ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-1133

SP-MCTS-based Intention Scheduling for BDI Agents

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

1 INTRODUCTION

Arguably the dominant paradigm in agent development is the *Belief-Desire-Intention* (BDI) model [2]. In BDI-based agent programming languages, the behaviour of an agent is specified in terms of beliefs, goals, and plans. *Beliefs* represent the agent's information about the environment (and itself). *Goals* represent desired states of the environment the agent is trying to bring about. *Plans* are the means by which the agent can modify the environment in order to achieve its goals. Plans may include sub-goals, and each sub-goal is in turn achieved by some other plan. The set of plans is pre-defined by the agent developer, and, together with the agent's initial beliefs and goals, form the program of the agent. The execution of a BDI agent consists of a repeated cycle of: updating the agent's beliefs and goals to reflect the current state of the environment, selecting plans to achieve the agent's current (sub)goals based on the agent's current beliefs, and finally executing one or more steps of the agent's currently intended plans. For each top-level goal, the agent selects a plan which forms the root of an intention. If the next step in an intention is a subgoal, a sub-plan is selected to achieve the subgoal and pushed onto the intention, and the steps in the sub-plan are then executed and so on. This process of repeatedly choosing and executing plans is referred to the agent's *deliberation cycle*.

In many BDI agent architectures, the execution of the plans comprising the agent's intentions is interleaved, e.g., when execution of a plan in one intention reaches a subgoal, the agent may switch to executing a plan in a different intention. Interactions between different intentions may result in conflicts, e.g., where the execution of a plan in one intention makes the execution of another plan in the same or another intention impossible or renders a goal unachievable. The task of anticipating and avoiding such conflicts is generally left to the agent developer. However, the run-time plan selection characteristic of BDI agents makes it difficult to anticipate all the ways in which an agent program may be executed, and harder to ensure that conflicts cannot arise. Ideally, the agent itself should be able to reason about possible conflicts between its intentions, and schedule their execution so as to avoid conflicts.

In this paper, we present a novel approach to intention scheduling for BDI agents based on Single-Player Monte-Carlo Tree Search (SP-MCTS) that avoids conflicts between intentions. We evaluate the performance of our approach and compare it to a previous approach to intention scheduling based on summary information [4]. Our preliminary experimental results indicate that our approach performs at least as well as approach to scheduling intention using summary information.

[1] School of Computer Science, University of Nottingham, Nottingham UK, email: {yvy,bsl}@cs.nott.ac.uk
[2] School of Computer Science & Information Technology, RMIT University, Melbourne, Australia, email: john.thangarajah@rmit.edu.au

2 SCHEDULING INTENTIONS

The plans to achieve a top-level goal of an agent naturally from a tree structure termed a *goal-plan tree* [5, 4]. The root of a goal-plan tree is the top-level goal, and its children are the plans that can be used to achieve the goal. In general, there may be several alternative plans to achieve a goal, hence, the child plan nodes are viewed as 'OR' nodes. On the other hand, plan execution involves performing all the steps in the plan. Thus, the children of a plan node are viewed as 'AND' nodes.

The execution of an intention to achieve a top-level goal amounts to choosing a path through the corresponding goal-plan tree — a sequence of plans, subgoals and sub-plans, ... that, when executed, achieves the top-level goal. In general, to achieve a (sub)goal or successfully execute a plan, certain conditions must hold in the agent's environment. We distinguish between *preconditions* (conditions that must be true in order to execute a plan) and *in-conditions* (conditions that must hold during the achievement of a goal or execution of a plan). If an in-condition becomes false during the achievement of a goal or execution of a plan, the goal or plan is dropped with failure. For example, a plan to buy groceries may have the in-condition 'at-store', and a precondition 'have-money'. If an agent were to leave the store during the execution of the plan, the plan would become unexecutable. Similarly, executing a plan or achieving a goal has *effects* in the environment (in addition to achieving the goal itself). Note that, as in In [5, 4], we abstract away from actions. The pre-conditions and effects of actions are captured at the plan level rather than at the level of the individual actions comprising the plan.

When executing a set of intentions, the interleaving of plans in different goal-plan trees can give rise to conflicts between plans and goals, e.g., if the the effects of a plan $P1$ in one goal-plan tree makes the precondition of a plan $P2$ in another tree false before $P2$ is executed. The *scheduling problem* is the problem of choosing a path through each goal-plan tree corresponding to one of the agent's current intentions, and an interleaving of the steps in these paths which ensures that the preconditions of each plan on a path are true when the plan begins execution, and that any in-conditions required by a goal or plan are true during the achievement of the goal or execution of the plan.

3 SINGLE-PLAYER MONTE-CARLO TREE SEARCH-BASED SCHEDULING

Our approach to intention scheduling is based on Single-Player Monte-Carlo Tree Search (SP-MCTS) [3]. SP-MCTS is a best-first search in which pseudorandom simulations are used to guide expansion of the search tree. It was originally developed to solve single-player puzzles (games against the environment), e.g. SameGame [3], however it has also been used successfully to solve reentrant scheduling problems [1]. SP-MCTS is an anytime algorithm — it iteratively

builds a search tree until some pre-defined computational budget (typically time, memory or number of expansions) is reached. The algorithm then halts, and the best performing action is returned. Each node in the search tree represents a state of the problem domain, and also records information that is used to select nodes for expansion. Edges represent actions leading to a subsequent state of the current node.

In the remainder of this section, we describe our approach to intention scheduling based on SP-MCTS. The input to the scheduling algorithm is a set of goal-plan trees T representing the current intentions of the agent, together with a set of condition variables C representing the current state of the agent's environment. Each goal-plan tree in T is associated with a pointer to the 'current step' in the tree, i.e., the point execution has reached in that goal-plan tree. The current step may be a plan or a subgoal, and the path from the the root of the goal-plan tree to the current step represents the choices made (and steps executed) so far in achieving the top-level goal. When an agent adopts an intention for a new top-level goal G, the current step of the goal-plan tree for G is initially set to G itself. The output of the scheduling algorithm is a next step of a goal-plan tree in T to be executed at the current deliberation cycle. The step returned is 'best' in the sense that the execution of no other next step of a goal-plan tree in T results in the achievement of a larger number of top-level goals.

Each node in the SP-MCTS search tree records the current step in each goal plan tree, and the state of the agent's environment and any active in-conditions resulting from the interleaved execution of the previous steps in each goal-plan tree on the path to this node. In addition, each node contains a record of the number of times it has been visited, the highest value simulation from the node, and the sum of all simulations from the node (see below).

Each iteration of the algorithm consists of 4 phases: selection, expansion, simulation and back-propagation.

Selection: in the selection phase, a leaf node, n_e, of the SP-MCTS search tree is selected for expansion. A node may be expanded if it represents a non-terminal state (a state in which it is possible to execute the next step of an intention). Nodes are selected using a modified version of Upper Confidence bounds applied to Trees (UCT) [3], which models the choice of node as a k-armed bandit problem. Starting from the root node, we recursively follow child nodes with highest UCT value until a leaf node is reached.

Expansion: In the expansion phase, the selected SP-MCTS node n_e is expanded by adding child SP-MCTS nodes representing the states reachable by performing each next step of an intention possible in n_e. Each child node therefore corresponds to a different choice of which intention to execute at this deliberation cycle. A next step is possible if the precondition of the step is true in the environment of n_e and the effects of the step do not violate the in-conditions that are active at n_e. Finally, one of the newly created child nodes, n_s, is selected at random for simulation in the next phase.

Simulation: in the *simulation* phase, the 'value' of the SP-MCTS node n_s is estimated by performing a series of pseudorandom simulations. Staring in the state representing by n_s, we randomly select a next step of an intention that can be executed in the environment of n_s and execute it, updating both the environment with its effects and the 'current step' of the selected intention. We keep doing this until no next steps can be executed or all top-level goals are achieved. The resulting value of the simulation is taken to be the number of top-level goals achieved by the random interleaving of the next steps of the intentions. We repeat the simulation a number of times (e.g., 100 times) and take the highest result as the value for this node.

Back-propagation: the last step is to back-propagate the simulation results for n_s to all nodes on the path to the root node and update their statistics. Any ancestor node with a value less than the value of n_s has their value replaced by the value of n_s.

4 EVALUATION

To evaluate the performance of SP-MCTS-based intention scheduling, we compared its performance to that of summary information-based scheduling proposed by Thangarajah et al [4]. In summary information-based scheduling, information is propagated up each goal-plan tree to allow reasoning about interactions between trees. For example, if a goal has two possible plan choices, and they both bring about an effect e, then it is possible to infer that the effect e will definitely occur as a result of achieving the goal, and that any other goal or plan in another intention that brings about $\neg e$ may cause a conflict. We generated goal-plan trees similar to the 'high level of interference' case in [4]. Each experiment involved scheduling 20 goal-plan trees, and we ran 50 experiments. The SP-MCTS scheduler was configured to execute 1000 iterations and perform 100 simulations at each iteration. In all 50 trials, SP-MCTS was able to schedule execution of the goal-plan trees so as to achieve all the top-level goals. For similar trees, scheduling using summary information is also able to achieve all 20 goals. This suggests that SP-MCTS is at least no worse than scheduling using summary information.

5 FUTURE WORK

As with summary information-based scheduling [5, 4] our current SP-MCTS scheduler abstracts away from actions in plans — the preconditions and effects of actions are captured at the plan level rather than at the level of the individual actions comprising a plan. While this reduces the number of points at which the execution of goal-plan trees can be interleaved, in some cases it may be necessary to interleave the execution of intentions at the action level to achieve the agent's goals. In future work, we plan to investigate the application of SP-MCTS to scheduling goal-plan trees in which actions are explicitly represented. We believe that SP-MCTS will outperform summary information-based scheduling when scheduling at the action level.

REFERENCES

[1] Shimpei Matsumoto, Noriaki Hirosue, Kyohei Itonaga, Nobuyuki Ueno, and Hiroaki Ishii, 'Monte-Carlo tree search for a reentrant scheduling problem', in *40th International Conference on Computers and Industrial Engineering (CIE)*, pp. 1–6, (2010). IEEE.

[2] Anand S. Rao and Michael P. Georgeff, 'Modeling rational agents within a BDI-architecture', in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 473–484, (April 1991). Morgan Kaufmann.

[3] 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).

[4] John Thangarajah and Lin Padgham, 'Computationally effective reasoning about goal interactions', *Journal of Automated Reasoning*, **47**(1), 17–56, (2011).

[5] John Thangarajah, Lin Padgham, and Michael Winikoff, 'Detecting & avoiding interference between goals in intelligent agents', in *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 721–726, (2003). Morgan Kaufmann.