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 an agent typically results in poor predictions of its behaviour. In this paper, we develop a new approach to intention-aware multi-agent scheduling in which an agent uses *online goal recognition* to identify the goals currently being pursued by 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 $\langle \phi, \psi \rangle$ 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 α_i (i.e., it must be the case that $B \models \phi$), and the literals that are true after the agent executes α_i. For simplicity, we assume that actions are deterministic: if $B \models \phi$, then ψ 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 $g : \chi \leftarrow s_1; \ldots; s_m$, where $\chi = con(p_i)$ 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 $T = \{t_1, ..., t_n\}$ is a set of goal-plan trees and $C = \{c_1, ..., c_n\}$ 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,[2] the dynamics of the environment, and possibly information about the agent's preferences over goals [Meneguzzi and Pereira, 2021].[3] 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 $\mathcal{P}_{MG}$ 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 I_{GR}.

[2] We focus on keyhole goal recognition, that is, where the agent being observed is not aware of the observing agent.

[3] Some approaches compute a probability distribution over an agent's possible goals, e.g., [Ramírez and Geffner, 2010; Sohrabi *et al.*, 2016; Masters and Vered, 2021], and assume the goal with the highest probability is the agent's current goal.

possible goals,[4] $\Omega = [s_0, a_0, s_1, a_1, \ldots, s_n, a_n]$ *is a sequence of observations where* $s_i \in S$ *and* $a_i \in Act$.

A solution to a multi-goal recognition problem $\mathcal{P}_{MG}$ *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' is 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' 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* $\mathcal{P}_{RP} = \langle \{\mathcal{P}_{MG_i} \mid i \in Agt\}, B, T, C \rangle$, *where* $\mathcal{P}_{MG_i}$ *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 $\mathcal{P}_{RP}$ *is a policy* $\Pi(\mathcal{P}_{RP})$, *that, at each deliberation cycle, selects a current step* $c_i \in C$ *to progress so as to maximise some overall utility function* $U^{\Pi}(\mathcal{P}_{RP})$*:*

$$\Pi(\mathcal{P}_{RP}) = c_i \textit{ and } \nexists\, \Pi' \textit{ s.t. } U^{\Pi'}(\mathcal{P}_{RP}) > U^{\Pi}(\mathcal{P}_{RP}).$$

In general, solving a multi-agent intention recognition and progression problem requires solving the multi-goal recognition problems $\mathcal{P}_{MG_i}$ 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^{Π}. For example, if U^{Π} 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^{Π}, then the agent may act to prevent the achievement of g, e.g., by denying i some resource necessary to achieve g.

[4] 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 $\{Q_g\}_{g \in G}$.[5] Given a sequence of observations Ω 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, $\textsc{Distance}(\Omega, Q_g)$, 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:

$$g^* = \operatorname*{arg\,min}_{g \in G} \textsc{Distance}(\Omega, Q_g) \tag{1}$$

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:

$$KL(\Omega, Q_g) = \sum_{i \in |\Omega|} \pi_\Omega(a_i \mid s_i) \log \frac{\pi_\Omega(a_i \mid s_i)}{\pi_g(a_i \mid s_i)} \tag{2}$$

where π_Ω is a pseudo-policy where $\pi_\Omega(a_i \mid s_i) = 1$ for each $\langle s_i, a_i \rangle \in \Omega$, and π_g 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 $\{Q_M\}_{M \in \wp(G)}$ 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 naive 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, δ, of the goal with the minimum KL divergence:

$$G' = \{g \in G \mid KL(\Omega, Q_g) \leq \min_{\hat{g} \in G} KL(\Omega, Q_{\hat{g}}) + \delta\} \tag{3}$$

In effect, all the goals for which the sequence of observations are within δ 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

[5] The Q-functions can, for example, be learned from the goal and the environment dynamics, or from previous traces of agent behaviour where the goal is known.

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 $\langle s_t, a_t \rangle$ under goal g:

$$KL(a_t, s_t, Q_g) = \pi_\Omega(a_t \mid s_t) \log \frac{\pi_\Omega(a_t \mid s_t)}{\pi_g(a_t \mid s_t)} \tag{4}$$

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

$$k_t(\Omega, Q_g) = \eta k_{t-1}(\Omega, Q_g) + (1 - \eta) KL(a_t, s_t, Q_g) \tag{5}$$

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]:

$$KL(\Omega, Q_g) = k_t(\Omega, Q_g)/(1 - \eta^t) \tag{6}$$

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 I_{GR}, is based on the state of the art intention-aware multi-agent scheduler I_{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, I_{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, I_{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. I_{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.

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

1: **function** ROLLOUT(s)
2: // Determine single goal rollout policy for other agent, π_o
3: $\pi_{single} \leftarrow \{\pi_g \mid g \in G'\}$
4: $\pi_o \leftarrow \arg\max_{\pi \in \pi_{single}} \max_{a \in Act} Q^\pi(s, a)$
5: **while** s is not terminal **do**
6: **if** other agent's turn to act **then**
7: **if** $\max_{a \in A} Q^{\pi_o}(s, a) < Q_{min}$ **then**
8: $\pi_o \leftarrow \arg\max_{\pi \in \pi_{single}} \max_{a \in Act} Q^\pi(s, a)$
9: **if** $\max_{a \in A} Q^{\pi_o}(s, a) < Q_{min}$ **then**
10: Select a uniformly at random from Act
11: **else**
12: $a \sim \pi_o$
13: **else**
14: // I_{GR}'s own turn to act
15: Select a based on I_{GR}'s rollout policy
16: $s.step(a)$
return s

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, I_{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. I_{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, I_{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 I_{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.

Goal Item	Ingredients	Tools Needed	Craft Location
Axe	Iron, stick	—	Toolshed
Bed	Grass, plank	—	Workbench
Bridge	Iron, wood	—	Factory
Cloth	Grass		Factory
Gem	—	Axe	—
Gold	—	Bridge	—
Plank	Wood	—	Toolshed
Rope	Grass	—	Toolshed
Stick	Wood	—	Workbench

Table 1: Item recipes in *Craft World*.

6 Evaluation

We evaluate our approach in the two-agent version of the well-known *Craft World* [Andreas *et al.*, 2017] environment developed by Dann *et al.* (2022) to evaluate I_{RM}.

In Craft World, agents must craft or gather certain *goal items*. The rules for acquiring items are summarised in Table 1. For example, to mine a gem, an agent must first acquire an axe, which can be crafted from iron and a stick at a toolshed. Raw ingredients (grass, iron and wood) can be collected directly from squares containing those resources. The starting locations of all objects in the environment, including the agents, are randomised at the start of each episode.

Agents have six possible actions: *movement* in the four cardinal directions, plus *collect* and *craft*. Actions that are currently inapplicable have no effect on the environment, e.g., performing *collect* at an empty square. We assume that actions are fully observable, i.e., agents can see the actions of all other agents. Goal-plan trees were generated algorithmically for each goal item.

We consider 10 scenarios, listed in Table 2. As in previous work [Dann *et al.*, 2020; Dann *et al.*, 2021], these are of three types: *selfish*, *allied* and *adversarial*. In selfish scenarios, agents seek to maximise achievement of their own goals. In allied scenarios, they seek to maximise the achievement of both agents' goals. In adversarial scenarios, they maximise *own_goals – other_agent_goals*. Goal items can be crafted multiple times, and each successful craft is worth 1 point.

The specific set of goal items for each agent depends on the scenario. In Table 2, the column "evaluation agent goals" shows the set of goal items for the agent under evaluation, "paired agent true goals" are the actual goals of the *paired agent*, i.e., the other agent in the environment, and "paired agent possible goals" are the set of possible goals given to I_{GR}. Note that in two of the scenarios (Selfish 3 and Selfish 4) there is a true goal that is not included in the possible set (indicated by an asterisk). This was done to evaluate the performance of I_{GR} when the assumed set of possible goals does not include all of the paired agent's actual goals.

We designed the goal sets and resource counts to yield interactions between the agents' plans. For example, in Selfish 1, there is insufficient iron and wood for the evaluation agent to craft both an axe and a bridge (assuming that the paired agent will craft an axe to mine a gem). Thus, it cannot mine both gems and gold and must decide which to pursue.

I_{GR} configuration. Since Craft World requires taking many movement actions, we configured I_{GR} to use plan-level interleaving. Unlike Amado *et al.* (2022), who use tabular Q-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_{min} parameter of the rollouts (see Algorithm 1) is set to 0.5. For MCTS, we use $\alpha = 100, \beta = 10, c = 2.5$.[6]

Baselines. We compare I_{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_{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_{RM}'s heuristic values.

To ensure a fair comparison with I_{GR}, we configured I_{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_{RM} has *a priori* knowledge of the paired agent's goals, we would expect it to exceed I_{GR}'s performance on average. On the other hand, since S_P is completely unaware of the other agent, I_{GR} ought to be able to outperform it, provided that I_{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*: $Q\text{-}learn$ and S_P. These agents simply pursue their own goals, ignoring potential interactions with other agents in the environment.
- *Intention-aware*: I_{RM} and I_{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_{GR} outperformed the intention-unaware S_P, but performed less well than I_{RM}, which has *a priori* knowledge of the paired agent's goals. Across all 40 combinations of scenario and paired agent, I_{GR} outperformed S_P in all cases. This clearly shows I_{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.

Scenario	Evaluation agent goals	Paired agent possible goals	Paired agent true goals	Grass	Iron	Wood	Gem	Gold
Selfish 1	Gem, gold	Gem, gold	Gem	2	2	2	5	4
Selfish 2	Bridge, gold, rope	Cloth, plank, rope, stick	Plank, stick	1	2	2	0	3
Selfish 3	Bridge, gold, rope	Cloth, plank	Plank, stick*	1	2	2	0	3
Selfish 4	Bridge, gold, rope	Cloth, plank	Stick*	1	2	2	0	3
Allied 1	Axe, bed	Cloth, bed, gold	Bed, gold	1	1	1	3	3
Allied 2	Cloth, gold, stick	Cloth, gem, gold, stick	Cloth, gold	1	2	2	2	2
Allied 3	Axe, bed	Axe, bridge, cloth, plank rope, stick	Bridge, cloth, rope	2	2	2	3	3
Adv. 1	Axe, bed, gold	Axe, bridge, rope	Rope	4	4	4	1	1
Adv. 2	Axe, bridge, cloth, rope	Cloth, gem, gold, rope	Gem, gold	2	4	1	2	2
Adv. 3	Cloth, plank, rope, stick	Axe, bed, bridge, cloth, rope	Axe, bed, bridge	3	6	3	2	2

Table 2: The 10 scenarios considered in Craft World.

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 $Q\text{-}learn$ 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 $Q\text{-}learn$. 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, $Q\text{-}learn$ 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 $Q\text{-}learn$ behaved particularly intelligently, but rather that it behaved *predictably* for I_{RM}, allowing I_{RM} to assist it better. Conversely, $Q\text{-}learn$ 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

		Paired agent			
		$Q\text{-}learn$	S_P	I_{RM}	I_{GR}
Selfish 1	$Q\text{-}learn$	2.59	2.25	2.18	2.23
	S_P	3.52	2.33	2.36	2.42
	I_{RM}	**4.41**	**3.96**	**3.96**	**3.97**
	I_{GR}	4.17	3.44	3.43	3.52
Selfish 2	$Q\text{-}learn$	2.47	1.88	1.81	1.86
	S_P	3.97	2.08	2.10	2.15
	I_{RM}	**4.98**	3.25	3.24	3.30
	I_{GR}	4.68	**4.08**	**4.06**	**4.17**
Selfish 3	$Q\text{-}learn$	2.44	1.83	1.80	1.85
	S_P	4.07	2.08	2.16	2.21
	I_{RM}	**4.98**	3.10	3.22	3.21
	I_{GR}	4.44	**3.29**	**3.28**	**3.36**
Selfish 4	$Q\text{-}learn$	2.39	2.10	2.09	2.10
	S_P	3.49	2.32	2.32	2.32
	I_{RM}	**5.06**	**4.37**	**4.33**	**4.39**
	I_{GR}	3.94	3.04	3.11	2.96

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

		Paired agent			
		$Q\text{-}learn$	S_P	I_{RM}	I_{GR}
Allied 1	$Q\text{-}learn$	0.98	1.34	2.15	1.98
	S_P	1.26	1.08	1.58	2.00
	I_{RM}	**1.95**	2.33	2.45	2.65
	I_{GR}	1.81	**2.38**	**2.54**	**2.66**
Allied 2	$Q\text{-}learn$	2.64	3.27	3.65	3.30
	S_P	3.26	3.29	3.40	3.43
	I_{RM}	**3.89**	**3.92**	**3.92**	**3.90**
	I_{GR}	3.42	3.71	3.79	3.79
Allied 3	$Q\text{-}learn$	3.38	3.64	**3.94**	3.58
	S_P	3.20	3.17	3.18	3.14
	I_{RM}	**3.91**	**3.91**	3.87	**3.83**
	I_{GR}	3.61	3.63	3.50	3.43

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

		Paired agent			
		$Q\text{-}learn$	S_P	I_{RM}	I_{GR}
Adv. 1	$Q\text{-}learn$	-0.99	-1.52	-1.49	-1.49
	S_P	0.23	-0.22	-0.22	-0.22
	I_{RM}	**1.14**	**0.53**	**0.57**	**0.56**
	I_{GR}	1.04	0.51	0.51	0.50
Adv. 2	$Q\text{-}learn$	0.97	0.58	0.32	0.46
	S_P	1.33	0.91	0.77	0.59
	I_{RM}	**2.09**	**1.83**	**1.62**	**1.29**
	I_{GR}	2.00	1.81	1.59	1.23
Adv. 3	$Q\text{-}learn$	2.30	1.57	-0.09	1.25
	S_P	2.69	2.05	1.70	1.92
	I_{RM}	**3.97**	**2.92**	**1.93**	**2.44**
	I_{GR}	3.77	2.88	1.82	2.26

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

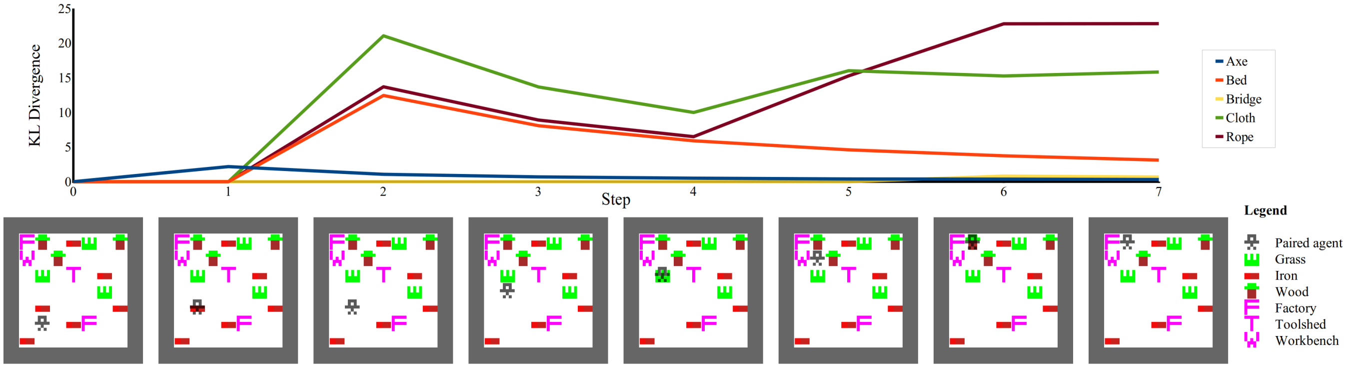

Figure 1: A partial trajectory from the *Adv. 3* scenario, showing how the KL divergences from I_{GR}'s goal recogniser evolved over time. Some details of the Craft World states (such as the position of the evaluation agent) have been omitted to aid readability.

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 ≥ 4 possible goals and ≥ 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, δ, 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 rollout policy, meaning that the complexity of the algorithm is $\mathcal{O}(\alpha\beta)$. 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 achieve 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

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

[Amado *et al.*, 2023] Leonardo R. Amado, Ramon F. Pereira, and Felipe Meneguzzi. Robust Neuro-Symbolic Goal and Plan Recognition. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*. AAAI Press, 2023.

[Andreas *et al.*, 2017] Jacob Andreas, Dan Klein, and Sergey Levine. Modular Multitask Reinforcement Learning with Policy Sketches. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 166–175. PMLR, 2017.

[Avrahami-Zilberbrand and Kaminka, 2005] Dorit Avrahami-Zilberbrand and Gal A. Kaminka. Fast and Complete Symbolic Plan Recognition. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 653–658. Professional Book Center, 2005.

[Browne *et al.*, 2012] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence andAI in games*, 4(1):1–43, 2012.

[Dann *et al.*, 2020] Michael Dann, John Thangarajah, Yuan Yao, and Brian Logan. Intention-Aware Multiagent Scheduling. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, pages 285–293, 2020.

[Dann *et al.*, 2021] Michael Dann, Yuan Yao, Brian Logan, and John Thangarajah. Multi-Agent Intention Progression with Black Box Agents. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*. IJCAI, 2021.

[Dann *et al.*, 2022] Michael Dann, Yuan Yao, Natasha Alechina, Brian Logan, and John Thangarajah. Multi-Agent Intention Progression with Reward Machines. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pages 215–222. IJCAI, 2022.

[de Silva *et al.*, 2020] Lavindra de Silva, Felipe Meneguzzi, and Brian Logan. BDI Agent Architectures: A Survey. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. IJCAI, 2020.

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

[Hu and Yang, 2008] Derek H. Hu and Qiang Yang. CIGAR: Concurrent and Interleaving Goal and Activity Recognition. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1363–1368. AAAI Press, 2008.

[Hu *et al.*, 2008] Derek H. Hu, Sinno J. Pan, Vincent W. Zheng, Nathan N. Liu, and Qiang Yang. Real world activity recognition with multiple goals. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 30–39. ACM, 2008.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[Logan *et al.*, 2017] Brian Logan, John Thangarajah, and Neil Yorke-Smith. Progressing Intention Progresson: A Call for a Goal-Plan Tree Contest. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, pages 768–772. IFAAMAS, 2017.

[Masters and Vered, 2021] Peta Masters and Mor Vered. What's the Context? Implicit and Explicit Assumptions in Model-Based Goal Recognition. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 4516–4523. IJCAI, 2021.

[Meneguzzi and Pereira, 2021] Felipe Meneguzzi and Ramon F. Pereira. A Survey on Goal Recognition as Planning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*. IJCAI, 2021.

[Mirsky *et al.*, 2019] Reuth Mirsky, Kobi Gal, Roni Stern, and Meir Kalech. Goal and Plan Recognition Design for Plan Libraries. *ACM Trans. Intell. Syst. Technol.*, 10(2):14:1–14:23, 2019.

[Mirsky *et al.*, 2022] Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V. Albrecht. A Survey of Ad Hoc Teamwork Research. In *Procceddings of the 19th European Conference on Multi-Agent Systems*, volume 13442, pages 275–293. Springer, 2022.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 2015.

[Pozanco *et al.*, 2018] Alberto Pozanco, Yolanda E-Martín, Susana Fernández, and Daniel Borrajo. Counterplanning using Goal Recognition and Landmarks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4808–4814. ijcai.org, 2018.

[Ramírez and Geffner, 2009] Miquel Ramírez and Hector Geffner. Plan Recognition as Planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1778–1783, 2009.

[Ramírez and Geffner, 2010] Miguel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1121–1126, 2010.

[Rao and Georgeff, 1992] Anand S. Rao and Michael P. Georgeff. An Abstract Architecture for Rational Agents. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 439–449. Morgan Kaufmann, 1992.

[Sohrabi *et al.*, 2016] Shirin Sohrabi, Anton V. Riabov, and Octavian Udrea. Plan recognition as planning revisited. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3258–3264, 2016.

[Stone *et al.*, 2013] Peter Stone, Gal A. Kaminka, Sarit Kraus, Jeffrey S. Rosenschein, and Noa Agmon. Teaching and leading an ad hoc teammate: Collaboration without pre-coordination. *Artificial Intelligence*, 203:35–65, 2013.

[Thangarajah and Padgham, 2011] John Thangarajah and Lin Padgham. Computationally Effective Reasoning About Goal Interactions. *Journal of Automated Reasoning*, 47(1):17–56, 2011.

[Thangarajah *et al.*, 2003] John Thangarajah, Lin Padgham, and Michael Winikoff. Detecting & Avoiding Interference Between Goals in Intelligent Agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 721–726. Morgan Kaufmann, 2003.

[Waters *et al.*, 2014] Max Waters, Lin Padgham, and Sebastian Sardiña. Evaluating Coverage Based Intention Selection. In *Proceedings of the 13th International Conference on Autonomous Agents and Multi-agent Systems*, pages 957–964. IFAAMAS, 2014.

[Waters *et al.*, 2015] Max Waters, Lin Padgham, and Sebastian Sardiña. Improving domain-independent intention selection in BDI systems. *Autonomous Agents and Multi-Agent Systems*, 29(4):683–717, 2015.

[Wu *et al.*, 2023] Di Wu, Yuan Yao, Natasha Alechina, Brian Logan, and John Thangarajah. Intention Progression with Maintenance Goals. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems*, pages 2400–2402. IFAAMAS, 2023.

[Yao and Logan, 2016] Yuan Yao and Brian Logan. Action-Level Intention Selection for BDI Agents. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*, pages 1227–1236. IFAAMAS, 2016.

[Yao *et al.*, 2014] Yuan Yao, Brian Logan, and John Thangarajah. Sp-mcts-based intention scheduling for bdi agents. In *Proceedings of the 21st European Conference on Artificial Intelligence*, pages 1133–1134. IOS Press, 2014.

[Yao *et al.*, 2016a] Yuan Yao, Lavindra de Silva, and Brian Logan. Reasoning about the Executability of Goal-Plan Trees. In *Proceedings of the 4th International Workshop on Engineering Multi-Agent Systems*, pages 181–196, 2016.

[Yao *et al.*, 2016b] Yuan Yao, Brian Logan, and John Thangarajah. Robust Execution of BDI Agent Programs by Exploiting Synergies Between Intentions. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2558–2565. AAAI Press, 2016.