Integrating Late Variable Binding with SP-MCTS for Efficient Plan Execution in BDI Agents

Frantisek Vidensky[1*][a], Frantisek Zboril[1*][b] and Petr Veigend [1][c]
[1]*Department of Intelligent Systems, Brno University of Technology, Bozetechova 2, Brno, Czech Republic*
[*]*These two authors contributed equally to this work*
{*ividensky, zborilf, veigend*}*@fit.vut.cz*

Keywords: BDI Agents, Agent Interpretation, AgentSpeak(L), Monte Carlo Tree Search

Abstract: This paper investigates the Late binding strategy as an enhancement to the SP-MCTS algorithm for intention selection and variable binding in BDI (Belief-Desire-Intention) agents. Unlike the Early binding strategy, which selects variable substitutions prematurely, Late binding defers these decisions until necessary, aggregating all substitutions for a plan into a single node. This approach reduces the search tree size and enhances adaptability in dynamic environments by maintaining flexibility during plan execution. We implemented the Late binding strategy within the FRAg system to validate our approach and conducted experiments in a static maze task environment. Experimental results demonstrate that the Late binding strategy consistently outperforms Early binding, achieving up to 150% higher rewards, particularly for the lowest parameter values of the SP-MCTS algorithm in resource-constrained scenarios. These results confirm that it is feasible to integrate Late binding into intention selection methods, opening opportunities to explore its use in approaches with lower computational demands than the SP-MCTS algorithm.

1 INTRODUCTION

The Belief-Desire-Intention (BDI) model (Rao and Georgeff, 1995) represents a dominant paradigm in agent development. Inspired by Bratman's theory of intentions (Bratman, 1987), BDI agents mimic human cognitive processes, enabling sophisticated reasoning, planning, and decision-making capabilities. This process, referred to as *practical reasoning*, involves selecting goals and determining the means to achieve them (Wooldridge, 1999).

In BDI-based agent programming languages (d'Inverno et al., 1998; Winikoff, 2005; Pokahr et al., 2005; Rao, 1996; Bordini et al., 2007), the agent's behaviour is defined by three key mental attitudes: beliefs, desires, and intentions. Beliefs represent the agent's information about its environment and itself. Desires capture the states the agent aims to achieve, while intentions embody the agent's commitments to specific actions or plans for achieving its desires. Plans are the means by which the agent modifies its environment to achieve its goals. A plan is comprised of steps that may include primitive actions directly altering the environment or subgoals addressed by other plans.

The execution of a BDI agent adheres to a repeated *deliberation cycle*. This cycle involves updating the agent's beliefs and goals to reflect the current environment, selecting plans for achieving active goals, and executing the next step of the chosen plans. For each top-level goal, a plan is selected, forming the root of an intention, and its steps are sequentially executed. If a step corresponds to a subgoal, a sub-plan is chosen and added to the intention, and the process continues recurrently.

Most researchers aiming to improve practical reasoning focus on addressing the *intention selection problem*. Practical reasoning, as realized in BDI agents, is operationalized through the deliberation cycle. The *intention selection problem* refers to the challenge of determining which intention to progress during the current deliberation cycle.

In many BDI architectures, intentions are executed in an interleaved manner (Winikoff, 2005; Bordini et al., 2007), enabling concurrent processing but introducing potential conflicts when steps in one intention block others. Researchers have proposed various strategies to address these conflicts, including

[a] https://orcid.org/0000-0003-1808-441X
[b] https://orcid.org/0000-0001-7861-8220
[c] https://orcid.org/0000-0003-3995-1527

Summary Information-based techniques (Thangarajah et al., 2003; Thangarajah et al., 2011), which reason about pre- and post-conditions, Coverage-based approaches (Thangarajah et al., 2012; Waters et al., 2014; Waters et al., 2015), which prioritize intentions most at risk due to environmental changes, and stochastic methods like Single-Player Monte Carlo Tree Search (SP-MCTS) (Yao et al., 2014; Yao and Logan, 2016), which optimize intention selection through simulation.

Our research group has taken a different approach, focusing on selecting variable substitutions in the well-known BDI programming language, AgentSpeak(L) (Rao, 1996). Its most widely used interpreter, Jason (Bordini et al., 2007), employs the *Early binding strategy* for selecting variable substitutions by default. This means that substitutions are selected during the evaluation of context conditions or the performing of test goals. In contrast, our proposed *Late binding strategy* (Zboril et al., 2022; Vidensky et al., 2023) defers the selection of variable substitutions until necessary, such as during the execution of actions. Until that point, the agent maintains a structure called the *context*, which contains all valid substitutions. This structure is dynamically updated as the plan is executed.

Both strategies have been implemented in the Flexibly Reasoning BDI Agent (FRAg)[1] system, and experimental evaluations (Vidensky et al., 2024) demonstrated that, despite increased computational overhead, the Late binding strategy outperformed the Early binding strategy in most scenarios. These results highlight the potential of Late binding for improving adaptability in dynamic settings.

Our previous work (Vidensky et al., 2025) extended the FRAg system by implementing a failure-handling mechanism inspired by the CAN (Sardina and Padgham, 2011) system. Experiments revealed that when combined with the Late binding strategy, the failure handling mechanism achieved better results than the Early binding strategy. Moreover, these results showed that the Early binding strategy can be effectively integrated with existing approaches. Building on these findings, we incorporated an SP-MCTS-based approach (Yao and Logan, 2016) for intention selection into the FRAg system. This approach is considered a state-of-the-art solution for addressing the intention selection problem.

In this paper, we explore the integration of the Late binding strategy into the SP-MCTS algorithm for intention selection in BDI agents. Section 2 introduces the Late binding strategy and its advantages over Early binding. Section 3 describes the SP-MCTS algorithm and its adaptation for action-level intention selection. Section 4 discusses the limitations of SP-MCTS, while Section 5 analyses the potential of Late binding to address computational challenges in intention selection. The experimental evaluation is presented in Section 6, and the paper concludes with a summary of findings and future research directions in Section 7.

[1] https://github.com/VUT-FIT-INTSYS/FRAg

2 LATE VARIABLE BINDING

In BDI agent systems, variable substitution plays a crucial role in ensuring effective and adaptive decision-making. Traditionally, BDI agents employ an *Early binding strategy*, where variable substitutions are determined when evaluating plan context conditions or test goals (Rao, 1996). While straightforward, this approach can lead to failures in dynamic environments, as it lacks the flexibility to adapt to changes occurring after plan selection.

The *Late binding strategy*, introduced in (Zboril et al., 2022), with its operational semantics detailed in (Vidensky et al., 2023), defers variable substitutions until they are strictly necessary, such as during the execution of actions. Instead of binding variables at an early stage, the strategy maintains all potential substitutions within a structure called the *context*. This context is dynamically updated during plan execution, discarding substitutions that no longer satisfy the agent's belief base or runtime conditions. By preserving valid options throughout the plan's lifecycle, the agent can adapt its behaviour to environmental changes without restarting plans unnecessarily.

The context is established when a plan is selected as part of an agent's intention. Unlike early binding systems, where variable substitutions are applied immediately, late binding systems maintain all possible variable substitutions. The context is represented as a set of possible substitutions called Possible Unifier Set (PUS) (Zboril et al., 2022), which encompasses all substitutions consistent with the agent's current belief base and the context conditions of the selected plan.

During plan execution, the context is continuously updated to reflect changes in the environment or the agent's beliefs. This process, known as the *restriction* (Zboril et al., 2022), systematically removes invalid substitutions. So, to be more precise, the restriction operation reduces the PUS to contain only valid substitutions Specifically, the restriction operation is applied in the following cases:

- **Test Goals:** When encountering a test goal, the context is restricted to retain only substitutions for

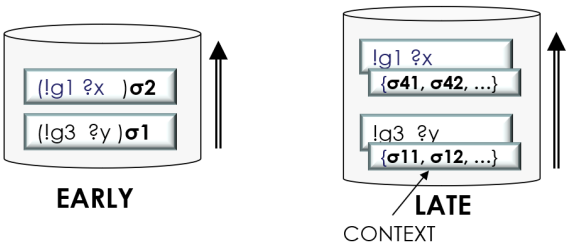


Figure 1: A comparison of Early and Late Binding Strategies. The left side illustrates Early binding, where intentions consist of stacks of plans with partially instantiated variables. In contrast, the right side illustrates Late binding, which retains uninstantiated plan bodies while maintaining variable substitutions separately within a context. Taken from (Vidensky et al., 2025).

which the tested predicate is true in the current belief base. This involves a *broad unification operation* (Zboril et al., 2022), which identifies all substitutions that unify the test goal with the agent's beliefs.

- **Actions:** When performing an action, the context is restricted to include only substitutions consistent with the action by binding free variables to specific atoms

By deferring the selection of variable substitutions until strictly necessary, late binding minimizes the risk of premature decisions that could lead to plan failure. This method allows the agent to dynamically adapt to environmental changes without restarting its plans.

This dynamic updating mechanism significantly enhances the robustness of BDI agents by reducing plan failures and improving adaptability in dynamic environments. As illustrated in Figure 1, early binding relies on partially instantiated plans, making intentions more rigid and prone to failure in volatile conditions. Conversely, late binding dynamically selects substitutions, enabling agents to respond effectively to changes without compromising the progress of their intentions.

3 ACTION-LEVEL INTENTION SELECTION WITH SP-MCTS

The Single-Player Monte Carlo Tree Search (SP-MCTS) algorithm (Schadd et al., 2008) has been adapted for use in BDI agents as a method for optimizing intention scheduling (Yao et al., 2014). This adaptation involves applying SP-MCTS to the goal-plan tree (GPT) (Thangarajah et al., 2011) structure instead of traditional game states. The goal-plan tree is a hierarchical representation of the agent's decision space. Goals are represented as parent nodes, with plans associated with their children. Subgoals within these plans expand into additional child nodes, creating a layered structure that captures the dependencies between goals and plans. By simulating various sequences of plan execution, SP-MCTS evaluates their effectiveness in achieving goals, making it a suitable approach for complex decision-making scenarios in dynamic environments.

SP-MCTS operates in iterative cycles, each consisting of four key steps:

- **Selection:** Starting from the root node, the algorithm selects a child node using an adapted version of the Upper Confidence Bounds for Trees (UCT) (Kocsis and Szepesvári, 2006) formula. This modification balances exploration (searching unexplored nodes) and exploitation (focusing on promising nodes) in the context of BDI decision-making.
- **Expansion:** A new node is added to the tree by selecting an unexplored action or goal from the current node.
- **Simulation:** A simulated sequence of actions is performed from the newly added node, using heuristic-based action selection to estimate outcomes.
- **Backpropagation:** The simulation results are propagated back through the selected path in the tree, updating the values of visited nodes.

This cycle allows SP-MCTS to iteratively refine its decision-making process, building a tree that captures the agent's possible future states and their associated outcomes. The algorithm uses two key parameters: α, which determines the number of node expansions (iterations), and β, which specifies the number of simulations conducted at each node. These parameters enable fine-tuning of the algorithm's performance, balancing computational effort and the quality of decision-making.

While the original SP-MCTS approach applied to entire plans, its adaptation in the Action-Level Intention Selection (SA) algorithm (Yao and Logan, 2016) introduces significant enhancements. Specifically, SA modifies the goal-plan tree (GPT) to include not only goals and plans but also primitive actions within those plans. This refinement enables action-level interleaving, addressing conflicts between steps of different plans more effectively.

By simulating these action sequences within the enhanced goal-plan tree, the SA algorithm identifies potential conflicts and selects the most promising actions to achieve goals efficiently. Each node in the tree represents a unique state of the environment and

the agent's progress. Simulations evaluate the potential outcomes of different action sequences, and the results are backpropagated to refine the tree. This enables the SA algorithm to dynamically adapt its decision-making process to changing environmental conditions.

These advancements underline the versatility of SP-MCTS in addressing conflicts and enhancing intention selection in BDI agents. Recent research has further explored the use of SP-MCTS in multi-agent environments (Dann et al., 2020; Dann et al., 2021; Dann et al., 2022). These extensions demonstrate the adaptability of SP-MCTS and its potential for broader applications in dynamic and collaborative scenarios.

4 LIMITATIONS AND CHALLENGES OF THE SA ALGORITHM

While innovative and effective in addressing conflicts at the action level in single-agent environments, the SA algorithm presents several limitations and challenges that must be considered for practical implementation.

One notable limitation is the necessity of explicitly defining pre- and post-conditions for actions. These conditions are crucial for evaluating the feasibility and consequences of action sequences during simulations. However, in programming environments like AgentSpeak(L), which do not natively require such specifications, this represents a significant departure from standard practices and could increase the development effort.

Another limitation lies in the reliance on propositional logic for goal-plan trees. While this approach is computationally efficient, it limits expressiveness in scenarios requiring more complex reasoning. Extending the algorithm to support predicate logic would allow for richer representations and introduce significant computational overhead, making its scalability more challenging.

Finally, the computational overhead associated with the algorithm, particularly the effort required to build and traverse the SP-MCTS search tree, is non-negligible. Although the authors claim that the complexity is manageable in their scenarios, dynamic environments requiring frequent updates may amplify this overhead, necessitating trade-offs between computational efficiency and the optimality of intention selection.

5 GOAL-PLAN TREE FOR LATE VARIABLE BINDING STRATEGY

Using the Goal-Plan Tree (GPT) structure for programs based on predicates can be computationally expensive, as it requires creating a separate node for each possible variable substitution. This approach, corresponding to Early binding, increases the number of nodes in the GPT, leading to higher computational costs in the SP-MCTS algorithm.

In contrast, the Late binding strategy aggregates all substitutions for a plan into a single node, reducing the number of nodes that must be traversed and simulated. This approach enhances computational scalability by minimizing redundant evaluations of invalid substitutions and makes the strategy particularly suitable for scenarios with many possible substitutions.

During simulation, substitutions are dynamically selected from the *context*, which maintains these substitutions. This approach ensures efficient use of the computational budget defined by β, allowing the algorithm to explore multiple possibilities while reducing redundant simulations.

To illustrate this concept, consider a simple example of an agent participating in a trading market for collectable cards. The agent's goal is to find a matching offer to satisfy a given demand. A plan for the agent, written in AgentSpeak(L), can be expressed as follows:

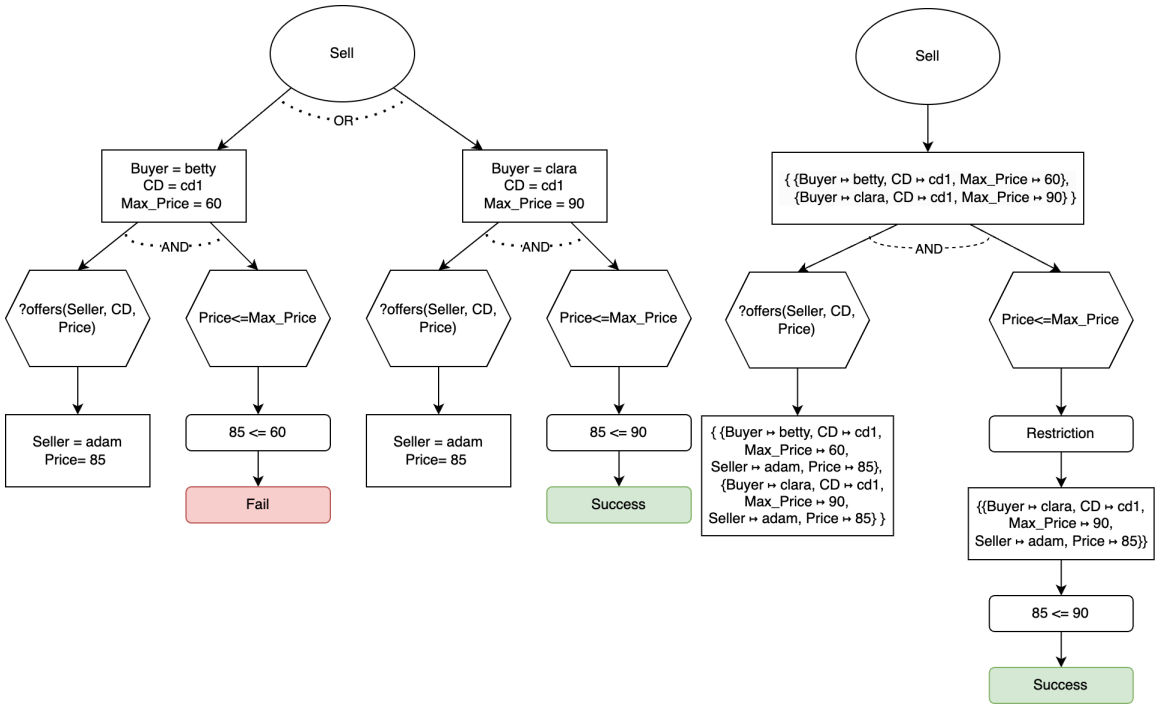
```
+!sell :wants(Buyer, Card, Max_Price)
  <- ?offers(Seller, Card, Price);
     Price<=Max_Price;
     sell(Seller, Buyer, Card, Price);
     !sell.
```

The environment includes one seller, *adam*, and two buyers, *betty* and *clara*, interested in the same card. The agent's belief base (BB) is defined as:

```
offers(adam, cd1, 85).
wants(betty, cd1, 60).
wants(clara, cd1, 90).
```

The example GPT can be seen in Figure 2. The trees are simplified as there is only one plan; under the goal node, child nodes represent the variable bindings created while evaluating the context condition. The plan nodes are truncated at the action that may fail, excluding subsequent actions for simplicity.

Figure 2 illustrates the difference between Early and Late binding strategies using the goal-plan tree. In the Early binding strategy (Figure 2a), each substitution generates a separate node, resulting in a larger



(a) Early binding: Separate nodes for each substitution.

(b) Late binding: Unified node with all substitutions.

Figure 2: Illustration of GPT construction under Early and Late binding strategies. Early binding creates multiple nodes for each substitution, whereas Late binding generates a unified node.

tree. For example, the agent must evaluate substitutions for both *betty* and *clara*, even though *betty*'s plan will fail due to price constraints. This leads to unnecessary computational effort during simulations.

In contrast, the Late binding strategy (Figure 2b) aggregates all potential substitutions into a single node. During plan execution, a *restriction operation* systematically refines the context by removing substitutions that no longer satisfy the belief base or plan conditions. For example, when the price constraint fails, the invalid substitution for *betty* is discarded.

The Late binding strategy offers several advantages:

- **Reduced Tree Size:** By aggregating substitutions, Late binding decreases the number of nodes in the GPT, improving computational efficiency.
- **Dynamic Flexibility:** Substitutions are evaluated only when necessary, allowing the agent to adapt to environmental changes.
- **Improved Robustness:** By deferring decisions, Late binding reduces the likelihood of plan failures caused by premature substitutions.

We hypothesize that the Late binding strategy will achieve better results than the Early binding strategy under identical parameter settings for the SA algorithm. By reducing the number of nodes that need to be traversed, the algorithm can explore more aggregated nodes within the same number of steps, potentially yielding an optimal outcome. This hypothesis will be validated through experimental evaluation.

We further anticipate that leveraging the Late binding strategy for SP-MCTS will prove particularly beneficial in scenarios involving a large number of possible variable substitutions. In such cases, Early binding requires the generation of separate nodes for each substitution, leading to exponential growth in the size of the goal-plan tree due to the combinatorial nature of substitutions across multiple variables. Late binding addresses this by dynamically narrowing down substitutions during plan execution, avoiding unnecessary computational overhead and invalid paths. This dynamic and adaptive behavior aligns with the requirements of complex and dynamic environments, where decision-making must remain flexible and efficient. Consequently, Late binding is expected to enhance the scalability and robustness of SP-MCTS in addressing such challenges effectively.

6 EXPERIMENTAL EVALUATION

This section presents the experimental evaluation of the SP-MCTS algorithm, focusing on its performance with the Late binding strategy. The primary objective of these experiments is to highlight the advantages of reducing the number of nodes expanded and simulated by the algorithm and the resulting improvements in the quality of the achieved results. The evaluation was conducted within a static maze environment, using varying values for parameters α (number of iterations) and β (number of simulations per iteration).

6.1 Environment Description

The *maze* environment is designed as a grid-based world, where an agent performs tasks involving the collection of materials. Materials such as gold, silver, or bronze can be found in various positions across the grid, and the agent's goal is to collect specific combinations of these materials in a defined sequence. The environment applies specific rules for material degradation, where materials degrade to lower-value versions, and evaluates the agent's performance based on its ability to complete tasks.

Grid Layout The environment consists of a grid with fixed dimensions (6x6). Each position is uniquely identified by its coordinates `[X, Y]` and can be associated with a material. The materials present on the grid are:

- **Gold**: The highest-value material, which degrades into silver when picked.
- **Silver**: An intermediate material which degrades into bronze when picked.
- **Bronze**: When picked, the lowest value material degrades to dust.
- **Dust**: A neutral material with no value, representing the final stage of degradation.

Material Degradation Each time the agent picks a material, it degrades according to the following rules:

- *gold* $\rightarrow$ *silver* $\rightarrow$ *bronze* $\rightarrow$ *dust*.

For example, if the agent picks a gold material, it degrades to silver and remains in the same position, waiting for further collection. Once a material degrades to dust, it can no longer be collected or contributed to task completion.

Tasks and Goals The environment provides predefined tasks, each represented as a combination of three materials in a specific order. For example, a task might require the agent to collect three instances of gold `(gold, gold, gold)` or a combination of gold, silver and bronze `(gold, silver, bronze)`. After collecting three materials, the agent's collected combination is evaluated against the predefined tasks. The agent is rewarded with a point if the combination matches any tasks. After collecting three materials, the agent's collected combination is evaluated against the predefined tasks. The agent is rewarded with a point if the combination matches any tasks.

Agent Interaction The agent interacts with the environment using two primary actions:

- `go(Direction)`: Moves the agent to an adjacent position in the specified direction (*up*, *down*, *left*, *right*).
- `pick`: Picks the material at the agent's current position, adding it to the agent's collection bag and triggering material degradation.

The agent's behaviour is guided by its belief base, which includes information about its position, perceived materials, and the current task.

Evaluation and Restart When the agent collects three materials, its bag is evaluated. If the collected combination matches a predefined task, the agent is rewarded with one point. Regardless of success or failure, the agent's bag is reset, and a new round begins. This process continues until the agent completes all tasks or exhausts the available materials.

The maze environment tests the agent's ability to adapt strategies by balancing navigation, resource management, and task adherence for optimal performance.

Suitability of the Material Collection Task The material collection task was chosen for its ability to highlight the differences between Early and Late binding strategies effectively. Unlike Early binding, which prematurely commits to a specific task, Late binding dynamically narrows down the set of viable tasks based on current conditions, maintaining flexibility throughout the decision-making process. This is particularly evident when the agent collects material and evaluates which tasks remain achievable based on the collected prefix. Instead of adopting a single task immediately, Late binding refines the task space to those still compatible with the agent's current progress and belief base.

Table 1: Rewards achieved by Early and Late variable binding strategies for different parameter settings of the SP-MCTS algorithm (α and β).

α	β	**Early**	**Late**
5	5	2	5
10	5	3	5
10	10	5	6
15	10	5	5
15	15	5	6
20	15	4	6
20	20	6	7
25	20	6	7

6.2 Evaluation Results

The evaluation results for the Early and Late binding strategies are summarized in Table 1. The table highlights the rewards achieved for various parameter combinations. All experiments were conducted with a limit of 60 agent steps per simulation. The Late strategy consistently outperforms the Early strategy, particularly for smaller values of α and β.

6.2.1 Analysis

The experimental results demonstrate that the Late binding strategy consistently outperforms the Early binding strategy across most tested parameter configurations. This advantage is particularly evident under constrained resource settings. For example, with $\alpha = 5$ and $\beta = 5$, the Late binding strategy achieved a reward of 5 points, compared to only 2 points for the Early binding strategy, an improvement of 150%. This trend highlights the Late binding strategy's ability to explore the decision space more effectively within limited computational budgets.

While some anomalies were observed, such as occasional performance drops at higher parameter values, these can be attributed to the stochastic nature of SP-MCTS and the limited number of experimental runs. For example, a suboptimal random choice during simulation or an unfinished task within the allocated steps may have contributed to these results. Increasing the number of experimental repetitions would provide a more robust statistical basis for these findings.

Overall, the Late binding strategy demonstrates a clear advantage in strengthening higher-quality solutions and efficiently navigating the search space. These results strongly support the hypothesis that Late binding enhances performance in scenarios with restricted resources and dynamic environments.

7 CONCLUSIONS

This paper presented an extension to the FRAg system by integrating the Late binding strategy into the SP-MCTS algorithm within the FRAg system, targeting variable binding and intention selection in BDI agents. The experimental results demonstrated that the Late binding strategy significantly outperforms the Early binding strategy, particularly under conditions with limited computational resources or small parameter values for α (iterations) and β (simulations).

The Late binding strategy's ability to aggregate multiple variable substitutions into a single node reduces the size of the search tree. It facilitates a more focused exploration of the decision space. This approach is expected to improve adaptability in dynamic environments by maintaining a broader range of options for variable substitutions throughout the execution of the plan, allowing agents to respond more effectively to changes.

While SP-MCTS represents a state-of-the-art approach to intention selection, its computational demands can make it unsuitable for scenarios requiring rapid decision-making or environments with highly constrained computational resources. Future research could explore integrating Summary Information-based and Coverage-based approaches to mitigate these limitations, potentially offering a more balanced trade-off between efficiency and adaptability in such contexts.

Additionally, a comparative analysis of the FRAg system against other BDI frameworks in more complex task environments would provide valuable insights into its practical advantages and areas for improvement. This direction could further validate the proposed Late binding strategy's robustness and applicability across diverse scenarios.

ACKNOWLEDGEMENTS

This work has been supported by the internal BUT project FIT-S-23-8151. Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

REFERENCES

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons.

Bratman, M. (1987). *Intention, plans, and practical reason*. Harvard University Press.

Dann, M., Thangarajah, J., Yao, Y., and Logan, B. (2020). Intention-aware multiagent scheduling. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 285–293, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Dann, M., Yao, Y., Alechina, N., Logan, B., and Thangarajah, J. (2022). Multi-agent intention progression with reward machines. In Raedt, L. D., editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 215–222. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Dann, M., Yao, Y., Logan, B., and Thangarajah, J. (2021). Multi-agent intention progression with black-box agents. In Zhou, Z.-H., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 132–138. International Joint Conferences on Artificial Intelligence Organization. Main Track.

d'Inverno, M., Kinny, D., Luck, M., and Wooldridge, M. (1998). A formal specification of dmars. In *Intelligent Agents IV Agent Theories, Architectures, and Languages: 4th International Workshop, ATAL'97 Providence, Rhode Island, USA, July 24–26, 1997 Proceedings 4*, pages 155–176. Springer.

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pokahr, A., Braubach, L., and Lamersdorf, W. (2005). *Jadex: A BDI Reasoning Engine*, pages 149–174. Springer US, Boston, MA.

Rao, A. S. (1996). Agentspeak(l): Bdi agents speak out in a logical computable language. In Van de Velde, W. and Perram, J. W., editors, *Agents Breaking Away*, pages 42–55, Berlin, Heidelberg. Springer Berlin Heidelberg.

Rao, A. S. and Georgeff, M. P. (1995). Bdi-agents: From theory to practice. In Lesser, V. and Gasser, L., editors, *Proceedings of the First International Conference on Multiagent Systems (ICMAS)*, volume 95, pages 312–319, Menlo Park, CA. AAAI Press.

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

Schadd, M. P. D., Winands, M. H. M., van den Herik, H. J., Chaslot, G. M. J. B., and Uiterwijk, J. W. H. M. (2008). Single-player monte-carlo tree search. In van den Herik, H. J., Xu, X., Ma, Z., and Winands, M. H. M., editors, *Computers and Games*, pages 1–12, Berlin, Heidelberg. Springer Berlin Heidelberg.

Thangarajah, J., Padgham, L., and Winikoff, M. (2003). Detecting & avoiding interference between goals in intelligent agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, page 721–726, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Thangarajah, J., Sardina, S., and Padgham, L. (2012). Measuring plan coverage and overlap for agent reasoning. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, page 1049–1056, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Vidensky, F., Zboril, F., Beran, J., Koci, R., and Zboril, F. V. (2024). Comparing variable handling strategies in bdi agents: Experimental study. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART*, pages 25–36. INSTICC, SciTePress.

Vidensky, F., Zboril, F., Koci, R., and Zboril, F. V. (2023). Operational semantic of an agentspeak(l) interpreter using late bindings. In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART*, pages 173–180. INSTICC, SciTePress.

Vidensky, F., Zboril, F., Koci, R., and Zboril, F. V. (2025). Advanced evaluation of variable binding strategies in bdi agents with integrated failure handling. *Lecture Notes in Artificial Intelligence*. Accepted for publication.

Waters, M., Padgham, L., and Sardina, S. (2014). Evaluating coverage based intention selection. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '14, page 957–964, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Waters, M., Padgham, L., and Sardina, S. (2015). Improving domain-independent intention selection in bdi systems. *Autonomous Agents and Multi-Agent Systems*, 29(4):683–717.

Winikoff, M. (2005). *Jack™ Intelligent Agents: An Industrial Strength Platform*, pages 175–193. Springer US, Boston, MA.

Wooldridge, M. (1999). *Intelligent Agents*, page 27–77. MIT Press, Cambridge, MA, USA.

Yao, Y. and Logan, B. (2016). Action-level intention selection for bdi agents. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, page 1227–1236, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

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, page 1133–1134, NLD. IOS Press.

Zboril, F., Vidensky, F., Koci, R., and Zboril, V. F. (2022). Late bindings in agentspeak(l). In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, pages 715–724. INSTICC, SciTePress.