approx$ 1000 iterations) in terms of GR quality with a shorter goal adaptation phase.

6.4 Training Efficiency

Appendix A [9] compares training efficiency. Despite higher initial costs, AURA significantly reduces adaptation time for new goals compared to retraining-based baselines like DRACO and GRAQL.

7 RELATED WORK

Over the past two decades, GR has been addressed primarily through symbolic approaches, such as planning [20, 24, 29]. These studies introduced the concept of GR by employing planners to infer the most likely goal based on given observations. in stochastic and continuous domains GR often benefits from learning-based approaches that operate without explicit models. Recent advancements have explored the integration of machine learning techniques into GR problems. Notably, the GR as RL framework [1] leverages RL to infer goals in discrete spaces, and DRACO [10, 22] extends this to continuous spaces. Recent geometric approaches [33] have also proposed approximate Euclidean metrics for fast recognition in 2D navigation. Other studies reframed GR as a supervised learning problem [2, 7, 21]. Further approaches utilize process-mining to learn the expected processes that are likely for each goal [17, 30]. These methods are all designed for static, single GR problems, limiting their applicability in real-world settings where agents must handle continuous and changing tasks. Recent work introduced Online Dynamic Goal Recognition for grid-based domains [28]. This definition encompasses multiple GR tasks within the same domain and provides a proof-of-concept for dynamic GR with changing goals in a simple, discrete navigational domain.

8 CONCLUSION

In this paper, we introduced the **General Dynamic Goal Recognition (GDGR)** problem, a generalization of the GR problem for dynamic environments where goals and settings change over time. To address this, we proposed the **Adaptive Universal Recognition Algorithm (AURA)**, an algorithmic framework for GDGR with three abstraction levels, each of which could benefit from mechanisms that support faster adaptation, such as caching to reuse past experiences for previously seen tasks. We further provide two example implementations: Meta RL (Meta-AURA) and GCRL (GC-AURA). These algorithms were evaluated against traditional GR baselines across multiple domains to showcase AURA's ability to adapt quickly to new goals and new domains.

Experimental results show AURA enables faster GDGR, which is essential for applications like autonomous vehicles and assistive robots. GC-AURA demonstrated scalability and adaptability to new goals without extensive retraining, while the analysis of both implementations highlighted trade-offs across abstraction levels. GC-AURA handles noise and continuous goal spaces well within a domain, whereas Meta-AURA facilitates rapid adaptation across domains. These algorithms support tailoring GR frameworks to application needs, balancing training time and inference efficiency.

Limitations and Assumptions. We currently focus on single-agent, simulator-based domains with access to rewards and dynamics for offline training. A natural next step is evaluation in richer multi-agent and real-world domains. Additionally, while GDGR is a continuous setting, our current experiments avoid catastrophic forgetting by freezing the shared meta-parameters during online adaptation; investigating continual learning updates remains a future direction. Meta-AURA assumes tasks share state and action spaces, and GC-AURA relies on effective sampling of the goal space. While current implementations of AURA are RL-based, AURA is defined as a general algorithm that can support symbolic or hybrid implementations.

Future Work. We plan to further integrate Meta-RL and Goal-Conditioned RL to improve GDGR and reach real-time GR when new goals and domains are introduced on-the-fly, and extend AURA to more realistic domains. By bridging theoretical advances with practical needs, we envision AURA as a foundation for next-generation GR systems in complex, dynamic settings.

REFERENCES

[1] Leonardo Amado, Reuth Mirsky, and Felipe Meneguzzi. 2022. Goal recognition as reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9644–9651.

[2] Leonardo Amado, Ramon Fraga Pereira, Joao Aires, Mauricio Magnaguagno, Roger Granada, and Felipe Meneguzzi. 2018. Goal recognition in latent space. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[3] Sébastien MR Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. 2020. learn2learn: A library for meta-learning research. *arXiv preprint arXiv:2008.12284* (2020).

[4] Saurabh Arora and Prashant Doshi. 2021. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence* 297 (2021), 103500.

[5] Gagan Bansal, Besmira Nushi, Ece Kamar, Walter S Lasecki, Daniel S Weld, and Eric Horvitz. 2019. Beyond accuracy: The role of mental models in human-AI team performance. In *Proceedings of the AAAI conference on human computation and crowdsourcing*, Vol. 7. 2–11.

[6] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. 2024. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems* 36 (2024).

[7] Mattia Chiari, Alfonso Emilio Gerevini, Francesco Percassi, Luca Putelli, Ivan Serina, and Matteo Olivato. 2023. Goal recognition as a deep learning task: the GRNet approach. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 33. 560–568.

[8] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. 2024. *Gymnasium Robotics*. http://github.com/Farama-Foundation/Gymnasium-Robotics

[9] Osher Elhadad and Reuth Mirsky. 2025. General dynamic goal recognition. *arXiv preprint arXiv:2505.09737* (2025).

[10] Zihao Fang, Dejun Chen, Yunxiu Zeng, Tao Wang, and Kai Xu. 2023. Real-Time Online Goal Recognition in Continuous Domains via Deep Reinforcement Learning. *Entropy* 25, 10 (2023), 1415.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[12] Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. 2021. panda-gym: Open-source goal-conditioned environments for robotic learning. *arXiv preprint arXiv:2106.13687* (2021).

[13] Christopher Geib and Pavan Kantharaju. 2018. Learning combinatory categorial grammars for plan recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[14] Karen Zita Haigh, John Phelps, and Christopher W Geib. 2002. An open agent architecture for assisting elder independence. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*. 578–586.

[15] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[16] Gal Kaminka, Jan Wendler, and Galit Ronen. 2001. New Challenges in Multi-Agent Intention Recognition. In *Proceedings of the Fall Symposium on Intention Recognition for Collaborative Tasks*.

[17] Jonghyeon Ko, Fabrizio Maria Maggi, Marco Montali, Rafael Peñaloza, and Ramon Fraga Pereira. 2023. Plan Recognition as Probabilistic Trace Alignment. In *2023 5th International Conference on Process Mining (ICPM)*. IEEE, 33–40.

[18] Minghuan Liu, Menghui Zhu, and Weinan Zhang. 2022. Goal-Conditioned Reinforcement Learning: Problems and Solutions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, (IJCAI-22)*. 5502–5511.

[19] Jean Massardi, Mathieu Gravel, and Éric Beaudry. 2020. Parc: A plan and activity recognition component for assistive robots. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3025–3031.

[20] Felipe Rech Meneguzzi and Ramon Fraga Pereira. 2021. A survey on goal recognition as planning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI), Canadá*.

[21] Wookhee Min, Eun Ha, Jonathan Rowe, Bradford Mott, and James Lester. 2014. Deep learning-based goal recognition in open-ended digital games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 10. 37–43.

[22] Ben Nageris, Felipe Meneguzzi, and Reuth Mirsky. 2024. Goal Recognition using Actor-Critic Optimization. *arXiv preprint arXiv:2501.01463* (2024).

[23] Irina Rabkina and Kenneth D Forbus. 2019. Analogical reasoning for intent recognition and action prediction in multi-agent systems. In *Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems*. Cognitive Systems Foundation Cambridge, 504–517.

[24] Miquel Ramírez and Hector Geffner. 2009. Plan recognition as planning. In *Twenty-First international joint conference on artificial intelligence*.

[25] Miquel Ramırez and Hector Geffner. 2011. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *IJCAI*. IJCAI/AAAI, 2009–2014.

[26] Brian Scassellati. 2002. Theory of mind for a humanoid robot. *Autonomous Robots* 12 (2002), 13–24.

[27] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. *arXiv preprint arXiv:1502.05477* (2015).

[28] Matan Shamir, Osher Elhadad, Matthew E Taylor, and Reuth Mirsky. 2024. ODGR: Online Dynamic Goal Recognition. *arXiv preprint arXiv:2407.16220* (2024).

[29] Shirin Sohrabi, Anton V Riabov, and Octavian Udrea. 2016. Plan Recognition as Planning Revisited.. In *IJCAI*. New York, NY, 3258–3264.

[30] Zihang Su, Artem Polyvyanyy, Nir Lipovetzky, Sebastian Sardiña, and Nick van Beest. 2023. Fast and accurate data-driven goal recognition using process mining techniques. *Artificial Intelligence* 323 (2023), 103973.

[31] Gita Sukthankar and Katia Sycara. 2011. Activity recognition for dynamic multi-agent teams. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 1 (2011), 1–24.

[32] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[33] Douglas Tesch, Leonardo Rosa Amado, and Felipe Meneguzzi. 2023. Real-time goal recognition using approximations in Euclidean space. *arXiv preprint arXiv:2307.07876* (2023).

[34] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulao, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. 2024. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032* (2024).

[35] Susanne Trick, Dorothea Koert, Jan Peters, and Constantin A Rothkopf. 2019. Multimodal uncertainty reduction for intention recognition in human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 7009–7016.

[36] Xiang Zhang, Lina Yao, Chaoran Huang, Quan Z Sheng, and Xianzhi Wang. 2017. Intent recognition in smart living through deep recurrent neural networks. In *Neural Information Processing: 24th International Conference, ICONIP, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24*. Springer, 748–758.