Anticipatory Thinking in Multi-Agent Contexts

Jomi F. Hübner[1], **Samuele Burattini**[2], **Alessandro Ricci**[2], **Simon Mayer**[3]

[1] Federal University of Santa Catarina, Brazil

[2]University of Bologna, Italy

[3]University of St.Gallen, Switzerland

`jomi.hubner@ufsc.br`

Abstract. *Anticipatory thinking allows agents to foresee and avoid future problems by exploiting their plan libraries. This paper extends previous work on anticipatory thinking for single agents to the domain of Multi-Agent Systems (MAS), where the environment's evolution depends on the combined actions of multiple autonomous agents. We detail how an agent can simulate potential futures in a multi-agent context, considering the preferred policies of all participants, to anticipate and react to undesirable outcomes. Our proposed strategies enable an agent to either abandon a goal if a problem is inevitable or dynamically select an alternative plan to avoid anticipated problems. Experiments in a Bridge Scenario, a classic Multi-Agent Pathfinding problem, yield a critical insight: anticipatory thinking, in isolation, only resolves the conflict when exactly one agent exercises this foresight. When multiple agents independently reason about the future, the problem re-emerges, indicating a fundamental limitation of purely individualistic anticipation and stressing the importance of future research into integrated anticipatory thinking and coordination.*

1. Introduction

In [Hübner et al. 2025], the authors propose an agent capable of anticipating future problems and exploiting its plan library to avoid them. The agent chooses applicable plans outside its usual preferences to avoid the problems and achieve its goals. Inspired by *anticipatory thinking* proposals [Jones and Laird 2023, Szpunar et al. 2014, Amos-Binks and Dannenhauer 2019, Amos-Binks et al. 2023], simulation is the main technique used to foresee the future, based on a given model of the environment.

Since their proposal considers only *one* agent, in this paper we address the case of *Multi*-Agent Systems (MAS). More agents bring some challenges to the process of computing possibilities in the future because the environment is shared among them. It is hence not only our agent that changes the environment, others also change, possibly conflicting with the objectives of our agent. As in [Hübner et al. 2025], we delimit the investigation to agents that have this knowledge available in a plan library, such as PRS and BDI architectures [Georgeff and Lansky 1987, Rao and Georgeff 1995, Silva et al. 2020]. In these architectures, the agent policy comes from a set of plans, and these plans represent options to achieve the agent's goals. The agent is autonomous in choosing among these options to better adapt its behaviour to run-time circumstances.

In Sections 2 and 3 we update the proposal of [Hübner et al. 2025] for the multi-agent case: how the future is simulated in the MAS case and how and agent can exploit its plan library to avoid future problems. The proposal is evaluated in Section 4 based on a simple scenario. We discuss the proposal, its results, limitations, main characteristics, and related work in Sections 5 and 6. Conclusions and future work are described in Section 7.

2. Future Simulation

The simulation of the future depends on how it evolves as agents act on the environment. Considering that we have n agents, a set of possible environment states S, and possible actions A, the evolution of the environment is modelled by the function $e : S \times A^n \rightarrow S$. Given a state $s \in S$ and the actions $a_i \in A$ of each agent i ($0 < i \leq n$), $e(s, a_1, a_2, ..., a_n)$ is the next state of the environment after the execution of all actions. For simplicity, we are initially assuming that the order in which these actions are executed or whether they are executed concurrently is irrelevant. We consider the environment as static, fully observable, deterministic, and discrete [Russell and Norvig 2010].

The simulation also depends on how every agent decides its action. Their decisions are based on their plan library and are modelled by the function $\pi_i : S \rightarrow A^r$ (agent *policy*). Given a state s, $\pi_i(s)$ is a sequence of r possible *options* of agent i, ordered by its *preference*. For example, if the state is s and $\pi(s) = \langle s, w, e, n \rangle$, the agent prefers to go south (s), then west (w), then east (e), then north (n).[1] This function represents the knowledge the developer gave to the agent: the preferred actions for every state ordered according to a strategy.

The result of a simulation is a history. We represent a history h ($h \in S^m$, $m \geq 0$) as a sequence of m states $\langle s_1, \ldots, s_m \rangle$. We denote the last state of the history as $h_{[-1]}$, an empty history as $\langle\rangle$, and the set of all histories as H ($H = \bigcup_{i=0}^{\infty} S^i$). The concatenation of two histories is produced by the operator $\oplus$. A possible history is illustrated in Fig. 1. The history $\langle s_1, s_2, s_3 \rangle$ is produced by every agent following their preferences departing from the state s_1. In this initial state, the preference of agent 1 is action a_1 ($\pi_1(s_1)_{[1]} = a_1$) and the preference of agent 2 is a_2 ($\pi_2(s_1)_{[1]} = a_2$). The alternative options in the figure (the dashed lines) are options for agent 1 only, the options of others are not shown.

Given these definitions, we have a discrete notion of time that evolves based on changes in states produced by the actions of the agents. By simulating the future with functions e and π, the future is linear, as well as the computational complexity to build it. For instance, from state s, the next state is given by $e(s, \pi_1(s)_{[1]}, ..., \pi_n(s)_{[1]})$. In our proposal, we are hence able to compute *the* future instead of *possible* futures. This is possible because we simulate the future based only on the first option of the agents' policy. In other techniques, such as automated planning [Ghallab et al. 2016], the future is branching for each possible action because the agent policy is not considered (indeed, their objective is to create a policy), which usually leads to exponential complexity.

3. Reasoning About Future Options

Taking the perspective of an agent with individual goals and the proposed model of how the environment evolves, this agent does not determine the future alone. The next state is

[1] We use $[i]$ to denote the i-th element of a sequence and $[-1]$ to refer to its last element. For example, if $\pi(s) = \langle s, w, e, n \rangle$, then $\pi(s)_{[1]} = s$ and $\pi(s)_{[-1]} = n$.

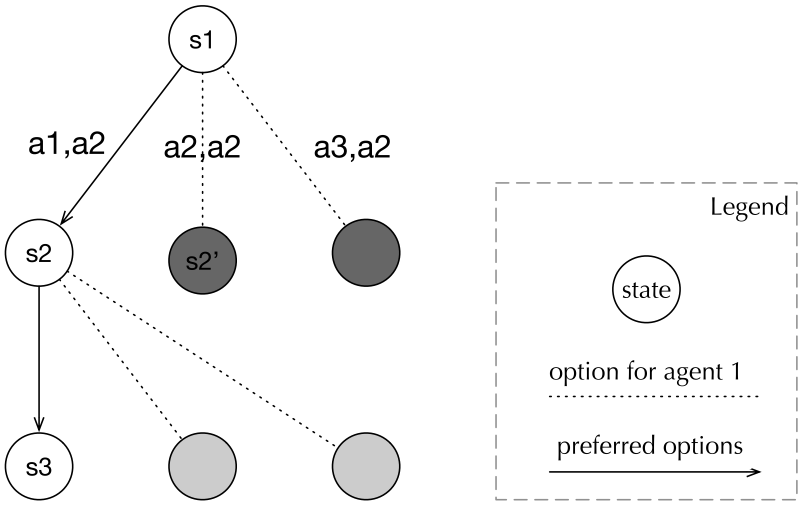

Figure 1. Example of future simulation in the perspective of agent 1. Four actions are possible ($A = \{a_1, a_2, a_3, a_4\}$) and we have two agents. Present is the state s_1 and the policies for this state are $\pi_1(s_1) = \langle a_1, a_2, a_3\rangle$ and $\pi_2(s_1) = \langle a_2, a_3\rangle$. The state s_1 is transformed into another state s_2 by actions a_1, a_2 ($e(s_1, a_1, a_2) = s_2$). These actions are the preferred options of both agents for state s_1. The produced history is $\langle s_1, s_2, s_3\rangle$.

also determined by the decision of others. To help this agent to reason about the success of its future, besides its own policy, the policies of all other agents, and the environment model e, we also assign to this agent a goal $g_i \in S$ and a boolean function p_i that classifies some histories as undesired ($p_i : H \rightarrow \mathbb{B}$). The domain of this function is a sequence of states that represent some *history* of the environment. The function maps a history to $true$ in cases where the history configures a problem. Possible problems include that the history contains an *undesired state*, that it is *too long* without the goal being achieved, that it *violates some norm*, etc.

Given both the environment model and the agents models, an agent i can use the following $matrix : S \times S \rightarrow H$ function to foresee whether its goal g_i will be achieved without any problem from the current state s:

$$
\begin{aligned}
matrix(g_i, s) &\stackrel{\text{def}}{=} mah(g_i, \langle s\rangle) \\
mah(g_i, h) &\stackrel{\text{def}}{=} \begin{cases} \langle\rangle & \text{if } p_i(h) \\ h & \text{if } s' = g_i \\ mah(g_i, h \oplus \langle e(s', \pi_1(s')_{[1]}, ..., \pi_n(s')_{[1]})\rangle) & \text{otherwise} \end{cases} \\
s' &= h_{[-1]}
\end{aligned}
$$

The $matrix$ function begins with the state s and builds the history incrementally by simulating the future step by step starting from the last state of a history h. The function returns either a history where the goal is achieved or an empty history if a problem is detected. The auxiliary function $mah : S \times H \rightarrow H$ maps a goal g_i and an history of decisions given by π and e into an history where the goal is the last state.[2]

[2]If both conditions of mah hold (i.e., $p_i(h)$ and $s' = g_i$), the function returns $\langle\rangle$. The problem precedes the achievement.

A simple decision process that this agent may now follow is: given the current state s, use the $matrix(g_i, s)$ to foresee if the goal g_i is achieved in the future. If so, keep going; otherwise, stop putting energy into pursuing g_i as it is preferable to do nothing rather than pursuing a goal that will not be achieved anyway. This strategy is called one in [Hübner et al. 2025] and is extended here for a matrix with multiple agents.

The matrix function considers that each agent will select its preferred options. However, although the decisions of others might be difficult to change, each agent may exploit its own option space to find an alternative, preferable, future. More specifically, the agent may select another option for the current state s (given by $\pi(s)_{[j]}$ with $j > 1$) instead of the most preferred option (given by $\pi(s)_{[1]}$). For example, in Fig. 1, Agent 1 may select its second option a_2 and change the future towards s'_2. [Hübner et al. 2025] investigates whether an alternative option should be considered just after s or later in the future. They also propose a search algorithm for that: a Breadth-First Search (BFS) [Russell and Norvig 2010] where the nodes of the tree search are expanded by the π options. The search ends when a state s with no anticipated problem ($matrix(g_i, s) \neq \langle\rangle$) is found.[3] As in [Hübner et al. 2025], we refer to this strategy as solve_f: instead of stopping to pursue its goal as in one, the agent continues to act toward g_i using alternative options. We can also note that the implementation of the solve_f strategy in the the multi-agent case does not require knowledge about the full policies of other agents; knowing their first preferred option is sufficient.

When applied to the *multi*-agent case, the proposed strategies simulate the future assuming that all agents will choose their first option (as defined in the $matrix$ function). Thus, they do not reason about the future while in "matrix mode." This assumption is taken to avoid the exponential complexity of a matrix function that would branch alternative futures based on all agents' options. As proposed, the future simulation (presented in Sec. 2) is linear. The solve_f strategy, however, searches for alternatives of *one* agent, considering only its options.

4. The Bridge Scenario

To evaluate our proposal, we use a Bridge Scenario, a particular case of a Multi-Agent Pathfinding (MAPF) problem [Stern et al. 2021, Moan et al. 2024]. This scenario is commonly used to assess cooperation among agents [Silver 2021], although here we are more focused on the implications of using anticipatory thinking. In our case, illustrated in Fig. 2, we have two agents. Agent 1 is on the left side of the river and wants to cross to the right side. Agent 2 is on the opposite side and also wants to cross. The problem is that the bridge is too narrow for both agents. If both agents enter the bridge, both will be blocked in the middle of the bridge trying to move on. A possible solution is that exactly one agent waits for the other to cross and then crosses itself.

We investigate this problem in the context of agents endowed with the strategies proposed in Sec. 3, considering the following cases:

c1 : no agent has the capability to reason about the future.

[3]Note that the termination condition for the search is a state s with a desirable future ($matrix(g_i, s) \neq \langle\rangle$) and not $s = g_i$ (as commonly used in BFS). The history from s to g_i is found by in linear time by the $matrix$ function. The search is similar to Monte Carlo Tree Search where the expansion phase is based on the agent options and the rollout phase is based on the matrix simulation [Yao et al. 2014].

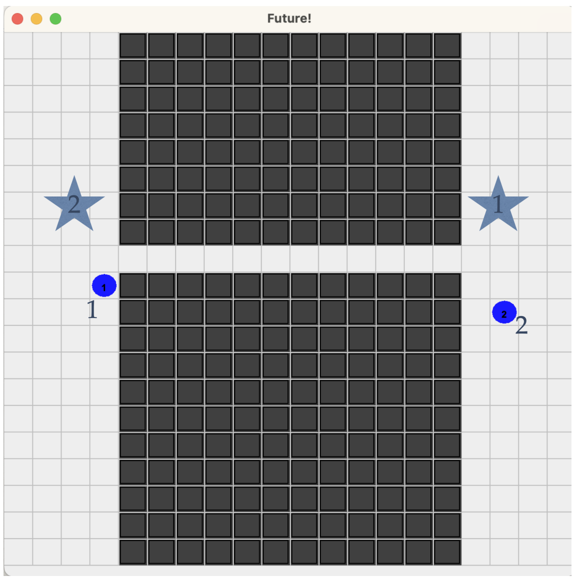

Figure 2. The Bridge Scenario. Blue circles are agents and stars are their corresponding destination. Dark squares represent the river; agents cannot enter these squares. Only one agent may occupy a grey location. Possible actions for the agents are $\{s, sw, se, w, e, n, nw, ne, idle\}$, i.e., moving in some direction or staying put.

- c2 : agent 1 uses the strategy one and agent 2 has no anticipatory capability.
- c3 : agent 1 uses the strategy one but does not give up the goal, it does $idle$ and tries again later hoping that the environment will be more favourable. Agent 2 has no anticipatory capability.
- c4 : agent 1 uses the strategy solve_f and agent 2 has no anticipatory capability .
- c5 : both agents use the strategy one.
- c6 : both agents use the strategy solve_f.

We also consider that no coordination mechanism is used. Agents decide their action based only on their own policy π. The options of these policies are ordered as the actions place them near the destination. For example, the options for Agent 1 in its initial location (line 10 and column 4 in Fig. 2) are $\pi_1(s_{10,4}) = \langle ne, n, nw, idle, w, s, sw\rangle$.

The following metrics were measured during the experiments and the results for each case are shown in Table 1:[4]

- *effectiveness*: the number of agents that have arrived at the destination;
- search *cost*: the number of states visited by the matrix function; and
- plan *efficiency*: the cost of the actions performed by the agents during the experiment (the idle cost is 0 and the cost of all other actions is 1);

For case c1, as expected, no agent achieves the goal. They enter the bridge at the same time and block each other. Moreover, following their policy, they continue trying to cross forever! In case c2, Agent 1 foresees the problem in the bridge and gives up the goal. Agent 2 crosses the bridge normally. In case c3, Agent 1 foresees the problem

[4]The experiment is implemented in Jason [Bordini et al. 2007] and available at `http://https://github.com/jomifred/future-bdi`. The implementation extends the matrix to run with several agents. This matrix is used to run either strategy one or solve_f. It basically clones the mental state of all agents and runs them in a simulated environment e instead of the real environment.

Table 1. Results for the Bridge Scenario

Case	Effectiveness	Search Cost	Plan Efficiency	
			Agent 1	Agent 2
c1	0	–	∞	∞
c2	1	25	0	15
c3	2	300	15	15
c4	2	7111	15	15
c5	0	2×25	0	0
c6	0	2×7111	∞	∞

and remains idle for 14 simulation steps, when Agent 2 has finished crossing the river. It then crosses the bridge. The matrix was executed 14 times, visiting 25 states during each execution. In case c4, Agent 1 foresees the problem, computes alternative options that make it remain idle for 14 steps, and then follows its usual preference. The cost of computing the alternative options is quite high (7111 states were visited in the search). In case c5, what happens with Agent 1 in c2 also happens with Agent 2, both foresee that their goals will not be achieved and they do nothing. Finally, in case c6, both agents compute alternative options, remain idle for 14 steps, and then both enter the bridge and block each other. They end up in the same situation as in case c1 with a lot of computation cost.

Although both cases c3 and c4 have ended with agents at their target destinations, c3 assumes that the environment will spontaneously improve in the future and requires a *default* action ($idle$ in case c3). The solve_f strategy of case c4 does not have these constraints, it uses options of the existing policy to avoid future problems. However, it pays for the result in terms of search cost.

5. Discussion

We notice that the bridge problem can be solved using anticipatory thinking only in cases c3 and c4, the cases where exactly *one* agent reasons about the future. If no agent has considered the future, or if both have done that, the problem is not solved. This is a significant limitation of the proposal and mirrors the dynamics in repeated games, where players adjust their strategies based on expectations about others' future actions. Specifically, cases c3 and c4 demonstrate that both agents can achieve their goals when only one agent anticipates potential conflicts and adapts accordingly. Conversely, when neither or both agents engage in anticipatory reasoning, they fail to achieve their goals.

If we are developing agents for an open system [Uez and Hübner 2017, da Rocha Costa et al. 1994], we cannot ensure that other agents will not reason about the future. As proposed in the research on MAPF, a proper solution of the problem can be obtained by having a centralised global view of the problem and/or a coordination mechanism. However, our question is whether anticipatory thinking alone could solve the problem. The answer so far is no. Anticipatory thinking is not completely useless, though. Agents can use the proposed one strategy to foresee the future problem and then use other techniques (e.g., planning) to solve the problem. A direction of future work is to investigate if there is another kind of *multi*-agent problem that can further benefit from anticipatory thinking. Since it seems that this capability is reasonably useful for human societies, it might also be useful for artificial agents.

In this paper, we are looking for improvements in agent decision-making considering the *future* of an environment being changed by several agents. A fundamental feature of the proposal is to be based on the knowledge provided by the developer to the agent. This knowledge equips an agent with options and strategies to achieve their goals, as represented by its π function. We are thus assuming that the developer is an expert in the domain application and that the more options s/he provides for the agent, the better. The solve_f strategy exploits this practical knowledge to better choose between options and avoid problems in the future. In the bridge scenario, we consider only options that are actions in the environment. A direction for future investigation is to consider coordination actions among the options of an agent policy. With coordination actions as an alternative type of action, the problem faced in case c6 might be mitigated.

This proposal requires (i) a model of the environment e and (ii) a model π of all agents. Although the requirement (i) is quite common in AI applications such as search and automatic planning [Ghallab et al. 2016], the requirement (ii) is not, especially in open systems. Indeed, having the agents' model is the reason for our proposal to simulate the future with linear complexity, since we know their decisions while computing the next state. Several approaches can be used by an agent to obtain these models from others:

- The agent may ask others for their models. Of course, the agent has to trust others and they should be capable of expressing their policies π (at least with respect to their first preferred action). In the case of Jason, for instance, it is simple to obtain such a model, as it can be inferred from the agent's mental state (beliefs and plans).
- The agent may assume that other agents behave homogeneously. Its own model, with minor changes, is thus used as the model of others. That could be the solution for the bridge case: the others are equal but with different destinations.
- The agent learns the policy of others by observing their behaviour.
- The agent developer provides or knows all models.

6. Related Work

Considering related work, we start with the similarities between our proposal and automated planning [Ghallab et al. 2016], including proposals that integrate planning and BDI, as in [Sardina and Padgham 2011, Meneguzzi and De Silva 2015]. Both approaches look for agents capable of finding a sequence of actions to achieve their goals, as a kind of "auto-programming." However, while planning is essentially based on a model of the environment, we also exploit the model of the agents. The agents' model allows us to anticipate a problem in the future with linear computational complexity while planning usually has exponential complexity. The use of other agents' models also distinguishes our proposal from adversarial search and algorithms like MiniMax [Russell and Norvig 2010]. Indeed, whether other agents are cooperating or competing is irrelevant for our proposal. Unless in the case where models are provided by others and they are dishonest.

Besides classical and adversarial search, and automatic planning, model checking also deals with foreseeing problems of an agent program [Bordini et al. 2003, Kamali et al. 2017]. The approach is design-time: when a problem is detected, the program is fixed. Our approach is run-time: we consider that the agent has to solve the problem without relying on the developer to upgrade its policy π. We are not looking for errors in the agent program; in contrast, we are exploiting such knowledge for run-time adaptation. The proposal of Ferrando et. al. [Ferrando and Cardoso 2023] also consider run-time verification. As in our proposal, theirs detects a problem and selects another option. However, they focus on problems in the present and do not consider problems in the future. In some sense, we extend that proposal considering the simulation of the future. Another approach that considers the future is to assign a probability of success to the agent's options (as proposed, for instance, in [Baitiche et al. 2017]). The agent then selects the option with better chances to succeed, which implies avoiding future problems. The main difficulty with this approach is how to properly assign such probabilities. Learning is a possible approach for that [Singh et al. 2010].

Considering our bridge scenario, MAPF problem has several proposals ranging from search-based methods, compilation-based approaches, and data-driven techniques [Surynek 2022, Wang et al. 2025]. While they are specifically focused on the path finding problem, our proposal is more general and the bridge scenario is just an example used to experiment with the proposal. The proposal introduced in Sections 2 and 3 is hence problem-agnostic. Another difference is that we take an agent perspective: one agent reasoning about its future. MAPF usually takes a system perspective: all data is know, it is centrally processed by a solver, a global solution is found and determined for the agents. Of course, our agent does something similar on its own. However, it focuses on better choices for itself, not for the others. We hence assume that other agents remain autonomous.

In the context of BDI agents and their programming languages [Boissier et al. 2013, Pokahr et al. 2005, Hindriks 2009, Dastani 2008], the notion of intentions is *future*-oriented. An intention represents a desired future state that the agent has committed to pursue [Bratman 1987, Cohen and Levesque 1987]. However, few programming platforms compute the future considering the agent's intentions, their achievements, and how they impact present decisions. Only the developer considers intention as future-oriented, not the agent itself. In the direction of endowing agents with future reasoning capabilities, we found only the research on Intention Progression [Logan et al. 2017, Thangarajah et al. 2003, Dann et al. 2023]. They are concerned about scheduling several intentions of *one* agent so that they do not negatively interfere with each other's achievements. The problem they are addressing is caused by several intentions being pursued concurrently. Problems are foreseen by computing the future of these intentions. We follow their approach by considering that the course of actions for an intention is derived from a plan library. However, our proposal also considers other agents. In terms of avoiding future problems, they skip them by improving the schedule, while we find alternative options.

Time is taken into account by more generic cognitive agent architectures. For example, the episodic memory of Soar and its planning capabilities [Laird 2019]. Regarding the future, they are more focused on finding a policy to achieve a goal, i.e., self-building a program. Our proposal departs from an existing agent program, usually provided by the developer. Currently, we are focused on evaluating whether this program will succeed or produce undesirable outcomes. It enables the agent to give up the goal prematurely or find alternatives.

Regarding the work on anticipatory thinking that served as inspiration for this research, our contribution relies on exploiting the policy of several agents. As well as proposed in [Amos-Binks and Dannenhauer 2019], we propose a metacognitive process, and, as proposed by [Jones and Laird 2023], the nature of this process is similar to normal agent reasoning. Our agent places a clone of itself in a "matrix" together with the clones of others and "sees" the outcomes (metacognition).

7. Conclusions and Future Work

Departing from a proposal of an agent capable of reasoning about the future using simulation, the contribution of this paper is to start investigating the consequences of having several agents acting in a shared environment. Not only the decisions of this agent have to be simulated, but also the decision of others. The conclusion so far is that in some configurations the agent can anticipate and avoid future problems. One of this configurations is characterised by other agents *not* reasoning about the future.

In future work, we plan to address some limitations initially considered for the scope of the research presented in this paper.

- The requirement of an environment model. This requirement is quite common in techniques that simulate the future. In some application, however, it is difficult to obtain this model [Söderström and Stoica 1989]. One alternative is to learn an environment model, for example, by trial-and-error experimentation, as in reinforcement learning [Sutton and Barto 1998].
- The requirement of the model of other agents. We plan to apply machine learning to learn other agents' policies from their behaviour.
- Experiment with other scenarios, possibly more complex than the bridge scenario. These new scenarios should comprise cooperative and competitive cases.
- Investigate how to include coordination actions in the options given by a policy. With these special actions, we might solve configurations that are not solved so far. These special actions could also consider communicative actions.
- In the matrix, consider agents that also reason about the future. Currently, the clones used inside the matrix chose their first option without considering alternatives.
- Consider the case where our agent, instead of foreseeing a problem for itself, realises a future situation that it knows that is not desired by others. This agent can thus share its anticipation with others in a cooperative attitude.
- Experiment the computational complexity of the proposal as the number of agents increase.
- Relax some current assumptions: deterministic, static, and fully observable environment; irrelevance of action order/concurrency.

Acknowledgment

This work was produced at the research group headed by Professor Alessandro Ricci at the University of Bologna during the sabbatical of Professor Jomi F. Hübner. He expresses gratitude to the team for the inspiring conversations that took place at that time, which results include this paper. The sabbatical year was partially funded by Brazilian Agencies for Higher Education and Research: CNPq and CAPES, under the project PrInt CAPES-UFSC "Automation 4.0".

References

Amos-Binks, A. and Dannenhauer, D. (2019). Anticipatory thinking: A metacognitive capability. In *Proc. of the Workshop on Cognitive Systems for Anticipatory Thinking*.

Amos-Binks, A., Dannenhauer, D., and Gilpin, L. H. (2023). The anticipatory paradigm. *AI Magazine*, 44(2):133–143.

Baitiche, H., Bouzenada, M., and Saïdouni, D. E. (2017). Towards a generic predictive-based plan selection approach for bdi agents. *Procedia Computer Science*, 113:41–48. The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops.

Boissier, O., Bordini, R., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747–761.

Bordini, R. H., Hübner, J. F., and Wooldrige, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. John Wiley & Sons, England.

Bordini, R. H., Visser, W., Fisher, M., Pardavila, C., and Wooldridge, M. (2003). Model checking multi-agent programs with CASP. In *Pre-proceedings of Third Workshop on Automated Verification of Critical Systems (AVoCS 2003), 2–3 April, Southampton, U.K.*

Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge.

Cohen, P. R. and Levesque, H. J. (1987). Intention = choice + commitment. In *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 410–415, Cambridge. Morgan Kaufmann.

da Rocha Costa, A. C., Hübner, J. F., and Bordini, R. H. (1994). On entering an open society. In *Anais do XI Simpósio Brasileiro de Inteligência Artificial*, pages 535–546, Fortaleza.

Dann, M., Thangarajah, J., and Li, M. (2023). Feedback-guided intention scheduling for bdi agents. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS'23*, pages 1173–1181.

Dastani, M. (2008). 2APL: a practical agent programming language. *Autonomous Agent and Multi-Agent Systems*, 16:241–248.

Ferrando, A. and Cardoso, R. C. (2023). Failure handling in BDI plans via runtime enforcement. In Gal, K., Nowé, A., Nalepa, G. J., Fairstein, R., and Radulescu, R., editors, *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 716–723. IOS Press.

Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In *Proc. of the Sixth National Conference on Artificial Intelligence (AAAI 87)*, pages 677–682.

Ghallab, M., Nau, D., and Traverso, P. (2016). *Automated Planning and Acting*. Cambridge University Press, Cambridge.

Hindriks, K. V. (2009). Programming rational agents in GOAL. In Bordini, R. H., Dastani, M., Dix, J., and Seghrouchni, A. E. F., editors, *Multi-Agent Programming*, pages 119–157. Springer, Cham.

Hübner, J. F., Burattini, S., Ricci, A., and Mayer, S. (2025). Reflexive anticipatory reasoning by BDI agents. *Auton. Agents Multi Agent Syst.*, 39(1):7.

Jones, S. J. and Laird, J. E. (2023). A cognitive architecture theory of anticipatory thinking. *AI Magazine*, 44(2):155–164.

Kamali, M., Dennis, L. A., McAree, O., Fisher, M., and Veres, S. M. (2017). Formal verification of autonomous vehicle platooning. *Science of Computer Programming*, 148:88 – 106. Special issue on Automated Verification of Critical Systems (AVoCS 2015).

Laird, J. E. (2019). *The SOAR Cognitive Architecture*. The MIT Press, Cambridge.

Logan, B., Thangarajah, J., and Yorke-Smith, N. (2017). Progressing intention progression: A call for a goal-plan tree contest. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS'17*, pages 768–772.

Meneguzzi, F. and De Silva, L. (2015). Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning. *The Knowledge Engineering Review*, 30(1):1–44.

Moan, R. A., McBeth, C., Morales, M., Amato, N. M., and Hauser, K. (2024). Experience-based multi-agent path finding with narrow corridors. In *Robotics: Science and Systems*.

Pokahr, A., Braubach, L., and Lamersdorf, W. (2005). Jadex: A BDI reasoning engine. In Bordini, R. H., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., editors, *Multi-Agent Programming: Languages, Platforms, and Applications*, number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations, chapter 6, pages 149–174. Springer, Cham.

Rao, A. S. and Georgeff, M. P. (1995). BDI agents: from theory to practice. In Lesser, V., editor, *Proceedings of the First International Conference on MultiAgent Systems (ICMAS'95)*, pages 312–319, San Francisco, USA. AAAI Pess.

Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 3 edition.

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

Silva, L. d., Meneguzzi, F., and Logan, B. (2020). BDI agent architectures: A survey. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4914–4921.

Silver, D. (2021). Cooperative pathfinding. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 1(1):117–122.

Singh, D., Sardina, S., and Padgham, L. (2010). Extending BDI plan selection to incorporate learning from experience. *Robotics and Autonomous Systems*, 58(9):1067–1075. Hybrid Control for Autonomous Systems.

Söderström, T. and Stoica, P. (1989). *System identification*. Prentice Hall, Saddle River, New Jersey.

Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., and Boyarski, E. (2021). Multi-agent pathfinding: Definitions, variants, and benchmarks. *Proceedings of the International Symposium on Combinatorial Search*, 10:151–158.

Surynek, P. (2022). Problem compilation for multi-agent path finding: a survey. In Raedt, L. D., editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5615–5622. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Bradford, Cambridge.

Szpunar, K. K., Spreng, R. N., and Schacter, D. L. (2014). A taxonomy of prospection: Introducing an organizational framework for future-oriented cognition. *Proceedings of the National Academy of Sciences*, 111(52):18414–18421.

Thangarajah, J., Padgham, L., and Winikoff, M. (2003). Detecting & Avoiding Interference Between Goals in Intelligent Agents. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 721–726.

Uez, D. M. and Hübner, J. F. (2017). Modeling for openness in MAS. In Fallah-Seghrouchi, A. E., Son, T. C., and Ricci, A., editors, *Pre Proc. of 5th International Workshop on Engineering Multi-Agent Systems (EMAS@AAMAS 2017)*, pages 174–189.

Wang, S., Xu, H., Zhang, Y., Lin, J., Lu, C., Wang, X., and Li, W. (2025). Where paths collide: A comprehensive survey of classic and learning-based multi-agent pathfinding. https://arxiv.org/abs/2505.19219.

Yao, Y., Logan, B., and Thangarajah, J. (2014). Sp-mcts-based intention scheduling for bdi agents. In *Proceedings of the Twenty-First European Conference on Artificial Intelligence*, ECAI'14, pages 1133–1134, NLD. IOS Press.