Efficient Selection of Consistent Plans Using Patterns and Constraint Satisfaction for Beliefs-Desires-Intentions Agents

Veronika Kurchyna*, Ye Eun Bae, Jan Ole Berndt and Ingo J. Timm
German Research Center for Artificial Intelligence (DFKI),
Cognitive Social Simulation, Behringstr. 21, 54296 Trier, Germany

Keywords: BDI Agent, Agent-Based Simulation, Agent Deliberation, Complexity Optimization.

Abstract: Agent-based models can portray complex systems that emerge from the actions of individual actors. The use of many agents with complex decision-making processes in a large action space is computationally intensive and leads to slow simulations. This work proposes an alternative approach to agent deliberation by pre-computing valid action sequences and simplifying decision-making at runtime to a linear problem. The method is demonstrated with pandemic self-protection as use case for an implementation of the concept. Additionally, a step-by-step guideline for application of this approch is provided.

1 INTRODUCTION

Agent-based models (ABMs) and multiagent systems (MAS) are widely used to study various complex real-life systems, addressing diverse questions ranging from industrial applications, such as agriculture and land use (Bert et al., 2011), to social phenomena, such as family planning (Rodermund et al., 2018) and health behaviour (Cuevas, 2020). Agents are designed to act autonomously while pursuing their goals, operating under constraints such as limited resources, interactions with other agents, or incomplete knowledge of their environment (Magessi and Antunes, 2015). To formalize the decision-making processes of individual agents, the Beliefs-Desires-Intentions (BDI) architecture is commonly used (Caillou et al., 2017).

The typical deliberation process of an agent involves taking into account its knowledge of the world and current state—alongside its desires to formulate intentions. These intentions represent the concrete actions the agent plans to take in order to achieve its objectives. A key challenge arises when conflicting behavioural options emerge. Various approaches have been proposed to address this issue and enhance the efficiency of agent deliberation. For instance, agents can prioritize their desires and find ways to reconcile the competing courses of action (Timm, 2004). A goal deliberation method is proposed, aiming at reducing computational resources and the burden of ensuring the consistency of agents' goal sets by making agents select their own goals based on the situation (Pokahr et al., 2005). Moreover, the ad-hoc computation of consistent and logical plans for agents is commonly used in practice despite being resource-intensive, particularly for models with large numbers of agents, such as used in digital twins in smart cities contexts (Caillou et al., 2017), as well as large action spaces (Tkachuk et al., 2023). Another attempt was made using a context-sensitive deliberation framework, which prompts agents to employ alternative deliberation methods, instead of following default behaviour, only when there is a relevant change in context. This approach helps reduce the computational costs of decision-making (Jensen et al., 2022).

In light of such challenge, this paper proposes an alternative approach that leverages best practices established in software engineering, so-called design patterns. By employing these techniques, we suggest pre-simulation generation of consistent plans for agents to follow. This approach eliminates the need for at-runtime filtering of options and consistency checks, providing a more efficient and effective means of agents' decision-making in ABMs.

We begin by introducing the fundamentals of ABMs, outlining the relevant software desing patterns applied in this work, and providing a brief overview of key terminology, along with examples of our method. We then delve into the procedure in detail, illustrating it with a practical use case and offering a step-by-step guide for applying the apporach to other contexts. Finally, the runtime efficiency of the proposed method

*Corresponding author

Kurchyna, V., Bae, Y. E., Berndt, J. O. and Timm, I. J.
Efficient Selection of Consistent Plans Using Patterns and Constraint Satisfaction for Beliefs-Desires-Intentions Agents.
DOI: 10.5220/0013141700003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1*, pages 333-341
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

is demonstrated using a model for choosing protective behaviours in the context of the COVID-19 pandemic with comparisons made to hypothetical alternative implementations of decision-making.

2 FOUNDATIONS

The foundation section of this paper introduces the basic principles of ABM and the BDI architecture with a particular focus on the formalisation of deliberation. Additionally, design patterns, specifically strategies, and their application within an ABM context are discussed.

2.1 Beliefs, Desires and Intentions

ABMs simulate complex systems using autonomous agents that interact with their environment and each other in a reactive or goal-directed manner. In such models, complex system-wide dynamics result from the individual-level behaviours of the agents (Wooldridge and Jennings, 1995). Different approaches for formulating agents exist, among which BDI is a widely used architecture to describe agents.

Agents operate on their *Beliefs*, which are composed of subjective observations, knowledge and assumptions, and *Desires*, which describe goal states agents wish to achieve. Based on their beliefs, agents identify possible courses of actions in an *Option Generation* function. These options are *filtered* and prioritized based on feasibility and compatibility with desires. Finally, agents select one option as *Intention* to fulfill a certain desire and act upon it. The resulting changes to the environment are fed back as sensor input, which leads to a revision of beliefs and checking if an agent remains committed to a chosen option (Saadi et al., 2020).

2.2 Design Patterns: Strategies

Design patterns serve the purpose of reusable, modular and maintanable software engineering and propose best practices in several domains of software. These design patterns are categorized by their purpose, such as object creation, composite structures and behaviour (Gamma et al., 1994). For this paper, *Behavioural Patterns* are important, with different means of defining program behaviour in a modular way. *Strategies* are the pattern that will be used to implement agent behaviour. The purpose of strategies is to provide variations of code which can be easily switched out programmatically without large, hard-coded condition-action structures (Christopoulou et al., 2012).

A strategy is typically implemented as an interface which defines methods, with each strategy implementing the required functions. As a result, executing objects require no knowledge of the inner workings of a strategy and preserve the principle of encapsulation (Wick and Phillips, 2002). The authors see a major benefit of using strategies for agent-behaviour in the possibility to have agents execute a list of strategies and move plausibility checks into the strategy instead of a higher-level code block with more hard-coded conditions.

2.3 Terminology and Definitions

To ensure a unified understanding and distinction of terminology used in this paper, the definitions are given as follows:

- **Strategy.** Using the term as used in the context of design patterns in software development, a strategy defines a *behaviour* or *action* which is executed if certain conditions are met (Christopoulou et al., 2012). Thus, a strategy is not a firm decision of specific actions that will be performed, but rather the willingness to perform them if the preconditions are met.
- **Behaviour.** While the strategies in this model determine *when* something should be done, behaviours define *what* needs to be done, such as increased adherence to safety recommendations or reduced contacts with others.
- **Behaviour Package.** A set of consistent strategies which covers all possible preconditions, ensuring that agent behaviour is well defined at each simulation step. Thus, a selected package resembles *Intentions* of an agent. In this context, we use the term **plan** synonymously.
- **Package Class.** Depending on the evaluation mechanism, several packages can be evaluated into the same (or a close) score which suggests high similarity between packages. Assuming that packages of the same class are equivalent means of equal value, agents can be indifferent and do not prefer a specific package, but rather a class from which any package can be chosen randomly. Programatically, these classes allow representing equivalent packages as elements of an unordered set.
- **Constraints.** A set of logical expressions which must all be evaluated to *True* for a plan to be valid (Brailsford et al., 1999).

Three types of constraints are likely to be encountered when modelling behaviours, which are exemplified with A, B and C as boolean variables representing three different strategies:

1. **Mutual Exclusivity.** $A \implies \neg B$, meaning that B can never co-occur with A.
2. **Co-Occurence.** $A \implies C$, meaning that whenever A holds true, C must necessarily also be true.
3. **Definedness.** $B \oplus C$, meaning that, using the XOR-Operator, either B or C must be true.

As shown in Table 1, the application of these constraints eliminates $5/8$ combinations, with each remaining table row being an executable behaviour package. Alternatively, the three sample constraints can also be expressed in a matrix, as shown in Table 2. 0 denotes mutual exclusivity and 1 forced co-occurence. Empty cells indicate independence between two variables. It is important to remember that this graphical representation cannot convey the definedness-requirement. However, this format can give an overview of necessary relationships and the graphical representation facilitates deriving corresponding logical constraints in a structured, communicable manner.

This approach is essentially a modified form of the *Boolean Satisfiability* (SAT) problem, which deals with the question of whether a given logic expression has at least one configuration of boolean values that evaluates to true. While the verification of one configuration is fast, an exhaustive search is an exponential endeavour. Our approach includes the solving of the *#-SAT* problem, which requires a full enumeration and evaluation of all possible boolean configurations that satisfy all constraints (Creignou and Hermann, 1996). As such, the complexity of the initial package generation and filtering is exponential. However, we also assume that the action space of agents is not only finite, but also small enough for the exponential nature of the underlying theoretical problem to be handleable. Further, we also assume that the action

Table 1: Truth Table summarising possible configurations and filtering out invalid rows.

	A	**B**	**C**	$((B\wedge\neg C)\vee(\neg B\wedge C))$ $\wedge(A \implies \neg B)\wedge(A \implies C)$
1	F	F	F	F
2	F	F	T	T
3	F	T	F	T
4	F	T	T	F
5	T	F	F	F
6	T	F	T	T
7	T	T	F	F
8	T	T	T	F

Table 2: The matrix of the example with mutual exclusivity between A and B, as well as necessary C in case of A.

	A	B	C
A		0	1
B	0		0
C	-	0	

space is largely static - no new actions or axioms are added between model runs. Thus, we can save the results of an exhaustive search to a file and perform the exponential enumeration only once.

3 USE CASE: PANDEMIC MODELLING

The use of ABMs has gained significant popularity in studying pandemics, particularly due to the ongoing COVID-19 pandemic (Lorig et al., 2021). It serves as an illustrative example of how agents need to handle intricate situations where a diverse range of actions exists, some of which may be partially or fully mutually exclusive. To model agent behaviours in response to the pandemic, researchers often rely on routines and needs models. However, applying the BDI architecture to formalize agent behaviours leads to an intricate decision process. Agents must possess the capability to identify and reject any combination of behaviours that are inconsistent or mutually exclusive. Thus, this use case is well suited for the demonstration of this strategy-based constraint approach.

In this study, we focus on a model that encompasses a wide range of behaviours available to agents. These behaviours include actions such as self-quarantine in case of illness, compliance with external quarantine orders, random rapid testing, testing in case of symptoms, avoiding crowded locations, and varying levels of contact frequency and preventive measures like wearing masks correctly and maintaining physical distancing during interactions.

We choose a total of 8 behaviours agents can choose from. However, since two of these behaviours have numeric levels (contact frequency and degree of preventive measures), they were coded into multiple binary choices to avoid fuzzy logic with increased bases for the exponential number of possible combinations. With a total of 14 strategies, there are over 16 thousand possible combinations of these behaviours that an agent could potentially implement or refrain from:

- Q_v: Agents will quarantine voluntarily if they develop disease symptoms
- Q_c: Agents will comply with a quarantine order due to an official positive test or contact tracing

- Cw_A: Agents will avoid crowded locations (with more than x visitors, assumed to be a limit at which distancing becomes no longer possible)
- T_r: Agents will occasionally randomly perform a rapid test
- T_s: Agents will perform a rapid test if they feel disease symptoms
- Q_T: Agents will quarantine voluntarily even if a rapid test is negative
- C_o: Agents cease all contacts entirely
- C_l: Agents have a low level of contacts during their daily routines
- C_m: Agents have a medium level of contacts during their daily routines
- C_h: Agents have a high level of contacts during their daily routine
- P_h: Agents put high efforts into preventing an infection or diseapse spread
- P_m: Agents put medium effort into protecting themselves and others
- P_l: Agents put low effort into protecting themselves and others of infection
- P_r: Agents refuse all types of protective behaviour

However, a substantial number of these combinations lacks practicality or coherence. For instance, an agent who restricts themselves to low contacts would never have a large number of contacts simultaneously, and an agent who willingly chooses to self-quarantine upon experiencing illness symptoms would be unlikely to refuse quarantine after receiving a positive test result.

Such inconsistencies can arise from conflicting goals. As such, the formalisation of forbidden or required combinations contributes to the resolution of goal conflicts in agents, since they can only choose from valid configurations without logical inconsistencies. While this does not solve the question of goal priorization, this step forbids attempts at satisfying conflicting goals through incompatible and contradictory actions.

3.1 Behavioural Packages

In classic BDI, agents construct a plan using a bottom-up approach, evaluating different options regarding feasibility, compatibility with own goals and possible outcomes. This is a process repeated by each agent multiple times over the course of a simulation, with agents typically operating on the same set of rules and action spaces under different individual conditions and desires.

Behavioural Packages advocate a top-down approach in which agents select from a list of behaviours, forming a consistent plan that can be executed. Using strategies, which are either enabled or disabled, agents gain flexibility in their decision-making by providing a larger array of choices than a manual selection. For that, one may also use aids such as different levels represented as individual behaviours - while this increases the number of available actions artificially, handling of different levels of the same behaviour type is easy with a set of binary, mutually exclusive decisions. Assuming that each behaviour can be both a consistent, repeated course of action (e.g. the number of contacts during work etc.) as well as the general willingness to do something (e.g. quarantining in case of illness symptoms), we can consider the deliberation step as a periodic re-evaluation of which behavioural combination to use.

As such, instead of deciding on individual behaviours, an agent would choose a package of behaviours, which corresponds to choosing a plan from the set of possible intentions. These sets are referred to as behavioural packages which must give unambiguous instructions to the agent. This means that, besides consistency, behavioural packages also must be *complete*: for behaviour types such as contacts and preventive measures, a plan in which nothing was chosen at all may be free from contradictions, but incomplete, for an agent's behaviour in all situations that may arise during simulation must be defined. As a result, one also must consider behaviour *types*, of which at least one level needs to be selected.

In the context of the BDI-architecture, as shown in Figure 1, pre-generated plans replace the option generation and filtering. Using a scoring or selection function, agents choose a plan from this existing set of options as their intention.

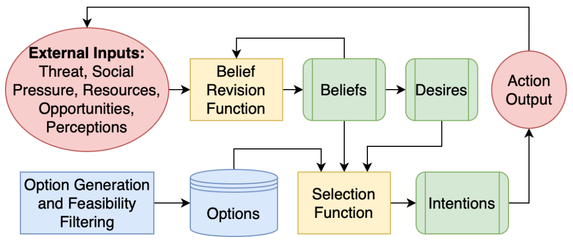
External Inputs: Threat, Social Pressure, Resources, Opportunities, Perceptions
Belief Revision Function
Beliefs
Desires
Action Output
Option Generation and Feasibility Filtering
Options
Selection Function
Intentions

Figure 1: Concept of how Behavioural Packages are integrated in the BDI cycle using a selection function.

3.2 Building the Matrix

For the generation of the set of valid plans, we can use a constraint satisfaction approach by first considering which behaviours are mutually exclusive, or logically co-occur. Table 3 summarises the differ-

Table 3: Matrix of behavioural combinations that always (1) or never (0) co-occur at the same time. Empty cells mean no rules for co-occurence are defined.

	Q_v	Q_c	Cw_A	T_r	T_s	Q_T	C_o	C_l	C_m	C_h	P_h	P_m	P_l	P_r
Q_v		1			1	1	0							0
Q_c							0							0
Cw_A							0							
T_r	1				1		0							0
T_s	1	1					0							0
Q_T	1	1		1	1		0							0
C_0	0	0	0	0	0	0		0	0	0	0	0	0	0
C_l							0		0	0				
C_m							0	0		0				
C_h								0	0					
P_h							0					0	0	0
P_n							0				0		0	0
P_l							0				0	0		0
P_r	0	0	0	0	0	0	0				0	0	0	

ent behavioural combinations and is to be read row-wise. Note that this matrix is not symmetrical, since some behaviours *can* co-occur, while the inversion may become a *must* co-occur, such as Q_v and Q_c. As an important disclaimer, it should be stated that this is merely a demonstration of how such a behavioural matrix can be set up, and further empirical research might be necessary to include probabilities of behavious co-occuring in real life. For the purpose of demonstration, however, we make basic common-sense assumptions.

The ranges $[0,1]$ score the probability of co-occurence, with 0 signifying mutual exclusivity and 1 meaning mandatory co-occurence.

3.3 Filtering Using Constraints

Once a matrix has been built, the generation of constraints may either be automated or, for small use cases, simply determined manually. For our example in Table 3, eight constraints are sufficient to express all conditions which a consistent plan must meet:

1. $Q_v \implies (Q_c \wedge T_s \wedge Q_T)$
2. $T_r \implies (Q_v \wedge T_s)$
3. $T_s \implies (Q_v \wedge Q_c)$
4. $Q_T \implies (Q_v \wedge Q_c \wedge T_r \wedge T_s)$
5. $C_0 \oplus C_l \oplus C_m \oplus C_h$
6. $C_0 \oplus P_h \oplus P_m \oplus P_l \oplus P_r$
7. $C_0 \implies \neg(Q_v \vee Q_c \vee Qw_A \vee T_r \vee T_s \vee Q_T \vee C_l \vee C_m \vee C_h \vee P_h \vee P_m \vee P_l \vee P_r)$
8. $P_r \implies \neg(Q_v \vee Q_c \vee Qw_A \vee T_r \vee T_s \vee Q_T \vee C_0 \vee P_h \vee P_m \vee P_l)$

By generating a 2^n binary table, with n being the number of possible behaviours, these constraints are used to filter out all rows with invalid combinations. After filtering, 58 behavioural packages remain.

3.4 From Plans to Intenions

The major benefit of our strategy-and-constraint approach is the pre-computation of valid action combinations into packages that define agent behaviour to reduce computational burden at runtime. However, agents still need to choose a plan which aligns with their desires, such as by computing utilities or using other means of selection. During the development of this method, several approaches have been explored by the authors, including decision trees, fuzzy sets and utility scoring. After careful examination of the advantages and disadvantages of each method, the plan selection function was not included as part of this approach, since the suitability of a given method depends on model-specific questions such as distinction between actions, data availability for comparison and empirical knowledge regarding decision mechanisms.

To complete the demonstration of this concept, this work will present a scoring-approach based on a psychological theory, which will allow agents to choose an appropriate behavioural package using constant-time operations as efficient mechanism which is unaffected by the size of the action space. In other words, the number of strategies, packages or agents will not affect on the calculation needed for each agent to choose a behavioural package, having the same number of computational steps.

4 CONCEPT SUMMARY

Before demonstrating the integration of behavioural packages into the agent decision mechanisms, we provide a summary of the individual steps to apply the proposed concept to an ABM:

1. **Define Behaviours.** Identify relevant behaviours agents can perform and the conditions under which it would make sense for them to execute those behaviours.
2. **Define Strategies and Levels.** Since strategies are binary, selected behaviours might have to be split into different, mutually exclusive strategies differing by the level/magnitutde of execution.
3. **Define Relationships.** For each combination of two strategies, define whether there exists mutual exclusivity or mandatory co-occurence.
4. **Define Constraints.** Define constraints to describe the relationships in the matrix using logic expressions. Remember that more complex conditions such as XOR might not be visualized.
5. **Filter Packages.** From a set of n_s^2 packages, filter out all that violate any of the defined con-

straints. Checking an individual combination for constraint satisfaction has a runtime of $O(n_s)$.

6. **Design Choice Mechanism.** Depending on the characteristics of the model, data availability and the action space, the choice mechanism can be designed accordingly, such as using decision trees, approximation heuristics or utility functions.
7. **Implementation.** Implement the strategies using design patterns as guideline. For higher efficiency, save valid packages to a file which is processed during model initialization. Implement the chosen mechanism for agents to choose the class of packages appropriate for their current state.

5 IMPLEMENTATION: CUSTOM SCORING FOR PANDEMIC SELF-PROTECTION

Using the same behavioural strategies as defined in Section 3, we build up a custom scoring of packages based on a psychological theory to demonstrate how a set of valid plans can be integrated into a model using constant-time operations.

The chosen psychological theory is the *Protection Motivation Theory* (PMT) (Hedayati et al., 2023), which delivers a framework for the explanation of seemingly irrational behaviour. When individuals confront a threat, they can either react adaptively to protect themselves, or deny the threat with maladaptive reactions. Factors such as fear, self-efficacy, perceived response costs and rewards are weighed up against each other to determine the reaction (Rogers, 1983). Previous works, such as Kurchyna et al. (2024) have demonstrated the application of PMT in the context of agents. Additionally, a variety of empirical studies regarding PMT and the COVID-19 pandemic is available to support modelling efforts (Hedayati et al., 2023).

Defining *Protection Motivation* as the difference between *Threat Appraisal* (Perceived danger) and *Coping Appraisal* (Ability to perform protective measures), the PMT offers a scale between adaptive (positive difference) and maladaptive (negative differendce) responses and yields a single score which allows identifying an individual's stance on the spectrum of possible responses.

To use the PMT for a custom scoring system, we take advantage of the adaptive-maladaptive axis and assign each behavioural package a rating which corresponds to its *package class* as defined in Section 2.3. This is achieved by attributing a partial score to each individual strategy, with the overall score of a package being the sum of strategies which are *true*. The scoring of individual actions is a debatable topic, since it is a subjective assessment: do scores reflect the *effectiveness* of an action, or the *effort* required? In this exemplary implementation, the scoring primarily reflects effort and attitude, rather than objective effectiveness. Table 4 displays the scores assigned to each strategy, which are not objective measures but evaluations that are relative to each other.

Table 4: (Mal-) Adaptivity Scores of Individual Strategies.

Strategy	Q_v	Q_c	Cw_A	T_r	T_s	Q_T	C_o
Score	2.5	1.0	1.5	0.5	3.0	3.0	3.0
Strategy	C_l	C_m	C_h	P_h	P_m	P_l	P_r
Score	2.5	-3	-5	2	1	0	-10

High scores indicate a high motivation to perform protective behaviours, while low or negative scores signify an inclination towards maladaptive behaviours. Packages of the same score will be considered as part of the same package class.

Assuming that the individual scores will only change rarely, the overall score can be included in the file saving valid strategy combinations. Additionally, by detaching the score computation from the identification of valid packages, new data regarding scoring can be used to replace previously computed scores without requiring a repeated identification of valid packages.

During runtime, each agent calculates their personal protection motivation score and chooses a behaviour package from the same package class that corresponds to their score. As mentioned in Section 3, the filtering of packages resolves conflicting goals by preventing invalid action combinations. Through the scoring, agents can prioritize different goals - agents who value their own safety will likely have high adaptivity scores. Social and conformist agents facing negative reactions from their surroudings will, due to high perceived costs of protective actions, likely veer towards maladaptive behaviours. Thus, goals that are based on needs and values are particularly suitable for this priorization method through their direct influence on the computation of the PMT score.

As a result, the complexity of decision-making is reduced to the computation of a singular score to choose which package class best matches the state of the agent.

5.1 Implementation of Strategies

While Section 2.2 introduces the concept of strategies in a general manner, a brief summary of how strategies are implemented in our model is provided to show an example of using this method. As shown in Table 5, agents have routines for actions and strategies which are applied to them. Strategies, generally, are tied to the executing agent (although a purely functional implementation with the agent as argument is possible) and whether they are currently enabled/active or not.

In the *Update Strategies* Function, the agent calls *Update* on each of its strategies, where additional logic may happen, such as state changes, triggering of immediate actions, or other logic which is state-dependent. One example from this use case is, in the case of Q_v, agents checking whether they are still feeling sick or if they may end their quarantine status. Another example is T_s, with check for disease symptoms resulting in the triggering and execution of a test-taking action. During the *Perform Routine*, the agent executes *Check Conditions* for each strategy and for each action until either one strategy yields a negative result (violation of conditions for execution) or all strategies evaluated successfully. For some strategies, such as T_r, *Check Conditions* will always yield *True*, since the strategy has no logical influence on individual actions in the context of routine behaviour.

5.2 Computational Complexity and Runtime Evaluation

While a classic approach with ad-hoc plan generation was not implemented, the pre-compilation of plans was tested and, using code runtime profiling, found to have no significant impact on the runtime performance of the simulation model due to the computation of agents' scores only using constant-time operations. This was tested on a model with 20.000 agents for 100 simulation steps.

Additionally, we also examined the runtime complexity of a hypothetical alternative implementation as visualised in Figure 2. The z-axis uses a logarithmic scale, since linear-scaled values are skewed beyond visibility by the range of z-values. In option **a**, each agent uses a greedy-search-strategy, identifies feasible plans, evaluates those using constant-time operations and selects the best option using linear algorithms. In comparison to that, **b** applies our method of binary strategies and filtering using constraints, followed by evaluation and selection. Finally, **c** depicts the proposed approach in which plans have been generated and filtered for feasibility ahead of simulation, rather than being a repeated step performed by each agent. Due to the exponential nature of the SAT problem, this package-based approach significantly outperforms greedy computation if the generation and filtering of plans are performed a single time ahead of simulations. As such, a mixed strategy still retains benefits regarding communication and modularity, but provides no better performance than ad-hoc planning without polynomial heuristics.

Table 5: Overview of Agents and the Strategy Interface.

Agent		StrategyInterface	
Id		*Name*	
Routine	List of Actions	*Agent*	Agent
Strategies	List of Strategies	*Active*	boolean
update_strategies()		update()	
perform_routine()		check_conditions(Action)	

These findings show that our method excels for models with large agent populations and large palettes of behaviours, since the computational effort primarily depends on the number of agents, rather than the size or complexity of the action space.

6 DISCUSSION

This contribution presents an efficient approach to agent-deliberation using pre-computation and strategy patterns as used in software design. With its integration within the BDI-architecture widely accepted by developers of ABMs, this method is compatible with existing standards and practices.

With the examination of voluntary self-protective behaviours during the COVID-19 pandemic, we demonstrated a practical implementation of this approach and compared it to ad-hoc planning by agents.

The main advantage of this method is twofold: as shown in Section 5.2, this method is efficient and scalable due to the one-time execution of the most resource-intensive task, the initial plan generation and filtering. Additionally, there is a strong abstraction between code, concept and meaning. Defining behaviours and building a matrix can be achieved in close interdisciplinary exchange with non-technical experts in a communicable and structured way.

This method's main downside is the requirement for expert knowledge to define possible behaviours and their relationships. With the current rise of Large Language Models and attempts at including semantic information in training data, reasoning on logical and inconsistent action combinations might be partially automated in the generation of constraints.

For actions with temporal dependencies, we can consider planning as NP-complete problem which can be translated into a satisfiability problem. Therefore, it is generally possible to include actions with temporal dependencies (Ullman, 1975). Due to the exponential nature of the #-SAT problem, the scaling for temporal sequences needs to be examined to verify performance against heuristic planning algorithms.

In general, performance is a major venue for fur-

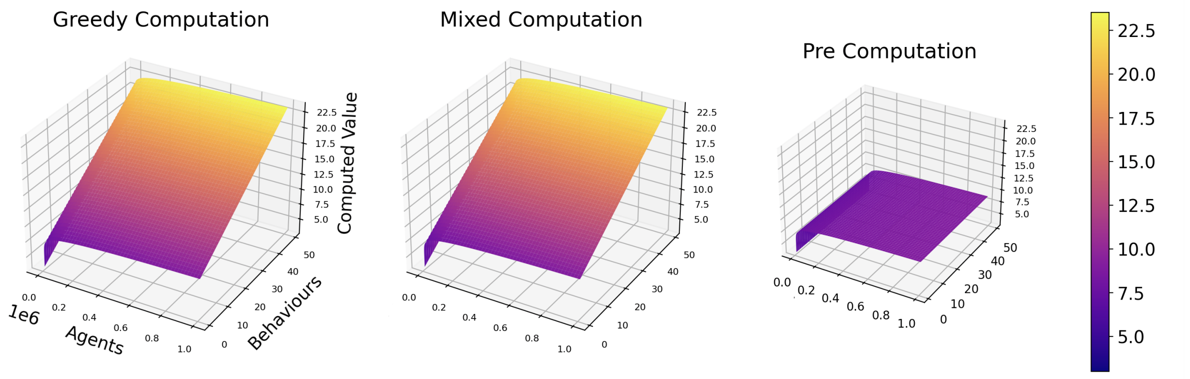

Figure 2: Comparison of Computation Steps depending on number of behaviours and agents using logarithmic scale for **(a)** greedy, **(b)** mixed and **(c)** pre-computation.

ther research: while we only presented package selection using a custom scoring system based on the PMT, there are also other methods for agents to select appropriate behavioural packages. Exploring different techniques, such as briefly adressed in Section 3.4, is an important contribution to understand the practical usefulness of this approach for use cases in which no appropriate pyschological theory with sufficient supporting data exists. Additionally, unoptimized planning was assumed in the comparison, without polynomial search heuristics in lieu of exhaustive search.

Moreover, further models with more complex agent behaviours, including interactions and temporal dependencies between behaviours, need to be investigated to examine the range of use cases that can be represented with our method.

Overall, the authors are optimistic that with more examples of successful usage, the method of precompiling plans using strategies provides opportunities to enhance the speed of large ABMs with complex agents displaying heteregoenous behaviours from a pre-evaluated selection of action combinations and sequences.

ACKNOWLEDGEMENTS

This work is the result of research in the context of the project SEMSAI, supported by the German Federal Ministry for Education and Research (BMBF) under the grant number 031L0295A.

REFERENCES

Bert, F. E., Podestá, G. P., Rovere, S. L., Menéndez, Á. N., North, M., Tatara, E., Laciana, C. E., Weber, E., and Toranzo, F. R. (2011). An agent based model to simulate structural and land use changes in agricultural systems of the argentine pampas. *Ecological Modelling*, 222(19):3486–3499.

Brailsford, S. C., Potts, C. N., and Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119(3):557–581.

Caillou, P., Gaudou, B., Grignard, A., Truong, C. Q., and Taillandier, P. (2017). A simple-to-use bdi architecture for agent-based modeling and simulation. In *Advances in Social Simulation 2015*, pages 15–28. Springer.

Christopoulou, A., Giakoumakis, E., Zafeiris, V. E., and Soukara, V. (2012). Automated refactoring to the strategy design pattern. *Information and Software Technology*, 54(11):1202–1214.

Creignou, N. and Hermann, M. (1996). Complexity of generalized satisfiability counting problems. *Information and Computation*, 125(1):1–12.

Cuevas, E. (2020). An agent-based model to evaluate the covid-19 transmission risks in facilities. *Computers in biology and medicine*, 121:103827.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Hedayati, S., Damghanian, H., Farhadinejad, M., and Rastgar, A. A. (2023). Meta-analysis on application of protection motivation theory in preventive behaviors against covid-19. *International Journal of Disaster Risk Reduction*, 94:103758.

Jensen, M., Verhagen, H., Vanhée, L., and Dignum, F. (2022). Towards efficient context-sensitive deliberation. In *Advances in Social Simulation: Proceedings of the 16th Social Simulation Conference, 20–24 September 2021*, pages 409–421. Springer.

Kurchyna, V., Rodermund, S. C., Bae, Y. E., Mertes, P., Flügger, P., Berndt, J. O., and Timm, I. J. (2024). Seeing through the smoke: An agent architecture for representing health protection motivation under social pressure. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, pages 315–325. INSTICC, SciTePress.

Lorig, F., Johansson, E., and Davidsson, P. (2021). Agent-based social simulation of the covid-19 pandemic: A systematic review. *Journal of Artificial Societies and Social Simulation*, 24(3).

Magessi, N. T. and Antunes, L. (2015). Modelling agents' perception: Issues and challenges in multi-agents based systems. In *Progress in Artificial Intelligence: 17th Portuguese Conference on Artificial Intelligence, EPIA 2015, Coimbra, Portugal, September 8-11, 2015. Proceedings 17*, pages 687–695. Springer.

Pokahr, A., Braubach, L., and Lamersdorf, W. (2005). A goal deliberation strategy for bdi agent systems. In *Multiagent System Technologies: Third German Conference, MATES 2005, Koblenz, Germany, September 11-13, 2005. Proceedings 3*, pages 82–93. Springer.

Rodermund, S. C., Berndt, J. O., and Timm, I. J. (2018). Social congtagion of fertility: An agent-based simulation study. In *2018 Winter Simulation Conference (WSC)*, pages 953–964.

Rogers, R. W. (1983). Cognitive and physiological processes in fear appeals and attitude change: A revised theory of protection motivation. *Social psychology: A source book*, pages 153–176.

Saadi, A., Maamri, R., and Sahnoun, Z. (2020). Behavioral flexibility in belief-desire- intention (bdi) architectures. *Multiagent and Grid Systems*, 16:343–377.

Timm, I. J. (2004). Dynamisches konfliktmanagement als verhaltenssteuerung intelligenter agenten. diski.

Tkachuk, V., Bakhtiari, S. A., Kirschner, J., Jusup, M., Bogunovic, I., and Szepesvári, C. (2023). Efficient planning in combinatorial action spaces with applications to cooperative multi-agent reinforcement learning. *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics,*.

Ullman, J. (1975). Np-complete scheduling problems. *Journal of Computer and System Sciences*, 10(3):384–393.

Wick, M. R. and Phillips, A. T. (2002). Comparing the template method and strategy design patterns in a genetic algorithm application. *SIGCSE Bull.*, 34(4):76–80.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152. Publisher: Cambridge University Press.