Using Multi-objective Optimization to Generate Timely Responsive BDI Agents

Doctoral Consortium

Márcio Fernando Stabile Junior
Universidade de São Paulo
São Paulo, Brazil
mstabile@ime.usp.br

ABSTRACT

A BDI agent's ability to perform well depends on its reasoning time. If the reasoning is slow, it is possible that the environment has changed and the action selected is no longer optimal by the time the agent has finished to deliberate. This work then builds a BDI architecture using Anytime Algorithms that can control the amount of time used by the agent to reason and act on the environment. I briefly describe the proposed architecture and its implementation in the Jason agent language.

KEYWORDS

BDI agents; Anytime algorithms; Multi-objective optimization

ACM Reference Format:
Márcio Fernando Stabile Junior. 2022. Using Multi-objective Optimization to Generate Timely Responsive BDI Agents: Doctoral Consortium. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 3 pages.

1 INTRODUCTION

The BDI model proposed by Bratman [2] and formalized by Rao and Georgeff [6] mainly consists of reviewing the agent's beliefs based on perceptions, listing and filtering desires to generate intentions, and creating and executing plans to achieve those intentions. However, Bratman et al. [3] states that one of the problems with agents is that reasoning is not immediate, as studied by Simon [8]. Hence, the agent risks that the environment changes and the selected action is no longer optimal when the agent finishes deliberating. Considering this problem, to plan actions in scenarios where the time available to generate the plan is variable, and the process of decision making is complex, Dean and Boddy [4] defined a class of algorithms called *anytime algorithms*. Based on the work of Dean and Boddy [4], Zilberstein [12] proposed ways to analyze and compose algorithms in order to create complex systems that allow the balancing between processing time and quality of results. Since one of the issues of the BDI architecture is the need for fast practical reasoning, and the main characteristic of Anytime Algorithms is the possibility of controlling the execution time, it seems logical to use anytime algorithms to design more efficient BDI agents.

This work then aims to answer the question: How can we build a BDI architecture using Anytime Algorithms that can control the amount of time used by the agent to reason and act on the environment while assuring a minimum quality in the actions?

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

2 ANYTIME BDI

In order to answer this question, in this section I present a brief description of the formal model called Anytime BDI. First, it is necessary to define three terms: Internal actions, external actions and plans. Based on the definitions made by Schut et al. [7], I formally define these elements below:

(1) An external action (α) is an action that is performed by the agent on the environment. Examples of external actions include agent movement, activating a button and carrying an item. (2) An internal action (β) is an action that is performed by the agent that does not directly change the environment. Instead, it affects the internal state of the agent. Examples of internal actions include creating a new belief or acquiring a new goal. (3) A plan (π) is a pre-defined sequence of internal (α) and external (β) actions.

Having made these definitions, I present a description of the model of the Anytime BDI Agent:

An Anytime BDI Agent is an intelligent agent architecture composed of two layers. One is the Agent Data Layer (ADL), which contains all the agent's data structures. The other is the Agent Control Layer (ACL), composed of the mechanisms that control the agent execution. The Anytime Data Layer is comprised of all the structures that store the necessary data for the execution of the agent. We define $ADL = \langle \mathcal{P}, \mathcal{B}, \Pi, \mathcal{D}, \mathcal{I}, \alpha_{default}, t_\Delta, \mathcal{DA}, \mathcal{PP} \rangle$ where:(1) The set of perceptions $\mathcal{P}$; (2) The set of beliefs $\mathcal{B}$; (3) The set of desires $\mathcal{D}$; (4) The set of plans Π; (5) The set of intentions $\mathcal{I}$; (6) The default action $\alpha_{default}$ executed if no better action was found; (7) The response time t_Δ; (8) The queue of delayed actions $\mathcal{DA}$; (9) The set of performance profiles $\mathcal{PP}$;

The Anytime Control Layer is comprised of the processes responsible for running the Anytime BDI Agent and ensuring its execution within the specified time. We define $ACL = \langle \mathcal{BM}, \mathcal{IG}, \mathcal{IE}, \mathcal{M}, \mathcal{F} \rangle$ where: (1) $\mathcal{BM}$: Belief Manager is the anytime component responsible for producing a new set of beliefs from perceptions; (2) $\mathcal{IG}$: Intention Generator is the anytime component responsible for based on current beliefs, desires, and intentions, produce a new set of intentions and plans; (3) $\mathcal{IE}$: Intention Executor is the anytime component responsible for based on current beliefs, intentions, utility function, and delayed actions, choose an external action to be executed on the environment; (4) $\mathcal{M}$: Monitor is the component responsible for based on the performance profiles and t_Δ, calculate time allocation and control the execution; (5) $\mathcal{F}$ is a function responsible for calculating the utility of an external action α.

The goal of this model is to make the agent perform an external action α in the environment at each time interval t_Δ. For this, three modules were defined ($\mathcal{BM}$, $\mathcal{IG}$, and $\mathcal{IE}$), which are anytime algorithms responsible for analyzing perceptions, beliefs, desires,

plans and intentions in order to generate an external action to be performed in the environment. To ensure that these modules run within the time limit, the Monitor $\mathcal{M}$ analyzes the performance profiles of each algorithm involved and executes them for the calculated time. At the end of the time, the Monitor $\mathcal{M}$ receives the generated external action and executes it in the environment.

The $\mathcal{BM}$ assumes that there is a list of perceptions to evaluate. As long as the processing time is not over and there are perceptions not yet analysed, the component will analyse the perceptions ordered by a user-defined policy based on the remaining perceptions and the time remaining. The $\mathcal{IG}$ analyses desires and intentions one at a time. Thus, it is possible to balance the frequency of creating new intentions and reconsidering existing ones. Lastly, $\mathcal{IE}$ executes the plans in a round-robin style, until it finds an external action. When one is found, it is scheduled for execution at the end of the allocated time, the plan is paused and the $\mathcal{IE}$ resumes executing the remaining plans. If multiple external actions are found, the ones with lower priority are stored in the Delayed Action queue ($\mathcal{DA}$) for future evaluation.

3 ANYTIME JASON

To validate the model and show that it allows achieving the research objectives, I chose to implement the model and carry out experiments using the BDI Jason agent programming language [1]. The main reasons are its popularity in the area of multi-agent systems and because it is a well-documented open-source language. Using an existing and well-accepted language allows us to compare the behavior of the agents and to identify the positive and negative points of the proposed architecture.

I then made the necessary changes to create a Jason architecture whose execution follows the theoretical model described in the Section 2. I named this architecture *AnytimeJason*. However, it is not possible to entirely change the way the language works. For example, Jason uses events as one of its primary mechanisms. Neither the model proposed in [11] nor the model I defined in the section 2 has any mention of events. On the other hand, ignoring the event mechanism could indicate that the model defined is not flexible enough to be implemented by the existing languages. I decided then to draw the best possible parallels to bring the language as close as possible to the theoretical model presented.

A Jason agent is executed through three main components (Sense, Deliberate, and Act). Each of these components consists of a set of functions responsible for the agent's execution. Rather than executing the steps in the reasoning cycle repeatedly, it is necessary to link the agent's reasoning cycle to the agent's expected response time. Thus, by implementing the behaviors defined for the $\mathcal{BM}$, $\mathcal{IG}$ and $\mathcal{IE}$ modules to the Sense, Deliberate and Act methods, we can control the execution of the agent's reasoning cycle. The next step was to make the calling of these functions and their execution time managed by the Monitor $\mathcal{M}$. For this, when an Anytime agent is created, the Jason compiler creates a Monitor $\mathcal{M}$ module for it and informs it of the response time t_Δ that is defined by the user in the agent's creation code. Once created, the Monitor $\mathcal{M}$ optimally splits the response time among the other components based on their performance profiles, determining t_{bm}, t_{ig}, t_{ie}, that are the amount of time allocated for each of the components.

To control the Sense runtime, I implemented a policy based on perception filters similar to those in [10]. By defining different filters, it is possible to priorize certain perceptions. For Deliberate, I modified Jason so that it analyzes more than one event in the same cycle, and it is possible to interrupt the analysis if the time runs out and continue at a later time. Finally, in the Act component, I implemented the proposed Delayed Action queue, which stores the external actions the agent is waiting to perform in the environment.

4 MULTI-OBJECTIVE OPTIMIZATION OF TIME ALLOCATION

Since I proposed a monitor that optimally splits the response time among the other components, it is necessary to describe how the Monitor $\mathcal{M}$ performs the split. I implemented a profiling mechanism in Jason. This mechanism records a series of information about the agent's execution. This information includes how many perceptions the agent receives, how long it takes to update each perception, how many internal actions it performs before finding an external action, etc. Based on this information, I generate the performance profiles of each component in each agent, which are estimates of the performance of each one. The quality of the Sense component is measured by how many percent of the total beliefs it was able to assess. Deliberate's measures how many percent of the events it would evaluate. The quality of the Act's response, on the other hand, is measured based on the probability of finding an external action within the given time. As the quality of the result of a component depends on the quality of the previous, I use for the Deliberate and Act components a type of performance profile called Conditional Performance Profile, described in [12]. Thus, we have three quality functions (performance profiles), and we aim to maximize the qualities of the responses given by the components.

I then use the ϵ-constraint method described in [5] to optimize the time allocation. This method optimizes one of the functions while defining constraints for the others. So, in our case, the agent designer can, as an example, state that the Sense component can never evaluate less than 20% of the total perceptions. Thus, the ϵ-constraint method will calculate a set of time allocations that maximizes the quality of the responses. Based on the user restrictions, the Monitor $\mathcal{M}$ selects one of the time allocations in the set and executes the other modules accordingly. If the Monitor $\mathcal{M}$ needs to change the allocations during the execution of a module (e.g., there are no perceptions to evaluate), the Monitor $\mathcal{M}$ can select another one based on the current status of the execution.

5 FUTURE WORK

I'm currently developing a set of experiments that seek to demonstrate the advantages and disadvantages of the proposed architecture compared to Jason's default architecture. These experiments will employ the agents created for the Multi-Agent Programming Contest [9]. The contest scenario is very conducive to this function as it already aims to compare teams of agents.

ACKNOWLEDGMENTS

This work was supported by CNPq, Brazil, Grant 140448/2016-0. The author would like to thank Jaime S. Sichman for his constant guidance and support as supervisor.

REFERENCES

[1] Rafael H. Bordini and Jomi F. Hübner. 2006. BDI Agent Programming in AgentSpeak Using Jason. In *Computational Logic in Multi-Agent Systems*, Francesca Toni and Paolo Torroni (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 143–164.

[2] M. Bratman. 1987. *Intention, plans, and practical reason.* Harvard University Press, Cambridge, MA. http://books.google.de/books?id=I0nuAAAAMAAJ

[3] Michael E. Bratman, David J. Israel, and Martha E. Pollack. 1988. Plans and resource-bounded practical reasoning. *Comput. Intell.* 4 (1988), 349–355. https://doi.org/10.1111/j.1467-8640.1988.tb00284.x

[4] Thomas Dean and Mark Boddy. 1988. An Analysis of Time-Dependent Planning. In *Seventh AAAI National Conference on Artificial Intelligence (AAAI)*, Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith (Eds.). AAAI Press / The MIT Press, 49–54. http://dl.acm.org/citation.cfm?id=2887974

[5] Kaisa Miettinen. 2008. *Introduction to Multiobjective Optimization: Noninteractive Approaches.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–26. https://doi.org/10.1007/978-3-540-88908-3_1

[6] Anand S. Rao and Michael P. Georgeff. 1991. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning* (Cambridge, MA, USA) *(KR'91)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 473–484.

[7] Martijn Schut, Michael Wooldridge, and Simon Parsons. 2004. *The theory and practice of intention reconsideration.* Vol. 16. 261–293 pages. https://doi.org/10.1080/09528130412331309277

[8] Herbert A. (Herbert Alexander) Simon. 1957. *Models of man, social and rational : mathematical essays on rational human behavior in a social setting.* Wiley, New York.

[9] Marcio Fernando Stabile and Jaime S. Sichman. 2021. The LTI-USP Strategy to the 2020/2021 Multi-Agent Programming Contest. In *The Multi-Agent Programming Contest 2021*, Tobias Ahlbrecht, Jürgen Dix, Niklas Fiekas, and Tabajara Krausburg (Eds.). Springer International Publishing, Cham, 108–133.

[10] Márcio Fernando Stabile Jr and Jaime S Sichman. 2015. *Melhorando o desempenho de agentes BDI Jason através de filtros de percepção.* Master's thesis. Universidade de São Paulo.

[11] M Wooldridge. 2000. *Reasoning About Rational Agents.* MIT Press.

[12] Shlomo Zilberstein. 1993. *Operational rationality through compilation of anytime algorithms.* Ph.D. Dissertation. http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1136