Progressing Intention Progression: A Call for a Goal-Plan Tree Contest

Brian Logan
University of Nottingham
United Kingdom
bsl@cs.nott.ac.uk

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

Neil Yorke-Smith
American University of Beirut
Lebanon
nysmith@aub.edu.lb

ABSTRACT

User-supplied domain control knowledge in the form of hierarchically structured Goal-Plan Trees (GPTs) is at the heart of a number of approaches to reasoning about action. Reasoning with GPTs connects the AAMAS community with other communities such as automated planning, and forms the foundation for important reasoning capabilities, especially *intention progression* in Belief-Desire-Intention (BDI) agents. Research on GPTs has a long history but suffers from fragmentation and lack of common terminology, data formats, and enabling tools. One way to address this fragmentation is through a competition. Competitions are increasingly being used as a means to foster research and challenge the state of the art. For example, the AAMAS conference has a number of associated competitions, such as the Trading Agent Competition, while agent research is showcased at competitions such as RoboCup. We therefore issue a call for a *Goal-Plan Tree Contest*, with the ambition of drawing together a community and incentivizing research in intention progression.

Keywords

intention progression; goal-plan trees; competition

1. INTRODUCTION

A key problem for an agent with multiple, possibly inconsistent, goals is: 'what should I do next'? What to do next can be formalized as the *intention progression problem* (IPP): what means (i.e., plan) to use to achieve a given (sub)goal; and which of the currently adopted plans (i.e., intentions), to progress at the current moment. An important capability of an intelligent agent is the ability to progress multiple intentions in parallel, by interleaving the steps in each intention to provide the best outcome for the agent. This problem is both central to agent reasoning and complex in its nature. For example, 'best outcome' may have different definitions depending on the application, while goals and plans may conflict given the resources available.

We call for a *Goal-Plan Tree Contest* to incentive research around the multi-faceted problem of intention progression. Competitions in specific areas of Computer Science are now familiar as a means to foster research. They range from 'grand challenges' (e.g., those from DARPA) to commercial contests (e.g., Netflix Challenge), to more academic contests (e.g., DIMACS, IPC, TAC) and competitions with long-term objectives (e.g., RoboCup). However, none of these existing competitions addresses intention progression as relevant for intelligent agents.

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

2. BACKGROUND

We delineate the Intention Progression Problem in the context of BDI agents that are characterised by having a *library* of (parameterized) plans. An agent chooses the most appropriate set of plans and how to execute them in order to achieve a set of goals – or more generally to maximise some objective function. The steps of a plan may contain (sub)goals which are in turn achieved by other plans.

This goal-plan decomposition is essentially the decomposition of high level tasks into smaller tasks, and is similar to *Hierarchical Task Network* (HTN) planning [4]. The difference between classical HTN planning and situated BDI agent planning, is that the former assumes that the set of (achievement) goals and the environment are static, while the paradigm of BDI agents is one of interleaved replanning and execution in a dynamic environment. This dynamism arises because of exogenous 'natural' events (such as weather) and the actions of other agents, and perhaps because of limitation in the agent's perception, its actions (e.g., action failure), or both.

These differences mean that BDI agents focus on the next step to progress while HTN planners aim to produce a complete solution that is conflict-free. HTN planning is complete when the resulting task network consists only of primitive tasks, together with a set of constraints that ensure the preconditions of a task hold when the task is executed. The primitive task network is then linearised yielding a fixed sequence of actions, which makes it difficult to handle exogenous events. No actions are executed until the plan is complete. As the process of ensuring solutions are conflict-free is costly (depending on the branching factor of the task hierarchy and the number of interacting conditions in each task in the network [7]), there can be a significant delay in responding to new goals or changes in the environment. In contrast, in the BDI approach, the focus is on timely (soft real-time) selection of which intention to progress.

We note that HTN planners could be configured to behave more like BDI agents, by decomposing high level tasks by a single step rather than attempting to find a complete conflict-free solution, and we hope that the Goal-Plan Tree

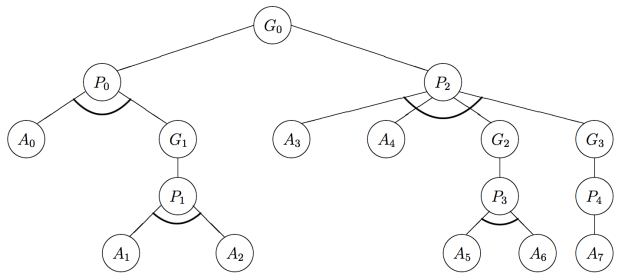

Figure 1: Example goal-plan tree

Contest would foster hybridisation with innovative techniques from the HTN planning community.

The IPP involves not only planning and replanning, but also online scheduling (e.g., see [20]). The agent has limited resources, goals may have deadlines, and plan steps may interfere with each other: hence the agent must decide in what order to execute plan steps. As with plan selection, action scheduling is a dynamic process subject to change according to the agent's desires and environmental changes.

3. GOAL-PLAN TREES AND INTENTIONS

Abstract agent programming languages such as AgentSpeak and CAN define intentions and their progression in terms of partially-executed programs; the agent configuration progresses by executing a single step of one of the programs [13]. However, which step of which intention is selected is a black box. To allow reasoning about intention progression, we instead define intentions and their progression in terms of goal-plan trees.

A BDI agent program consists of a set of pre-defined plans that are used to achieve the agent's goals. Each plan consists of steps which are either basic actions or sub-goals. Each sub-goal is in turn achieved by some other plan. This relationship is naturally represented as a tree structure termed a *goal-plan tree* [3, 21, 4, 20]. The root of a GPT is a top-level goal (goal-node), and its children are the plans that can be used to achieve the goal (plan-nodes). Usually there are several alternative plans to achieve a goal: hence, the child plan-nodes are viewed as 'OR' nodes. By contrast, plan execution involves performing all the steps in the plan: hence, the children of a plan-node are viewed as 'AND' nodes. As in Yao et al. [29, 26], we consider goal-plan trees in which plans may contain primitive actions in addition to sub-goals.

Figure 1 shows a simple goal-plan tree. The top-level goal G_0 can be achieved by either of the two plans P_0 or P_2 ('OR' nodes). The plan P_0 involves performing the action A_0 and achieving the sub-goal G_1 ('AND' nodes), while plan P_2 involves executing the actions A_3 and A_4 and achieving the sub-goals G_2 and G_3 ('AND' nodes) and so on.

Formally, we define a goal-plan tree by the BNF in Figure 2 [25]. A *GoalType* is a template for a goal. A *GoalInstance* is created when an agent chooses to pursue a particular instance of goal-type. Similarly, a *PlanType* is a template for a plan, and a *PlanInstance* is created when the agent executes a particular plan. An *ActionType* is a template for an action, and an *ActionInstance* is created when a particular action is chosen for execution by the agent. *GoalTypeName*, *PlanTypeName* and *ActionTypeName* are labels that indicate the type of the goal, the plan or the action respectively. *Plans* represents the set of plan-types that may be used to satisfy a goal of the corresponding *GoalType*. We assume that it is possible to generate a GPT corresponding to each top-level goal that can be achieved by an agent program.[1]

⟨GoalType⟩ ::= ⟨GoalTypeName⟩ ⟨Precondition⟩ ⟨In-condition⟩ ⟨Postcondition⟩ ⟨Plans⟩
⟨GoalTypeName⟩ ::= ⟨Label⟩
⟨Plans⟩ ::= ⟨PlanTypeName⟩ (, ⟨PlanTypeName⟩)*

⟨PlanType⟩ ::= ⟨PlanTypeName⟩ ⟨Precondition⟩ ⟨In-condition⟩ ⟨Postcondition⟩ ⟨PlanBody⟩
⟨PlanTypeName⟩ ::= ⟨Label⟩
⟨PlanBody⟩ ::= ⟨ExecutionStep⟩ (; ⟨ExecutionStep⟩)*
⟨ExecutionStep⟩ ::= ⟨ActionTypeName⟩ | ⟨GoalTypeName⟩ | (⟨ExecutionStep⟩ ‖ ⟨ExecutionStep⟩)

⟨ActionType⟩ ::= ⟨ActionTypeName⟩ ⟨Precondition⟩ ⟨In-condition⟩ ⟨Postcondition⟩
⟨ActionTypeName⟩ ::= ⟨Label⟩

⟨Precondition⟩ ::= ϵ | ⟨Condition⟩ (, ⟨Condition⟩)*
⟨In-condition⟩ ::= ϵ | ⟨Condition⟩ (, ⟨Condition⟩)*
⟨Postcondition⟩ ::= ϵ | ⟨Condition⟩ (, ⟨Condition⟩)*
⟨Condition⟩ ::= ⟨Statement⟩ | *NOT* ⟨Statement⟩
⟨Statement⟩ ::= *string* | ⟨Variable⟩ = ⟨Value⟩
⟨Label⟩ ::= *unique string*
⟨Variable⟩ ::= *unique string*
⟨Value⟩ ::= *string*

⟨GoalInstance⟩ ::= ⟨InstanceName⟩ ⟨GoalType⟩
⟨PlanInstance⟩ ::= ⟨InstanceName⟩ ⟨PlanType⟩
⟨ActionInstance⟩ ::= ⟨InstanceName⟩ ⟨ActionType⟩
⟨InstanceName⟩ ::= ⟨Label⟩

Figure 2: BNF Syntax of GPTs with actions [25]

3.1 Intention Progression

Following Yao et al. [28], we define intentions and the IPP in terms of goal-plan trees, as follows.

The intentions of an agent are represented by a set T of goal-plan trees, where the root goal g_i of each GPT $t_i \in T$ corresponds to a top-level goal of the agent. The progression of an intention to achieve a top-level goal g_i amounts to traversing a path through the goal-plan tree t_i. The path specifies a sequence of plans, actions, sub-goals and sub-plans that, if executed successfully, will achieve g_i. The execution of an agent program thus corresponds to an interleaving of paths through each of the GPTs in T.

More precisely, let $T = \{t_1, \ldots, t_n\}$ be the set of goal-plan trees corresponding to the agent's intentions, and $S = \{s_1, \ldots, s_n\}$ be a set of *pointers to the current step* of each intention. The current step s_i of a goal-plan tree t_i is either a primitive action or a sub-goal, and is initially set to the root goal of t_i, g_i. We define $next(s_i)$ as the step of t_i following the current step s_i. If s_i is a primitive action, then $next(s_i)$ is the primitive action or sub-goal following s_i in the same plan, or, if s_i is the last action in a plan, $next(s_i)$ is the next primitive action or sub-goal in the parent plan of the current

[1]Note that the set of goal-plan trees corresponding to an agent program can be computed offline, from the code of the program itself. Some approaches incorporate online planning [5, 13] to allow dynamic extension or customization of the plan library.

plan. If s_i is a sub-goal, advancing the current step involves choosing a plan for the sub-goal, and setting $next(s_i)$ to be the first action or sub-goal of the selected plan. A current step s_i is *progressable* if $next(s_i)$ points to a step which is either an action whose precondition holds, or a sub-goal for which at least one plan is applicable in the current state of the agent and environment.

We can then define a function to progress the current step s_i of an intention as follows:

$$prog(S, s_i) = S' \text{ s.t. } S' = S \setminus \{s_i\} \cup \{next(s_i)\}.$$

Note that *prog* is a partial function; it is not defined if s_i is not progressable. (In practice, attempting to, e.g., execute an action whose precondition does not hold would typically result in an error condition; we do not consider this here.)

At each deliberation cycle, the agent has a set of intentions T corresponding to its current top-level goals, and their associated current step pointers S. Note that T may vary from cycle to cycle, e.g., if the agent adopts a new top-level goal, or the progression of an intention at the previous cycle resulted in the achievement of a goal.

The *intention progression problem* can now be stated as that of specifying a policy $\Pi(T, S)$, that, at each deliberation cycle, selects a current step $s_i \in S$ to progress so as to maximise some overall utility function $U^{\Pi}(T, S)$. Formally:

$$\Pi(T, S) = s_i \text{ and } \nexists\, \Pi' \text{ s.t. } U^{\Pi'}(T, S) > U^{\Pi}(T, S).$$

Deciding on an appropriate utility function (or set of utility functions) is a key task in defining the rules of a Goal-Plan Tree Contest. For example, the utility function could be such that progressing s_i would achieve more goals than progressing any other step, or, if they achieve the same number of goals, that progressing s_i results in an interleaving that is at least as fair, or achieves more high priority goals, or more goals by their deadline, etc.

3.2 Reasoning over GPTs

Approaches to reasoning over GPTs include recursive tree algorithms [20, 6], Petri nets [15], constraint programming [14], and stochastic approaches based on sampling [29, 26, 28]. However, despite the long history of research on GPTs, the state of the art in solving the IPP in current BDI agent programming languages is limited to simplistic (e.g., round robin) approaches to intention scheduling, or involves significant effort by the programmer to hand-craft a solution for the application at hand. While there are a number of promising automated approaches in the literature, current work is fragmented: e.g., using GPT coverage and overlap [22], decision-theoretic approaches [1], and Monte-Carlo Tree Search [27].

Further aspects and consequences of the IPP, include positive and negative interactions between the steps in intentions [20, 29], deadlines and temporal reasoning [10], and aborting or suspending goals or plans [16, 17]. The 'best outcome' may have different definitions depending on the application – for example, the number of goals achieved, the time taken to achieve the goals, and so on [23, 18, 19] – and depend on the timeframe considered and amount of predictive lookahead. Extensions to GPT include first-order representations [10] and multi-agent planning [2].

As in HTN planning [3, 9], GPTs can have information associated with nodes, such as costs or deadlines, and this information can be propagated through the tree. Information can be summarized and the summaries used as a basis for the agent's reasoning [20, 19]. Similar approaches are found elsewhere in the agents literature, such as in the TAEMS multi-agent coordination framework [8], indicating the generality of the GPT representation.

4. PROPOSAL FOR A COMPETITION

We propose the inaugural Goal-Plan Tree Contest to be held in 2018, co-located with AAMAS 2018.

4.1 Rationale

Competitions are increasingly being used as a mechanism to incentive research in computer science. A partial list of competitions familiar to AAMAS participants includes:

- RoboCup (Soccer, Rescue, @Home, etc) – since 1997
- International Planning Competition (IPC) – since 1998
- Trading Agent Competition – since 2000
- Competition on Knowledge Engineering for Planning and Scheduling – since 2005
- Agent Reputation and Trust Competition – 2006–08
- Power Trading Agent Competition – since 2012

In addition to these largely academic competitions, there are various Grand Challenges and contests sponsored by DARPA, other funding bodies, and commercial entities. None of these existing competitions addresses the Intention Progression Problem.

Competitions serve at least four purposes: 1) providing common terminology, data formats, and problem instances; 2) comparing approaches in a scientific manner; 3) fostering software engineering and robustness of approaches; and 4) raising the academic and public profile of an area.

The specific rationale for Goal-Plan Tree Contest is the importance of GPT representation, not least for the IPP, contrasted with the current fragmentation of research on GPTs; the lack of common terminology, data formats, and enabling tools; and the need to focus on a motivating challenge, which we identify as intention progression.

4.2 Tasks

Organising the competition involves five main tasks:

1. **Common data format.** The first task to is to define a data format for GPTs and the execution environment sufficient to capture problems of interest. We propose an XML-based format.

2. **Problems.** The second task is to define the precise variants of Intention Progression Problem that entrants will address, including specifying the utility function U^{Π}. At a high level, the problem is to specify, within a given computation time limit, what decisions the agent should take over its intentions. The decisions can include progressing an intention step, adopting a new intention, changing the plan for an intention, etc. We envision potentially two dimensions of problems: 1) the nature of the environment (e.g., deterministic vs. non-deterministic), and 2) the degree and type of dynamism (e.g., environmental, arrival of new goals).

3. **Instances.** The third task is to collect potential problem instances. The recent *GenGPT* of Yao [24] is the first publicly-available open source GPT generator. GenGPT generates synthetic (i.e., random) GPTs of arbitrary depth/width. Plans may contain both sub-goals and (deterministic or nondeterministic) actions, and a simple form of parallel construct is supported (all actions and/or sub-goals in a plan are executed in parallel). It is also possible to control the number of conflicts and interactions between trees. The Contest will however also require non-random problem instances. Sources for extracting such instances include published examples of BDI agent programs; solutions to the Multiagent Programming Contest, etc.; instantiations of, e.g., Prometheus designs [11]; translations of HTN planning domains from, e.g., the IPC (see, for example, [12]).

4. **Call for Entrants.** The fourth task is to announce the Contest, solicit entrants, and prepare the contest framework. To ensure fairness, we envisage that code submitted by entrants will be run on a virtual machine.

5. **Scoring.** The fifth task is to run the entries on the instances, using automated scoring mechanisms. The results will then be announced, perhaps with an associated prize. The problems used and source code for all entries would be published on the Contest website.

We envision that a new workshop, or session within an established AAMAS workshop such as EMAS, could provide a forum for entrants to discuss the techniques embodied in their Contest entries, and for the announcement of results.

5. CALL TO ACTION

This paper provided a crisp characterisation of the Intention Progression Problem, a reasoning task that is central to intelligent agents, especially those in the BDI tradition, and one that connects with the HTN planning community. To foster research on the IPP, we call for a Goal-Plan Tree Contest to develop the best 'intention progression' mechanism that provides the most optimal outcome for the agent. Our ambition is to draw together a community and incentivize research.

We welcome from the AAMAS community: 1) expressions of interest and support, 2) contributions of ideas, and 3) volunteers to help organize the inaugural contest in 2018.

Acknowledgements. We thank the AAMAS 2017 reviewers for their encouraging comments.

REFERENCES

[1] R. H. Bordini, A. L. C. Bazzan, and et al. AgentSpeak(XL): Efficient intention selection in BDI agents via decision-theoretic task scheduling. In *Proc. of AAMAS'02*, pages 1294–1302, 2002.

[2] S. Brahimi, R. Maamri, and Z. Sahnoun. Partially centralized approach for dynamic hierarchical-plans merging. *Neurocomputing*, 146:187–198, 2014.

[3] B. J. Clement and E. H. Durfee. Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proc. of AAAI'99*, pages 495–502, 1999.

[4] B. J. Clement, E. H. Durfee, and A. C. Barrett. Abstract reasoning for planning and coordination. *J. of Artificial Intelligence Research*, 28:453–515, 2007.

[5] L. de Silva, S. Sardiña, and L. Padgham. First principles planning in BDI systems. In *Proc. of AAMAS'09*, pages 1105–1112, 2009.

[6] L. de Silva, S. Sardiña, and L. Padgham. Summary information for reasoning about hierarchical plans. In *Proc. of ECAI'16*, pages 1300–1308, 2016.

[7] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *Proc. of AAAI'94*, pages 1123–1128, 1994.

[8] V. R. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. E. Neiman, R. M. Podorozhny, M. V. N. Prasad, A. Raja, R. Vincent, P. Xuan, and X. Zhang. Evolution of the GPGP/TÆMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):87–143, 2004.

[9] A. Lotem and D. S. Nau. New advances in GraphHTN: Identifying independent subproblems in large HTN domains. In *Proc. of ICAPS'00*, pages 206–215, 2000.

[10] D. N. Morley, K. L. Myers, and N. Yorke-Smith. Continuous refinement of agent resource estimates. In *Proc. of AAMAS'06*, pages 858–865, 2006.

[11] L. Padgham, J. Thangarajah, and M. Winikoff. Prometheus design tool. In *Proc. of AAAI'08*, pages 1882–1883, 2008.

[12] S. Sardiña, L. de Silva, and L. Padgham. Hierarchical planning in BDI agent programming languages: A formal approach. In *Proc. of AAMAS'06*, pages 1001–1008, 2006.

[13] S. Sardiña and L. Padgham. A BDI agent programming language with failure handling, declarative goals, and planning. *Autonomous Agents and Multi-Agent Systems*, 23(1):18–70, 2011.

[14] P. H. Shaw and R. H. Bordini. An alternative approach for reasoning about the goal-plan tree problem. In *Proc. of LADS'10*, pages 115–135, 2010.

[15] P. H. Shaw, B. Farwer, and R. H. Bordini. Theoretical and experimental results on the goal-plan tree problem. In *Proc. of AAMAS'08*, pages 1379–1382, 2008.

[16] J. Thangarajah, J. Harland, D. N. Morley, and N. Yorke-Smith. Aborting tasks in BDI agents. In *Proc. of AAMAS'07*, pages 8–15, 2007.

[17] J. Thangarajah, J. Harland, D. N. Morley, and N. Yorke-Smith. Operational behaviour for executing, suspending, and aborting goals in BDI agent systems. In *Proc. of DALT'10*, pages 1–21, 2010.

[18] J. Thangarajah, J. Harland, D. N. Morley, and N. Yorke-Smith. Towards quantifying the completeness of BDI goals. In *Proc. of AAMAS'14*, pages 1369–1370, 2014.

[19] J. Thangarajah, J. Harland, and N. Yorke-Smith. Estimating the progress of maintenance goals. In *Proc. of AAMAS'15*, pages 1645–1646, 2015.

[20] J. Thangarajah and L. Padgham. Computationally effective reasoning about goal interactions. *Journal of Automated Reasoning*, 47(1):17–56, 2011.

[21] J. Thangarajah, L. Padgham, and M. Winikoff.

Detecting and avoiding interference between goals in intelligent agents. In *Proc. of IJCAI'03*, pages 721–726, 2003.

[22] J. Thangarajah, S. Sardiña, and L. Padgham. Measuring plan coverage and overlap for agent reasoning. In *Proc. of AAMAS'12*, pages 1049–1056, 2012.

[23] M. B. van Riemsdijk and N. Yorke-Smith. Towards reasoning with partial goal satisfaction in intelligent agents. In *Proc. ProMAS'10*, pages 41–59, 2010.

[24] Y. Yao. GenGPT: A package to generate synthetic goal-plan trees. `https://github.com/yvy714/GenGPT`, 2016.

[25] Y. Yao, L. de Silva, and B. Logan. Reasoning about the executability of goal-plan trees. In *Proc. of EMAS'16*, pages 181–196, 2016.

[26] Y. Yao and B. Logan. Action-level intention selection for BDI agents. In *Proc. of AAMAS'16*, pages 1227–1236, 2016.

[27] Y. Yao, B. Logan, and J. Thangarajah. SP-MCTS-based intention scheduling for BDI agents. In *Proc. of ECAI'14*, pages 1133–1134, 2014.

[28] Y. Yao, B. Logan, and J. Thangarajah. Intention selection with deadlines. In *Proc. of ECAI'16*, pages 1700–1701, 2016.

[29] Y. Yao, B. Logan, and J. Thangarajah. Robust execution of BDI agent programs by exploiting synergies between intentions. In *Proc. of AAAI'16*, pages 2558–2565, 2016.