Intention Progression with Maintenance Goals

Extended Abstract

Di Wu
Zhejiang University of Technology
Hangzhou, China
wudi@zjut.edu.cn

Yuan Yao
University of Nottingham
Ningbo, China
yuan.yao@nottingham.edu.cn

Natasha Alechina
Utrecht University
Utrecht, The Netherlands
n.a.alechina@uu.nl

Brian Logan
Utrecht University
Utrecht, The Netherlands
University of Aberdeen
Aberdeen, United Kindom
b.s.logan@uu.nl

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

KEYWORDS

BDI Agents; Intention Progression Problem; Maintenance Goals

ACM Reference Format:
Di Wu, Yuan Yao, Natasha Alechina, Brian Logan, and John Thangarajah. 2023. Intention Progression with Maintenance Goals: Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.

1 INTRODUCTION

One of the key advantages of Belief-Desire-Intention (BDI) agents [7] is their ability to pursue multiple goals in parallel. When multiple goals are pursued at the same time, an agent has to decide which of its intentions should be progressed, and if the next step of the selected intention is a subgoal, the agent also has to decide which plan should be used. These two choices together form the *intention progression problem* [5]. Previous work on the intention progression problem is limited to scheduling achievement goals [8–13, 16]. In addition to achieving certain states, in many applications agents must also *maintain* particular states of the environment, e.g., not running out of power, avoiding collisions, etc. Such goals are termed *maintenance goals*, as they specify a state of the environment an agent should maintain, and maintenance goals are supported by many BDI systems, including Jadex [6] and JAM[4]. Previous approaches to [2] proactively reasoning about maintenance goals are based on summary-information [9]. However, the approach in [2] assumes that preventive measures to maintain a goal do not interact with the agent's other intentions.

In this paper, we present SA_M, a new approach to intention progression for BDI agents with both achievement and maintenance goals which extends *SA* scheduler [15]. We compare the performance of our approach to that of [2]. The results suggest our approach significantly improves the performance of agents with maintenance goals.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2 SCHEDULING MAINTENANCE GOALS

We consider agents with two types of goals: achievement goals and maintenance goals. An achievement goal represents the state of the environment an agent is trying to bring about, A maintenance goal specifies a state of the environment the agent should maintain for a period of time [3], and is defined as follows: $\langle G_m, C_m, Pls, t\rangle$, where G_m is the name of the goal, C_m is the maintenance condition, Pls are the plans that can be used to re-establish C_m, and t is the time after which G_m should be dropped. Unlike achievement goals which simply require the successful execution of a plan, maintenance goals may be implemented in different ways, depending on whether the violation of C_m can be predicted.

If violation of C_m can be reliably predicted given the agent's other intentions, the agent can adopt a plan in Pls **proactively**, i.e., before the violation occurs. For example, if we can predict the battery consumption resulting from the agent's intentions, we can specify as a maintenance condition that the battery level should always be greater than zero. On the other hand, if violation of C_m cannot be reliably predicted, the agent must adopt a plan in Pls **reactively**, i.e., after the violation of C_m occurs. In this case, the maintenance condition C_m typically needs to be more "conservative". For example, if we cannot predict the battery consumption resulting from the agent's intentions, the maintenance condition may specify a minimum battery level, e.g., 20%, that ensures that the agent is always able to return to a recharging point from the location at which the maintenance condition is violated.

3 SA_M SCHEDULER

SA_M extends the *SA* scheduler [15] in explicitly taking account of the agent's reactive and proactive maintenance goals. SA_M is based on Monte-Carlo Tree Search (MCTS) [1] which uses pseudo-random simulations to guide the expansion of the search tree. Edges in the search tree represent a choice of which action to execute in one of the agent's intentions. Each node represents the state of the agent and its environment following the execution of the action. Starting from the root node representing the current state of the agent's intentions I and state of the environment E, SA_M iteratively builds a search tree until a pre-defined computational budget is reached. The scheduler then halts and the best action is returned. The agent's intentions I are represented by a tuple (T, S), where

$T = \{t_1, ..., t_n\}$ is a set of goal-plan trees [8, 9, 14] corresponding to the agent's top-level goals and $S = \{s_1, ..., s_n\}$ is a set of pointers to the current step of each goal-plan tree in T. As with MCTS and *SA*, each iteration of SA_M consists of four main phases: selection, expansion, simulation, and back-propagation.

To support intention progression with maintenance goals, we modified the expansion and the simulation phases of the *SA* scheduler. In particular, we include a mechanism that decides when recovery plans should be considered and how they should be executed. For reactive maintenance goals, recovery plans can only be applied when the corrsponding maintenance condition is violated. That is, the agent can choose either to continue achieving its top-level goals as before, or to execute recovery plans to re-establish the violated maintenance condition. For reactive maintenance goals, recovery plans can only be executed if the corresponding maintenance goals are triggered. However, for proactive maintenance goals, SA_M assumes maintenance goals are always active, i.e., it is always possible to execute a recovery plan for a maintenance goal G_m, even if there is no existing intention to recover G_m.

The recovery plans for maintenance goals may be interleaved with other intentions so as to avoid conflicts or to exploit synergies as in [15, 16]. In the special case when an agent does not have any achievement goals and none of the maintenance conditions are violated, no further action will be taken by the agent. That is, if the agent is not pursuing any goals, then waiting is the best way to maintain the current state.

In SA_M, a terminal state is reached when any of the following situations occur: 1) all achievement goals are achieved and no maintenance condition is violated; 2) all the remaining achievement goals cannot be achieved in the current state; or 3) a maintenance condition cannot be re-established.

The above modifications are used when SA_M expands new nodes in the expansion phase and when randomly simulating the execution of the agent in the simulation phase.

4 EVALUATION

We evaluate the performance of SA_M in the Mars rover scenario from [2]. In the scenario, the environment is a grid of 20×20 cells. The rover can move up, down, left and right, and has achievement goal(s) to visit different locations in the grid. Each movement action consumes 1 unit of battery power, and the rover can return to the depot in the centre of the environment to recharge its battery. In the experiments reported below, we assume the Mars rover starts in the depot cell, all its goals (i.e., where to visit) are randomly generated, and the battery capacity of the rover is 40.

We compare the performance of SA_M for reactive maintenance goals (RMCTS) and proactive maintenance goals (PMCTS), with the reactive (RMG) and proactive (PMG) approaches from [2] for an increasing number of achievement goals. Both RMCTS and PMCTS use SA_M to schedule the progression of the agent's intentions. In RMCTS the maintenance goals are implemented reactively and in PMCTS proactively. In the case of RMG, when the reactive maintenance goal is triggered, the rover must first head to the depot to recharge its battery before attempting to achieve other goals. On the other hand, with PMG, the maintenance condition is predicted by using summary information. The reactive maintenance condition for RMG and RMCTS is set to be $batteryLevel > 20$. For PGM and PMCTS, the condition is set to be that the *batteryLevel* is greater than the minimal number of moves required for the agent to return to the charging station from its current position. As all approaches can achieve all the achievement goals, we evaluate their performance on the amount of battery consumed.

In the experiments, the SA_M scheduler is configured to perform 100 iterations in each deliberation cycle and 10 simulations per iteration, and we report the average performance in 100 runs. The results are shown in Figure 1.

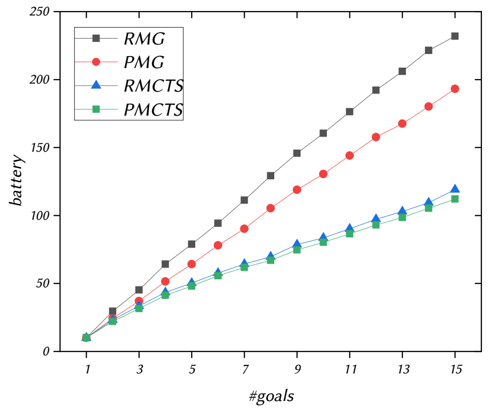

Figure 1: Battery consumption with battery capacity of 40

As can be seen, for all approaches, as the number of goals increases the amount of battery consumption increases. However the MCTS-based approaches (PMCTS, and RMCTS) outperform RMG and PGM, and approaches using proactive maintenance goals have better performance than those using reactive maintenance goals. As expected, RMG has the worst performance, i.e., it consumes more battery power than the other approaches. PMG has better performance than RMG, as it can estimate the battery consumption before pursuing an achievement goal: if the agent will trigger the maintenance goal half way to achieving its next achievement goal, then it will choose to recharge first. PMCTS and RMCTS have a clear advantage over PMG, particularly when the number of achievement goals increases. The reason is that the MCTS-based approach can predict not only the possible violation of maintenance conditions during the execution but also possible synergies between different intentions (i.e., the agent can merge the same actions from different intentions to save time and resources).

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (61906169) and Yongjiang Talent Introduction Programme (2022A-234-G).

REFERENCES

[1] 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*, Christian Darken and Michael Mateas (Eds.). The AAAI Press, Stanford, California, USA.

[2] Simon Duff, James Harland, and John Thangarajah. 2006. On proactivity and maintenance goals. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*. ACM, 1033–1040. https://doi.org/10.1145/1160633.1160817

[3] Simon Duff, John Thangarajah, and James Harland. 2014. Maintenance Goals in Intelligent Agents. *Comput. Intell.* 30, 1 (2014), 71–114. https://doi.org/10.1111/coin.12000

[4] Marcus J. Huber. 1999. JAM: A BDI-Theoretic Mobile Agent Architecture. In *Proceedings of the Third Annual Conference on Autonomous Agents, AGENTS 1999, Seattle, WA, USA, May 1-5, 1999*. ACM, 236–243. https://doi.org/10.1145/301136.301202

[5] Brian Logan, John Thangarajah, and Neil Yorke-Smith. 2017. Progressing Intention Progresson: A Call for a Goal-Plan Tree Contest. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017),*, S. Das, E. Durfee, K. Larson, and M. Winikoff (Eds.). IFAAMAS, IFAAMAS, Sao Paulo, Brazil, 768–772.

[6] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. 2005. Jadex: A BDI Reasoning Engine. In *Multi-Agent Programming*, RafaelH. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni (Eds.). Multiagent Systems, Artificial Societies, and Simulated Organizations, Vol. 15. Springer US, 149–174. https://doi.org/10.1007/0-387-26350-0_6

[7] A. S. Rao and M. P. Georgeff. 1992. An abstract architecture for rational agents. In *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*, C. Rich, W. Swartout, and B. Nebel (Eds.). 439–449.

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

[9] 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.

[10] John Thangarajah, Lin Padgham, and Michael Winikoff. 2003. Detecting & exploiting positive goal interaction inintelligent agents. In *The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003*. ACM, Melbourne, Victoria, Australia, 401–408.

[11] John Thangarajah, Michael Winikoff, Lin Padgham, and Klaus Fischer. 2002. Avoiding Resource Conflicts in Intelligent Agents. In *Proceedings of the 15th Eureopean Conference on Artificial Intelligence*. IOS Press, Lyon, France, 18–22.

[12] Max Waters, Lin Padgham, and Sebastian Sardina. 2014. Evaluating Coverage Based Intention Selection. In *Proceedings of the 13th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2014)* (Paris, France), Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (Eds.). IFAAMAS, 957–964. http://dl.acm.org/citation.cfm?id=2617388.2617398

[13] 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

[14] 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)*, Matteo Baldoni, Jorg P. Muller, Ingrid Nunes, and Rym Zalila-Wenkstern (Eds.). Singapore, 181–196.

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

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