Implementing the Adaptation Procedure in Misinformation Games

Merkouris Papamichail
Institute of Computer Science,
Foundation for Research and
Technology - Hellas
Heraklion, Greece
mercoyris@ics.forth.gr

Constantinos Varsos
Institute of Computer Science,
Foundation for Research and
Technology - Hellas
Heraklion, Greece
varsosk@ics.forth.gr

Giorgos Flouris
Institute of Computer Science,
Foundation for Research and
Technology - Hellas
Heraklion, Greece
fgeo@ics.forth.gr

ABSTRACT

In this paper we consider scenarios where players interact having a possibly incorrect information regarding the interaction they participate. Building upon the recently proposed framework of misinformation games, we develop a time-discrete iterative procedure, where in each step players make their decisions according to the (possibly erroneous) information they have. Then the joint decision is revealed to anyone, alongside the payoffs it has for each player, according to actual specifications of the interaction. With this at hand, players update their information and (possibly) adapt their behaviour in the next step. We call this process the *Adaptation Procedure*. After the formal establishment of the Adaptation Procedure we implement this methodology using Logic Programming, and more specifically Answer Set Programming. Hence, we provide an integrated pipeline that analyses interactions where participants engulf new information and re-evaluate their behaviour.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent systems**; • **Theory of computation** → *Solution concepts in game theory*.

KEYWORDS

Misinformation games, Adaptation Procedure, stable misinformed equilibrium, Answer Set Programming

ACM Reference Format:
Merkouris Papamichail, Constantinos Varsos, and Giorgos Flouris. 2022. Implementing the Adaptation Procedure in Misinformation Games. In *Proceedings of 12th EETN Conference on Artificial Intelligence (SETN '22).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Multi-agent systems are ubiquitous in diverse scientific fields especially when real-life interactions among rational individuals need to be modeled and studied. Game theory has proved to be a useful tool in the analysis of such systems, where a usual assumption is that the game specifications (i.e., the rules of interaction) are common knowledge among the players, who have correct (although not necessarily complete) information regarding the game.

However, in many real-life interactions the correct information assumption does not hold. Usually, participants interact while suffering from bounded rationality, computational restrictions, and cognitive limitations, for various reasons; the agents may hold an incorrect belief about the environment due to misperception, deception and sensory or communication errors; or the environment may evolve without the players' knowledge. These are some of the reasons that may lead the players to incorporate incorrect knowledge regarding the actual interaction without being aware of it; in other words to be *misinformed*. Hence, in order to analyse multi-agent interactions in such settings, it is more realistic to consider that players have a subjective and misinformed view about reality [28].

Consequently, the study of settings where any participant has (possibly) wrong knowledge with regards to the real situation is important. Towards this direction, we use the concept of *misinformation games* as provided in [28], which fall under the hood of games with misperception, see [2, 10, 19]. In misinformation games, each player has a subjective view of the game's specifications (number of players, number of strategies, payoffs etc), which may not coincide with the specifications of the real interaction. Hence, the interactions are modeled considering both the real situation (*actual game*), and the *subjective (misinformed) views* of the players. A key characteristic of misinformation games is that the equilibrium (i.e., the set of strategic choices where no player wants to deviate from) is determined by the subjective views of the players, rather than the actual game; this is called the *natural misinformed equilibrium* in [28]. On the other hand, the payoffs received by the players for their actions are determined by the actual game.

The above observation introduces an element of surprise for the players, as they receive unexpected payoffs for their actions (i.e., those in the actual game, rather than those in their subjective views). A limitation of the approach of [28], is that this fact is not considered; in other words, the approach of [28] does not consider the *reaction* of the players upon the realisation that the received payoffs are different than expected. In some sense we can say that the framework of [28] addresses only one-shot games, but players' interactions are hardly ever one-shot, so it is important to consider their reactions as they observe unexpected outcomes.

To incorporate this element in misinformation games, we develop an iterative methodology, called the *Adaptation Procedure*, which models the evolution of the strategic behaviour of rational

SETN '22, September 07–09, 2022, Corfu, Greece

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

players in a misinformation game, as they obtain new information and update their (erroneous) game specifications.

We consider the following setting: time is discrete. At each time point, players choose a strategic action, receive the corresponding payoffs, and update their game specifications according to the payoff they receive. The procedure is then repeated. Importantly, the information received in each time point may lead players to a different choice in the next time point (because they now operate under a different payoff matrix), so the procedure is iterative and stabilises when the players have no incentives to deviate from their current choices, based on what they know so far.

Apart from defining the Adaptation Procedure theoretically (Section 4), a further contribution of this paper is the development (Section 5), implementation (Section 6), and experimental evaluation (Section 7) of an algorithm modelling the methodology.

2 RELATED WORK

There is an extensive literature regarding multi-agent systems with adaptive participants. Here, we present works that are closer to our setting, namely interacting agents with subjective views, where the views are updated during the interaction.

A significant stream of works aims in the analysis of players' beliefs in misspecified models (e.g., [1, 6–9, 13, 16, 24, 27]), where players update their beliefs using probabilistic formulas and techniques. Contrary to these works in our study we do not have any notion of beliefs.

In [1], a learning framework regarding objective and subjective games is provided, but with the limitation that players are completely ignorant regarding opponents' decisions. In our work, we drop the limitation regarding the opponents' decisions, and any new information is automatically integrated.

Further, in [18] a class of Bayesian processes for iterated normal form games is studied, where each player knows her own payoff function, but is uncertain about the opponents' payoffs. In [7] authors provide a time dependent learning mechanism in games where players have subjective views of an objective game. In the same manner, in [24], behavioral-based model are provided in order to model the ignorance of each player about the opponent. As opposed to the stream of works that rely on probabilistic or Bayesian techniques, in this work we take no probabilistic considerations regarding the beliefs of the players. Our study focuses on the effect of new information in the misinformed views of the players.

Authors in [7, 12] use the concept of Nash-Berk equilibrium, that is differ from stable misinformed equilibrium concept. Technically, they rely heavily in statistical mechanics tools, such as Kullback-Leibler divergence, as oppose to our study where we do not use any statistical tool. Further, we do not consider any randomization in our model (besides mixed strategies) in contrast with [7–9, 12], thus allowing all different outcomes that can be occur. This provides a substantial conceptual difference.

Another equilibrium concept closely related to stable misinformed equilibrium is that of self-confirming equilibrium [5, 11]. Although capture the idea of "stable" joint decisions in misspecified models, in our model players revise their information according to actual specifications, while in self-confirming case players revise their information according to the choices of their opponents.

A significant number of works focuses on the limit behaviour of players whose strategic choices are evolving, for various reasons. In [22] an example was presented where the agent's actions cycle ad infinitum. Authors in [25] based on a continuous time model, provide characterization of asymptotic behaviour. Moreover, in [15] a convergence analysis was introduced in case where players bias their observations, while in [16] authors establish convergence of beliefs, and actions, in a misspecified model with endogenous actions. Further, in [26] the author gives sufficient conditions for the convergence of the posterior without assuming that the subjective views are part of actual specifications. Also, in [12] a complete characterization of the limit behavior of actions in cases with misspecified Bayesian players is provided.

Besides that, in [13] authors introduce a swap learning method in 1-level Hypergames, where players engulf the information they gain by observing the opponents' actions, and decrease their misperception at the cost of potentially incurring inconsistencies in their perception. In our framework, we make no considerations as to the revealing inconsistencies. Additionally, authors in [3] provide an approach for learning in adaptive dynamic systems, where a player learns an efficient policy over the opponents' adaptive dynamics. However, they did not consider any concept of incorrect or subjective information.

An interesting approach proposed in [17] where evolutionary Hypergame dynamics were introduced, but the analysis constrained in prototypical cases, e.g., either two or three pure strategies per player. In our work, we consider bounded yet abstract number of pure strategies.

Finally, this paper provides a further important innovation. To the best of our knowledge, there is no other work in the area of misspecified models where the evolution of the strategic behaviour of the players is analysed using Logic Programming methods. Hence, the techniques, methodologies and pipelines we develop in order to study iterative interactions between misinformed players, are fundamentally different from any other stream of works. Consequently, there is no algorithmic methodology to, either theoretically or experimentally, compare it with our model.

3 PRELIMINARIES

We start by describing normal-form games, the most commonly-studied class of games [23]. A game in normal-form is represented by a *payoff matrix* that defines the payoffs of all players for all possible combinations of pure strategies:

Definition 3.1 (Normal-form games). A normal-form game G is a tuple $G = \langle N, S, P \rangle$, where:

- N is the set of players
- $S = S_1 \times \cdots \times S_{|N|}$, S_i is the set of pure strategies of player $i \in N$
- $P = (P_1; \ldots; P_{|N|})$, $P_i \in \mathbb{R}^{|S_1| \times \ldots \times |S_{|N|}|}$ is the payoff matrix of player $i \in \{1, 2, \ldots, |N|\}$

If player i has m_i pure strategies, then a strategy for player i is a tuple $\sigma_i = (\sigma_{i1}, \ldots, \sigma_{im_i})$ where $\sigma_{ij} \geq 0$ and $\sum_j \sigma_{ij} = 1$. We denote by $supp(\sigma_i) = \{s_{ij} \in S_i \mid \sigma_{ij} > 0\}$, i.e., the pure strategies chosen by the player with non-zero probability. In a game with $|N|$ players, a strategy profile is an $|N|$-tuple $\sigma = (\sigma_1, \ldots, \sigma_{|N|})$, where

σ_i is a strategy for player i. Analogously, $supp(\sigma) = supp(\sigma_1) \times \cdots \times supp(\sigma_{|N|})$.

The players' behaviour in a normal-form game is predicted through the *Nash* equilibrium:

Definition 3.2 (Nash equilibrium [21]). A strategy profile $\sigma^* = (\sigma_1^*, \ldots, \sigma_{|N|}^*)$ is a Nash equilibrium, if and only if, for any i and for any $\hat{\sigma}_i \in \Sigma_i$, $f_i(\sigma_i^*, \sigma_{-i}^*) \geq f_i(\hat{\sigma}_i, \sigma_{-i}^*)$, where f is the payoff function of player i and defined as: $f_i : \Sigma \to \mathbb{R}$, such that:

$$f_i(\sigma_i, \sigma_{-i}) = \sum_{k \in S_1} \cdots \sum_{j \in S_{|N|}} P_i(k, \ldots, j) \cdot \sigma_{1,k} \cdot \ldots \cdot \sigma_{|N|,j},$$

We denote with *ne* a Nash equilibrium and with $NE(G)$ the set of *ne*s in G. Misinformation games were introduced in [28] as follows:

Definition 3.3. A *misinformation normal-form game* (or simply *misinformation game*) is a tuple $mG = \langle G^0, G^1, \ldots, G^{|N|} \rangle$, where all G^i are normal-form games and G^0 contains $|N|$ players.

G^0 is called the *actual game* and represents the game that is actually being played, whereas G^i ($i \in N$) represents the game that player i thinks that is being played (called the *game of player i*).

In Definition 3.3, no assumptions are made as to the relation among G^0 and G^i, thereby allowing all types of misinformation to occur. Nevertheless, in [28] the case of *canonical misinformation games* was also studied, where the subjective views and the actual game differ only in the values of the elements of the payoff matrices (but not on the players or available strategies). Further, the authors provided a methodology that transforms any misinformation game into an equivalent canonical one. Hence, without loss of generality we will focus on canonical misinformation games here as well.

A key characteristic of misinformation games is the concept of the misinformed strategy and the consequent equilibrium solution:

Definition 3.4. A *misinformed strategy*, $m\sigma_i$ of a player i is a strategy of i in game G^i. We denote the set of all possible misinformed strategies of player i as Σ_i^i. A *misinformed strategy profile* of mG is an $|N|$-tuple of misinformed strategies $m\sigma = (m\sigma_1, \ldots, m\sigma_{|N|})$, where $m\sigma_i \in \Sigma_i^i$.

Definition 3.5. A misinformed strategy of player i, $m\sigma_i$, is a *misinformed equilibrium strategy* iff it belongs to a Nash equilibrium strategy profile in game G^i. A misinformed strategy profile $m\sigma$ is called a *natural misinformed equilibrium*, or *nme* for short, iff it consists of misinformed equilibrium strategies.

We denote by $NME(mG)$ the set of *nme*s in mG.

4 ADAPTATION PROCEDURE

4.1 Constructing the Adaptation Tree

As explained above, the Adaptation Procedure occurs in discrete time steps $t \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$. It starts from $t = 0$ where we have the misinformation game mG. Then, in each time step $t \geq 0$ the following happen:

- The players choose a strategy to play; we assume that the strategy chosen by each player is a natural misinformed equilibrium strategy.
- The players receive their payoffs and update their payoff matrices based on this payoff.

As we will explain later, the process is not necessarily linear, but may include branching, essentially resulting in a tree of misinformation games. To describe formally the above procedure, we will first define some notions that will be useful in the following.

Consider a multidimensional matrix A, and a vector $\vec{v}$. We denote by $A_{\vec{v}}$ the element of A in position $\vec{v}$. For example, $A_{(1,2)}$ is the top right element of a 2×2 matrix A. We define the operation of replacing element $A_{\vec{v}}$ with b as follows:

Definition 4.1. Consider a matrix $A \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_m}$, a vector $\vec{v}$ indicating a position in A and some $b \in \mathbb{R}$. We denote by $A \oplus_{\vec{v}} b$ the matrix $B \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_m}$, such that $B_{\vec{v}} = b$ and $B_{\vec{u}} = A_{\vec{u}}$ for all $\vec{u} \neq \vec{v}$.

The Adaptation Procedure leads to the replacement of an element in the subjective payoff matrices of players with the respective element in the actual payoff matrix; formally:

Definition 4.2. Consider a canonical misinformation game $mG = \langle G^0, G^1, \ldots, G^{|N|} \rangle$, where $G^i = \langle N, S, P^i \rangle$ (for $0 \leq i \leq |N|$), and some vector $\vec{v}$. We define the $\vec{v}$-update of mG, denoted by $mG_{\vec{v}}$, to be the misinformation game $\langle G^0, G^{1\prime}, \ldots, G^{|N|\prime} \rangle$, where $G^{i\prime} = \langle N, S, P^i \oplus_{\vec{v}} P^0_{\vec{v}} \rangle$, for $1 \leq i \leq |N|$.

Definition 4.2 tells us how to perform the update process that the Adaptation Procedure requires. The position where the update takes place (denoted by $\vec{v}$ in Definition 4.2) is determined by the strategic choices of the players, and can be "extracted" using the following definition:

Definition 4.3. Consider a strategy profile $\sigma = (\sigma_1, \ldots, \sigma_N)$, where $\sigma_i = (\sigma_{i1}, \ldots, \sigma_{im_i})$. The characteristic strategy set of a strategy σ_i is defined as $\chi(\sigma_i) = \{j | s_{ij} \in supp(\sigma_i)\}$. Abusing notation, we denote by $\chi(\sigma)$ the set $\chi(\sigma_1) \times \ldots \chi(\sigma_N)$.

To simplify notation, we will sometimes use χ over sets of strategy profiles (say Σ), and set $\chi(\Sigma) = \bigcup_{\sigma \in \Sigma} \chi(\sigma)$. Note that χ essentially returns the indices associated with the strategies in the support, as clarified in the following example:

Example 1. *Assume a* 4×3 *bimatrix game. Then, the characteristic strategy set of vectors of* $\sigma = ((1/2, 0, 1/3, 1/6), (0, 0, 1))$ *is* $\chi(\sigma) = \{(1, 3), (3, 3), (4, 3)\}$. □

The Adaptation Procedure is based on continuous applications of the $\vec{v}$-updating procedure (Definition 4.2), for the vector(s) $\vec{v}$ that correspond to the strategic choices of the players (i.e., $\vec{v} \in \chi(\sigma)$ for $\sigma \in NME(mG^{(t)})$).

Example 2 (Running Example). *Consider the misinformation game* $mG^{(t)} = \langle G^0, G^{1,(t)}, G^{2,(t)} \rangle$, *where* $G^{i,(t)} = \langle \{1, 2\}, S = \{s_1, s_2\}, P^{i,(t)} \rangle$, *with* $i \in \{1, 2\}$ *and*

$$P^0 = P^{1,(0)} = \begin{pmatrix} (2,2) & (0,3) \\ (3,0) & (1,1) \end{pmatrix}, \quad P^{2,(0)} = \begin{pmatrix} (1,1) & (3,0) \\ (0,3) & (2,2) \end{pmatrix}$$

In $t = 0$, *player* 1 *has equilibrium strategy* s_2 *in* $G^{1,(0)}$, *while player* 2 *has equilibrium strategy* s_1 *in* $G^{2,(0)}$. *Thus the nme corresponds to* (s_2, s_1) *and has strategy profile* $((0, 1), (1, 0))$. *Using the characteristic strategy vector we take* $\chi((0, 1), (1, 0)) = \{(2, 1)\}$.

The update operator gives $mG^{(1)} = \langle G^0, G^{1,(1)}, G^{2,(1)} \rangle$ *with the following payoff matrices (note how the bottom-left payoff has been*

updated):

$$P^{1,(1)} = \begin{pmatrix} (2,2) & (0,3) \\ (3,0) & (1,1) \end{pmatrix}, \quad P^{2,(1)} = \begin{pmatrix} (1,1) & (3,0) \\ (3,0) & (2,2) \end{pmatrix}$$

■

The fact that $\chi(\sigma)$ is not necessarily a singleton set complicates matters, and forces us to consider branches in the Adaptation Procedure:

Definition 4.4. For a set M of misinformation games, we set:

$$\mathcal{AD}(M) = \{mG_{\vec{u}} \mid mG \in M, \vec{u} \in \chi(\sigma), \sigma \in NME(mG)\}$$

We define as Adaptation Procedure of a set of misinformation games M to be the iterative process such that:

$$\begin{cases} \mathcal{AD}^{(0)}(M) = M \\ \mathcal{AD}^{(t+1)}(M) = \mathcal{AD}^{(t)}(\mathcal{AD}(M)) \end{cases}$$

for $t \in \mathbb{N}_0$.

Note that the Adaptation Procedure is defined over a set of misinformation games. Although our intent is basically to apply it over a single misinformation game, the branching process, along with the recursive nature of the definition, forces us to consider the more general case right from the start. Note also that we will often abuse notation and write $\mathcal{AD}(mG)$ (or $\mathcal{AD}^{(t)}(mG)$) instead of $\mathcal{AD}(\{mG\})$ (or $\mathcal{AD}^{(t)}(\{mG\})$).

The following example shows how the Adaptation Procedure of Example 2 continues in its second step. Interestingly, $mG^{(1)}$ includes a hybrid natural misinformed equilibrium, thus illustrating the branching process mentioned above.

Example 2. *[continued] In $t = 1$, player 1 has equilibrium strategy s_2 in $G^{1,(1)}$, while player 2 has a mixed equilibrium strategy (randomizes between s_1 and s_2) in $G^{2,(1)}$. The corresponding nme has strategy profile $((0,1),(1/3,2/3))$. Using the characteristic strategy vector we take $\chi(nme) = \{(2,1),(2,2)\}$. Notice that, as one player randomized, $\chi(nme)$ has more than one elements, and the Adaptation Procedure branches, resulting to two new misinformation games, say $mG^{(2a)}$, $mG^{(2b)}$.*

Let us first consider the element $(2,1)$ of $\chi(nme)$ (which leads to $mG^{(2a)}$)). We note that the payoff matrices of $mG^{(1)}$ are already updated with the correct value with respect to the bottom-left element, therefore $mG^{(2a)} = mG^{(1)}$.

Similarly, for the element $(2,2)$ of $\chi(nme)$, we update the bottom-right element of $P^{1,(1)}$ and $P^{2,(1)}$, so $mG^{(1)}$ leads to $mG^{(2b)} = \langle G^0, G^{1,(2b)}, G^{2,(2b)}\rangle$ with payoff matrices:

$$P^{1,(2b)} = \begin{pmatrix} (2,2) & (0,3) \\ (3,0) & (1,1) \end{pmatrix}, \quad P^{1,(2b)} = \begin{pmatrix} (1,1) & (3,0) \\ (3,0) & (1,1) \end{pmatrix}$$

Hence, we conclude that $\mathcal{AD}^{(2)}(\{mG^{(0)}\}) = \{mG^{(1)}, mG^{(2b)}\}$. ■

4.2 Stabilisation of the Adaptation Procedure

The following definition determines when the procedure is assumed to have "terminated"; this corresponds to the time point where any further iterations do not provide new information to the players:

Definition 4.5. We say that the Adaptation Procedure terminates at step $t \in \mathbb{N}_0$, if and only if t is the smallest non-negative integer for which:

$$\mathcal{AD}^{(t+1)}(M) = \mathcal{AD}^{(t)}(M)$$

We call t the *length of the Adaptation Procedure* and we denote it as $\mathfrak{L}_{\mathcal{AD}}(M)$.

In other words, the Adaptation Procedure terminates at the first time step in which all of the spawned misinformation games already appear in the recursive tree (not necessarily in the same branch).

To simplify presentation in the following, we will denote by $\mathcal{AD}^*(M)$ the set of all misinformation games created by M, i.e., $\mathcal{AD}^*(M) = \bigcup_{t=0}^{\infty} \mathcal{AD}^{(t)}(M)$. Moreover, we will denote by $\mathcal{AD}^{\infty}(M)$ the misinformation games that $\mathcal{AD}()$ produces after its termination point, i.e., $\mathcal{AD}^{\infty}(M) = \mathcal{AD}^{(t)}(M)$ for $t = \mathfrak{L}_{\mathcal{AD}}(M)$. $\mathcal{AD}^{\infty}(M)$ will be called the *Stable Set.*

Definition 4.6. Consider a misinformation game mG. Then, σ is a *stable misinformed equilibrium* (or *sme* for short) of mG, iff there exists some $\widehat{mG} \in \mathcal{AD}^{\infty}(\{mG\})$ such that $\sigma \in NME(\widehat{mG})$ and, for all $\vec{v} \in \chi(\sigma)$, $\widehat{mG}_{\vec{v}} = \widehat{mG}$.

We denote by $SME(mG)$ the smes of mG. Note that for a strategy profile to be a stable misinformed equilibrium, it is not enough to be an nme of some game in the Stable Set; it should also be such that, when the players choose it, no further information will be obtained.

Example 2. *[continued] For $t = 2$, let us first consider $mG^{(2a)}$. As mentioned, $mG^{(2a)} = mG^{(1)}$ so, for reasons analysed in Example 2, it branches into $mG^{(3a)} = mG^{(1)}$, $mG^{(3b)} = mG^{(2b)}$.*

As regards $mG^{(2b)}$, we note that the nme of $mG^{(2b)}$ is $\sigma = ((0,1),(1/2,1/2))$, for which $\chi(\sigma) = \{(2,1),(2,2)\}$. Observe that both positions in $\chi(\sigma)$ (namely, $(2,1)$, $(2,2)$) are known to the players, i.e., it holds that $P^{i,(2b)}_{\vec{v}} = P^0_{\vec{v}}$, for $i \in \{1,2\}$, $\vec{v} \in \{(2,1),(2,2)\}$. Thus, $mG^{(3c)} = mG^{(2b)}$.

Combining the above, we observe that $\mathcal{AD}^{(3)}(\{mG^{(0)}\}) = \{mG^{(1)}, mG^{(2b)}\} = \mathcal{AD}^{(2)}(\{mG^{(0)}\})$, so the Adaptation Procedure terminates at step 2, i.e., $\mathfrak{L}_{\mathcal{AD}}(\{mG^{(0)}\}) = 2$.

Now let us identify the smes of $mG^{(0)}$. As explained above, and in Example 2 $NME(mG^{(1)}) = \{((0,1),(1/3,2/3))\}$, $NME(mG^{(2b)}) = \{((0,1),(1/2,1/2))\}$. As regards $\sigma_1 = ((0,1),(1/3,2/3))$, we note that it is not an sme, because there exists a position $(2,2) \in \chi(\sigma_1)$, for which $mG^{(1)}_{(2,2)} = mG^{(2b)}$. On the other hand, $\sigma_2 = ((0,1),(1/2,1/2))$ is an sme, because, as mentioned above, $P^{i,(2b)}_{\vec{v}} = P^0_{\vec{v}}$, for $i \in \{1,2\}$, $\vec{v} \in \{(2,1),(2,2)\}$, so $mG^{(2b)}_{\vec{v}} = mG^{(2b)}$ for $\vec{v} \in \{(2,1),(2,2)\}$. ■

Next we provide a useful property of the Adaptation Procedure:

Proposition 4.7. *For any set of misinformation games M:*

$$\mathcal{AD}(M) = \bigcup_{mG \in M} \mathcal{AD}(mG)$$

Proposition 4.8. *For any two misinformation games mG, mG' and $t \geq 0$, if $mG' \in \mathcal{AD}^{(t)}(mG) \cap \mathcal{AD}^{(t+1)}(mG)$ then $mG' \in \mathcal{AD}^{\infty}(mG)$.*

From Definition 4.6, not all misinformation games in the Stable Set contribute an sme. In other words, some of the games in $\mathcal{AD}^{\infty}(mG)$ are irrelevant when it comes to computing the smes.

Moreover, the end criterion in Definition 4.5 describes a "global" situation, where the set of misinformation games in two consecutive steps of the procedure are equal. The set-theoretic comparison of

two sets would be quite expensive computationally, so we would like to state a more "local" end criterion. Can we know, while examining a single misinformation game, whether it is safe to discard it from our future iterations?

Both problems above are resolved through the notion of the Terminal Set:

Definition 4.9. Let $mG^{(0)}$ be a misinformation game. We define the *Terminal Set* of the Adaptation Procedure on $mG^{(0)}$ as follows:

$$L = \{mG \in \mathcal{AD}^*(mG^{(0)}) \mid mG \in \mathcal{AD}(mG)\}.$$

The next proposition shows that the games in the Terminal Set are also in the Stable Set; in addition, they are the only members of the Stable Set that matter when it comes to computing smes:

Proposition 4.10. *Let L be the Terminal Set of the Adaptation Procedure on $mG^{(0)}$. Then:*

- $L \subseteq \mathcal{AD}^\infty(mG^{(0)})$
- *For any $\sigma \in SME(mG^{(0)})$, there exists $mG \in L$ such that $\sigma \in NME(mG)$.*

Note how this proposition affects the computational properties of the Adaptation Procedure. If a game mG is not in the Terminal Set, then we are only interested in what it "produces" (i.e., $\mathcal{AD}(mG)$), but not in the game itself (as it will not produce an sme). On the other hand, if mG is in the Terminal Set, then we just "keep" it, but we are not further interested in processing it: we know that it will be repeated forever, thus it is an unnecessary overhead to keep processing it. On the other hand, the games in $\mathcal{AD}(mG) \setminus \{mG\}$ may be relevant, if any of them (or any of their descendants) is in the Terminal Set.

An important question regarding the Adaptation Procedure is whether it always terminates. As we show below, this is true for finite misinformation games:

Proposition 4.11. *For any finite misinformation game mG, $\mathfrak{L}_{\mathcal{AD}}(mG)$ and $\mathcal{AD}^\infty(\{mG\})$ are finite.*

Moreover, all finite misinformation games have an sme. To show this, we use two intermediate lemmas:

Lemma 4.12. *If $mG' \in \mathcal{AD}^*(\{mG\})$ and $\mathcal{AD}(\{mG'\}) = \{mG'\}$ then $NME(mG') \subseteq SME(mG)$.*

Lemma 4.13. *Take some finite $mG_1, \ldots, mG_n$ such that $mG_{i+1} \in \mathcal{AD}(\{mG_i\})$, for $i = 1, \ldots, n-1$ and $mG_1 \in \mathcal{AD}(\{mG_n\})$. Then, $mG_i = mG_j$ for all i, j.*

Proposition 4.14. *If mG is finite, then $SME(mG) \neq \emptyset$.*

4.3 Adaptation Graphs

Until this point, we studied the Adaptation Procedure from the perspective of misinformation games. We will now consider an alternative view, where the Adaptation Procedure is studied from the perspective of position vectors which correspond to the payoffs that are being learnt by the players during the Adaptation Procedure; as we will see in Section 5, this viewpoint is more suitable from the algorithmic perspective.

Before explaining this, we introduce some new symbols: we denote by $\mathcal{U}$ the set of all position vectors in the payoff matrix and by $\mathcal{U}^*$ the set containing all sequences that can be generated from elements in $\mathcal{U}$ (including the empty sequence, $\epsilon = [\,]$)[1].

4.3.1 Sequence-based Adaptation Graph. Now, given some misinformation game mG, from Definition 4.4 it is clear that each of the misinformation games in $\mathcal{AD}^*(mG)$ is the result of a sequence of applications of the operation in Definition 4.1 upon mG. In other words, if $mG' \in \mathcal{AD}^*(mG)$, then there exists a sequence of position vectors $w = [\vec{v}_1, \ldots, \vec{v}_k] \in \mathcal{U}^*$ such that $mG' = (((mG)_{\vec{v}_1}) \ldots)_{\vec{v}_k}$ (note also that $mG_\epsilon = mG$). To simplify notation, we will write mG_w to denote $(((mG)_{\vec{v}_1}) \ldots)_{\vec{v}_k}$.

To formalise this idea, we first observe that not all possible sequences are relevant to any given Adaptation Procedure. We denote by V_{mG} the sequences that do occur in a given Adaptation Procedure initiated by mG, called the *reachable* sequences. Note that we restrict reachable sequences to contain any given position vector at most twice (i.e., allow a single repetition); the reasons for this choice will be made clear later. Formally:

Definition 4.15. For a misinformation game mG we define $V_{mG} \subseteq \mathcal{U}^*$ to be the following set of position sequences:

- $\epsilon \in V_{mG}$
- If $w = [\vec{v}_1, \ldots, \vec{v}_k] \in V_{mG}$, $\vec{v}_i \neq \vec{v}_j$ for all $i \neq j$ and $\vec{v} \in \chi(NME(mG_w))$ then $[w, \vec{v}] \in V_{mG}$

The sequences in V_{mG} will be called *reachable from mG*. The subscript will be omitted from V_{mG} when mG is obvious from the context.

Elements in V_{mG} are connected, through the Adaptation Procedure, as follows:

Definition 4.16. Let mG be a misinformation game and V its reachable position sequences. We define the *adaptation relation on sequences* to be the set $E_{mG} \subseteq V \times V$, where:

$$E_{mG} = \{(w, r) \mid w, r \in V, r = [w, \vec{v}], \text{where } \vec{v} \in \chi(NME((mG)_w))\}.$$

The subscript will be omitted from E_{mG} when mG is obvious from the context.

By Definition 4.16, each pair in E encodes a single step of the Adaptation Procedure, i.e., if $(w, r) \in E$ then we can arrive from mG_w to mG_r in a single step. As a result, $G_{seq} = (V, E)$ is a graph (actually, a tree), which contains all the information relevant for the Adaptation Procedure (see also Figure 1b).

Definition 4.17. Consider some misinformation game mG, and let V, E as in Definitions 4.15, 4.16 respectively. The graph $G_{seq} = (V, E)$ is called the *sequence-based adaptation graph of mG*.

The sequence-based adaptation graph is meant to model the Adaptation Procedure. As is obvious by the above discussion, each sequence w that is a node in G_{seq} corresponds to a game $mG_w \in \mathcal{AD}^*(mG)$, and vice-versa; moreover, each link between two sequences w, w' corresponds to an application of the Adaptation Procedure on mG_w, producing $mG_{w'}$, i.e., $mG_{w'} \in \mathcal{AD}(mG_w)$.

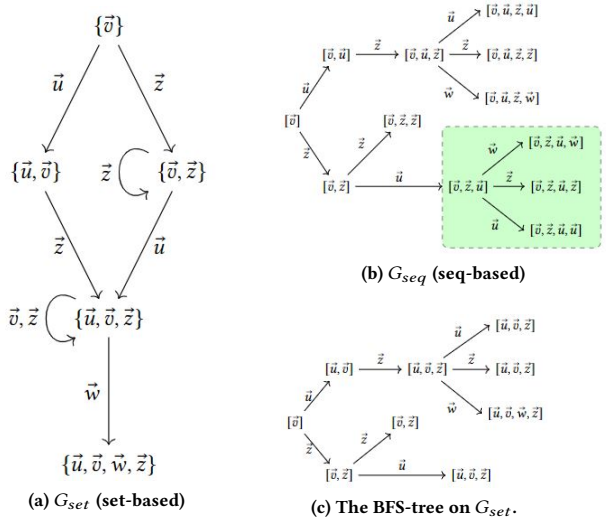
We formalise these ideas with the following proposition, which clarifies the properties of the sequence-based adaptation graph:

Proposition 4.18. *Consider a misinformation game mG and its respective sequence-based adaptation graph $G_{seq} = (V, E)$. Then:*

[1] Here we make use of the Kleene star notation.

- G_{seq} *is a tree*
- *If* w *is a leaf in* G_{seq} *then* mG_w *is in the Stable Set of* mG*, i.e.,* $mG_w \in \mathcal{AD}^{\infty}(mG)$ *whenever there is no* $r \in V$ *such that* $(w, r) \in E$
- *If* $w = [\vec{v}_1, \ldots, \vec{v}_{k-1}, \vec{v}_k] \in V$ *and* $\vec{v}_{k-1} = \vec{v}_k$*, then* mG_w *is in the Terminal Set of* mG*.*

Since the sequence graph G_{seq} is actually a tree, and in order to emphasise this fact, we will be using often the term *sequence tree*. Also, we will denote the sequence tree with T_{seq}, or simply T, instead of G_{seq}.



(a) G_{set} (set-based)

(b) G_{seq} (seq-based)

(c) The BFS-tree on G_{set}.

Figure 1: Sequence-based and set-based adaptation graphs.

Now we can also explain why repetitions are allowed. Indeed, consider the graph in Figure 1. We observe that $[\vec{v}, \vec{z}]$ is connected to $[\vec{v}, \vec{z}, \vec{z}]$, which includes a repetition. Let us view this process from the perspective of misinformation games. Set $mG_1 = (mG_{\vec{v}})_{\vec{z}}$ and $mG_2 = ((mG_{\vec{v}})_{\vec{z}})_{\vec{z}}$. It is easy to see that $mG_1 = mG_2 \in \mathcal{AD}(mG_1)$. If we disallowed repetitions in G_{seq}, then the position corresponding to mG_1 would not be a leaf in G_{seq}, and, thus, we would lose the information that mG_1 repeats itself, i.e., that it is in the Terminal Set. On the other hand, repetitions are allowed only once because if a position repeats itself, then it would not produce a new misinformation game, and, thus, need not be considered further (because this position vector, and its corresponding misinformation game, will appear in all subsequent steps of the Adaptation Procedure).

4.3.2 Set-based Adaptation Graph. The sequence-based adaptation graph is a suitable tool for implementation, but contains many more elements than necessary. To see this, observe that the update operation (Definition 4.1) is idempotent and commutative, i.e., for any $\vec{v}_1, \vec{v}_2$:

- $(mG_{\vec{v}_1})_{\vec{v}_1} = mG_{\vec{v}_1}$
- $(mG_{\vec{v}_1})_{\vec{v}_2} = (mG_{\vec{v}_2})_{\vec{v}_1}$

Therefore, we don't really need to consider sequences of position vectors; sets of position vectors are sufficient. Indeed, take any given sequence w. Suppose that we change the order of the positions in w, and/or add/delete duplicates, getting a new sequence w'. Then, by the above properties, $mG_w = mG_{w'}$. In other words, only the actual set of elements consisting a sequence w matters, not the sequence itself.

For any given sequence $w = [\vec{v}_1, \ldots, \vec{v}_k]$, let us denote by S^w the set $\{\vec{v}_1, \ldots, \vec{v}_k\}$. Abusing notation, for a set of positions $S = \{\vec{v}_1, \ldots, \vec{v}_k\}$, we denote by mG_S the game $mG_S = (\ldots(mG_{\vec{v}_1})_{\vec{v}_2}\ldots)_{\vec{v}_k}$. Given the properties above, the notation mG_S is well-defined (although S contains no fixed order), and, for any w, $mG_w = mG_{S^w}$.

We can now repeat the above exercise to define an adaptation graph analogous to the sequence-based adaptation graph, but based on sets.

Definition 4.19. Consider a misinformation game mG and its reachable sequences V_{mG}. We define $\mathcal{V}_{mG} \subseteq 2^{\mathcal{U}}$ to be all the sets that correspond to reachable sequences, i.e.: $\mathcal{V}_{mG} = \{S^w \mid w \in V_{mG}\}$. The sets of positions in $\mathcal{V}_{mG}$ will be called *reachable from* mG.

The subscript will be omitted from $\mathcal{V}_{mG}$ when mG is obvious from the context.

Definition 4.20. Let mG be a misinformation game and E its adaptation relation on sequences. We define $\mathcal{E}_{mG} \subseteq 2^{\mathcal{V}} \times 2^{\mathcal{V}}$ to be all the pairs of sets that correspond to the adaptation relation on sequences, i.e.: $\mathcal{E}_{mG} = \{(S^w, S^r) \mid (w, r) \in E\}$. The subscript will be omitted from $\mathcal{E}_{mG}$ when mG is obvious from the context.

We observe that, like G_{seq}, $G_{set} = (\mathcal{V}, \mathcal{E})$ is a graph which contains all the information relevant for the Adaptation Procedure (see also Figure 1a):

Definition 4.21. Consider some misinformation game mG, and let $\mathcal{V}, \mathcal{E}$ as in Definitions 4.19, 4.20 respectively. The graph $G_{set} = (\mathcal{V}, \mathcal{E})$ is called the *set-based adaptation graph of* mG.

The following proposition is the counterpart of Proposition 4.18 for the set-based adaptation graph:

Proposition 4.22. *Consider a misinformation game* mG *and its respective set-based adaptation graph* $G_{set} = (\mathcal{V}, \mathcal{E})$*. Then:*

- *All cycles in* G_{set} *have length equal to 1 (i.e., self-loops).*
- *There exist* $S_1, \ldots, S_n \in \mathcal{V}$, $n \geq 1$, *such that* $S_n = S$, $(S_1, S_1) \in \mathcal{E}$ *and* $(S_i, S_{i+1}) \in \mathcal{E}$ *for all* $1 \leq i < n$ *if and only if* mG_S *is in the Stable Set of* mG*.*
- $(S, S) \in \mathcal{E}$ *if and only if* mG_S *is in the Terminal Set of* mG*.*

5 ALGORITHM

The two notions defined in Subsection 4.3 (G_{seq}, G_{set}) describe the relations among misinformation games that appear in an adaptation procedure. As stated in Proposition 4.18, G_{seq} is actually a tree, i.e., connected and acyclic. Therefore, an algorithm can traverse this graph "blindly", without risk of infinite iterations, since there are no loops. However, the price to pay for the lack of cycles is the introduction of extra, redundant nodes, compared to the G_{set} graph. On the other hand, the set-based adaptation graph G_{set} provides us with a much more compact description of the misinformation games' relations in an adaptation procedure, but breaks the tree-like structure of the graph (see, Figure 1a) by introducing loops and situations where a node has multiple parents. Nevertheless,

with a little caution we can traverse the set-based adaptation graph bypassing these obstacles. As it turns out, a simple variation of the standard Breadth-First Search algorithm would do the trick.

Our algorithms explore both scenarios, and work in two different "modes". We call the first, naive method, based on G_{seq} "slow-mode", whereas the second, more elaborate method, based on G_{set} "fast-mode". The main difference of the two methods lies on an *ordering* function, i.e. the order we introduce to the sequence of choices made by the agents. Algorithm 1 implements the Adaptation Procedure, and employs two other algorithms, each corresponding to one algorithmic phase. In the first phase we traverse the adaptation graph (either G_{seq} or G_{set}), in order to compute the Terminal Set. In the second phase, we traverse the elements of the Terminal Set to identify the stable misinformed equilibria (Algorithm 3).

Algorithm 1: Adaptation Procedure

Input: A root misinformation game mG.
Output: The set of smes SME.

```
if fast-mode then
    ordering ← ordering-fast
else
    ordering ← ordering-slow
L ← TraverseAdaptationGraph(mG, ordering)
SME ← ComputeSME(L)
return SME
```

In Algorithm 1 we choose between the two modes, by choosing an ordering function. The ordering function is used to "manipulate" the position vectors during the execution of Algorithm 2.

In particular, for the case of `ordering-fast`, we sort the position vectors in the sequence w (lexicographically), and then delete the duplicate position vectors, if any, i.e.:

$$\texttt{ordering} - \texttt{fast}(w) = \texttt{unique}(\texttt{sort}(w)).$$

This manipulation essentially instills set semantics to sequences, i.e., if there are two sequences with the same elements, they will be mapped to the same sequence.

On the other hand, the function `ordering-slow` is the identical function, i.e., it does not perform any manipulation on the sequences, retaining the sequence semantics of its input.

$$\texttt{ordering} - \texttt{slow}(w) = w$$

Using these two functions, we *implicitly* define the *state space* of our agents. In the first case, with the `ordering-fast` function, we allow the agents to move on a *set-based* state space, that is the set $\mathcal{V}$ of Definition 4.19. In the second case, with the `ordering-slow` function, we allow the agents to move on a *sequence-based* state space, namely the set V of Definition 4.15.

The main functionality of our algorithm is handled in phase 1, i.e., Algorithm 2, which traverses *breadth-first* the states of the adaptation graph (see Figure 1) and computes the Terminal Set. Depending on the ordering function, Algorithm 2 will traverse either the *sequence based* graph G_{seq}, or the *set based* graph G_{set}. In both cases we keep the set V of the visited nodes, which corresponds to the states we have already discovered while traversing the graph. After the completion of the algorithm, when we have discovered all

Algorithm 2: Traverse Adaptation Graph

Input: A root misinformation game mG. An ordering function $\texttt{ordering}: \mathcal{U}^* \to \mathcal{U}^*$.
Output: The Terminal Set L.

```
1  L ← ∅, Q ← {ϵ}, V ← {ϵ}
2  while Q ≠ ∅ do
3      w ← pop(Q)
4      foreach position vector v⃗ ∈ χ(NME(mG_w)) do
5          r ← ordering(wv⃗)
           /* Update operation, see Definition 4.1 */
6          mG_r ← (mG_w)_v⃗
           /* Detect loop, see Definition 4.9.     */
7          if mG_r = mG_w then
8              L ← L ∪ {mG_r}
           /* Check if r unvisited.                */
9          if r ∉ V then
10             V ← V ∪ {r}, Q ← Q ∪ {r}
11 return L
```

states, the set of visited nodes will be equal to the set of reachable sequences (V = V – slow mode) or sets (V = $\mathcal{V}$ – fast mode). The Terminal Set L is updated in steps 7, 8, where we detect if a state has a "self-loop", with respect to Definition 4.9.

A technicality worth mentioning is that we check whether a node belongs to the Terminal Set, *before* checking whether we have visited this node. This is because the parent node, and the child node, may result in the same encoding in "fast-mode", i.e., $\texttt{ordering} - \texttt{fast}(w) = \texttt{ordering} - \texttt{fast}(r)$. In Figure 1 we depict with the green box the redundant part that we omit in "fast-mode". Observe that in Subfigure 1b the subtrees below the nodes $[\vec{v}, \vec{u}, \vec{z}]$ and $[\vec{v}, \vec{z}, \vec{u}]$ are isomorphic, *with respect to the* `fast-ordering`. Hence, when we arrive to the node $[\vec{v}, \vec{z}, \vec{u}]$, we don't explore the subtree any further.

Algorithm 3: Compute-Stable-Misinformed-Equilibria

Input: The Terminal Set L
Output: The set of stable misinformed equilibria SME of the root misinformation game mG.

```
1  SME ← ∅
2  foreach mG ∈ L do
3      foreach σ ∈ NME(mG) do
           /* Following Definition 4.6          */
4          if for all position vectors v⃗ ∈ χ(σ), we have
            mG = mG_v⃗ then
5              SME ← SME ∪ {σ}
6  return SME
```

Lastly, we implement phase 2 in Algorithm 3, which traverses the Terminal Set and checks whether each nme of the misinformation games in the Terminal Set satisfies Definition 4.6. Proposition 4.10 guarantees that this procedure will not miss any of the stable misinformed equilibria.

We close our discussion on Algorithm 1 by observing that, due to Proposition 4.11, the algorithm will always terminate.

6 IMPLEMENTATION

In this section we discuss the implementation of the algorithms described in Section 5, provide details regarding our program and explain the technologies and techniques we have utilized. The main part of the program is implemented in Python, for it is a simple and expressive programming language suitable for artificial intelligence applications. In order to compute the Nash equilibria in normal form games we used the package Gambit [20], which provides fast and accurate algorithms to compute the Nash equilibria in $|N|$-player normal form games. Further, we implemented the update operation of Definition 4.1 using the Answer Set Programming language Clingo[2]. As the update operation captures the main "logic" of the program, it is suitable to write the axioms of Definition 4.1 in a Logic Programming language. Another functionality of our algorithm is that the clingo-program informs us whether the child misinformation game has changed from its parent. Therefore, Steps 6 and 7 of Algorithm 2 are implemented in the logic program. Both the Gambit and Clingo subsystems are implemented as subprocesses of the main python program and communicate with it through plain text and pipes.

We have organized our program in four classes, namely:

- `NormalFormGame` class, which defines the notion of a normal form game.
- `MisinformationGame` class, which groups together $|N|+1$-normal form games, where the 0-game is the objective game, and the games $1,\dots,|N|$ are the subjective games of the players.
- `AdaptationTreeNode` class, which represents a node on the adaptation graph (either G_{seq}, or G_{set}).
- `AdaptationProcedure` class, which organizes all the other classes and implements the Algorithm 1 as a method.

Note that, as established in Proposition 4.18, G_{seq} is actually a tree. On the other hand, as we explore breadth-first the graph G_{set}, we also construct a tree (see Figure 1). Therefore, we will use the term tree here (rather than graph).

In Figure 2 we present a high level description of our implementation. We begin with a file containing a single misinformation game. From this file we initialize an instance of the `MisinformationGame` class, which is used to initialize an instance of the `AdaptationTreeNode` class, and insert it to the queue of the `AdaptationProcedure` class. Next, we pop the first element of the queue and get a misinformation game `MG_0`. For each normal form game G_i, $i \in [|N|] \cup \{0\}$ of `MG_0` we compute its Nash Equilibria using the Gambit software package. Thereupon, we import the output of the Gambit to a gambit output interpreter and get the Nash equilibria as tuples. Then, we combine the Nash equilibria to compute the natural misinformed equilibria. For each nme, we compute the position vectors it represents. Observe that the function $\chi(NME(mG))$ is many-to-many and if σ_1, σ_2 are two misinformation games, then, in general $\chi(\sigma_1) \cap \chi(\sigma_2) \neq \emptyset$. For each position-vector, we enter a clingo-predicate along with a description of the misinformation game to the Clingo compiler and apply the update operation of

[2]https://potassco.org/clingo/

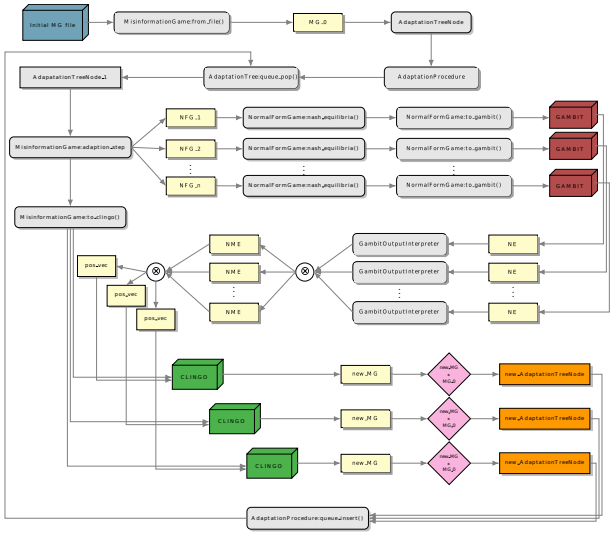


Figure 2: Flow chart.

Definition 4.1. Thus, we obtain k new misinformation games. If the child-misinformation games are different from their parent, we construct an instance of the `AdaptationTreeNode` class. Lastly, we insert the `AdaptationTreeNodes` again to the queue, and repeat the above procedure.

6.1 Virtualization & Memoization

We now present the data structure `mg_pool`, which encodes the relation between the state space V explored in Algorithm 2 and the set of misinformation games. A state is encoded in the `AdaptationTreeNode` class, which represents either a node of the sequence-graph G_{seq} (which is actually a tree), or a node of the set-based graph G_{set}. In the latter case, while we explore G_{set}, we again construct the breadth-first tree (see Figure1). On the other hand a misinformation game is encoded in the class `MisinformationGame`.

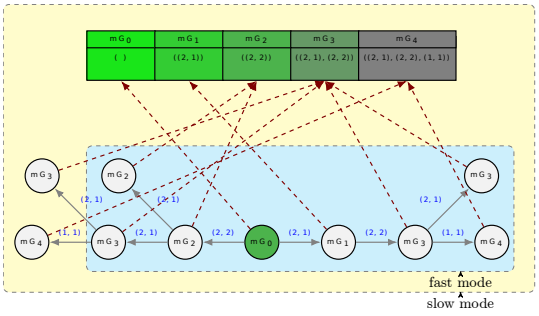


Figure 3: The `mg_pool` dictionary, in the two modes.

As we noted before, the state space of the sequence-based graph V contains some redundancy, as many sequences in V may correspond to the same misinformation game. On the other hand, the state space of the set-based graph $\mathcal{V}$ is minimal, in the sense that $|\mathcal{V}| = |\mathcal{AD}^*(mG)|$, but due to the multiple self loops, we may again encounter a node (at most) twice (see Figure 1). For the sake

of efficiency, we don't want to recompute a misinformation game if we encounter it a second time. Hence, in the `AdaptationTreeNode` class, we include a *label*[3] to a misinformation game as data member, rather than an instance of the `MisinformationGame` class (see Figure 3). The data structure `mg_pool` is a *dictionary*, from the set of labels to instances of the `MisinformationGame` class. Since $|\mathcal{V}| = |\mathcal{AD}^*(mG)|$, the most appropriate set of labels is $\mathcal{V}$.

6.2 Answer Set Programming

In this subsection we discuss the logic program that implements the update operation of Definition 4.1. We encode a payoff function with the predicate `u(G, P, SP, U)` of arity four, which represents the value of the payoff (`U`) that the player `P` will receive, according to the normal form game of player `G`, when the strategy profile SP is played. Note that Clingo does not support lists natively[4]. Therefore, we use symbolic functions to encode finite lists. For example, we encode the strategy profile $(1, 2, 3)$, with $\mathtt{sp}(1, \mathtt{sp}(2, \mathtt{sp}(3, \mathtt{nul})))$, where `sp/2` is a symbolic function with two arguments and `nul/0` is a symbolic function with arity zero.

Listing 1: The Update Operation's Logic program (Pseudocode).

```
% the new payoff function if pos(SP).
v(G, P, SP, V) :- pos(SP), u(0, P, SP, U), V = U.

% the new payoff function if not pos(SP).
v(G, P, SP, V) :- not pos(SP), u(G, P, SP, U), V = U.

% check if child-mG != parent-mG
changed :- v(G, P, SP, V), u(G, P, SP, U), V != U.
unchanged :- not changed.
```

In Listing 1 we present the logic program for applying the update operation of Definition 4.1. The logic program takes as input a list of payoff predicates `u/4` and a single predicate `pos(SP)` describing a position vector as a finite list. Then, outputs (proves) a list of updated payoff predicates `v/4`, which follow the syntax of `u/4`. For each strategy profile `SP` we check whether the predicate `pos(SP)` holds. If it does, we force the new payoff entry `v/4` of this strategy profile to equal the *actual* game G^0. If there is an instance where `v` $\neq$ `u`, then the logic program proves the predicate `changed/0`. Otherwise, the predicate `unchanged/0` holds. This way the python program can tell whether $mG \neq mG'$, where mG' is a child of mG.

6.3 Computation of Nash Equlibria

To compute the Nash equilibria of a normal form game we utilized the Gambit package. In particular, we used the `gambit-gnm` command which implements the Generalized Newton Method for computing the Nash equilibria in $|N|$-player normal form games, [14]. For the perturbation parameter used by Generalized Newton Method we used the value $n = 100$. Since the computational complexity for computing a Nash equilibrium (and also a natural misinformed equilibrium) is **PPAD**-complete ([4] and [28] respectively), we inherit this complexity in our algorithm.

[3]The technically correct term would be a *pointer*. Note however that the Python programming language does not allow an explicit use of pointers, like C or C++, but with careful programming we can enforce this behaviour implicitly.
[4]Unlike other Logic Programming languages, e.g., Prolog.

7 EXPERIMENTS

In this section we experimentally evaluate our algorithm. Given the intractability of computing a Nash equilibrium (being in the **PPAD**-complete class), and the fact that such a computation takes place several times during the Adaptation Procedure, we restricted our experimentation to relatively small misinformation games. In particular, we considered 2-player misinformation games, where the number of strategies per player ranged as follows:

(1) Both players have 2 strategies.
(2) Player-1 has 3 strategies, while player-2 2.
(3) Both players have 3 strategies.
(4) Player-1 has 4 strategies, while player-2 has 3 strategies.
(5) Both players have 4 strategies.

Moreover, we considered a 3-player misinformation game with two strategies per player, but only under "fast-mode".

All experiments were run on an Ubuntu 20.04 machine with an Intel-i5-4200M processor at 3.100GHz. The version of the Python language was 3.8.10, Gambit's version was 15.1.1, and Clingo's version was 5.4.0. We ran each of the experiments five times with different random generator seeds. In order to eliminate any caching effects we discarded the first and the last measurement and took the average of the remaining three measurements. For the seed of the random experiments, we chose the first five prime numbers, i.e., 2, 3, 5, 7 and 11.

In Table 1 we present our results regarding the performance of our algorithm. "CPU time" refers to the time consumed only by the Python program, whereas "total time" refers to the total elapsed time including the time consumed by the sub-processes, Clingo and Gambit. Observe that only a small fraction of the elapsed time is consumed by the Python program. As a matter of fact, the calls to the Gambit package (which undertake the computationally heavy task of computing the Nash equilibria) constitute the main bottleneck of our pipeline, by far.

Also, we compare the "fast" and the "slow" methodology of our algorithm. As the strategy space of the considered misinformation games gets richer, the performance difference between the two methodologies becomes more substantial. In the case with 12 strategy profiles of a 2-player misinformation game, the fast mode takes less than half the time compared to the slow one. This is because the fast mode allows us to omit several Clingo-calls. Moreover, for a 2-player game with 16 strategy profiles, as well as for the 3-player game with 8 strategy profiles, the slow methodology was unable to terminate in a reasonable amount of time.

To understand why the experiments are so time consuming, we need to keep in mind the exponential growth of the search space as the available strategies increase, in tandem with the fact that each node in the adaptation tree requires the computation of several Nash equilibria, an inherently intractable process.

In Table 2 we give the characteristics of the adaptation tree for the considered 2-player games in the slow and fast methodology respectively. Recall that we have a node for each position vector sequence that came up during the execution of Algorithm 1. On the other hand, two nodes may correspond to the same misinformation game. Consider the case of the 2-player game with 12 strategy profiles. In the slow methodology, we compute 9.008 nodes, which correspond to only 181 *unique* misinformation games. If we are not

Table 1: Efficiency experiments

	no. strategy profiles	slow mode		fast mode	
		total time	CPU time	total time	CPU time
2-pl.	2×2	0.170s	0.050s	0.151s	0.045s
	3×2	0.439s	0.090s	0.437s	0.090s
	3×3	0.883s	0.134s	1.129s	0.154s
	4×3	68.81s	12.658s	31.373s	3.791s
	4×4	–	–	3634.635s	454.232s
3-pl.	$2 \times 2 \times 2$	–	–	26.963s	0.563s

Table 2: Properties of the Adaptation Procedure.

mode	no. strategy profiles	no. nodes	no. unique *mGs*	no. leafs	no. (unique *mGs* ∧ leafs)	no. *smes*
slow	2×2	13.667	4.000	8.667	2.333	4.667
	3×2	20.333	7.333	11.667	6.000	2.000
	3×3	42.333	9.000	29.333	7.667	1.333
	4×3	9008.0	181.667	4694.00	144.00	4.000
fast	2×2	9.667	4.000	5.000	2.333	4.667
	3×2	17.667	7.333	9.333	6.000	2.000
	3×3	28.000	9.000	17.333	7.667	1.333
	4×3	543.00	180.00	191.667	143.00	4.000
	4×4	46285.333	9950.333	26830.667	9392.333	1584.667

interested in the number of different choice sequences that lead to a specific game, we may discard many of these nodes. In fast mode we do exactly that and only need to compute 543 nodes for the same experiment. The reason for the number of nodes computed in fast mode not being equal to the number of unique misinformation games is because we still allow parent-child repetitions, as established in Section 4. The number of leaves corresponds to the number of different sequences that generate the Terminal Set. Lastly, we observe that the number of smes does not correlate with the other quantities of the experiment.

8 CONCLUSIONS

We provided a methodology for studying game-playing scenarios where misinformed players revise their game-related information as they interact with their environment. This idea builds on the notion of misinformation games [28], and, unlike the setting of [28], considers scenarios where the players revise their games (thereby reducing their misinformation), based on the received payoffs that are publicly announced in each game iteration. Our analysis unfolded into two directions, a theoretical and a practical one.

From the theoretical perspective, we defined the *Adaptation Procedure*, which describes the changes in the decisions of the players as they obtain new information. This leads to a new equilibrium concept, the *stable misinformed equilibrium*, which is the strategy profile(s) that the players choose when the procedure stabilizes.

From the practical perspective, we presented two algorithms, the naive "slow-mode", which is based on position-vector sequences graph G_{seq}, and the more elaborate "fast-mode", which is based on position-vector set graph G_{set}. We provided an implementation of the two modes and an experimental evaluation.

There are many directions for future work. From the theoretical perspective we could consider dropping the assumption that players fully update their information (e.g., they may not be able to observe other players' payoffs). From the practical perspective, we could consider approximate methods, in order to compute the stable misinformed equilibria, without fully exploring the graph G_{set}.

REFERENCES

[1] Kenneth J. Arrow and Jerry R. Green. 1973. Notes on Expectations Equilibria in Bayesian Settings Institute for Mathematical Studies in the Social Sciences. Working Paper No. 33, Stanford University.

[2] Peter G. Bennett. 1980. Hypergames: Developing a model of conflict. *Futures* 12, 6 (1980), 489–507.

[3] Andriy Burkov and Brahim Chaib-draa. 2009. Effective learning in the presence of adaptive counterparts. *J. Algorithms* 64, 4 (2009), 127–138.

[4] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. 2009. The Complexity of Computing a Nash Equilibrium. *SIAM J. Comput.* 39 (2009), 195–259.

[5] Eddie Dekel, Drew Fudenberg, and David K. Levine. 1999. Payoff Information and Self-Confirming Equilibrium. *Journal of Economic Theory* 89, 2 (1999), 165–185.

[6] David Easley and Nicholas M Kiefer. 1988. Controlling a Stochastic Process with Unknown Parameters. *Econometrica* 56, 5 (1988), 1045–1064.

[7] Ignacio Esponda and Demian Pouzo. 2016. Berk-Nash Equilibrium: A Framework for Modeling Agents with Misspecified Models. 84, 3 (2016), 1093–1130.

[8] Ignacio Esponda and Demian Pouzo. 2016. Equilibrium in Misspecified Markov Decision Processes. *ERN: Other Microeconomics: Decision-Making under Risk & Uncertainty (Topic)* (2016).

[9] Ignacio Esponda and Demian Pouzo. 2019. Asymptotic Behavior of Bayesian Learners with Misspecified Models. *arXiv: Theoretical Economics* (2019).

[10] Yossi Feinberg. 2020. Games with Unawareness. *The B.E. Journal of Theoretical Economics* (2020).

[11] Drew Fudenberg and David K. Levine. 1993. Self-Confirming Equilibrium. *Econometrica* 61, 3 (1993), 523–545.

[12] D. Fudenberg, G. Romanyuk, and P. Strack. 2016. Active Learning with Misspecified Beliefs. *ERN: Search* (2016).

[13] Bahman Gharesifard and Jorge Cortés. 2012. Evolution of Players' Misperceptions in Hypergames Under Perfect Observations. *IEEE Trans. Autom. Control.* 57, 7 (2012), 1627–1640.

[14] Srihari Govindan and Robert Wilson. 2003. A global Newton method to compute Nash equilibria. *J. Econ. Theory* 110 (2003), 65–86.

[15] Paul Heidhues, Botond Kőszegi, and P. Strack. 2017. Unrealistic Expectations and Misguided Learning. *CSN: Economics (Topic)* (2017).

[16] Paul Heidhues, Botond Kőszegi, and P. Strack. 2018. Convergence in Misspecified Learning Models with Endogenous Actions. *Econometrics: Econometric Model Construction* (2018).

[17] Junjie Jiang, Yu-Zhong Chen, Zi-Gang Huang, and Ying-Cheng Lai. 2018. Evolutionary hypergame dynamics. *Physical Review E* (2018).

[18] J.S Jordan. 1991. Bayesian learning in normal form games. *Games and Economic Behavior* 3, 1 (1991), 60–81.

[19] R. Duncan Luce and Howard Raiffa. 1957. *Games and Decisions: Introduction and Critical Survey.* John Wiley and Sons, New York.

[20] Richard D. McKelvey, Theodore L. Turocy, and Andrew McLennan. 2014. Gambit: Software Tools for Game Theory, Version 16.0.0.

[21] John F. Nash. 1951. Non-Cooperative Games. *The Annals of Mathematics* 54, 2 (1951), 286–295.

[22] Yaw Nyarko. 1991. Learning In Mis-Specified Models And The Possibility Of Cycles. *Journal of Economic Theory* 55 (1991), 416–427.

[23] Martin J. Osborne and Ariel Rubinstein. 1994. *A Course in Game Theory.* Vol. 1. The MIT Press.

[24] Debajyoti Ray, Brooks King-Casas, P. Read Montague, and Peter Dayan. 2008. Bayesian Model of Behaviour in Economic Games. In *Advances in Neural Information Processing Systems 21, NIPS08.* 1345–1352.

[25] G. Romanyuk, P. Strack, and D. Fudenberg. 2017. Active learning with a misspecified prior.

[26] Cosma Rohilla Shalizi. 2009. Dynamics of Bayesian Updating with Dependent Data and Misspecified Models. *Electronic Journal of Statistics* 3 (2009), 1039 – 1074.

[27] Ran Spiegler. 2016. Bayesian Networks and Boundedly Rational Expectations *. *The Quarterly Journal of Economics* 131, 3 (2016), 1243–1290.

[28] Constantinos Varsos, Giorgos Flouris, Marina Bitsaki, and Michail Fasoulakis. 2021. A Study of Misinformation Games. In *PRICAI 2021: Trends in Artificial Intelligence - 18th Pacific Rim International Conference on Artificial Intelligence*, Vol. 13031. Springer, 76–87.