# Intention Progression with Maintenance Goals

**Di Wu & Yuan Yao & Natasha Alechina & Brian Logan & John Thangarajah**

Zhejiang University of Techonology & University of Nottingham Ningbo

## Introduction

One of the key advantages of Belief-Desire-Intention (BDI) agents [3] 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* [2]. Existing works on intention progression problem are limited to scheduling achievement goals. In many applications agents must also *maintain* particular states of the environment, e.g., not running out of power, avoiding collisions, etc., in addition to achieving certain states. Such goals are termed *maintenance goals*, as they specify a state of the environment an agent should maintain. Previous work [1] on proactively reasoning about maintenance goals is summary-information-based. However, that approach assumes that the preventive measure to maintain the goal doesn't interact with the agent's other intentions.

Here, we present $SA_M$, a new approach to intention progression for BDI agents with both achievement and maintenance goals.

## Methods

$SA_M$ extends the *SA* scheduler [4] in explicitly taking account of the agent's reactive and proactive maintenance goals. $SA_M$ is based on the Monte-Carlo Tree Search(MCTS) 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 is the state of the agent and its environment following the execution of the action. 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.

**Modification to the expansion phase of *SA*** In the expansion phase of $SA_M$, different mechanisms are used for reactive and proactive maintenance goals. For reactive maintenance goals, $SA_M$ first checks if any of the maintenance conditions are violated in $n_s$ (the leaf node selected in the selection phase). For each maintenance goal $G_m$ which has its maintenance condition violated in $n_s$, a set of nodes representing the states resulting from executing each of its applicable recovery plans are generated, and added as the children of $n_s$. 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 $G_m$ if there is no existing intention for recovering $G_m$ yet. In a special case when an agent does not have any achievement goals and all the maintenance conditions are not violated, no child node of $n_s$ will be generated. That is, if the agent is not pursuing any goals, then waiting is the best way to maintain the current state.

**Modifications to the Simulation Phase of *SA*** During the simulation phase, $SA_M$ randomly simulates the possible actions in achieving the agent's goals as in *SA*. When the maintenance condition of a reactive maintenance goal $G_m$ is triggered, in addition to achieving the agent's achievement goals, $SA_M$ can also select to execute the actions in one of $G_m$'s recovery plans. For proactive maintenance goals, $SA_M$ is able to execute their recovery plans anytime when there is no existing intention to maintain the same goals. The recovery plans for maintenance goals can also be interleaved with other intentions so as to avoid conflicts or to exploit synergies. The simulation ends when a terminal state is reached which contains one of the following situations: 1) All achievement goals are achieved and the maintenance condition is not violated. 2) All the remaining achievement goals cannot be achieved in the current state. 3) A maintenance condition cannot be re-established.

## Results

We evaluate the performance of $SA_M$ based on a Mars rover example [1]. The environment is a grid consisting of $20 \times 20$ cells (see Figure 1).

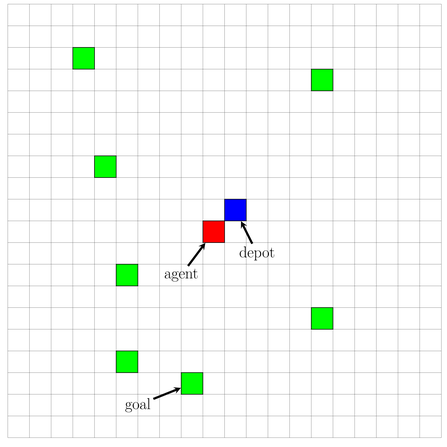


**Figure 1:** A Mars rover example

We measure the performance of the Mars rover agent based on two criteria: the number of goals achieved and the amount of battery consumed. To evaluate the performance of $SA_M$, the experiments were conducted in 4 cases: RMG, PMG, RMCTS and PMCTS. Both RMG and PMG are from [1].

The results are shown in Figure 2.

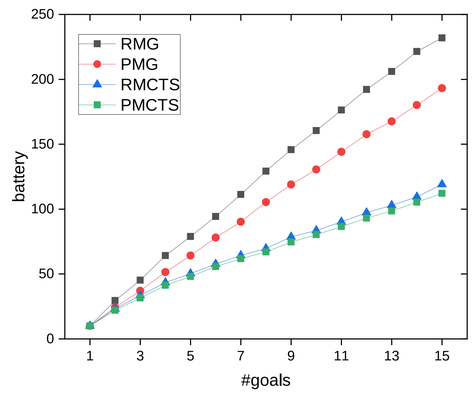


**Figure 2:** Battery consumptions with fixed capacity of 40

As can be seen, for all approaches the amount of battery consumption increases, as the number of goals increases. MCTS-based approaches (PMCTS, and RMCTS) outperform all other approaches, and approaches using proactive maintenance goals have better performance than those using reactive maintenance goals. As expected, RMG has the worst performance in all testing cases, i.e., it consumes more battery power than any other approaches. The PMG has better performance than RMG, as it can estimate the battery consumption before pursuing a goal. If the agent will trigger the maintenance goal half way achieving its next achievement goal, then it will choose to recharge first. Finally, both PMCTS and RMCTS have a clear advantage over PMG. Especially, when the given number of goals becomes larger, the differences between MCTS-based approaches and other approaches are more significant. 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).

## Conclusion

The preliminary results suggested that even without giving any reliable prediction of future violation, $SA_M$ with reactive maintenance goal can still outperform both reactive and proactive approaches proposed in [1] in all cases.

The advantage of using $SA_M$ comes from not only the ability to avoid violation between achievement goals and maintenance goals, but also the nature that MCTS can interleave recovery plans and other intentions so as to avoid negative interaction and to exploit synergies if maintenance goals are implemented proactively.

## Future work

- Extend $SA_M$ scheduler to work with nondeterministic actions in uncertain environment.
- Extend the current $SA_M$ to additional richer types of goals that are explicitly represented as Linear Temporal Logic formula.
- Investigate how to schedule maintenance goals that are only valid for a period of time.
- Investigate how to schedule multiple maintenance goals with different priorities or urgency.

## Acknowledgements

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

[2] Brian Logan, John Thangarajah, and Neil Yorke-Smith. Progressing intention progresson: A call for a goal-plan tree contest. In S. Das, E. Durfee, K. Larson, and M. Winikoff, editors, *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017),*, pages 768–772, Sao Paulo, Brazil, May 2017. IFAAMAS, IFAAMAS.

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

[4] Yuan Yao and Brian Logan. Action-level intention selection for BDI agents. In *15th International Conference on Autonomous Agents and Multiagent Systems*, pages 1227–1236. IFAAMAS, 2016.