Intention-Aware Multiagent Scheduling

Michael Dann
RMIT University
michael.dann@rmit.edu.au

John Thangarajah
RMIT University
john.thangarajah@rmit.edu.au

Yuan Yao
Zhejiang University of Technology
yaoyuan@zjut.edu.cn

Brian Logan
University of Nottingham
brian.logan@nottingham.ac.uk

ABSTRACT

The Belief Desire Intention (BDI) model of agency is a popular and mature paradigm for designing and implementing multiagent systems. There are several agent implementation platforms that follow the BDI model. In BDI systems, the agents typically have to pursue multiple goals, and often concurrently. The way in which the agents commit to achieving their goals forms their intentions. There has been much work on scheduling the intentions of agents. However, most of this work has focused on scheduling the intentions of a single agent with no awareness and consideration of other agents that may be operating in the same environment. They schedule the intentions of the single-agent in order to maximise the total number of goals achieved. In this work, we investigate techniques for scheduling the intentions of an agent in a multiagent setting, where an agent is aware (or partially aware) of the intentions of other agents in the environment. We use a Monte Carlo Tree Search (MCTS) based approach and show that our intention-aware scheduler generates better outcomes in cooperative, neutral (selfish) and adversarial settings than the state-of-the-art schedulers that do not consider other agents' intentions.

KEYWORDS

Intention scheduling; Multiagent scheduling; Goal reasoning

ACM Reference Format:
Michael Dann, John Thangarajah, Yuan Yao, and Brian Logan. 2020. Intention-Aware Multiagent Scheduling. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

The Belief-Desire-Intention (BDI) agent paradigm is a well established agent development paradigm with solid theoretical foundations [18] and practical implementations in the form of agent programming languages (e.g. JACK [35], JadeX[17] and Jason [3]). In the BDI paradigm, the behaviour of an agent is specified in terms of beliefs, goals, and plans – *beliefs* are what the agent knows about the environment and itself; *goals* are the states the agent desires to achieve; and *plans* are the means by which the agent achieve its goals. For each top-level goal the agent has one or more pre-defined plans that can be executed to achieve the goal. Each plan can include (primitive) actions and/or sub-goals. Each subgoal in turn has plans that can be used to achieve it. This relationship between a top level goal, its plans and sub-goals defines a tree structure for each top-level goal, which is termed the goal-plan tree (GPT) [26, 27].

During the execution of a goal, the agent will select plans to achieve the goal and subsequently further plans to achieve any subgoals, which in essence is progressively selecting a path through the goal's GPT structure. The path that the agent commits to forms the intentions of the agent and thus the GPT structures can be viewed as a model of the agent's potential intentions. The GPT structures can therefore be used to schedule the agent's intentions in order to maximise the number of goals achieved, for example. Logan et al. [15] proposed the Intention Progression Competition, which aims to evaluate state-of-the-art approaches to the intention scheduling problem that utilise these GPT structures.

Previous work has used GPTs to schedule the intentions of an agent using techniques such as *summary-information-based scheduling* [25, 26], *coverage-based scheduling* [28, 32], and *Monte Carlo Tree Search (MCTS)-based scheduling* [37, 40].

All of these existing works however, have focused on scheduling the intentions of a *single agent* with no awareness and consideration of other agents that may be operating in the same environment. The schedulers work on the assumption that the intentions of the single agent are to be scheduled cooperatively, i.e. so as to maximise the total number of goals achieved.

In this paper, we present the multiagent intention-aware scheduler, I_A, a novel framework for scheduling the intentions of an agent in a multiagent setting, where an agent is aware (or partially aware) of the intentions of other agents in the environment. I_A adapts and extends the work of Yao et al. [37] to a multiagent setting. We do this not only because their work was shown to outperform other approaches, but because the MCTS-based approach is inherently well-suited to the multiagent case. In the multiagent setting, we consider the case where each agent is aware of the other agents' intentions, in the form of the set of GPT structures. In this work, we do not explore how these may be attained or inferred, and assume they are a true reflection of the other agent's intentions. We do however consider the case of *partial vision* – where only some of the other agent's intentions are known.

We evaluate our approach in three types of multiagent setting that are characterised by the nature of the agents' behaviour towards one another:

- *allied*, where the agents collaborate to maximise the combined goal achievement;
- *neutral*, where they neither collaborate nor compete but are only interested in pursuing their own goals (i.e. they act selfishly); and
- *adversarial*, where the agents compete with each other.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Whilst there could be systems with a combination of agent behaviour types, we present these distinct cases for simplicity and clarity to better understand the effect of our approach in these different types of systems. We benchmark I_A in terms of goal completions, against the single-agent schedulers mentioned above and also the standard round-robin, first-in-first-out, and random schedulers. We show that I_A outperforms all of them.

The remainder of the paper is structured as follows. In Section 2 we present a brief overview of BDI agents and closely related work on intention scheduling, including the MCTS-based scheduler of Yao et al. We describe our approach and the I_A scheduler in Section 3. The evaluation of our approach, including the experimental setup and findings are presented in Section 4. We discuss related work in Section 5 and conclude with some future directions.

2 BACKGROUND

As explained above, the behaviour of a BDI agent is specified in terms of beliefs, goals, and plans. We assume that the agent's beliefs and goals are represented by sets of literals. Each plan is of the form: $e : \psi \leftarrow \delta$, where e is the event goal the plan is going to achieve, ψ is the context condition, and δ is plan body. The context condition is a set of literals that must be true for the plan to be considered an applicable means of achieving e in the current context. The plan body consists of a sequence of steps that are either primitive actions or subgoals. Each primitive action is specified by its precondition and postcondition which are respectively the states of the environment that must hold immediately before the action is executed, and the set of literals that are brought about by executing the action.

The relationship between goals, plans and actions can be represented by a goal-plan tree [25, 26, 36]. The root of a goal-plan tree is a top-level goal (goal-node), and its children are plan nodes

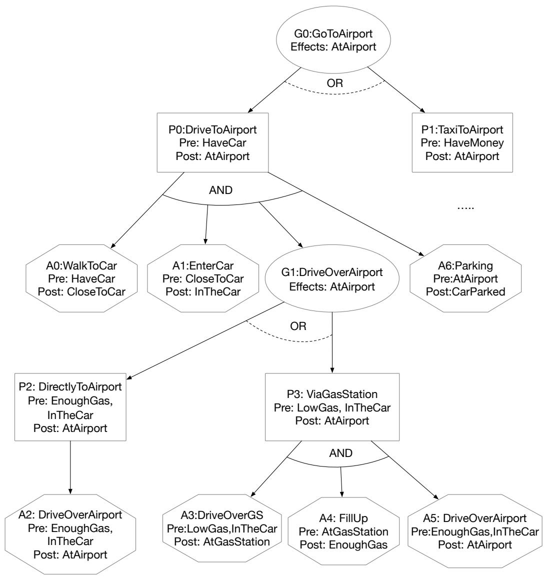

Figure 1: Example GPT; circles, rectangles and octagons represent respectively goal, plan and action nodes.

representing the potential plans to achieve the top-level goal. Each plan may contain actions (action nodes) and subgoals (goal-nodes). The leaf nodes are action-nodes representing the primitive actions that directly change the state of the environment. An example goal-plan tree for the goal $G0$ of going to the airport is shown in Figure 1. There are two plans to achieve $G0$, $P0$ and $P1$, only one of which needs to be executed to achieve the goal. Therefore, in a goal-plan tree, the children of a goal-node can be seen as "OR" nodes. In contrast, all the subgoals and actions in a plan must be achieved or executed successfully in order for the plan to succeed. Hence, the children of a plan-node can be seen as "AND" nodes. For example, the plan $P0$ contains four execution steps, i.e., WalkToCar, EnterCar, DriveOverAirport and Parking. If we want $P0$ to succeed such that the top-level goal $G0$ is achieved, then all these steps need to be executed in sequence. For an action to be executable, its preconditions must hold in the current environment.For example, the action $A1$ has the precondition that the agent is close to the car and executing the action has the effect of the agent being in the car.

The progression of an agent's intentions can be seen as choosing an appropriate interleaving of paths through the goal-plans trees corresponding to the agent's top-level goals.

2.1 Single-Agent Intention Scheduling

A number of GPT-based approaches to scheduling the intentions of a single agent have been proposed in the literature. (We discuss other approaches to intention scheduling in Section 5.)

One strand of work involves reasoning about possible interactions between intentions. Thangarajah et al. [25, 26] describe an approach based on *summary information* that avoids conflicts by reasoning about necessary and possible preconditions and postconditions of different ways of achieving a goal. They give algorithms for computing summary information at compile time, and for dynamically updating it at run-time. They also present mechanisms to determine whether a newly adopted (sub)goal will definitely be safe to execute without conflicts, will *definitely* result in conflicts, or *may* result in conflicts. If the goal cannot be executed safely, execution of the intention is deferred.

Another approach that uses the goal-plan tree structure is the *coverage-based approach* proposed by Thangarajah et al. [28]. The coverage of an intention is defined as the percentage of world states for which there is some applicable plan for any subgoal within an intention. In the coverage-based approach, the intention with the lowest coverage, i.e., the highest probability of becoming non-executable due to changes in the environment, is selected for execution. Waters et al. [32, 33] implemented two versions of the coverage-based approach, and showed a significant improvement compared to round-robin and FIFO in scheduling agent's intentions in dynamic environments.

Arguably the state-of-the-art in GPT-based intention scheduling for a single agent is that of Yao et al. [37–40]. They present an anytime stochastic approach to intention progression based on Monte-Carlo and Single-Player Monte-Carlo Tree Search [6, 14, 20], which uses random sampling to guide the expansion of the search tree. Given a set of goal-plan trees representing an agent's intentions, the MCTS-based scheduler estimates how good a next step will be based on random sampling of possible interleaving of paths through each goal-plan tree. The estimation results of each node become more and more accurate as the search tree is iteratively expanded. When the algorithm halts, the step which leads to the best child node of the root node is returned for execution. In [37], Yao et al. show that their approach outperforms most existing methods used for intention progression including first-in-first-out, round-robin, summary information approaches and coverage-based approaches.

2.2 MCTS-based Intention Scheduling

In this section, we briefly recall the MCTS-based scheduler of Yao et al. [37] which forms the starting point for our approach to scheduling in a multiagent setting, as presented in Section 3.

The MCTS-based intention scheduler takes 4 parameters as input: (1) The current state of the environment, E. (2) A set of goal-plan trees and their step pointer, corresponding to the agent's intentions, I. (3) The number of node expansions to be performed, α. (4) The number of simulations performed per node expansion, β. The parameters E and I represent the agent's current state, while α and β together define the computational budget, i.e., how much effort is to be spent on intention selection.

The root node of the MCTS tree, n_0, captures the agent's current intentions I and the current state of the environment E. The algorithm then iteratively builds a search tree from n_0 based on stochastic simulations. The edges in the search tree represent either the selection of a plan or the execution of an action, which connect the (parent) node representing the original state and the (child) node, which is the state resulting from the plan selection or subgoal selection. In addition, each node in the MCTS tree contains not only the agent's current intention base and the environment states, but also the statistics resulting from the random simulations, including the total simulation value of the node and the number of times the node has been visited.

The algorithm for building the search tree consists of 4 phases:

Selection: In the selection phase, the leaf node n_e with the greatest urgency is selected for expansion. A policy called UCT (Upper Confidence Bound for Trees) is used to determine the urgency, by balancing the exploration of less visited nodes with the exploitation of high reward nodes.

Expansion: A list of nodes representing the state after executing one step from n_e are generated and added as the children of n_e. Each child represents a different choice of plan or intention.

Simulation: After the expansion phase, one of the newly generated child nodes n_s is selected for random simulation. The simulation starts from the selected node n_s, and continues via the random selection of executable steps until a terminal state is reached, i.e., a state where all intentions have been achieved or the remaining intentions cannot be executed any further. An action or a plan is executable if its precondition or context condition holds in the current environment. When a terminal state is reached, the simulation stops and a reward value based on the evaluation function is returned. The evaluation function represents the criteria used to measure the performance of the agent and can be tailored for different problem domains. For example, one criteria used in [37] is the *fairness* in achieving the agent's top-level goals.

Back-propagation: Finally, after β simulations, the results are back-propagated from n_s to all nodes on the path to the root node.

The MCTS-based intention scheduler runs α iterations and then returns the step leading to the child node with the most favourable simulation statistics.

3 OUR APPROACH

In order to extend previous work to the multiagent setting as seamlessly as possible, we assume that the agents act in a turn-based manner, as outlined in Algorithm 1. The agents are also assumed to act in a decentralised manner, such that they may use completely different scheduling strategies, and no agent knows exactly how any other agent is going to schedule its intentions. For a given agent acting in this environment, the intention scheduling problem boils down to finding a *scheduleAction*() method that yields strong performance with respect to its own objective (which may differ from the objectives of the other agents).

As explained in the introduction, previous methods for scheduling BDI agents are not designed for multiagent environments. However, since Yao and Logan's approach [37] is based on MCTS, which is known to perform well in many multiplayer games [4, 23], we hypothesise that it ought to be adaptable to this setting. Our central idea is just to replace the single player logic in Yao and Logan's scheduler with the logic in multiplayer MCTS. However, this high-level explanation belies a number of subtle changes that must be made to the agent model, which we describe in detail below.

3.1 I_A: Intention-Aware MCTS Scheduling

Since our approach is based on multiplayer MCTS, which involves simulating the behaviour of other agents, it requires a model of how the other agents act. In multiplayer MCTS, it is assumed during the *selection* phase that external agents will seek to maximise their own objective functions. Therefore, our model must include assumptions about what these objectives are. For example, in a team-oriented task, one might assume that all agents have the same objective; namely, to maximise the total number of team goals completed. However, in a less team-oriented task, such as driving a car in heavy traffic, one might expect other agents to behave somewhat

Algorithm 1 Environment Loop

```
1:  initialise environment state, $s$
2:  while any agent has valid actions remaining do
3:      for each $agent \in agents$ do                ▷ turn-based scheduling
4:          $a \leftarrow agent.scheduleAction()$
5:          if $a \neq null$ and preconditions of $a$ are met then
6:              update $s$ according to $a$
7:          else
8:              do nothing                            ▷ ignore null/invalid actions
9:          end if
10:     end for
11: end while
```

selfishly, i.e. to favour the completion of their own goals over the completion of other agents' goals.

Under our approach, we capture this information via two components. First, we assume that for each of the n agents in the environment, the scheduler has a model of that agent's plans, structured as a forest of goal-plan trees. We denote the forest modelling agent i's plans as $\mathcal{T}_i$:

$$\mathcal{T}_i = \{t_i^1, t_i^2, \ldots, t_i^{N_i}\} \tag{1}$$

where N_i is the total number of top-level goals attributed to agent i, and each t_i^k is a goal-plan tree for achieving the kth goal. Note that agent i may in reality have fewer or greater than N_i top-level goals, and that even if the goal of t_i^k corresponds to a true goal of agent i, some details of the modelled GPT may be inaccurate. The question of how well our scheduler performs under faulty assumptions about the other agents' intentions is explored in our later experiments.

The second component of our model, which aims to capture the overall objective of each agent (with respect to both its own goals *and* the goals of the other agents acting in the environment) is a payoff matrix, $\mathbf{P}$:

$$\mathbf{P} = \begin{bmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \mathcal{P}_{13} & \ldots & \mathcal{P}_{1n} \\ \mathcal{P}_{21} & \mathcal{P}_{22} & \mathcal{P}_{23} & \ldots & \mathcal{P}_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \mathcal{P}_{n3} & \ldots & \mathcal{P}_{nn} \end{bmatrix} \tag{2}$$

where $\mathcal{P}_{ij} = \langle p_{ij}^1, p_{ij}^2, \ldots, p_{ij}^{N_j} \rangle$ are the assumed payoffs that agent i receives upon the completion of agent j's goals. Payoffs may be positive (implying a shared objective), zero (implying neutrality) or negative (implying that agent i would prefer agent j *not* to complete the corresponding goal).

Under this model, the total payoff to an agent at the end of a simulation depends on the completion status of *all* agents' goals. Let C_j be a binary vector that captures the completion status of agent j's goals, i.e.

$$C_j = \langle c_j^1, c_j^2, \ldots, c_j^{N_j} \rangle \tag{3}$$

$$\text{where} \quad c_j^k = \begin{cases} 1, & \text{if } t_j^k \in \mathcal{T}_j \text{ has been completed} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Agent i's total payoff can then be written as:

$$\text{payoff}_i = \sum_{j=1}^{n} \mathcal{P}_{ij} \cdot C_j \tag{5}$$

where the sum is taken over all agents in the environment.

In the *selection* phase of MCTS, actions are determined by maximising the *UCB1* value [4], which is calculated by summing the average payoff with an exploration bonus:

$$\text{UCB1 value} = \frac{\mathit{sum}(\text{payoffs})}{n} + c\sqrt{\frac{2 \ln N}{n}} \tag{6}$$

where c is an exploration constant, n is the number of times the action has been tried, N is the total number of simulations performed so far, and payoffs are calculated according to Equation 5. In our experiments we set c to the standard value of $\sqrt{2}$; we also tried doubling and halving this value to little effect.

We name the overall scheduling approach, I_A.

3.2 Assumptions

One of the main assumptions made in I_A is that the scheduling is turn-based. An alternative approach would be to add execution times to actions, as in Yao et al.'s work [39], and let each agent act whenever it is free. However, this would require an approach to handling environment variable conflicts, i.e. when two agents are attempting to modify the same environment variable simultaneously. Since temporal aspects are not the main focus of this work, we deliberately avoid this complication.

Another requirement of our approach is that the scheduler can somehow infer the progress of the other agents' GPTs, so that it knows the starting point for the simulations. In this work, we take a simple approach: First, we assume that the scheduler can observe all environment variable changes. Then, when an action is taken by some external agent, it either:

(i) conforms to the scheduler's model of that agent, i.e. the action taken was possible under the scheduler's model, and implies that the external agent progressed a particular intention. The scheduler then records the progress of that intention; or
(ii) does not conform to the scheduler's model of the external agent, in which case the scheduler does not record an update.

A more sophisticated approach would be to somehow maintain a probability distribution over the other agents' statuses and sample from that distribution at the beginning of the simulations, though we leave this as an idea for future work.

Finally, one additional consideration in multiagent scheduling is that it may not always be beneficial for an agent to progress an intention, since doing so may obstruct an allied agent, or assist an adversary. To account for this, we allow schedulers to pass. In preliminary experiments, this occasionally led to stalemates, where all schedulers passed repeatedly. We resolved this by terminating the task if all schedulers passed three times in a row.

4 EXPERIMENTS

In this section, we investigate the properties of our proposed scheduler by conducting a series of experiments.

4.1 Experimental Setup

While I_A can in theory handle environments containing any number of agents, we limited our study to two-agent environments in order to simplify analysis. To investigate the behaviour of I_A under different payoff schemes, we considered three configurations:

- **Allied** (Section 4.2). I_A is incentivised to behave in a team-oriented manner, receiving a +1 payoff whenever *either* agent completes a goal. An example of where such scenarios might arise is a team robotics task where the behaviour of all robots is known, but not all of them can be reprogrammed to act under a single scheduler.
- **Neutral** (Section 4.3). I_A is configured to behave "selfishly", receiving a +1 payoff whenever it completes a goal, but 0 payoff whenever the other agent completes a goal. This scenario is motivated by tasks involving a limited resource, where agents will naturally interfere with each other by pursuing their intentions.
- **Adversarial** (Section 4.4). I_A still receives a +1 payoff for completing its own goals, but is also encouraged to obstruct the other agent via a -1 payoff whenever the other agent completes a goal.

An example scenario is a capture-the-flag game, where each team knows roughly the high-level strategies of the other team.

Note that the individual payoff values need not be in {-1, 0, +1}, nor are the above the only possible alliance types; they were merely chosen as a diverse set of scenarios.

To generate goal-plan trees for our experiments, we used the same generator as Yao and Logan [37]. This generator was also used for the 2019 Intention Progression Competition[1], and is designed to generate non-trivial tasks that include conflicting goals and actions. In keeping with our assumption that the environment is turned-based, the actions in the generated GPTs are not durative. It is possible for an agent to block another agent's future action by violating its preconditions, but it is impossible to violate the preconditions of an action during its execution. For the most part, we used the same generator settings as in Section 6.1.1 of Yao and Logan's work [37]. The generated GPTs had a depth of 5 and one subgoal per plan. Each forest contained 12 GPTs, with 6 assigned to each scheduler in our two-agent experiments. Each GPT contained either one or two plans, with a 50% chance of each.[2]

The first key question we sought to answer was whether I_A's intention-awareness would actually give it an advantage in practice. To answer this, we benchmarked against a number of different single-agent schedulers from previous work. These "naïve" schedulers simply progress their own intentions, ignoring those of other agents. They still witness environment variable changes caused by other agents, but cannot anticipate these changes. The specific schedulers we benchmarked against were as follows:

- **Naïve MCTS**. Our implementation of this scheduler was taken from Yao and Logan [37]. It can be thought of as an "intention-unaware" version of I_A.
- **First in, First out (FIFO)**. FIFO selects the first intention in its GPT forest that is progressable, then continues progressing that intention until it is completed or no longer progressable. It keeps repeating this process until no intentions are progressable.
- **C0** and **C1**. These schedulers are based on the *coverage* heuristic, as explained in Section 2.1. C1 always progresses the intention with the least coverage, while C0 is essentially a hybrid between C1 and FIFO: It selects the intention with the least coverage, then progresses that intention until it is completed or no longer progressable. Our implementations are exactly the same as in Yao and Logan [37].
- **Round robin (RR)**. Iterates through the intentions in turn, skipping intentions that are unprogressable.
- **Random**. Randomly selects a progressable intention. This is the same scheduling approach used to generate the rollouts in the *simulation* phase of I_A.

Besides serving as baselines, the above schedulers also serve as different partner types in our experiments.

On the presumption that intention-awareness would prove beneficial in at least some cases, we sought to investigate whether the edge would persist if I_A was given only partial knowledge of the other agent's intentions. To do so, we configured two versions of I_A: *Full vision*, which sees all of the other agent's GPTs, and *partial vision*, which sees only half of the other agent's GPTs. The partial vision scheduler sees 9 GPTs in total: Its own 6 GPTs, plus 3 belonging to the other agent. Moreover, it only receives payoffs for the completion of goals that it is aware of.

Finally, we sought to investigate how much I_A's performance would be affected by its assumption regarding the other agent's objective; namely, by whether I_A expects the other agent to behave selfishly or in a team-oriented manner. Accordingly, we ran the allied and neutral experiments twice: Once with I_A expecting the other agent to maximise the neutral objective, and once with it expecting the other agent to maximise the allied objective. In the adversarial experiments, I_A was configured to expect adversarial behaviour from the other agent, since this is arguably the most natural assumption in adversarial domains.

Following Yao and Logan [37], we set $\alpha = 100$ and $\beta = 10$ for all MCTS-based schedulers (full vision I_A, partial vision I_A and naïve MCTS). To account for the fact that first agent to act will generally complete more intentions on average, we ran two-sided "mirror matches", with the GPT assignments and first agent to act swapped for the second leg. All results were obtained by averaging over 100 randomly generated GPT forests.

4.2 Allied

Results for the *allied* experiment are displayed in Table 1. The top third of the table shows the average total team score when the scheduler to the left was teamed with the various allies along the top. The remaining sections of the table break down the score into the average number goals that the scheduler itself completed, and the average number of goals that its ally completed.

As indicated by the figures in bold, the highest total team score for each ally type was achieved by either full vision I_A or partial vision I_A. The edge held by the intention-aware schedulers was particularly pronounced when partnered with weak allies. For example, when partnered with the random scheduler, full vision I_A's average team score was 10.0, which is almost a full point higher than what any other scheduler achieved. As the score breakdown highlights, this was not only because I_A completed more of its own goals; it was also because I_A enabled its allies to complete more goals, *despite having no direct control over its allies' decisions*. In the case above, the random scheduler completed roughly a full goal more on average when allied with full vision I_A compared to when it was allied with naïve MCTS. This is a very strong result, as it implies that including an intention-aware agent on a team can potentially elevate the performance of other agents, even without reconfiguring them.

The results shown in Table 1 were generated with I_A configured to expect allied behaviour from its partner. That is, it was assumed in the MCTS simulations that the other agent would seek to maximise the total team score. In reality though, the naïve schedulers are inherently neutral, since they are unaware of other agents. For this reason, we suspected that I_A might perform better if configured to expect neutral behaviour from its partner. However, when we reran the experiment with this configuration, the results were much the same. We omit the full results here to save space, but to briefly summarise: Under the neutral behaviour assumption, full vision I_A achieved ≈ 0.2 more team goals when allied with round robin, but

[1] https://www.intentionprogression.org/

[2] Yao and Logan vary the chance of there being one plan through 0–75%. We fix this value to 50% as a happy medium, and to avoid an excessive number of configurations.

		Ally							
		I_A full	I_A partial	Naïve MCTS	FIFO	C0	C1	Round Rob.	Random
Total Score	I_A full	11.125	**11.239**	**11.188**	10.148	**10.631**	**10.420**	**9.233**	**10.034**
	I_A partial	**11.239**	11.153	10.842	**10.158**	10.468	10.061	8.767	9.176
	Naïve MCTS	11.188	11.000	10.775	9.967	10.396	9.983	8.733	9.172
	FIFO	10.148	10.222	9.967	9.350	9.783	9.256	7.711	8.422
	C0	10.631	10.585	10.396	9.783	10.272	9.683	7.539	8.850
	C1	10.420	10.278	9.983	9.256	9.683	9.261	7.522	8.589
	Round Rob.	9.233	8.938	8.733	7.711	7.539	7.522	8.406	7.783
	Random	10.034	9.688	9.172	8.422	8.850	8.589	7.783	7.544
Own Score	I_A full	5.563	5.591	5.489	5.409	5.369	5.426	5.193	5.398
	I_A partial	5.648	5.577	5.455	5.460	5.358	5.398	5.284	5.420
	Naïve MCTS	5.699	5.545	5.387	5.269	5.286	5.320	5.250	5.417
	FIFO	4.739	4.761	4.698	4.675	4.683	4.639	4.017	4.639
	C0	5.261	5.227	5.110	5.100	5.136	5.089	4.022	5.061
	C1	4.994	4.881	4.663	4.617	4.594	4.631	4.094	4.722
	Round Rob.	4.040	3.653	3.483	3.694	3.517	3.428	4.203	3.506
	Random	4.636	4.267	3.756	3.783	3.789	3.867	4.278	3.772
Ally Score	I_A full	5.563	5.648	5.699	4.739	5.261	4.994	4.040	4.636
	I_A partial	5.591	5.577	5.387	4.698	5.110	4.663	3.483	3.756
	Naïve MCTS	5.489	5.455	5.387	4.698	5.110	4.663	3.483	3.756
	FIFO	5.409	5.460	5.269	4.675	5.100	4.617	3.694	3.783
	C0	5.369	5.358	5.286	4.683	5.136	4.594	3.517	3.789
	C1	5.426	5.398	5.320	4.639	5.089	4.631	3.428	3.867
	Round Rob.	5.193	5.284	5.250	4.017	4.022	4.094	4.203	4.278
	Random	5.398	5.420	5.417	4.639	5.061	4.722	3.506	3.772

Table 1: Allied setting. The best total team score with each ally type is bolded. Since each scheduler was assigned 6 GPTs, the maximum possible team score was 12.

≈ 0.2 less team goals when allied with naïve MCTS and random. All other differences were negligible. The results for partial vision I_A differed by similarly small magnitudes. Despite not matching our expectations, these results are positive because they imply that I_A is not overly reliant on the accuracy of the payoff matrix.

One final point worth noting about the allied setting is that even in scenarios where the entire team can theoretically be scheduled collaboratively, it may not be tractable to do so, especially if there is a large number of GPTs. This is especially pertinent to MCTS-based scheduling, since the more GPTs, the larger the branching factor of the search tree and the greater the length of the simulations. To illustrate this point, we ran a side experiment using the same randomly generated GPTs as above, but with a single MCTS scheduler responsible for scheduling the entire forest. The single scheduler completed a fractionally larger percentage of goals than any of the pairings in Table 1, achieving 11.28 goals on average. However, its total computation time was almost double that of the dual I_A partial + I_A partial alliance (precisely, it took 92% longer), and even compared to the dual *full vision* alliance, it took 36% longer.[3] An idea for future work is to consider ways of splitting a team's goals between partial vision schedulers so as to achieve the best possible trade-off between average goal completions and computation time.

[3]This second result might seem counter-intuitive, given that none of the GPTs are excluded from the simulations in the full vision case. However, forcing the schedulers to select from one half of the forest makes them behave more FIFO-like, which in turn leads to less drawn out simulations.

4.3 Neutral

The results of the *neutral* experiment are summarised in Table 2. For this experiment, we report only the scheduler's own number of goals completed, since it is the only measure that matters in this scenario. The results clearly show the benefit of intention-awareness: No matter the type of other agent acting in the environment, full vision I_A completed the most goals on average, while partial vision I_A completed the second most goals. As in the *allied* experiments, it mattered little whether I_A was configured to expect allied behaviour or neutral behaviour from the other agent. (The I_A results in Table 2 are based on assuming neutral behaviour from the other agent; results under the other setting differed by no more than ± 0.1)

4.4 Adversarial

In the *adversarial* experiment, full vision I_A achieved a net positive score against all other schedulers (see Table 3). Partial vision I_A achieved a positive score against all schedulers other than full vision I_A, again illustrating that I_A was able to exploit even limited knowledge of the other agent's intentions.

For context, it is important to note that the GPT generator we used does not purposely create "wrecker plans", i.e. plans whose goal is the negation of another agent's goal. As such, the *adversarial* scenario should not be interpreted as typical head-to-head combat; rather, it resembles a situation where there is a "mole" trying to obstruct the plans of another agent while seemingly only following its

		Other Scheduler							
		I_A full	I_A partial	Naïve MCTS	FIFO	C0	C1	Round Rob.	Random
Score	I_A full	**5.561**	**5.578**	**5.606**	**5.528**	**5.517**	**5.528**	**5.388**	**5.506**
	I_A partial	5.506	5.511	5.539	5.489	5.444	5.478	5.315	5.461
	Naïve MCTS	5.344	5.404	5.424	5.330	5.330	5.284	5.307	5.358
	FIFO	4.706	4.697	4.727	4.690	4.693	4.642	4.028	4.614
	C0	5.094	5.118	5.108	5.102	5.139	5.108	4.028	5.051
	C1	4.672	4.685	4.665	4.614	4.631	4.625	4.085	4.688
	Round Rob.	3.713	3.629	3.489	3.710	3.528	3.403	4.170	3.557
	Random	3.944	3.860	3.983	3.864	3.790	3.670	4.290	3.784

Table 2: Neutral setting. The best results achieved with respect to the other scheduler type are bolded.

		Opponent							
		I_A full	I_A partial	Naïve MCTS	FIFO	C0	C1	Round Rob.	Random
Net Score	I_A full	**0.000**	**0.363**	**0.706**	**1.029**	**0.512**	**1.424**	**2.118**	**2.329**
	I_A partial	-0.363	0.000	0.465	0.876	0.417	1.083	1.804	2.006
	Naïve MCTS	-0.706	-0.465	0.000	0.577	0.179	0.631	1.738	1.554
	FIFO	-1.029	-0.876	-0.577	0.000	-0.423	0.024	0.298	0.845
	C0	-0.512	-0.417	-0.179	0.423	0.000	0.488	0.512	1.190
	C1	-1.424	-1.083	-0.631	-0.024	-0.488	0.000	0.732	0.923
	Round Rob.	-2.118	-1.804	-1.738	-0.298	-0.512	-0.732	0.000	-0.738
	Random	-2.329	-2.006	-1.554	-0.845	-1.190	-0.923	0.738	0.000
Score For	I_A full	5.049	5.152	5.418	5.476	5.359	5.388	5.253	5.329
	I_A partial	4.789	5.124	5.424	5.400	5.310	5.393	5.286	5.411
	Naïve MCTS	4.712	4.959	5.360	5.298	5.280	5.321	5.262	5.399
	FIFO	4.447	4.524	4.720	4.699	4.696	4.631	4.048	4.637
	C0	4.847	4.893	5.101	5.119	5.158	5.113	4.060	5.065
	C1	3.965	4.310	4.690	4.607	4.625	4.658	4.173	4.661
	Round Rob.	3.135	3.482	3.524	3.750	3.548	3.440	4.268	3.554
	Random	3.000	3.405	3.845	3.792	3.875	3.738	4.292	3.845
Score Against	I_A full	5.049	4.789	4.712	4.447	4.847	3.965	3.135	3.000
	I_A partial	5.152	5.124	4.959	4.524	4.893	4.310	3.482	3.405
	Naïve MCTS	5.418	5.424	5.360	4.720	5.101	4.690	3.524	3.845
	FIFO	5.476	5.400	5.298	4.699	5.119	4.607	3.750	3.792
	C0	5.359	5.310	5.280	4.696	5.158	4.625	3.548	3.875
	C1	5.388	5.393	5.321	4.631	5.113	4.658	3.440	3.738
	Round Rob.	5.253	5.286	5.262	4.048	4.060	4.173	4.268	4.292
	Random	5.329	5.411	5.399	4.637	5.065	4.661	3.554	3.845

Table 3: Adversarial setting. The best net score achieved against each opponent type is bolded.

own plans. Nonetheless, our framework is certainly capable of accommodating wrecker plans, and I_A's edge would almost certainly be stronger if they were included.

Even without wrecker plans, I_A was effective at interfering with the plans of other agents. For example, naïve MCTS completed roughly 0.5 goals less against full vision I_A than it did against any of the non-intention-aware agents. Interestingly though, I_A had relatively less success obstructing C0 and FIFO. (Recall from earlier that C0 also behaves in a FIFO-like manner.) One explanation for this is that the policy of completing one intention before starting another is an effective defense mechanism against plan interruption. An idea for future work is to bias I_A's rollouts via the FIFO heuristic to see if it yields better performance than uniform random rollouts.

A final point to note regarding I_A's performance is that the α and β values we used for the MCTS simulations ($\alpha = 100$ and $\beta = 10$), are relatively small compared to those usually employed in game-playing agents. We chose these values partly so that we could run a large number of experiments, and partly to match Yao and Logan [37]. In a side experiment, we tried increasing α to 500, and ran the agent against the default configuration with $\alpha = 100$. The $\alpha = 500$ scheduler achieved a net score of +0.11, completing an average of 5.08 goals versus 4.97 by the $\alpha = 100$ scheduler. On our hardware, the $\alpha = 500$ scheduler still only took at most a few seconds to calculate each action, which is reasonable for many applications. In domains that allow more deliberation time, it ought to be possible to improve I_A's performance by further increasing α.

5 RELATED WORK

In addition to the work of Thangarajah et al. [25, 26, 28], Waters et al. [32, 33] and Yao et al. [38] discussed previously, a number of other approaches to scheduling intentions to avoid conflicts have been proposed in the literature.

Shaw and Bordini have proposed approaches to intention selection based on Petri nets [21] and constraint logic programming [22]. In their work (as in [25, 26]), the plans and sub-goals in a goal-plan tree are regarded as basic steps, and interleaving is at the level of sub-plans and subgoals. They do not consider interactions between actions in plans.

The TÆMS (Task Analysis, Environment Modelling, and Simulation) framework [10] together with Design-To-Criteria (DTC) scheduling [30] have been used in agent architectures such the Soft Real-Time Agent Architecture [29] and AgentSpeak(XL) [2] to schedule intentions. TÆMS provides a high-level framework for specifying the expected quality, cost and duration of of methods (actions) and relationships between tasks (plans). DTC decides which tasks to perform, how to perform them, and the order in which they should be performed, so as to satisfy hard constraints (e.g. deadlines) and maximise the agent's objective function. DTC can produce schedules that allow interleaved or parallel execution of tasks and can be used in an anytime fashion. In the work closest to that presented here [2], DTC was used to schedule execution of AgentSpeak intentions at the level of individual plans. The TÆMS relations between plans required to generate a schedule (*enables*, *facilitates* and *hinders*) were specified as part of the agent program. In contrast, I_A interleaves intentions at the level of actions, and information about possible conflicts between intentions is extracted automatically from GPTs generated from the agent program.

In [19] Sardina et al. show how an HTN planner can be integrated into a BDI agent architecture. However their focus is on finding a hierarchical decomposition of a plan that is less likely to fail by avoiding incorrect decisions at choice points, and they do not take into account interactions with other concurrent intentions.

In [34], Wilkins et al. presented the Cypress architecture, which combines the Procedural Reasoning System reactive executor PRS-CL, and the SIPE-2 look-ahead planner. A Cypress agent uses PRS-CL to pursue its intentions using a library of procedures (plans). If a failure occurs during the execution of the plan due to an unanticipated change in the agent's environment, the executor calls SIPE-2 to produce a new plan to achieve the goal, and continues executing those portions of plans which are not affected. However, their approach focuses on generating new plans to recover from plan failures, rather than interleaving intentions so as to avoid conflicts.

There has also been work on avoiding conflicts in a multiagent setting. For example, Clement and Durfee [7–9] propose an approach to coordinating concurrent hierarchical planning agents using summary information and HTN planning. However in this work, summary information is used to identify when conflicts may arise between two or more agents rather than to avoid conflicts between the intentions of a single agent. Moreover, it is assumed that the agents plan offline in a static environment. In [11], Ephrahi et al. present an approach to planning and interleaving the execution of tasks by multiple agents. The task of each agent is assigned dynamically, and the execution of all tasks achieves a global goal. They show how conflicts between intentions can be avoided by appropriate scheduling of the actions of the agents.

There has also been work on using MCTS to coordinate the activities of multiple agents. Baker et al. [1] present Co-MCTS, a factored coordinated Monte Carlo tree search algorithm to perform decentralised path planning for multiple coordinated UAVs, and show that it out-performs both conventional path planning and MCTS on real-world examples. However, unlike I_A, Co-MCTS takes a centralised approach.

At a high level, multiagent scheduling methods are also related to multiagent reinforcement learning (MARL) [5]. Within MARL, there is extensive work on learning to communicate [12, 24], coordinating exploration [16, 31], and handling non-stationarity (which arises if other agents in the environment also change their behaviour over the course of training) [13]. However, these issues are less relevant to multiagent scheduling, where it is generally assumed that the agents are already capable of performing various behaviours, and the focus is more on achieving synergy. While synergy is also important in multiagent RL, it is often implicit, i.e. by striving for greater expected returns, the agents naturally learn to perform synergistic actions. Moreover, in multiagent scheduling, it is commonly assumed that there is explicit structure in place that allows agents to reason about conflicts, e.g. GPTs.

6 CONCLUSION

In this paper we introduced I_A, an MCTS-based framework for performing intention-aware scheduling in multiagent environments. We tested our approach in three dual-agent scenarios: (i) *allied*, where the agents collaborate to maximise the total number of goals achieved; (ii) *neutral*, where the agents pursue the selfish objective of maximising only their own goal achievement; and (iii) *adversarial*, where the agents strive to complete their own goals whilst interfering with the other agent.

Compared to several previously proposed schedulers that act without regard to other agents, I_A's awareness of other agents' plan structures gave it a clear edge. In the *allied* and *adversarial* experiments, it was not just that I_A completed more of its own goals; it also enabled its allies – particularly the weaker ones – to complete more goals, and was effective at blocking the plans of its adversaries, despite not possessing "wrecker" plans.

When configured so that it could only see some of the other agent's goal-plan trees, the scheduler's performance expectedly dropped off compared to full vision I_A, but remained above that of naïve MCTS (the state-of-the-art approach for the single agent setting), thus confirming that our approach was able to exploit partial information. Moreover, in the *allied* and *neutral* experiments, I_A's performance proved robust to different assumptions about the selfishness of the other agent.

Besides the several extension ideas already outlined throughout the paper, a natural direction for future work is to try to reduce I_A's reliance on upfront knowledge about the other agents' goal-plan trees. For example, one idea is to start with a candidate set of goal-plan trees for the other agents, and try to infer by their actions which trees they are actually following.

REFERENCES

[1] Chris A. B. Baker, Sarvapali Ramchurn, W. T. Luke Teacy, and Nicholas R. Jennings. 2016. Planning Search and Rescue Missions for UAV Teams. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI-2016)*, G. A. Kaminka, M. Fox, P. Bouquet, Hullermeijer E., Dignum F., Dignum V., and van Harmalen F. (Eds.). ECCAI, IOS Press, The Hague, The Netherlands, 1777–1782.

[2] Rafael H. Bordini, Ana L. C. Bazzan, Rafael de O. Jannone, Daniel M. Basso, Rosa M. Vicari, and Victor R. Lesser. 2002. AgentSpeak(XL): Efficient Intention Selection in BDI Agents via Decision-Theoretic Task Scheduling. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*. 1294–1302.

[3] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. 2007. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. Vol. 8. John Wiley & Sons.

[4] Cameron Browne, Edward J. Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intellig. and AI in Games* 4, 1 (March 2012), 1–43.

[5] L. Busoniu, R. Babuska, and B. De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[6] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. 2008. Monte-Carlo Tree Search: A New Framework for Game AI. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*. Stanford, California, USA.

[7] Bradley J. Clement and Edmund H. Durfee. 1999. Theory for Coordinating Concurrent Hierarchical Planning Agents Using Summary Information. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*, Jim Hendler and Devika Subramanian (Eds.). AAAI Press / The MIT Press, Orlando, Florida, USA, 495–502.

[8] Bradley J. Clement and Edmund H. Durfee. 2000. Performance of Coordinating Concurrent Hierarchical Planning Agents Using Summary Information. In *4th International Conference on Multi-Agent Systems*. IEEE Computer Society, Boston, MA, USA, 373–374.

[9] Bradley J. Clement, Edmund H. Durfee, and Anthony C. Barrett. 2007. Abstract Reasoning for Planning and Coordination. *J. Artif. Intell. Res. (JAIR)* 28 (2007), 453–515.

[10] K. S. Decker and V. R. Lesser. 1993. Quantitative Modeling of Complex Environments. *International Journal of Intelligent Systems in Accounting, Finance and Management* 2 (1993), 215–234.

[11] Eithan Ephrati and Jeffrey S. Rosenschein. 1993. A Framework for the Interleaving of Execution and Planning for Dynamic Tasks by Multiple Agents. In *From Reaction to Cognition, 5th European Workshop on Modelling Autonomous Agents, MAAMAW '93, Neuchatel, Switzerland, August 25-27, 1993, Selected Papers*, Cristiano Castelfranchi and Jean-Pierre Müller (Eds.). Springer, Neuchatel, Switzerland, 139–153.

[12] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 2137–2145.

[13] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning*. 1146–1155.

[14] Levente Kocsis and Csaba Szepesvári. 2006. Bandit Based Monte-Carlo Planning. In *17th European Conference on Machine Learning*, Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (Eds.). Springer, Berlin, Germany, 282–293.

[15] Brian Logan, John Thangarajah, and Neil Yorke-Smith. 2017. Progressing Intention Progression: A Call for a Goal-Plan Tree Contest. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*. ACM, 768–772.

[16] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-Agent Variational Exploration. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 7613–7624.

[17] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. 2005. *Jadex: A BDI Reasoning Engine*. Springer US, Boston, MA, 149–174.

[18] Anand S. Rao and Michael P. Georgeff. 1991. Modeling Rational Agents within a BDI-Architecture. In *Proc. of KR'91*. Cambridge,MA, 473–484.

[19] Sebastian Sardiña, Lavindra de Silva, and Lin Padgham. 2006. Hierarchical Planning in BDI Agent Programming Languages: A Formal Approach. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone (Eds.). ACM, Hakodate, Japan, 1001–1008.

[20] Maarten P. D. Schadd, Mark H. M. Winands, Mandy J. W. Tak, and Jos W. H. M. Uiterwijk. 2012. Single-Player Monte-Carlo Tree Search for SameGame. *Knowl.-Based Syst.* 34 (2012), 3–11.

[21] Patricia H. Shaw and Rafael H. Bordini. 2007. Towards Alternative Approaches to Reasoning About Goals. In *Declarative Agent Languages and Technologies V, 5th International Workshop*, Matteo Baldoni, Tran Cao Son, M. Birna van Riemsdijk, and Michael Winikoff (Eds.), Vol. 4897. Springer, Honolulu, HI, USA, 104–121.

[22] Patricia H. Shaw and Rafael H. Bordini. 2010. An Alternative Approach for Reasoning about the Goal-Plan Tree Problem. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, Helder Coelho, Rudi Studer, and Michael Wooldridge (Eds.), Vol. 215. IOS Press, Lisbon, Portugal, 1035–1036.

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[24] Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 2244–2252.

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

[26] John Thangarajah, Lin Padgham, and Michael Winikoff. 2003. Detecting & Avoiding Interference Between Goals in Intelligent Agents. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, Georg Gottlob and Toby Walsh (Eds.). Morgan Kaufmann, Acapulco, Mexico, 721–726.

[27] John Thangarajah, Lin Padgham, and Michael Winikoff. 2003. Detecting & Exploiting Positive Goal Interaction in Intelligent Agents. In *The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003*. ACM, Melbourne, Victoria, Australia, 401–408.

[28] John Thangarajah, Sebastian Sardina, and Lin Padgham. 2012. Measuring Plan Coverage and Overlap for Agent Reasoning. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 1049–1056.

[29] Régis Vincent, Bryan Horling, Victor Lesser, and Thomas Wagner. 2001. Implementing Soft Real-Time Agent Control. In *Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS'01)*. ACM Press, New York, NY, USA, 355–362. https://doi.org/10.1145/375735.376329

[30] T. Wagner, A. Garvey, and V. Lesser. 1998. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning* 19 (1998), 91–118.

[31] Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. 2020. Influence-Based Multi-Agent Exploration. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. https://arxiv.org/pdf/1910.05512.pdf

[32] Max Waters, Lin Padgham, and Sebastian Sardina. 2014. Evaluating Coverage Based Intention Selection. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (Eds.). 957–964.

[33] Max Waters, Lin Padgham, and Sebastian Sardiña. 2015. Improving domain-independent intention selection in BDI systems. *Autonomous Agents and Multi-Agent Systems* 29, 4 (2015), 683–717. https://doi.org/10.1007/s10458-015-9293-5

[34] David E Wilkins, Karen L Myers, John D Lowrance, and Leonard P Wesley. 1995. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental & Theoretical Artificial Intelligence* 7, 1 (1995), 121–152.

[35] Michael Winikoff. 2005. JACK Intelligent Agents: An Industrial Strength Platform. In *Multi-Agent Programming*. Springer, New York, NY, 175–193.

[36] Yuan Yao, Lavindra de Silva, and Brian Logan. 2016. Reasoning About the Executability of Goal-Plan Trees. In *Proceedings of the 4th International Workshop on Engineering Multi-Agent Systems (EMAS 2016)*. Singapore, 181–196.

[37] Yuan Yao and Brian Logan. 2016. Action-Level Intention Selection for BDI Agents. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, J. Thangarajah, K. Tuyls, C. Jonker, and S. Marsella (Eds.). IFAAMAS, Singapore, 1227–1235.

[38] Yuan Yao, Brian Logan, and John Thangarajah. 2014. SP-MCTS-based Intention Scheduling for BDI Agents. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI-2014)*, Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan (Eds.). ECCAI, IOS Press, Prague, Czech Republic, 1133–1134.

[39] Yuan Yao, Brian Logan, and John Thangarajah. 2016. Intention Selection with Deadlines. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI-2016)*, G. A. Kaminka, M. Fox, P. Bouquet, Hullermeijer E., Dignum F., Dignum V., and van Harmalen F. (Eds.). ECCAI, IOS Press, The Hague, The Netherlands, 1700–1701. https://doi.org/10.3233/978-1-61499-672-9-1700

[40] Yuan Yao, Brian Logan, and John Thangarajah. 2016. Robust Execution of BDI Agent Programs by Exploiting Synergies Between Intentions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, Dale Schuurmans and Michael P. Wellman (Eds.). AAAI Press, Phoenix, USA, 2558–2564. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12148