ODGR: Online Dynamic Goal Recognition

Matan Shamir*
matan.shamir@live.biu.ac.il
Department of Computer Science
Bar-Ilan University, Ramat Gan, Israel

Osher Elhadad*
osher.elhadad@live.biu.ac.il
Department of Computer Science
Bar-Ilan University, Ramat Gan, Israel

Matthew E. Taylor
matthew.e.taylor@ualberta.ca
Department of Computing Science
Alberta Machine Intelligence Institute (Amii)
University of Alberta, Edmonton, Canada

Reuth Mirsky
mirskyr@cs.biu.ac.il
Department of Computer Science
Bar-Ilan University, Ramat Gan, Israel

Abstract

Traditionally, Reinforcement Learning (RL) problems are aimed at *optimization* of the behavior of an agent. This paper proposes a novel take on RL, which is used to learn the policy of another agent to allow real-time *recognition* of that agent's goals. Goal Recognition (GR) has traditionally been framed as a planning problem where one must recognize an agent's objectives based on its observed actions. Recent approaches have shown how RL can be used as part of the GR pipeline, but are limited to recognizing predefined goals and lack scalability in domains with a large goal space. This paper formulates a novel problem, "Online Dynamic Goal Recognition" (ODGR), as a first step to address these limitations. It introduces the concept of dynamic goals into the standard GR problem definition and demonstrates the feasibility of solving ODGR in a navigation domain using transfer learning.

1 Introduction

Goal Recognition plays a crucial role in a variety of domains in Human-Robot Interaction (Massardi et al., 2020; Jiang et al., 2021; Shvo et al., 2022; Geib, 2002; Inam et al., 2018) and Multi-Agent Systems (Avrahami-Zilberbrand & Kaminka, 2005; Freedman & Zilberstein, 2017; Su et al., 2023). Reasoning about the goal of another agent presents an important challenge that complements an ego agent's policy learning and can inform and leverage RL techniques by facilitating significant breakthroughs in how machines perceive and interpret other agents, especially human intentions.

Existing GR approaches commonly assume that a fixed set of goals is provided as part of the problem formulation (Mirsky et al., 2021; Meneguzzi & Pereira, 2021). In recent GR approaches called *GR as RL* that leverage Reinforcement Learning (RL) (Amado et al., 2022), a policy is learned for each goal during a dedicated learning phase, often occurring offline before the GR task. Then, during the inference phase, the algorithm is given a sequence of observed actions and uses inference methods to perform GR. Unlike planning-based GR (Ramírez & Geffner, 2010; Meneguzzi & Pereira, 2021), these approaches have two fundamental limitations: first, all possible goals must be specified before the inference phase. This requirement may not hold in dynamic environments where goals can emerge or change over time. Second, an issue of scalability arises when dealing with domains that encompass many goals. GR requires significant computational effort and resources for considering each individual goal, posing a challenge for large-scale applications. This problem is further exacerbated in learning-based GR methods, where training an agent for each goal becomes prohibitively costly and infeasible due to the computational burden and time required for training.

*These authors contributed equally.

Considering these two limitations, the first contribution of this paper is conceptual: **it formulates a problem called *Online Dynamic Goal Recognition* (ODGR) with a temporal setting**, where the time steps in which inputs are given are specified. By generalizing the definition of GR, this paper can help evaluate various GR frameworks across different settings and assumptions. The second contribution of this paper is algorithmic: we offer **a general algorithm that leverages Transfer Learning between RL tasks** and demonstrates the feasibility of a framework that would implement this solution by presenting a proof-of-concept for simple navigational environments. The algorithm expands the capacity of traditional learning-based GR methods by adapting to "dynamic goals" – goals not set before the inference phase. For this navigational domain, we propose a method that aggregates existing Q-functions into new Q-functions for the dynamic goals, which can adequately facilitate recognition. In the proposed solution, existing Q-functions are learned before any observations are given, and they are designed to represent the expected policies for a predefined set of base goals. The success of our solution is measured by the ability of the recognizer to use the generated policies when performing recognition and succeed in inferring the correct goal.

2 Preliminaries

The basic model used to describe a domain is a **Markov Decision Process (MDP)** M, which is a 4-tuple $\langle S, A, P, R \rangle$ such that S represents a set of states in the environment, A is the set of actions the agent can execute, P is a transition function, and R is a **reward** function. A **transition** function $P(s'|s, a)$ returns the probability of transitioning from state s to state s' after taking action a, and a reward function $R(s, a, s')$ returns the reward obtained when an agent transitions from s to s' using action a. The **utility function** Q (or Q-function) is a function $Q : S \times A \to \mathcal{R}$ that defines the expected utility of executing an action $a \in A$ in state $s \in S$. A Q-learning agent estimates the values of $Q(s, a)$ from its experiences. A **Policy** $\pi : S \times A \to [0, 1]$ represents the probability that an agent would choose to perform an action $a \in A$ in state $s \in S$.

The Goal Recognition (GR) problem consists of two agents: an *actor* and an *observer*. The GR problem is defined from the observer's perspective, who needs to recognize the actor's goal. Recognizing someone's goal can be challenging, particularly when these goals undergo sudden and unanticipated transformations. This work relies on a line of work in which a GR algorithm produces a set of policies or behaviors, one per goal (Polyvyanyy et al., 2020; Ko et al., 2023; Chiari et al., 2023). We follow the utility-based GR problem formulation from Amado et al. (2022):

Definition 1 *A Goal Recognition problem is a tuple $\langle S, A, O, G, Q_G \rangle$, such that S is a set of states and A is a set of actions an observed actor can take, O is a sequence of observations which are states and actions tuples, $G \subseteq S$ is a set of goals (the actor pursues only one goal at a time), and Q_G is a set of Q-functions $\{Q_g\}_{g \in G}$. The output of a GR problem is a goal $g \in G$ that best explains O.*

Addressing GR through existing RL methodologies, such as those proposed by Amado et al. (2022), involves learning a distinct policy for each goal within a predefined goal set and then comparing these policies against observed behaviors. Importantly, policies should not be perfectly optimal to tackle many GR challenges. Hence, this article will introduce the application of zero-shot transfer learning and heuristic strategies in simple navigational domains as a first step in developing sub-optimal policies that nevertheless provide precise GR outcomes. This paper focuses on goal-directed reward functions (i.e., a reward function-oriented towards a goal g). In addition, we focus on a specific implementation of GR using Q-Learning, but we highlight that the ODGR problem, in general, is not limited to this implementation. To represent the behavior of agents pursuing each potential goal in G, we use a set Π_G. Each policy π_g for an actor pursuing goal g is a softmax based on the Q values over all actions in state s: $\pi_g(a|s) = \frac{Q_g(s,a)}{\sum_{a' \in \mathcal{A}} Q_g(s,a')}$. Then, the policy for an agent pursuing a goal π_g chooses an action based on this probability distribution. Using this formulation, we can say that a goal g explains an observation sequence by evaluating the likelihood that the actor will produce o from π_g. As this paper focuses on learning dynamic goals, we use an existing underlying approach to learn Π_G and follow a similar learning procedure as Amado et al. (2022) to learn Π_G.

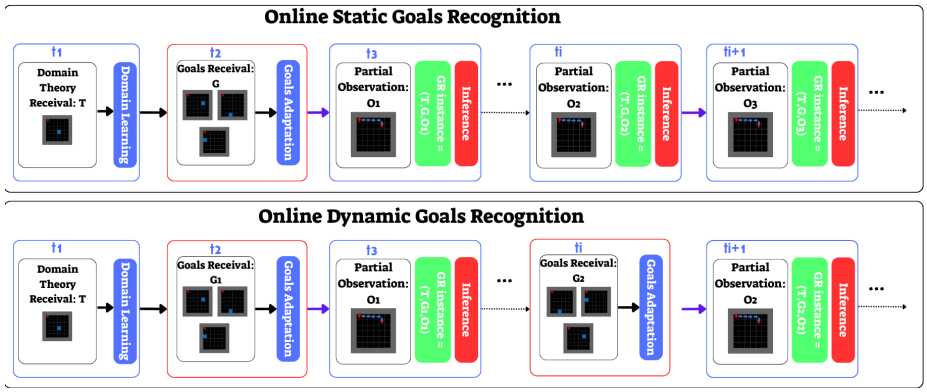

Figure 1: OSGR and ODGR; the symbol t_i denotes the initiation time of each process.

3 Online Dynamic Goal Recognition

Certain methodologies for GR rely on the notion that inputs to the problem arrive at distinct time steps, along with varying assumptions regarding their nature, such as whether they constitute goals or new observations. For example, in Plan Recognition as Planning (Ramírez & Geffner, 2010), it is expected that the description of the environment will be decoupled from a specific problem instance, and different GR problems can be defined for the same environment. We now present a formulation that aims to identify the different components that can change within each problem, potentially facilitating the selection of a more appropriate framework based on the expected scenarios the user may encounter, and the requirements from the expected solution.

Definition 2 *An Online Dynamic Goals Recognition (ODGR) problem is a tuple* $\langle T, \langle G^i, \{O\}^i \rangle_{i \in 1..n} \rangle$, *where* $T = \langle S, A \rangle$ *is a domain theory such that* S *is the set of all possible states and* A *is the set of actions,* G^i *is a set of goals such that* $\forall i \in \{1..n\}, G^i \subseteq S$, *and* $\{O\}^i$ *represents a set of observations sequences (which might contain gaps), where each* $O_j \in \{O\}^i$ *is a single observation sequence* $O_j = \langle o_j^1, o_j^2 \ldots \rangle = \langle \langle s_j^1, a_j^1 \rangle, \langle s_j^2, a_j^2 \rangle, \ldots \rangle$. *Aligning with other online problems, each input is given at an increasing time-step, lexicographically ordered, while it is inherent that because of the input dependency,* T, G^i *and* $O_j \in \{O\}^i$ *arrive at increasing time-steps* $\forall i, j$. *For each two sequences of observations* $o_t^l, o_t^m \in O_t$ $o_j^z, o_j^k \in O_j$ *where* $t < j$, $l < m$, *and* $z < k$, *it holds that the observations are given chronologically, such that* $o_t^l < o_t^m < o_j^z < o_j^k$.

An algorithm for the ODGR problem is expected to return a goal $g \in G^i$ that best explains $O_j \in \{O\}^i$ upon its arrival for all i, j. The final output is a set of goals $G^* = \{\{g_1^1, g_2^1, ...\}, \{g_1^2, g_2^2, ...\}, ..., \{g_1^n, g_2^n, ...\}\}$ where g_j^i is the goal returned by the algorithm upon receiving $O_j \in \{O\}^i$. Using this formulation, one can define a general GR *domain* according to T. In this work, the domain consists of an MDP, but the formulation allows for different representations as well. For each domain, several *instances* can be constructed at different times according to the set of potential goals G^i the actor might be pursuing. For each instance, a single recognition *problem* is an observation trace that needs to be explained by G^i. We can split an ODGR problem into several time intervals, depending on the reception of each input type, T, G, and O.

Domain Learning Time is the duration necessary from receiving the domain theory T until concluding the domain-specific processing and being prepared to receive G.

Goals Adaptation Time is the duration from receiving G until completing the inner-state changes and becoming ready to perform inference based on observation trajectories O.

Inference Time is the time it takes to process an observation and produce a goal in response.

A GR framework can receive a new input before completing the processing of the previous one. Following Allen's interval algebra (Allen, 1983), we say that one input may *overlap* another in-

put. These definitions remain independent of the time steps assigned to each input. However, for simplicity, in this work, we assume that inputs must *precede* one another.

We identify the family of problems where $n = 1$ as *Online Static GR (OSGR)* problems. They are static in the sense that in all instances, the set of goals remains the same. Later in this section, we will show that many GR solutions solve OSGR problems rather than their dynamic counterparts. Figure 1 illustrates OSGR and ODGR problems. The next section reviews recent work, accompanied by an evaluative analysis associated with each methodology in light of the novel definitions.

Model-Based GR (MBGR) In MBGR, the recognition process entails utilizing a pre-defined model that encompasses the characteristics of an environment along with the actions that can be executed within that environment (Mirsky et al., 2021; Meneguzzi & Pereira, 2021; Masters & Vered, 2021). Traditional MBGR often exploits planning and parsing techniques (Avrahami-Zilberbrand & Kaminka, 2005; Geib & Goldman, 2009; Geib, 2009; Ramirez & Geffner, 2009; Mirsky & Gal, 2016; Son et al., 2016). A major limitation of such approaches is their inflexibility in dynamic domains and that a complete model is crucial to solving the GR problem. Such algorithms require complex computation processes to calculate the likelihood of the goal given the observations. These computations are required for every given observation sequence, making these solutions less suitable for real-time performance.

Model-Free GR (MFGR) In MFGR problems (Geffner, 2018; Amado et al., 2024), the recognizer does not have access to the underlying model that describes the properties and dynamics of the environment. While some approaches learn the model dynamics and then employ MBGR methods (Asai & Fukunaga, 2018; Amado et al., 2018), other approaches perform GR directly, without learning the model of the world. Among these approaches, the main techniques include leveraging RL (Amado et al., 2022) or employing deep neural networks to perform GR using a classification network (Min et al., 2014; Borrajo et al., 2020; Chiari et al., 2023). These methods typically have no Domain Learning Time (as the learning process depends on the set of goals), a fairly long Goals Adaptation Time, and a short Inference Time when given an observation sequence.

Different approaches have been proposed for learning an MFGR classification model using languages like PDDL (Geib & Kantharaju, 2018; Maynard et al., 2019; Borrajo et al., 2020; Min et al., 2014; 2016; Fang et al., 2023). These approaches aim for generality by employing a simple domain file that specifies actions using labels and defines a state space based on a finite set of fluents (facts). Chiari et al. (2023) proposed a framework (termed GRnet) that generalizes to any set of goals without additional learning, presenting a model-free solution suitable for environments with changing goals. Compared to the other approaches, GRnet excels in ODGR, demonstrating immediate adaptation to new goal sets and fast inference for new observations through a forward pass of the trained network. While this approach is promising, its reliance on fluent enumeration is critical for its success.

A separate line of work uses MDPs instead of planning languages to model the environment. Amado et al. (2022) presented an MDP-based method termed GR as RL, which trains RL agents during Goals Adaptation Time and computes the likelihood of such agents to produce the sequence of observations given at Inference Time. In this method, information sharing across various GR problems constructed by different observation traces with identical domain theories and goals is inherent. This sharing facilitates short Inference Time. This framework demonstrates high effectiveness in an OSGR setting, However, as these approaches are not designed to handle dynamic environments if employed in an ODGR context, every new set of goals necessitates re-training for each distinct goal.

4 Dynamic Goal Recognition using Transfer Learning

To tackle ODGR problems, we introduce a general algorithm based on transfer learning between source and target RL tasks (Taylor & Stone, 2009). It facilitates learning in new tasks by transferring relevant parts from the policies of the source tasks. This algorithm selects specific goals, G_b, after receiving the domain theory T, and trains agents with policies Π_{G_b} for each of them as part of the Domain Learning time. Upon receiving a set of goals G_d, the algorithm leverages transfer learning

to create Π_{G_d} as part of the Goals Adaptation time. When given an observation trace of an actor, the algorithm returns the most likely goal by using distance metrics as part of the Inference time. For convenience, we refer to the original set of goals, G_b, and the policies of their agents, Π_{G_b}, as *base goals* and *base policies*. Similarly, the new goals, G_d, in the adaptation phase are referred to as *dynamic goals*, and the policies generated for them, Π_{G_d}, are called *dynamic policies*.

Algorithm 1 DomainLearningPhase

Require: $T = \langle S, A \rangle$ - a domain theory
$G_b = \text{SelectBaseGoals}(T)$
for $g \in G_b$ **do**
    $q_g = \text{Learn}(T, g)$
$Q_{G_b} = \{q_g\}_{g \in G_b}$
$T_{G_b} = \langle S, A, Q_{G_b} \rangle$
return T_{G_b}

Algorithm 2 GoalsAdaptationPhase

Require: $T_{G_b} = \langle S, A, Q_{G_b} = \{q_g\}_{g \in G_b} \rangle$
Require: $g_d \in G_d$: a set of new goals
for $g \in G_d$ **do**
    $q_g = \text{Transfer}(T_{G_b}, g)$
$Q_{G_d} = \{q_g\}_{g \in G_d}$
$T_{G_d} = \langle S, A, Q_{G_d} \rangle$
return T_{G_d}

Algorithm 3 InferencePhase

Require: $T_{G_c} = \langle S, A, Q_{G_c} \rangle, Q_{G_c} = \{q_g\}_{g \in G_c}$ - State and action spaces, Q-functions per goal (g_c)
Require: O: a sequence of states and actions ($\langle s_0, a_0 \rangle, \ldots, \langle s_t, a_t \rangle$), where t is the length of O
Sample/receive set of dynamic goals G_d
$m_g^* \leftarrow \infty$ ▷ Init shortest distance
for all g in G_c **do** ▷ Compute distances from O
    q_g^O is the Q-function of goal g from $\{q_g\}_{g \in G_c}$ for O states and actions
    $m_g \leftarrow \text{DISTANCE}(q_g^O, O)$ ▷ Use distance measure
    if $m_g \leq m_g^*$ **then**
        $g^* \leftarrow g$ and $m_g^* \leftarrow m_g$
return g^*

Goal Adaptation using Transfer Learning (GATLing) consists of three parts that are essential to solving the ODGR problem. The *Domain Learning Time* consists of building a domain theory using Algorithm 1. Then, the system receives inputs sequentially. If the input is a set of new goals, it creates a new domain theory using algorithm 2, which uses transfer learning from the base policies to train agents to the new goals, with the objective of minimizing the *Goals Adaption Time* in this phase. The framework keeps the most updated domain theory for future recognition, and if the input is a trace of observations, it performs recognition using the most updated domain theory by calling Algorithm 3 as part of the *Inference Time*, which takes the dynamic goals' Q-functions, compares them to the observation, and chooses the most likely dynamic goal using the DISTANCE metric.

4.1 Implementation in a Navigational Domain

We implement GATLing using a simple navigational domain without obstacles to show feasibility. We use the Gym MiniGrid navigational domain (Chevalier-Boisvert et al., 2023) as a case study, in which we conduct a comparative analysis of the different techniques' respective performances.

We implement Algorithm 1 using Q-Learning, as suggested in the GRAQL framework of Amado et al. (2022). As part of Algorithm 2, for policy transfer, we assign utilities to states and actions by applying weights to the expected utilities of actions from states in the Q-functions of the base goals. The weights assigned to these Q-functions are determined by the similarity of the path from the state to the base goal and the path to the dynamic goal, as perceived at each state individually. In the navigational domain, we used the distance metric as a similarity metric (where a lower distance means greater similarity). We employed two different methods to calculate this distance:

Static weights are assigned between each pair of base and dynamic goals, according to their Euclidean distance. The weight assignment is considered static as the weight for each action remains

unchanged regardless of the states' spatial proximity to the dynamic goal. However, this approach introduces challenges. In certain states, there may be a tendency to move in directions contrary to those leading to the dynamic goal, deviating from the optimal policy, because of the proximity of the dynamic goal to another foundational goal in the opposite direction within these states.

Dynamic assigns variable weights to each state's utility. This weight assignment is based on the cosine similarity between the trajectory from the current state at which the Q-function was being calculated to the dynamic goal and the trajectory from this state to each base goal. The equation that describes the dynamic approach based on cosine similarity is denoted as follows:

$$\text{Cosine\_Similarity}(s, g_b, g_d) = \frac{\text{traj}(s, g_b) \cdot \text{traj}(s, g_d)}{\|\text{traj}(s, g_b)\| \cdot \|\text{traj}(s, g_d)\|}$$

where s is the current state, g_b is a base goal, g_d is a dynamic goal, and $\text{traj}(s, g_b)$ represents a 2D vector originating in s and ending at g_b. The cosine similarity between two trajectories ranges from -1 (perfectly opposite directions) to 1 (perfectly aligned directions), with 0 indicating orthogonality.

To combine the policies of the base goals into a single policy for the dynamic goal, we employed three different aggregation techniques of the Q-values. **Weighted Average (normalize)** is a fair and accurate approach, but slow: $\frac{\sum_{i=1}^{n} w_i \cdot Q_i(s,a)}{\sum_{j=1}^{n} w_j}$. **Softmax** is an accurate approach, but slow, and highly variable: $\frac{\sum_{i=1}^{n} e^{w_i} \cdot Q_i(s,a)}{\sum_{j=1}^{n} e^{w_j}}$. **Max (Hard Weighted Average)** is a greedy approach, fast but less accurate: $\max_{i=1}^{n} Q_i(s, a)$.

4.2 Empirical Evaluation

We present the results of our experiments which evaluate GATLing in different scenarios. Our evaluation focuses on four key evaluation metrics: accuracy, precision, recall, and F-score. As the output of GATLing is a distribution over a set of goals rather than a label, we explain the implementation of each metric in Appendix A.

Evaluating policy transfer can be complex, as it varies greatly given the domain's dynamics and the properties of the source policies. Instead of running batches of runs on an arbitrary subspace of policies, we focus our experiments on two clear and controlled use cases: the first presents a completely smooth environment (without sudden changes in Q-values) and the second breaks this smoothness property. Both experiments employed Q-learning for learning, and during the inference stage, we leveraged KL-divergence to compare each policy with a pseudo-policy based on the observations reported by Amado et al. (2022). The hyperparameters were set according to that work, and were consistent across both experiments: $\alpha = 1 \times 10^{-3}$, $\epsilon = 1 \times 10^{-8}$, $\gamma = 0.99$, and $Episodes = 10^7$. The base goals used for the first experiment were (1,6), (6,6), and (6,1), and for the second one (1,7), (7,7), and (7,1). The results are reported for partial traces of 0.1, 0.3, and 0.5, as traces with a completeness score of 0.7 or higher achieved a 100% success rate across all four metrics in all our experiments, for both GRAQL and GATLing. (fuller traces showed absolute success for both GRAQL and GATLing). We also implemented a scaling factor that prioritizes and emphasizes each state's highest-ranked action (or actions) while reducing the Q-values of other actions in the same state. Appendix B shows an example of the Q-table generated with and without this scaling.

4.2.1 Results

Tables 1 and 2 show the results from Experiment 1 and Experiment 2, respectively. The dynamic approach significantly outperformed the *static* approach, highlighting the importance of adaptability in ODGR. When using the dynamic approach, the softmax and normalized techniques were always able to effectively combine policies from base goals' agents into coherent policies for dynamic goals, while the max aggregation could not rank the true goal as the most likely one. We therefore show the performance results of the *softmax* aggregation method with a *dynamic* distance metric that performed best, leaving out the results using other approaches. Regarding runtime, GATLing policy

		Empty 8x8 Minigrid							
		GATLing				GRAQL			
OBS	Goals	Acc	Prec	Rec	F-score	Acc	Prec	Rec	F-score
0.1	2	1.0±0.0	0.95±0.15	0.95±0.15	0.95±0.15	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	3	0.95±0.15	0.9±0.2	0.9±0.2	0.9±0.2	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	4	0.9±0.2	0.8±0.4	0.8±0.4	0.8±0.4	0.95±0.15	0.9±0.2	0.9±0.2	0.9 ±0.2
0.3	2	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	3	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	4	0.95±0.15	0.95±0.15	0.95±0.15	0.95±0.15	1.0±0.0	0.95±0.15	0.95±0.15	0.95 ±0.15
0.5	2	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	3	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	4	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
Average		0.97 ±0.05	0.95 ±0.1	0.95 ±0.1	0.95 ±0.1	0.99 ±0.01	0.98 ±0.03	0.98 ±0.03	0.98 ±0.03

Table 1: A comparison of dynamic GATLing and GRAQL in Experiment 1. The setup varies by the number of goals and observability level (OBS). We present the accuracy (Acc), Precision (Prec), Recall (Rec), and F-score. Cells show the average result (± standard dev.) over 10 executions.

		Simple Crossing (with 2 walls) 9x9 Minigrid							
		GATLing				GRAQL			
OBS	Goals	Acc	Prec	Rec	F-score	Acc	Prec	Rec	F-score
0.1	2	0.9±0.2	0.9±0.2	0.9±0.2	0.9±0.2	0.8±0.4	0.8±0.4	0.8±0.4	0.8±0.4
0.3	2	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
0.5	2	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
Average		0.96 ±0.06	0.96 ±0.06	0.96 ±0.06	0.96 ±0.06	0.93 ±0.13	0.93 ±0.13	0.93 ±0.13	0.93 ±0.13

Table 2: A comparison of dynamic GATLing and GRAQL in Experiment 2.

construction comes only after a Domain Learning Phase, in which runtime compares with GRAQL's. Similarly, the inference times are swift for both GATLing and GRAQL because both don't require any form of learning or planning. However, constructing a dynamic goal policy using the GATLing framework required only 1.56 seconds on average across 10 runs on a GPU. In contrast, learning a single goal with the GRAQL approach took significantly longer, averaging 485.45 seconds across 10 runs on the same GPU, emphasizing the potential of transfer learning in requiring shorter goal adaptation times. While this evaluation cannot precisely determine the performance of different approaches due to implementation variability, it provides some insight into the potential benefits of considering ODGR problems.

5 Conclusion

This paper presented the novel problem of ODGR, which emphasizes the time in which different parts of the input are received, and the potential gain from policy transer. It formally defines ODGR problems and how they compare to most existing GR problems. Then, the paper provides an overview of existing GR algorithms and their computational effort distribution according to ODGR's components. Lastly, the paper introduces GATLing, a general algorithm for performing ODGR in 2D navigational domains, with an implementation using two distance metrics and three aggregation techniques for the policy ensemble. Our experiments demonstrate that in navigational domains without obstacles, GATLing offers effective ODGR capabilities, with accurate recognition performance and significantly reduced run-time compared to existing RL recognition approaches.

While GATLing provides a foundation for future algorithms, there are several clear areas for potential improvements. First, the proposed aggregation techniques are expected to fail in domains that consist of rapid changes in Q values. Moreover, it is not straightforward how this approach can be extended to continuous state and action spaces. Nevertheless, by avoiding the lengthy policy learning phases. GATLing ensures that the recognition process remains efficient and scalable, even in environments where goals may emerge or evolve. As the field of data-driven GR develops, we anticipate that the need for faster goal learning will increase. This research is expected to contribute to developing adaptable and efficient GR systems in dynamic, changing environments.

References

James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26 (11):832–843, 1983.

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

Leonardo Amado, Reuth Mirsky, and Felipe Meneguzzi. Goal recognition as reinforcement learning. In *Tthe AAAI Conference on Artificial Intelligence*, volume 36, pp. 9644–9651, 2022.

Leonardo Amado, Sveta Paster Shainkopf, Ramon Fraga Pereira, Reuth Mirsky, and Felipe Meneguzzi. A survey on model-free goal recognition. In *IJCAI 2024: 33rd International Joint Conference on Artificial Intelligence*, 2024.

Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *Proceedings of the aaai conference on artificial intelligence*, volume 32, 2018.

Dorit Avrahami-Zilberbrand and Gal A Kaminka. Fast and complete symbolic plan recognition. In *IJCAI*, pp. 653–658, 2005.

Daniel Borrajo, Sriram Gopalakrishnan, and Vamsi K. Potluru. Goal recognition via model-based and model-free techniques, 2020.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *arXiv preprint arXiv:2306.13831*, 2023.

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

Zihao Fang, Dejun Chen, Yunxiu Zeng, Tao Wang, and Kai Xu. Real-time online goal recognition in continuous domains via deep reinforcement learning. *Entropy*, 25(10):1415, 2023.

Richard Freedman and Shlomo Zilberstein. Integration of planning with recognition for responsive interaction using classical planners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Hector Geffner. Model-free, model-based, and general intelligence, 2018.

Christopher Geib. Delaying commitment in plan recognition using combinatory categorial grammars. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

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

Christopher W Geib. Problems with intent recognition for elder care. In *Proceedings of the AAAI-02 Workshop "Automation as Caregiver*, pp. 13–17, 2002.

Christopher W Geib and Robert P Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, 2009.

Rafia Inam, Klaus Raizer, Alberto Hata, Ricardo Souza, Elena Forsman, Enyu Cao, and Shaolei Wang. Risk assessment for human-robot collaboration in an automated warehouse scenario. In *International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pp. 743–751. IEEE, 2018.

Yu-Sian Jiang, Garrett Warnell, and Peter Stone. Goal blending for responsive shared autonomy in a navigating vehicle. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 5939–5947, 2021.

Jonghyeon Ko, Fabrizio Maria Maggi, Marco Montali, Rafael Peñaloza, and Ramon Fraga Pereira. Plan recognition as probabilistic trace alignment. In *5th International Conference on Process Mining*, 2023.

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

Peta Masters and Mor Vered. What's the context? implicit and explicit assumptions in model-based goal recognition. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4516–4523. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/615. URL https://doi.org/10.24963/ijcai.2021/615. Survey Track.

Mariane Maynard, Thibault Duhamel, and Froduald Kabanza. Cost-based goal recognition meets deep learning. *arXiv preprint arXiv:1911.10074*, 2019.

Felipe Rech Meneguzzi and Ramon Fraga Pereira. A survey on goal recognition as planning. In *International Joint Conference on Artificial Intelligence (IJCAI), 2021, Canadá.*, 2021.

Wookhee Min, Eun Ha, Jonathan Rowe, Bradford Mott, and James Lester. Deep learning-based goal recognition in open-ended digital games. In *The AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 10, pp. 37–43, 2014.

Wookhee Min, Bradford W Mott, Jonathan P Rowe, Barry Liu, and James C Lester. Player goal recognition in open-world digital games with long short-term memory networks. In *IJCAI*, pp. 2590–2596, 2016.

Reuth Mirsky and Ya'akov Gal. Slim: semi-lazy inference mechanism for plan recognition. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 394–400, 2016.

Reuth Mirsky, Sarah Keren, and Christopher Geib. *Introduction to symbolic plan and goal recognition*, volume 16. Springer, 2021.

Artem Polyvyanyy, Zihang Su, Nir Lipovetzky, and Sebastian Sardina. Goal recognition using off-the-shelf process mining techniques. In *International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1072–1080, 2020.

M. Ramirez and H. Geffner. Plan recognition as planning. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009.

Miguel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *The AAAI conference on artificial intelligence*, volume 24, pp. 1121–1126, 2010.

Maayan Shvo, Ruthrash Hari, Ziggy O'Reilly, Sophia Abolore, Sze-Yuh Nina Wang, and Sheila A McIlraith. Proactive robotic assistance via theory of mind. In *International Conference on Intelligent Robots and Systems (IROS)*, pp. 9148–9155. IEEE, 2022.

Tran Son, Orkunt Sabuncu, Christian Schulz-Hanke, Torsten Schaub, and William Yeoh. Solving goal recognition design using asp. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

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

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

A Evaluation Metrics Definition

As the output of our GATLing is a distribution over a set of goals rather than a label, we explain how we implemented each measure:

- **Accuracy** measures the proportion of correctly recognized dynamic goals. A goal is considered correctly recognized if it has the highest likelihood of being the actor's goal in comparison to other goals.
- **Precision** indicates how many of the most likely goals were the actor's goal.
- **Recall** indicated how many of the actor's true goals were correctly identified.
- **F-score** is the harmonic mean of Precision and Recall.

B Scaling Factor for Q-function Aggregation

Sometimes, the desired action may not have the highest Q-value after aggregation. This phenomenon can sometimes be attributed to how two base goals' policies bias the dynamic policy in a slightly wrong direction. To mitigate this issue, we implemented a scaling factor that prioritizes and emphasizes each state's highest-ranked action (or actions) while reducing the Q-values of other actions in the same state. Figures 2 (left) and 2 (right) show the crafting of a dynamic goal (4,4) Q-table for a 8×8 Minigrid environment without obstacles. The experimental setup is the same as experiment 1. These scaling methods fine-tune the Q-values for particular actions for every state to ensure consistent and optimal behavior of the agent.

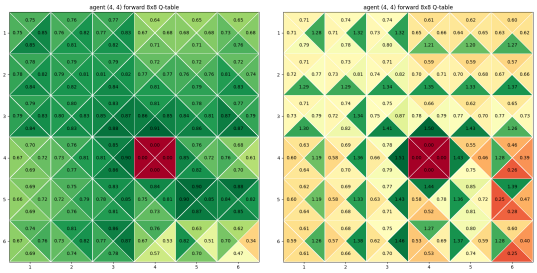

Figure 2: The heuristic Q-table generated from Q-tables whose base goals are at (1,6), (6,1), and (6,6) using cosine similarity and softmax (without scaling) at the 8×8 environment. Each cell shows the Q-values for taking the actions up, right, down, and left. The left (right) grid shows the outcome Q-table without (with) scaling.