Intention Selection with Deadlines

Yuan Yao[1] and **Brian Logan**[1] and **John Thangarajah**[2]

1 INTRODUCTION

In BDI agent programming, an *intention* is the combined plan steps an agent commits to in order to achieve a goal. One of the key features of the BDI approach is the ability of an agent to pursue multiple goals concurrently, by interleaving the steps of multiple intentions. Choosing the next step to progress (execute) from these concurrent intentions is critical, as the wrong choice can result in failure to achieve one or more goals. Conversely, appropriate scheduling of the steps in intentions can maximise the number of goals achieved by the agent. Deciding which intention to progress next becomes more challenging in settings where goals must be achieved before a deadline. An interleaving of steps in the agent's intentions that avoids conflicts may still result in failure to achieve a goal by its deadline.

There has been relatively little work on intention selection with deadlines. One recent exception is AgentSpeak(RT). AgentSpeak(RT) [3, 2] is a real-time agent programming language based on AgentSpeak(L) in which top-level goals may have deadlines and priorities. Given the estimated execution time of plans, AgentSpeak(RT) schedules intentions so as to achieve a priority-maximal set of intentions by their deadlines with a specified level of confidence. However AgentSpeak(RT) avoids conflicts by scheduling potentially conflicting intentions in FIFO fashion, which can make it more difficult for an agent to achieve its goals by their deadlines.

In this paper, we present S_R, a novel approach to intention selection with deadlines. S_R extends the stochastic scheduling approach of Yao et al. [5, 6, 4] in two ways. First, goals may have both a preferred achievement time (the time by which the goal should ideally be achieved) and a deadline (the time by which the goal must be achieved). Second, S_R supports the concurrent execution of durative actions in different intentions. S_R schedules intentions so as to maximise the number of goals achieved and minimise tardiness (i.e, the difference between the time a goal is achieved and its preferred achievement time). We evaluate the performance of S_R and compare it to that of AgentSpeak(RT) in a simple blocks world domain. Our results suggest S_R outperforms AgentSpeak(RT) in this scenario.

2 INTENTIONS WITH DEADLINES

We extend the notion of goal found in BDI-based languages by introducing two temporal constructs: a preferred achievement time (soft deadline) p, and a deadline (hard deadline) d, where $p \leq d \leq \infty$. The *preferred achievement time* specifies the time by which a goal should ideally be achieved; however achieving the goal after the preferred achievement time still has value. The *deadline* specifies the time by which a goal must be achieved; achieving the goal after the deadline has no value. Goals which are not achieved by their (hard) deadline are dropped. If a top-level goal with preferred achievement time p is achieved at time $t \leq d$, we define the *tardiness* in achieving the goal as $max(0, t - p)$.

Each action a an agent may perform also has an *estimated execution time*, $e(a, \Phi)$, which is a function of the agent's beliefs Φ. The estimated execution time may be specified by the developer, e.g., the expected execution time of a 'move' action may be based on the distance to be moved, or learnt from experience, e.g., the agent may learn that traversing a hallway during a busy period takes longer than in the middle of the night. In general, the actual time it takes an action to execute, t, will differ from $e(a, \Phi)$. We define the estimation error in the execution time of an action a by $\frac{|e(a,\Phi)-t|}{e(a,\Phi)}$.

We assume that the agent executes its intentions in parallel. Progressing intentions in parallel allows the agent to achieve the maximum number of goals by their deadlines. (Executing intentions in parallel is often desirable for other reasons, e.g., to ensure more predictable response times for users or other agents.) Since external actions will, in general, require more time than a single deliberation cycle to execute, this means that multiple actions will typically be executing concurrently.

3 INTENTION SELECTION

In this section, we present our approach to intention selection, S_R. S_R attempts to find an interleaving of steps in the agent's intentions which (a) maximises the number of goals achieved (by their hard deadline), and (b), where the number of goals achieved is the same, minimises the total tardiness (i.e, the difference between the time goals are achieved and their preferred achievement time).

We use *goal-plan trees* to represent the relations between goals, plans and actions, and to reason about the interactions between intentions. The root of a goal-plan tree is a top-level goal (goal-node), and its children are the plans that can be used to achieve the goal (plan-nodes). Plans may in turn contain subgoals (goal nodes), giving rise to a tree structure representing all possible ways an agent can achieve the top-level goal. As in [6, 4] goal-plan trees may contain actions in plans (action nodes). We assume that each top-level goal is associated with a preferred achievement time and deadline, and each action is associated with the expected execution time of the action.

The agent's current intentions are represented by a set of pairs, $I = \{(T_1, c_1), \ldots, (T_n, c_n)\}$, where each T_i is a goal-plan tree corresponding to a top-level goal of the agent, and c_i is the *current step* (primitive action or a subgoal) of T_i. The *next step* of a goal-plan tree is the step after the current step. For example, if the current step is a subgoal, advancing the current step involves choosing a plan for the subgoal, and setting the next step to be the first action of the plan. If the current step is a primitive action, the next step is the basic action or subgoal following that action in the same plan. An intention is *progressable* if its current step does not point to an action that is

[1] University of Nottingham, email: {yvy | bsl}@cs.nott.ac.uk
[2] RMIT University, email: john.thangarajah@rmit.edu.au

currently executing, and the next-step (step after the current step) of the intention is either an action whose precondition holds and its pre- and postcondition does not conflict with the pre- or postcondition of any currently executing action, or a sub-goal for which there is at least one applicable plan in the current state.

If there is at least one progressable intention at the current deliberation cycle, the S_R algorithm is invoked to determine which intention to progress (which current-step pointer to advance). S_R is based on Single-Player Monte-Carlo Tree Search (SP-MCTS) [1], a best-first search in which pseudorandom simulations are used to guide expansion of the search tree. Each node in the search tree represents a possible interleaving of steps from the goal-plan trees in I, and records the state of the agent's environment resulting from the execution of this interleaving, the estimated time at which that state is achieved, the current-step of each intention and the completion time for the goals achieved on the path to this node. Edges represent the selection of a plan for a subgoal or the execution of primitive action in a plan.

Starting from the current step in each intention and the current environment, S_R iteratively builds a search tree. Each iteration consists of four phases. In the *selection* phase, a leaf node, n_e, representing a non-terminal state $s(n_e)$ is selected for expansion using a modified version of Upper Confidence bounds applied to Trees (UCT) [1]. In the *expansion* phase, n_e is expanded by adding child nodes representing the environment state (and the new current step of each intention) resulting from executing each next step of a goal-plan tree that is executable in state $s(n_e)$. Each child node therefore corresponds to a different choice of which intention to progress at this cycle. In the *simulation* phase, a developer-specified number of simulations are performed from a randomly selected child node, n_s, to determine the maximum number of goals that can be achieved from n_s, v_g, and the total tardiness of the interleaving, v_t. Starting in the environment represented by n_s, a next step of a goal-plan tree that is executable in $s(n_s)$ is randomly selected, and the environment and the current step pointer updated. This process is repeated until a terminal state is reached in which no next steps can be executed or all top-level goals are achieved. The value of the simulation is taken to be the values of v_g and v_t in the terminal state. Finally, in the *back-propagation* phase, v_g and v_t are back-propagated from n_s to all nodes on the path to the root node n_0 to inform selection at the next iteration.

After a developer-specified number of iterations, the step that leads to the best child of the root node is returned for execution at this deliberation cycle. S_R selects the best step using $<_{g,t}$, i.e., first on the number of top-level goals achieved, and, where two or more steps result in the achievement of the same number of goals, it prefers steps that also minimise total tardiness.

4 EVALUATING S_R

To evaluate the ability of our approach to schedule intentions with deadlines, we compared the performance of S_R with AgentSpeak(RT) in a simple Blocks World scenario in which an agent must pick blocks and stack them to build towers. Each top-level goal involves picking up two blocks from different locations and building a tower at a specified location. Each top-level goal has a preferred achievement time and deadline, specifying the times by which the task should and must be completed. In the experiments reported below, we used an environment with 30 cells. Initially, 20 randomly selected cells contain a block (numbered from 1 to 20), and the remaining 10 cells (the locations of the towers) are empty. The actions of picking up a block and stacking a block require 100 time units. The time required by a move action is determined by the distance the agent has to move (moving from one cell to another requires 100 time units). For simplicity, we assume the blocks and tower positions are located on a straight line, and the number of blocks the agent can carry is unlimited. The Blocks World scenario is very simple, but representative of a large class of real-world problems in which an agent must collect and use resources in order to achieve goals.

In our experiment, we used 50 sets of 10 goals. Each goal involves collecting two specified blocks and stacking them in a specified empty cell (the block ids and tower positions are different for each goal). Three goals are given initially, and the remaining top-level goals are posted every 1650 time units. As the average time for achieving a top-level goals is 3300 time units, posting a new goal every 1650 time units means that, typically, the agent must achieve more than one goal in parallel. We varied the deadline d for each top-level goal from 7500 to 30000, and the preferred achievement time p was 2500 time units less than the deadline.

Table 1: Goals achieved and tardiness with decreasing deadline

		S_R		AS(RT)	
d	p	#Goals	Tardiness	#Goals	Tardiness
30000	27500	10	0	10	0
15000	12500	10	0	6.96	2386
10000	7500	10	2340	5.38	3195
7500	5000	9.34	3874	5.04	3220

The results are shown in Table 1. As can be seen, as the deadline decreases, the performance of AgentSpeak(RT) declines rapidly. In contrast, S_R was able to achieve at least 9.34 goals on average, though with tighter deadlines, tardiness increases significantly.

5 FUTURE WORK

S_R takes deadline and preferred achievement time of goals into account when choosing which intention to progress. However, while the choice of action at each deliberation cycle is based on the current state of the agent's environment at that cycle, the simulation of the possible outcomes of actions assumes a static environment. In future work we plan to investigate the incorporation of simple environment models to allow the prediction of likely environment changes during simulation, and evaluate S_R in dynamic environments.

REFERENCES

[1] Maarten P. D. Schadd, Mark H. M. Winands, Mandy J. W. Tak, and Jos W. H. M. Uiterwijk, 'Single-player Monte-Carlo tree search for SameGame', *Knowledge-Based Systems.*, **34**, 3–11, (2012).

[2] Konstantin Vikhorev, Natasha Alechina, Rafael Bordini, and Brian Logan, 'An operational semantics for AgentSpeak(RT) (preliminary report)', in *Proceedings of the Ninth International Workshop on Declarative Agent Languages and Technologies (DALT 2011)*, pp. 79–94, (May 2011).

[3] Konstantin Vikhorev, Natasha Alechina, and Brian Logan, 'Agent programming with priorities and deadlines', in *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pp. 397–404, (May 2011).

[4] 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 (AAMAS 2016)*, pp. 1227–1235, (May 2016).

[5] 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*, (August 2014).

[6] Yuan Yao, Brian Logan, and John Thangarajah, 'Robust execution of BDI agent programs by exploiting synergies between intentions', in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pp. 2558–2564, (February 2016).