Multi-Agent Intention Recognition and Progression

Michael Dann\(^1\), Yuan Yao\(^2\), Natasha Alechina\(^3\), Brian Logan\(^3,4\), Felipe Meneguzzi\(^4\) and John Thangarajah\(^1\)
\(^1\)RMIT University
\(^2\)University of Nottingham, Ningbo China
\(^3\)Utrecht University
\(^4\)University of Aberdeen
michael.dann@rmit.edu.au, yuan.yao@nottingham.edu.cn, \{n.a.alechina, b.s.logan\}@uu.nl, felipe.meneguzzi@abdn.ac.uk, john.thangarajah@rmit.edu.au

Abstract

For an agent in a multi-agent environment, it is often beneficial to be able to predict what other agents will do next when deciding how to act. Previous work in multi-agent intention scheduling assumes a priori knowledge of the current goals of other agents. In this paper, we present a new approach to multi-agent intention scheduling in which an agent uses online goal recognition to identify the goals currently being pursued by other agents while acting in pursuit of its own goals. We show how online goal recognition can be incorporated into an MCTS-based intention scheduler, and evaluate our approach in a range of scenarios. The results demonstrate that our approach can rapidly recognise the goals of other agents even when they are pursuing multiple goals concurrently, and has similar performance to agents which know the goals of other agents a priori.

1 Introduction

The Belief-Desire-Intention (BDI) model [Rao and Georgeff, 1992] is a popular approach to implementing autonomous agents that must act in complex and dynamic environments [de Silva et al., 2020]. In the BDI approach, beliefs represent the agent’s information about the environment and its own state, goals (desires) represent states of the environment the agent should achieve, and intentions represent commitments to achieving particular goals. The program of a BDI agent consists of a set of initial beliefs and a set of plans for achieving goals. Each plan consists of a sequence of primitive actions that change the state of the environment, and subgoals which in turn are achieved by their own plans.

A key advantage of the BDI approach is that agents are capable of pursuing multiple goals concurrently, by interleaving steps (actions or sub-plans) in the intentions for each goal. For example, consider an agent in a Craft World [Andreas et al., 2017] environment with top-level goals to craft a stick and a plank. Both items require wood, but are crafted at different locations: sticks can only be crafted at a workbench while planks are crafted at a toolshed. A BDI agent agent may collect wood for both items before crafting either a stick or a plank. To pursue multiple goals concurrently, at each decision cycle, a BDI agent must solve the intention progression problem (IPP) [Logan et al., 2017], i.e., which of its multiple intentions it should progress next, and, if the next step in the selected intention is a subgoal, which plan should be used to achieve it. The IPP has been extensively studied in the single agent setting, and a number of approaches have been proposed in the literature, including summary-information-based (SI) [Thangarajah et al., 2003; Thangarajah and Padgham, 2011], coverage-based (CB) [Waters et al. 2014; 2015] and Monte-Carlo Tree Search-based (MCTS) [Yao et al., 2014; Yao and Logan, 2016; Yao et al., 2016b].

In recent work [Dann et al., 2020; Dann et al., 2021; Dann et al., 2022], the IPP has been extended to the multi-agent setting. In the multi-agent setting, solutions to the IPP must take into account the implications of action scheduling for both the agent’s own goals and the achievement of the goals of other agents, e.g., when the execution of a step in a plan of one agent makes the execution of a step in a plan of another agent impossible. This is termed intention-aware multi-agent scheduling by Dann et al. [2020]. Work to date in intention-aware scheduling assumes an agent knows the goals currently being pursued by other agents a priori. For example, the MCTS-based ‘intention-aware’ scheduler \(I_A\) developed by Dann et al. [2020] assumes that agents have access both to the current goals of other agents and the plans used to achieve them. In [Dann et al., 2022] agents predict the actions of other agents based on a high-level declarative specification of the tasks performed by an agent rather than its program; however knowledge of current goals of other agents is still assumed.

In many cases, the assumption that the current goals of other agents are known a priori is unrealistic. For example, in a disaster-response scenario the possible goals of other agents may be known (searching for survivors, providing first aid, etc.), but their current intentions may not. In such situations, simply ascribing all possible goals as the current goals of another agent impossible. This is termed intention-aware scheduling by Dann et al. [2020]. Work to date in intention-aware scheduling assumes an agent knows the goals currently being pursued by other agents a priori. For example, the MCTS-based ‘intention-aware’ scheduler \(I_A\) developed by Dann et al. [2020] assumes that agents have access both to the current goals of other agents and the plans used to achieve them. In [Dann et al., 2022] agents predict the actions of other agents based on a high-level declarative specification of the tasks performed by an agent rather than its program; however knowledge of current goals of other agents is still assumed.
other agents at runtime based on their actions in the environment. Goal recognition occurs while the agent acts in pursuit of its own goals, allowing agents to anticipate the future actions of other agents in ‘one-shot’ scenarios, e.g., ad-hoc teamwork.

We formally define the multi-goal recognition and multi-agent intention recognition and progression problems, and extend recent work on goal recognition as reinforcement learning [Amado et al., 2022] to settings where agents may have multiple concurrently active goals. This is essential when interacting with more complex agents, e.g., when the other agents in the environment are BDI agents. A key contribution of our approach is avoiding the exponential explosion inherent to a naive application of the techniques from Amado et al. [2022]. We show how online goal recognition can be incorporated into an MCTS-based intention scheduler, and evaluate our approach in range of scenarios, including cooperative, neutral and adversarial settings. The results demonstrate that our approach can rapidly recognise the goals of other agents even when they are pursuing multiple goals concurrently, and has similar performance to agents which know the goals of other agents a priori.

2 BDI Agents

Before we define the problems we address in this paper, we briefly recall the key components of a BDI agent including beliefs, goals, actions, plans, goal-plan trees and intentions.

Beliefs and goals. We assume agents encode their beliefs and goals using a finite set of propositions $P$. The state space $S \subseteq \wp(P)$ induced by this language consists of all truth assignments to propositions in $P$. Beliefs $B = \{b_1, \ldots, b_n\}$ encode the agent’s information about the environment, and consists of a finite set of ground literals defined over $P$. For simplicity, we assume the environment is fully observable, and the agent’s beliefs are updated when the state of the environment changes. The agent’s desires, or top-level goals $G = \{g_1, \ldots, g_m\}$, consist of a finite set of literals representing the states of the environment desired by the agent. The agent’s goals need not be consistent, since goals $g$ and $\neg g$ can be achieved at different times. For simplicity, and to allow comparison with previous work, we consider only BDI agents with achievement goals in what follows. However, Wu et al. [Wu et al., 2023] show how MCTS-based scheduling can be extended to handle maintenance goals (i.e., goals to maintain a particular condition).

Actions and plans. We define the agent’s action space as a set $Act = \{\alpha_1, \ldots, \alpha_k\}$ of STRIPS-style actions. Each action $\alpha_i \in Act$ is a tuple $\langle \phi, \psi, \psi' \rangle$ consisting of a set of preconditions $\phi = pre(\alpha_i)$, and effects $\psi = eff(\alpha_i)$. These represent, respectively, literals that must be true before the agent can execute $\alpha_i$ (i.e., it must be the case that $B \models \phi$), and the literals that are true after the agent executes $\alpha_i$. For simplicity, we assume that actions are deterministic: if $B \models \phi$, then $\psi$ holds after the agent executes the action. $^1$The set of fluents $F \subseteq P$ are the propositions whose truth value may change as the result of an action. A BDI agent achieves its goals by employing a set of hierarchical plans $H = \{p_1, \ldots, p_n\}$. Each goal $g$ is associated with one or more plans $p_i \in H$ of the form $g \leftarrow s_1; \ldots; s_m$, where $\chi = \text{goal}(p_i)$ is the context condition (i.e., a set of literals which must be true for $p_i$ to be applicable), and $s_1; \ldots; s_m$ is a sequence of steps which are either actions or subgoals. A plan can be executed if its context condition holds, the precondition of each of its action steps holds when the step is reached, and each of its subgoal steps has an executable plan when the subgoal is reached. We assume that successful execution of any plan for $g$ achieves $g$.

Goal-plan trees and intentions. We represent the relationship between the plans, actions, and subgoals that can be used to achieve a goal by a hierarchical structure termed a goal-plan tree (GPT) [Thangarajah et al., 2003; Thangarajah and Padgham, 2011; Yao et al., 2016a]. Each top-level goal is represented by a goal-node that forms the root of a goal-plan tree representing a state of the environment an agent may try to bring about. Its children are plan nodes representing the plans associated with the top-level goal. As only one plan needs to be executed to achieve the goal, goal nodes can be viewed as or-nodes. In contrast, the children of a plan node are the action and subgoal nodes corresponding to the steps in the plan body. As these must be executed sequentially, plan nodes can be viewed as (ordered) and-nodes. Each subgoal node has its associated plans as children, giving rise to a hierarchical tree structure representing all possible ways an agent can achieve the top-level goal. The intentions of an agent at each deliberation cycle are represented by a pair $\langle T, C \rangle$ where $T = \{t_1, \ldots, t_n\}$ is a set of goal-plan trees and $C = \{c_1, \ldots, c_n\}$ are indexes to the current step of each $t_i$.

3 Multi-Agent Intention Recognition and Progression

In this section, we formally define the problems we address in this paper. Previous work on goal recognition focuses on single goals. In single goal recognition the task is to infer the current goal of an agent, given observations of the agent’s behaviour in the environment, the dynamics of the environment, and possibly information about the agent’s preferences over goals [Meneguzzi and Pereira, 2021]. However, agents, e.g., BDI agents, may pursue multiple goals concurrently. We therefore generalise the single-goal recognition problem to the multi-goal recognition problem, which is defined as:

Definition 1 (Multi-Goal Recognition). A multi-goal recognition problem $P_{MG}$ is a tuple $\langle \Xi, s_0, G, \Omega \rangle$, where: $\Xi = \langle F, Act \rangle$ is the planning domain, $F$ is a set of fluents, and $Act$ is a set of actions; $s_0$ is the initial state; $G$ is the set of

$^1$Yao et al. [2016] present an MCTS-based scheduling approach which is able to handle nondeterministic actions. It would be straightforward to integrate their approach into IGR.
possible goals, \(\Omega = [s_0, a_0, s_1, a_1, \ldots, s_n, a_n]\) is a sequence of observations where \(s_i \in S\) and \(a_i \in \text{Act}\).

A solution to a multi-goal recognition problem \(P_{MG}\) is a set of goals \(G' \subseteq G\). A solution is correct if \(G' = G^*\) where \(G^*\) is the set of current goals of the agent that generated the observations.

It is important to note that, for intention-aware multi-agent scheduling, in many cases goal recognition does not have to be perfectly correct in this sense. The agent is not trying to identify the goals of the other agent per se, but to allow it to choose its own actions based on predictions of what the other agent will do next. Ascribing an incorrect set of goals \(G'\) acceptable if the actions chosen by the agent are the same as the actions it would have chosen had \(G^*\) been ascribed to the other agent. In many cases, it is sufficient to correctly predict only the next \(k\) actions of the other agent: for example when agents may interact only briefly; the other agent’s goals change at runtime, e.g., due to changes in the environment or the agent being asked to do something else, reducing the value of predictions more than a few steps ahead. Moreover, goal recognition is an ongoing process, and future observations may be used to discriminate between “similar” goals.

Finally, we extend the definition of the single agent intention progression problem given in [Logan et al., 2017] to include multi-goal recognition. Below we state the problem of multi-agent intention recognition and progression for the general case where there is a set of \(m \geq 1\) ‘other agents’ \(\text{Agt}\) in the environment, each having multiple concurrent goals. However, in the rest of this paper, to simplify the presentation and analysis of the experimental results, \(m = 1\) (there is only one ‘other agent’).

**Definition 2** (Multi-Agent Intention Recognition and Progression). A multi-agent intention recognition and progression problem is a tuple \(P_{RP} = \langle (P_{MG}_i \mid i \in \text{Agt}) \}, B, T, C\rangle\), where \(P_{MG}_i\) follows Definition 1, \(B\) are the agent’s current beliefs, and \(T, C\) are the agent’s intentions.

A solution to a multi-agent intention recognition and progression problem \(P_{RP}\) is a policy \(\Pi(P_{RP})\), that, at each deliberation cycle, selects a current step \(c_t \in C\) to progress so as to maximise some overall utility function \(U^\Pi(P_{RP})\):

\[
\Pi(P_{RP}) = c_t \text{ and } \exists \Pi' \text{ s.t. } U^\Pi(P_{RP}) > U^\Pi'(P_{RP}).
\]

In general, solving a multi-agent intention recognition and progression problem requires solving the multi-goal recognition problems \(P_{MG}_i\), for each of the other agents in the environment, using the inferred goals to predict their likely actions, and then deciding how to act to maximise \(U^\Pi\). For example, if \(U^\Pi\) would be increased if another agent \(i\) achieves a particular goal \(g\), then the agent should choose actions that (at a minimum) do not prevent \(i\) achieving \(g\) while still achieving its own goals. Conversely, if \(i\) achieving \(g\) reduces \(U^\Pi\), then the agent may act to prevent the achievement of \(g\), e.g., by denying \(i\) some resource necessary to achieve \(g\).

\footnote{To allow comparison with previous work, we focus on recognising the achievement goals of the other agent(s). However, our approach requires only the rewards of the other agent and is insensitive to whether the reward results from achieving a goal or maintaining a condition.}

### 4 Recognising Multiple Goals

In this section we present our approach to the multi-goal recognition problem. Our approach extends recent work on Goal Recognition as Reinforcement Learning (GRAQL) [Amado et al., 2022] for the single goal recognition task. GRAQL represents the possible goals \(G\) in a single goal recognition problem by a set of Q-functions \(\{Q_g\}_{g \in G}\).

Given a sequence of observations \(\Omega\) of an agent’s behaviour, the Q-functions are used to infer which reward function (i.e., implicit goal in the MDP formalisation) the agent is likely to be following. Inference is based on a distance measure, \(\text{DISTANCE}(\Omega, Q_g)\), to determine the degree of divergence between the observation sequence, \(\Omega = \langle s_0, a_0, s_1, a_1, \ldots \rangle\), and the behaviour expected from an agent pursuing goal \(g\).

The inferred goal \(g^*\), then, is the one with the smallest distance:

\[
g^* = \arg \min_{g \in G} \text{DISTANCE}(\Omega, Q_g) (1)
\]

In what follows, we use KL divergence as the distance measure, as this was found to give good performance in [Amado et al., 2022]. KL divergence is defined as:

\[
\text{KL}(\Omega, Q_g) = \sum_{i \in [2]} \pi_g(a_i \mid s_i) \log \frac{\pi_g(a_i \mid s_i)}{\pi_\Omega(a_i \mid s_i)} (2)
\]

where \(\pi_g\) is a pseudo-policy where \(\pi_g(a_i \mid s_i) = 1\) for each \(s_i, a_i \in \Omega\), and \(\pi_\Omega\) is a softmax policy derived from the Q-values \(Q_g\).

One possible way of extending GRAQL to multi-goal recognition is to generate a set of Q-functions \(\{Q_M\}_{M \in \mathcal{V}(G)}\) for all possible sets of current goals, and then find the multi-goal Q-function that minimises the distance measure from the observations. However, such a naïve approach requires \(2^{|G|}\) Q-functions to be trained, and rapidly becomes impractical as the set of possible goals grows. Moreover, even small changes in the set of possible goals requires the regeneration of many Q-functions. We therefore adopt a heuristic approach that requires only single-goal Q-functions, in which we take the set of inferred current goals, \(G'\), to be those goals whose KL divergence is within some threshold, \(\delta\), of the goal with the minimum KL divergence:

\[
G' = \{ g \in G \mid \text{KL}(\Omega, Q_g) \leq \min_{g \in G} \text{KL}(\Omega, Q_g) + \delta \} (3)
\]

In effect, all the goals for which the sequence of observations are within \(\delta\) of being optimal are inferred to be the current goals of the agent. A potential weakness of this approach is that, when the current goals of an agent must be pursued sequentially (e.g., because they must be achieved in different parts of the environment), some of the agent’s actual current goals may not be recognised initially. In general, the extent to which later goals can be recognised depends on the number of actions “characteristic” of the goal in the sequence of actions observed so far. However, as noted above, in many cases even inaccurate goal recognition that allows the correct
prediction of the next few actions is sufficient for effective intention progression.

Our aim is therefore to infer the goal(s) an agent may be actively pursuing (i.e., that may give rise to the next few actions), and we rely on MCTS (see Section 5) to determine which actions (and hence which goal(s)) the agent is likely to pursue next, given the inferred goals. Thus, rather than using the definition of KL divergence given in Equation 2 (where KL divergence is summed over the entire observation sequence), we use an exponential moving average of the KL divergence which is more sensitive to recent observations.

Let \(KL (a_t, s_t, Q_g)\) denote the KL divergence for a single state-action observation \((s_t, a_t)\) under goal \(g\):

\[
KL (a_t, s_t, Q_g) = \pi (a_t \mid s_t) \log \frac{\pi (a_t \mid s_t)}{\pi_g (a_t \mid s_t)}
\]

Let \(\eta \in (0, 1)\), and define the sequence \(k_t (\Omega, Q_g)\) as:

\[
k_t (\Omega, Q_g) = \eta k_{t-1} (\Omega, Q_g) + (1 - \eta) KL (a_t, s_t, Q_g)
\]

If \(k_t\) is initially set to zero for all goals, then this gives a zero-biased moving average. To debias it, we need to divide by \((1 - \eta^t)\), as in the Adam optimiser [Kingma and Ba, 2015]:

\[
KL (\Omega, Q_g) = k_t (\Omega, Q_g) / (1 - \eta^t)
\]

We use this as the KL divergence in Equation 3.

## 5 Intention Scheduling with Goal Recognition

In this section, we explain how we incorporate our multi-goal recognition approach into a multi-agent intention scheduler.

The new scheduler, which we call \(\text{GR}\), is based on the state of the art intention-aware multi-agent scheduler \(\text{RM}\) [Dann et al., 2022].

Much of the recent work on multi-agent intention progression [Dann et al., 2020; Dann et al., 2021; Dann et al., 2022] is based on the MCTS algorithm [Browne et al., 2012]. Briefly, MCTS works by iteratively building a search tree. Each node in the tree is evaluated by averaging the outcomes of stochastic rollouts (i.e., possible future executions). Nodes which have been visited less often and which have better average outcomes are favoured for expansion, yielding an asymmetric tree where promising action sequences are analysed in greater depth. In order to predict the behaviour of other agents in rollouts, previous multi-agent intention schedulers based on MCTS require a priori knowledge of the current goals of other agents. For example, \(\text{RM}\) uses the current goals of the other agents to calculate a tactic set for each agent, which is essentially a multi-goal policy for achieving all of the other agent’s goals as quickly as possible.

In contrast, \(\text{GR}\) uses the multi-goal recognition approach described in Section 4 to infer the current goals of other agents. However, using the inferred goals to predict the behaviour of other agents in rollouts is non-trivial. \(\text{RM}\) only has to calculate each agent’s tactic set once, as the agent’s goals are known initially and assumed not to change during execution (except through achievement). Calculating each agent’s tactic set online based on inferred goals is potentially much more computationally demanding, since the inferred goals of other agents may change frequently.

To address this, we use an alternative rollout approach (see Algorithm 1), in which the single-goal policies used by the goal recogniser are also used to predict the actions of other agents. The rollout model assumes that the other agent will pursue the inferred goal with the greatest Q-value, and commits to that goal until its Q-values drop below a certain threshold, \(Q_{min}\) (indicating that the goal has either been achieved or is no longer achievable). The agent then switches to pursuing the goal that currently has the greatest Q-value, and so on. This process repeats until there are no goals with Q-values exceeding \(Q_{min}\), at which point the other agent is assumed to pick actions uniformly at random.

In addition, \(\text{GR}\) generalises how an agent interleaves its intentions. Previous approaches either interleave intentions at the plan level, e.g., [Thangarajah et al., 2003; Yao et al., 2014], or at the action level e.g., [Yao and Logan, 2016; Dann et al., 2022]. Which approach is better depends on the structure of the agent’s plans and the application. For example, for goals that require moving to a particular location in the environment, action-level interleaving is often suboptimal. Conversely, when actions can be interleaved effectively, plan-level interleaving may delay or even prevent the achievement of goals. \(\text{GR}\) can therefore be configured to interleave an agent’s intentions at both the plan and action level. For action-level interleaving, in the rollout phase, \(\text{GR}\) randomly chooses an action from one of its progressible intentions (line 15 in Algorithm 1). For plan-level interleaving, line 15 randomly chooses an available plan and selects actions from that plan until the plan is complete or no longer progressible. Implementing plan-level interleaving for the tree policy phase is more challenging, as the multi-player variant of MCTS used by \(\text{RM}\) assumes a turn-based environment. With plan-level interleaving, each step in the search tree is temporally-extended, so the agents no longer take “turns” but act concurrently in the environment. We therefore use the single-player version of MCTS for plan-level interleaving (as in [Yao et al., 2014]), treating other agents as if they are part of the environment. During the tree policy phase, instead of following a UCT-based policy [Browne et al., 2012], the other agents’ behaviour is simulated in the same manner as in the rollouts.

**Algorithm 1 Rollout phase for \(I_{GR}\) (one other agent).**

1: \textbf{function ROLLOUT}(s)
2: // Determine single goal rollout policy for other agent, \(\pi_0\)
3: \(\pi_{\text{single}} \leftarrow \{ \pi_g \mid g \in G \} \)
4: \(\pi_0 \leftarrow \arg \max_{g \in \pi_{\text{single}}} \max_{a \in \text{Act}} Q^g(s, a)\)
5: \textbf{while} \(s\) is not terminal \textbf{do}
6: \textbf{if} other agent’s turn to act \textbf{then}
7: \(\textbf{if} \max_{a \in \text{Act}} Q^g(s, a) < Q_{min} \textbf{then}\)
8: \(\pi_0 \leftarrow \arg \max_{g \in \text{GR}_{a \neq \text{act}}} \max_{a \in \text{Act}} Q^g(s, a)\)
9: \textbf{else}
10: Select \(a\) uniformly at random from \(\text{Act}\)
11: \(a \sim \pi_0\)
12: \textbf{else}
13: \(// GR’s own turn to act\)
14: Select \(a\) based on \(I_{GR}\)’s rollout policy
15: \(s, \text{step}(a)\)
16: \textbf{return} \(s\)
functions for goal recognition, we use deep function approximation. To obtain generalising Q-functions that do not need to be trained separately for each scenario, we apply the DQN algorithm [Mnih et al., 2015] across randomly generated levels. For the goal recogniser, we set $\delta = 2.5$ and $\eta = 0.95$. The $Q_{\text{min}}$ parameter of the rollouts (see Algorithm 1) is set to 0.5. For MCTS, we use $\alpha = 100, \beta = 10, c = 2.5$.

**Baselines.** We compare $I_{\text{GR}}$ against three baselines:

- **Q-learn**: A DQN agent [Mnih et al., 2015], trained in a single-agent version of the environment.
- **$S_P$**: Yao et al.’s (2016b) scheduler, based on single-player Monte Carlo Tree Search.
- **$I_{\text{RM}}$**: A reimplementation of Dann et al.’s (2022) state-of-the-art multi-agent scheduler that assumes a priori knowledge of the paired agent’s goals. We made some small changes to the implementation to facilitate a fair comparison with the other schedulers: the main difference is that we use deep function approximation to estimate $I_{\text{RM}}$’s heuristic values.

To ensure a fair comparison with $I_{\text{GR}}$, we configured $I_{\text{RM}}$ and $S_P$ to use plan-level interleaving. These schedulers can be thought of roughly as best-case and worst-case baselines for our approach. Since $I_{\text{RM}}$ has a priori knowledge of the paired agent’s goals, we would expect it to exceed $I_{\text{GR}}$’s performance on average. On the other hand, since $S_P$ is completely unaware of the other agent, $I_{\text{GR}}$ ought to be able to outperform it, provided that $I_{\text{GR}}$’s goal recognition is sufficiently accurate to predict some interactions between the agents.

**Paired agents.** Ideally, a multi-agent scheduling approach ought to perform well when paired with a variety of agents. Therefore, we consider two different classes of paired agent:

- **Intention-unaware**: $Q$-learn and $S_P$. These agents simply pursue their own goals, ignoring potential interactions with other agents in the environment.
- **Intention-aware**: $I_{\text{RM}}$ and $I_{\text{GR}}$. These agents are aware of other agents in the environment, and thus, when paired, both agents in the environment (the evaluation agent and the paired agent) are attempting to predict the other agent’s behaviour.

All paired agents are configured to pursue the “paired agent true goals” in Table 2.

### 6.1 Results

The experiment results are summarised in Tables 3, 4 and 5. All results are averaged over 500 randomly generated task instances, with the best results highlighted in bold.

As expected, $I_{\text{GR}}$ outperformed the intention-unaware $S_{\text{P}}$, but performed less well than $I_{\text{RM}}$, which has a priori knowledge of the paired agent’s goals. Across all 40 combinations of scenario and paired agent, $I_{\text{GR}}$ outperformed $S_{\text{P}}$ in all cases. This clearly shows $I_{\text{GR}}$ was able to predict some interactions with the paired agent, despite only being provided with the set of the paired agent’s possible goals. As expected,

---

6 Code is available at https://github.com/mchldann/IJCAI_GR.
Table 2: The 10 scenarios considered in Craft World.

Table 3: Results for the selfish scenarios (own_score).

Table 4: Results for the allied scenarios. The score reported is own_score + other_agent_score.

Table 5: Results for the adversarial scenarios. The score reported is own_score – other_agent_score.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Evaluation agent goals</th>
<th>Paired agent possible goals</th>
<th>Paired agent true goals</th>
<th>Grass</th>
<th>Iron</th>
<th>Wood</th>
<th>Gem</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selfish 1</td>
<td>Gem, gold</td>
<td>Gem, gold</td>
<td>Gem, gold</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Selfish 2</td>
<td>Bridge, gold, rope</td>
<td>Cloth, plank, rope, stick</td>
<td>Plank, stick</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Selfish 3</td>
<td>Bridge, gold, rope</td>
<td>Cloth, plank</td>
<td>Plank, stick</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Selfish 4</td>
<td>Bridge, gold, rope</td>
<td>Cloth, plank</td>
<td>Stick</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Allied 1</td>
<td>Axe, bed</td>
<td>Cloth, bed, gold</td>
<td>Bed, gold</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Allied 2</td>
<td>Cloth, gold, stick</td>
<td>Cloth, gem, gold, stick</td>
<td>Cloth, gold</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Allied 3</td>
<td>Axe, bed</td>
<td>Axe, bridge, cloth, plank</td>
<td>Bridge, cloth, rope</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Evaluation agent goals</th>
<th>Paired agent possible goals</th>
<th>Paired agent true goals</th>
<th>Grass</th>
<th>Iron</th>
<th>Wood</th>
<th>Gem</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selfish 1</td>
<td>Gem, gold</td>
<td>Gem, gold</td>
<td>Gem, gold</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Selfish 2</td>
<td>Bridge, gold, rope</td>
<td>Cloth, plank, rope, stick</td>
<td>Plank, stick</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Selfish 3</td>
<td>Bridge, gold, rope</td>
<td>Cloth, plank</td>
<td>Plank, stick</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Selfish 4</td>
<td>Bridge, gold, rope</td>
<td>Cloth, plank</td>
<td>Stick</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

I_{GR} performed less well than I_{RM} overall, although the difference in performance is fairly small. While there is variation across the individual results, on average, I_{GR} scored 0.79 points more than S_{P}, but only 0.20 points less than I_{RM}. In other words, I_{GR} achieved most of the advantages of full intention-awareness by inferring the goals of the paired agent.

Interestingly, I_{GR} actually outperformed I_{RM} in 8 cases (3 cases in each of Selfish 2, Selfish 3 and Allied 1). Given I_{RM}'s complete knowledge of the other agent’s goals, this may seem surprising. However, recall that, in the MCTS rollouts, I_{RM} assumes that the paired agent will follow a policy based on the conjunction of its goals, whereas I_{GR} assumes that the paired agent will follow the single-goal policy with the largest Q-value. When paired with the Q-learn agent, which actually does follow a policy based on the conjunction of its goals, I_{RM} therefore performs very well: in all 10 scenarios, it achieved the highest score of any agent paired with Q-learn. However, when I_{RM} is paired with agents that do not conform as well to its rollout model, it performed less well. The results suggest that I_{GR}'s rollout model, based on Q-values for individual goals, is better at predicting the paired agent’s behaviour in some settings.

Other results further support this analysis. For example, Q-learn performs broadly the worst, but achieves the best score in Allied 3 when paired with I_{RM}. The most likely explanation for this is not that Q-learn behaved particularly intelligently, but rather that it behaved predictably for I_{RM}, allowing I_{RM} to assist it better. Conversely, Q-learn performed very poorly against I_{RM} in the adversarial scenarios (especially in Adv. 3), probably because I_{RM} could anticipate its behaviour and so obstruct it effectively.

The performance of I_{GR} in scenarios Selfish 2 – Selfish 4 illustrates the impact of incorrect assumptions about the paired agent’s possible goals. In Selfish 2, the assumed possible goals (cloth, plank, rope, stick) include the paired agent’s true goals (plank, stick). In Selfish 3, however, one of the true goals (stick) is not included in the possible goal set. I_{GR} still performed well here, surpassing I_{RM}, but by a much smaller margin than in Selfish 2. In Selfish 4, the assumed possible
goals no longer include any of the true goals. Unsurprisingly, $I_{GR}$ performs less well than $I_{RM}$ in this case, although it still outperforms the intention-unaware $S_P$. This is likely because one of the possible goals (plank) has a similar plan to the true goal (stick). $I_{GR}$ could therefore anticipate that the paired agent was competing for wood, but could not predict all of its movements accurately.

Lastly, note that $I_{GR}$ performed well in scenarios with $\geq 4$ possible goals and $\geq 2$ true goals, indicating that its goal recogniser is capable of handling multiple goals.

**Goal Recognition Example.** To illustrate the operation of $I_{GR}$’s goal recogniser, we provide a partial trajectory in Figure 1, showing how the KL divergences evolve over time. The paired agent (black sprite, initially located near the bottom-left of the world) moves upwards, stopping to collect a piece of iron at step 2, then later collecting a piece of wood at step 7. Out of the set of possible goals (axe, bed, bridge, cloth, rope), iron is only required for axes and bridges, so when the agent collects iron, the goal recogniser becomes confident that it is not pursuing beds, cloth or rope. The divergences for cloth and rope reduce over steps 3 and 4, as the agent moves closer to a piece of grass (which these items require), but increase again when the agent skips over the grass at step 5. While beds also require grass, they require wood too, and it is plausible from the trajectory that the agent has decided to collect wood before grass; hence the bed divergence continues to decline. The small increase in divergence for axe at step 1 is more difficult to explain (as the agent has moved to a piece of iron, which axes require), and may reflect a quirk in the deep RL policy that the goal recogniser uses for axes. This illustrates the usefulness of the threshold, $\delta$, in our goal recognition approach: for $\delta = 2.5$ (as in the experiments), the axe divergence remains just within the threshold, so the goal recogniser still considers it to be an inferred goal at step 1.

**Computational Cost.** At each deliberation cycle, the time that $I_{GR}$ spends on goal recognition is negligible (less than a millisecond) compared to the time spent on MCTS rollouts (around 4.5 seconds on a Ryzen 9 5900X, with $\alpha = 100, \beta = 10$). The most expensive operation, by far, is the neural network forward pass in the computation of the roll-out policy, meaning that the complexity of the algorithm is $O(\alpha \beta)$. Since $I_{GR}$ and the reimplemented $I_{RM}$ both use deep learned rollout policies, their computational costs are near-identical.

7 Related Work

The problems we address in this paper overlap with three key areas of research on agent behaviour: Ad Hoc teamwork, goal recognition, and counterplanning.

In Ad Hoc teamwork, agents try to collaborate efficiently and robustly with unknown agents without any explicit communication protocol [Stone et al., 2013]. Research on in this area has yielded a number of techniques, some of which also include inferring the task or goal currently being pursued by other agents [Mirsky et al., 2022]. Unlike our work, these techniques all assume that agents are either wholly cooperative, or have no conflicting objectives. However, we make no such assumptions, and our experiments show that our approach performs well, even when the agents involved are adversarial.

In goal recognition, the agent’s behaviour consists of a sequence of actions performed by the agent or snapshots of the current environment state or both. The sequence may be incomplete (e.g., observations may not include some actions, or state descriptions may be only partial) and/or noisy (e.g., incorrect action labels or fluents in the state descriptions). Goal recognition approaches often encode the agent preferences in an exhaustive enumeration of the potential goals an agent can be pursuing, or as a plan library/goal-plan tree. The former representation is common in goal recognition as planning [Ramírez and Geffner, 2009; Meneguzzi and Pereira, 2021], whereas the latter is common in plan library-based approaches to goal recognition [Avrahami-Zilberbrand and Kaminka, 2005; Mirsky et al., 2019]. In contrast, in $I_{GR}$ the preferences of the observed agent are encoded as Q-functions rather than explicit goals or GPTs. This can be seen as closer to the GPT approach, but with potentially greater coverage of the action space. While we have not evaluated $I_{GR}$ in scenarios with partial observability or noisy observations, goal recognition as reinforcement learning has been shown to be robust to partial and noisy observation sequences [Amado
et al., 2022], which suggests $I_{GR}$ may be similarly robust. However, evaluating this is future work.

Some approaches to activity and plan recognition based on hierarchical plan libraries have considered the problem of agents with multiple goals. For example, approaches to mixed activity and plan recognition directly from sensor data [Hu and Yang, 2008; Hu et al., 2008] have used skip-chain conditional random fields to successfully deal with agents executing plans in parallel towards different goals. These approaches rely on learning not only the likelihood of observations, but also the way in which goals may interact. In contrast, $I_{GR}$ learns policies associated with each goal through a reward function. It is not clear how we could convert between the two formalisms to allow a direct comparison. Similarly, approaches based on grammar parsing [Geib and Goldman, 2009] developed to recognise multiple concurrent goals require a precondition-free goal-plan tree representation of each goal, augmented with probabilities about agent choices. This is significantly more information than our approach requires for each potential goal of an agent, which makes a direct comparison difficult. However, more recent approaches to learn agent preferences over specific plans [Amado et al., 2023] may allow such a comparison in the future.

Finally, the adversarial setting, in which an agent tries to prevent another agent achieving its goals, overlaps with recent work in counterplanning [Pozanco et al., 2018]. Applying such techniques directly in our scheduler would be a non-trivial extension, and we leave this for future work.

8 Discussion and Conclusion

In this paper, we introduced the multi-agent intention recognition and progression problem, that is, the problem of identifying the goals currently being pursued by other agents at runtime to allow the more effective scheduling of an agent’s intentions. Our key contributions are threefold. First, we formally define the multi-agent intention recognition and progression problem, connecting the intention scheduling problem with that of goal recognition. As part of our formalisation, we expand the definition of goal recognition problems to situations in which an agent may pursue multiple goals rather than a single goal. Second, we extend reinforcement learning-based goal recognition techniques to the multi-goal recognition problem. Third, we present $I_{GR}$, an approach to intention scheduling that uses the output of a goal recogniser to predict the actions that may be taken by other agents, allowing an $I_{GR}$ agent to choose its own actions so as to maximise its utility. We show experimentally that $I_{GR}$ agents perform as well (and sometimes better) as agents which know the goals of other agents a priori.

Acknowledgements

For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

References


