# Finite-horizon Equilibria for Neuro-symbolic Concurrent Stochastic Games


## Abstract

We present novel techniques for neuro-symbolic concurrent stochastic games, a recently proposed modelling formalism to represent a set of agents operating in a probabilistic, continuous-space environment using a combination of neural network based perception mechanisms and traditional symbolic methods. To date, only zero-sum variants of the model were studied, which is too restrictive when agents have distinct objectives. We formalise notions of equilibria for these models and present algorithms to synthesise them. Focusing on the finite-horizon setting, and (global) social welfare subgame-perfect optimality, we consider two distinct types: Nash equilibria and correlated equilibria. We first show that an exact solution based on backward induction may yield arbitrarily bad equilibria. We then propose an approximation algorithm called frozen subgame improvement, which proceeds through iterative solution of nonlinear programs. We develop a prototype implementation and demonstrate the benefits of our approach on two case studies: an automated car-parking system and an aircraft collision avoidance system.


## 1 INTRODUCTION

Stochastic games [Shapley, 1953] are a well established model for the formal design and analysis of probabilistic multi-agent systems. In particular, *concurrent stochastic games* (CSGs) provide a natural framework for modelling a set of interactive, rational agents operating concurrently within an uncertain or probabilistic environment. For finite-state CSGs, algorithms for their solution are known [Chatterjee et al., 2013, de Alfaro and Majumdar, 2004, De Alfaro et al., 2007] and, more recently, techniques and tools for their formal modelling, analysis and verification have been developed [Kwiatkowska et al., 2020, 2021] and applied to examples across robotics, computer security and networks.

In more complex scenarios, for example sequential decision making in continuous-state or mixed discrete-continuous state environments, CSGs are again a natural formalism for problems such as multi-agent reinforcement learning [Papoudakis et al., 2021, Yan et al., 2021]. A recent trend in this setting is the use of neural networks (NNs), to represent learnt approximations to value functions [Omidshafiei et al., 2017] or strategies [Lowe et al., 2017] for CSGs. However, the scalability and efficiency of such approaches are limited when NNs are used to manage multiple, complex aspects of the system. To overcome this, a further promising direction is the use of *neuro-symbolic* approaches. These deploy NNs within certain data-driven components of the control problem, e.g., for perception modules, and traditional symbolic methods for others, e.g., nonlinear controllers.

In this paper, we work with the recently proposed formalism of *neuro-symbolic concurrent stochastic games* (NS-CSGs) [Yan et al., 2022], designed to model probabilistic multi-agent systems comprising neuro-symbolic agents operating concurrently within a shared, continuous-state environment. In [Yan et al., 2022], the *zero-sum* control problem is considered, namely to synthesise strategies for one set of agents who are aiming to maximise their (discounted, infinite-horizon) expected reward, while the other agents aim to minimise this value. However, in practice, this is limiting: even for the case of just two coalitions of agents, they will often have distinct, but not directly opposing goals, which cannot be modelled in a zero-sum fashion.

To tackle this problem, we work with *equilibria*, defined by a separate, independent objective for each agent. These are particularly attractive since they ensure stability against deviations by individual agents, improving the overall system outcomes. We formalise the equilibrium synthesis problem for NS-CSGs, considering two distinct variants: *Nash equilibria* (NEs), which aim to ensure that no agent has an incentive to deviate unilaterally from their strategy, and

*Submitted to the 38th Conference on Uncertainty in Artificial Intelligence* (UAI 2022). **To be used for reviewing only**.

*correlated equilibria* (CEs), which allow agent coordination, e.g., through public signals, and where agents have no incentive to deviate from the resulting actions. The latter can both simplify strategy synthesis and improve performance.

Our focus is on (undiscounted) *finite-horizon* objectives, which simplifies the analysis (note that the existence of infinite-horizon NE for CSGs is an open problem [Bouyer et al., 2014], and the verification of non-probabilistic infinite-horizon reachability properties for neuro-symbolic games is undecidable [Akintunde et al., 2020a]), but also has a number of useful applications, e.g. in receding horizon control. Since multiple equilibria may exist, we target *social welfare (SW)* optimal equilibria, which maximise the sum of the individual agent objectives.

We also work with *subgame-perfect equilibria* (SPE), which are equilibria in every state of the game, ensuring that optimality remains as later states of the game are reached [Abreu et al., 2020, Fudenberg and Levine, 2009, Littman et al., 2006, Osborne et al., 2004]. Crucially, we consider *globally optimal* equilibria which, from a fixed initial state, are optimal over the chosen time horizon. This is in contrast to techniques for equilibria in finite-state CSGs [Kwiatkowska et al., 2021, 2022], which consider only local optimality at each time step in the finite-horizon setting.

We first adapt (classical) backward induction to NS-CSGs based on local optimality, but show that it may find an arbitrarily bad SPE. Then, for a fixed initial state, we show how to compute optimal equilibria by unfolding the game tree (including invocation of the NN perception function) and solving a *nonlinear program*. However, this suffers from limited scalability. So we then propose *frozen subgame improvement* (FSI), an approximation algorithm which iteratively solves nonlinear programs to monotonically improve the social welfare. Our approach is wholly different from the zero-sum (discounted, infinite-horizon) solution of NS-CSGs in [Yan et al., 2022], which applies value/policy iteration to finite model abstractions that rely on assumptions about the functions used to specify the model.

Finally, we implement our algorithms and evaluate them on two case studies, a car-parking example and the VerticalCAS (VCAS) aircraft system for collision avoidance, showing that they are capable of automatically generating equilibria that can improve over zero-sum strategies.

**Related Work.** Several papers have considered verification and synthesis of equilibria for stochastic games [Fernando et al., 2018, Horák and Bošanskỳ, 2019, Kwiatkowska et al., 2021, Mari et al., 2009], aiming to prove that a game satisfies a given equilibrium-related requirement specification and also to find such an equilibrium. However, none of these support CSGs whose agents are partly realized via NNs. The PRISM-games tool [Kwiatkowska et al., 2020] provides modelling, verification and equilibria synthesis for (discrete-state) CSGs, including finite-horizon analysis via backward induction, but for the simpler case of local optimality, as discussed abvove. [Kwiatkowska et al., 2020] also includes infinite-horizon $\epsilon$-optimal social welfare Nash equilibria, and [Kwiatkowska et al., 2022] correlated equilibria with two types of optimality conditions, computed using value iteration, but again only for discrete models.

Numerous methods have been proposed to compute SPEs since their introduction in the 1970s [Selten, 1975]. Most of these address the infinite horizon, for which fixed-point algorithms are the most common methods, from operator design for SPE payoff correspondence [Abreu et al., 2020, Brihaye et al., 2020, Burkov and Chaib-draa, 2010, Kitti, 2016, Yeltekin et al., 2017], to homotopy methods [Li and Dang, 2020]. For the finite horizon, which we consider here for reasons of decidability, backward induction is a simple and common bottom-up algorithm for finding an SPE efficiently. However, all these approaches fail to identify SW-SPEs over a finite horizon. In [Littman et al., 2006], a polynomial algorithm is proposed for computing optimal SPEs for turn-based games played over trees, which cannot deal with the concurrency in CSGs.

Neuro-symbolic computing has been attracting attention recently, see [Kahneman, 2011] and the surveys [De Raedt et al., 2020, Lamb et al., 2020]. The works of [Akintunde et al., 2020a,b] consider neuro-symbolic multi-agent systems represented as neural interpreted systems and study the finite-horizon verification problem for Alternating Temporal Logic, solved through reduction to an MILP problem, but no equilibria properties. The agents are endowed with perception similarly to what we do here, but are not stochastic.

## 2 NEURO-SYMBOLIC CSGS

We begin by describing *neuro-symbolic concurrent stochastic games (NS-CSGs)* [Yan et al., 2022], the modelling formalism that we use in this paper, for which we then define our notions of equilibria.

An NS-CSG comprises a number of interacting neuro-symbolic agents acting in a shared environment. Each agent has finitely many local states and actions, and is additionally endowed with a perception mechanism implemented as a neural network (NN), through which it can observe the state of the environment, storing the observations locally in *percepts*. For the purposes of this paper it suffices to assume that an NN is a function $f : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ over finite real vector spaces. Formally, an NS-CSG is defined as follows.

**Definition 1** (NS-CSG)**.** *A neuro-symbolic concurrent stochastic game (NS-CSG)* $\mathsf{C}$ *comprises agents* $(\mathsf{Ag}_i)_{i \in N}$*, for* $N = \{1, \ldots, n\}$*, and an environment* $E$ *where:*

$$\mathsf{Ag}_i = (S_i, A_i, \Delta_i, obs_i, \delta_i) \textit{ for } i \in N, \quad E = (S_E, \delta_E)$$

*and we have:*

- $S_i = Loc_i \times Per_i$ *is a set of states for* $\mathsf{Ag}_i$*, where* $Loc_i \subseteq \mathbb{R}^{b_i}$ *and* $Per_i \subseteq \mathbb{R}^{d_i}$ *are finite sets of local states and percepts, respectively;*
- $S_E \subseteq \mathbb{R}^e$ *is a finite or infinite set of environment states;*
- $A_i$ *is a nonempty finite action set for* $\mathsf{Ag}_i$*, and* $A := (A_1 \cup \{\bot\}) \times \cdots \times (A_n \cup \{\bot\})$ *is the set of* joint actions*, where* $\bot$ *is an idle action disjoint from* $\cup_{i=1}^{n} A_i$*;*
- $\Delta_i : S_i \to 2^{A_i}$ *is an available action function, defining the actions* $\mathsf{Ag}_i$ *can take in each state;*
- $obs_i : (S_1 \times \cdots \times S_n \times S_E) \to Per_i$ *is an observation function for* $\mathsf{Ag}_i$*, mapping the state of all agents and the environment to a percept of the agent, implemented via an NN classifier;*
- $\delta_i : S_i \times A \to \mathbb{P}(Loc_i)$ *is a probabilistic transition function for* $\mathsf{Ag}_i$*, where* $\mathbb{P}(X)$ *denotes the set of probability distributions over a set* $X$*, determining the probability of moving to local states given its current state and joint action;*
- $\delta_E : S_E \times A \to S_E$ *is a deterministic environment transition function determining the environment's next state given its current state and joint action.*

Formally, the semantics of an NS-CSG is an infinite state CSG, as described in [Yan et al., 2022]. For brevity, we present here an informal description of the semantics.

Each (global) state $s$ of NS-CSG $\mathsf{C}$ comprises the state $s_i = (loc_i, per_i) \in S_i$ of each agent $\mathsf{Ag}_i$ and the state $s_E \in S_E$ of the environment. Starting from some initial state, the game evolves as follows. First, each agent $\mathsf{Ag}_i$ observes the state of the agents and the environment to generate a new percept $per'_i$ according to its observation function $obs_i$ implemented via an NN. Then, each agent $\mathsf{Ag}_i$ synchronously chooses one of the actions from the set $\Delta_i(s_i)$, which are available in its state $s_i$. This results in a joint action $\alpha = (a_1, \ldots, a_n) \in A$. Each agent $\mathsf{Ag}_i$ then updates its local state to $loc'_i \in Loc_i$ according to the probabilistic local transition function $\delta_i$, applied to the state of agent $(loc_i, per'_i)$ and joint action $\alpha$. The environment updates the environment state to $s'_E \in S_E$ according to the environment transition function $\delta_E$, applied to its state $s_E$ and joint action $\alpha$. Thus, the game reaches the state $s' = (s'_1, \ldots, s'_n, s'_E)$, where $s'_i = (loc'_i, per'_i)$ for $i \in N$.

For simplicity, we consider here deterministic environments, but the results can be directly extended to discrete probabilistic environments with finite branching.

To illustrate NS-CSGs, we model the VerticalCAS Collision Avoidance Scenario [Julian and Kochenderfer, 2019, Julian et al., 2019] presented as a two-agent neurosymbolic system (VCAS[2]) in [Akintunde et al., 2020a]. Our model differs in that we separate the states of the agents and the environment state by adding to the agents' states a variable that measures their trust in the advisory's output, whereas [Akintunde et al., 2020a] replicates the climb rates in both agents' local states and the environment state. We update the agents' trust level probabilistically to account for possible uncertainty.

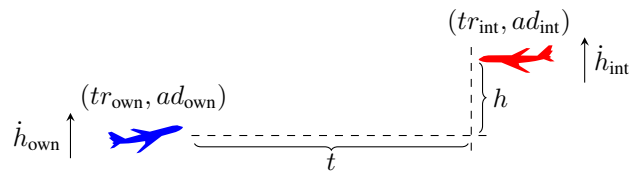


Figure 1: Geometry for the VCAS[2] example with trust level $tr_{\text{i}}$ and advisory $ad_{\text{i}}$, for $i \in \{\text{own}, \text{int}\}$.

**Example 1.** *In the VCAS[2] system (Figure 1) there are two aircraft (ownship and intruder* $\mathsf{Ag}_i$ *for* $i \in \{\text{own}, \text{int}\}$*), each of which is equipped with an NN-controlled collision avoidance system called VCAS. Each second, VCAS issues an advisory (*$ad_i$*) from which, together with the current trust level (*$tr_i$*) in the previous advisory, the pilot needs to make a decision about accelerations, aiming at avoiding a near mid-air collision (NMAC) [Akintunde et al., 2020b].*

*The input of the VCAS is* $(h, \dot{h}_{\text{own}}, \dot{h}_{\text{int}}, t)$ *recording the relative altitude* $h$ *of two aircraft, the climb rate* $\dot{h}_{\text{own}}$ *of the ownship, the climb rate* $\dot{h}_{\text{int}}$ *of the intruder, and the time* $t$ *until loss of their horizontal separation. VCAS is implemented via nine feed-forward NNs* $F = \{f_i : \mathbb{R}^4 \to \mathbb{R}^9 \mid i \in [9]\}$*, each of which corresponds to an advisory and outputs the scores of nine possible advisories, where* $[k]$ *is the set* $\{1, \ldots, k\}$*. Each advisory will provide a set of accelerations for the agent to select from. There are four trust levels* $\{4, 3, 2, 1\}$ *indicating the trust scores. The trust level is increased probabilistically if the current advisory is compliant with the executed action, and decreased otherwise. We formulate VCAS[2] as an NS-CSG with the agents* $\mathsf{Ag}_i$ *for* $i \in \{\text{own}, \text{int}\}$ *and the environment defined as follows:*

- $s_i = (tr_i, ad_i)$ *is a state of the agent* $\mathsf{Ag}_i$ *with local state* $tr_i \in [4]$ *and percept* $ad_i \in [9]$*;*
- $s_E = (h, \dot{h}_{\text{own}}, \dot{h}_{\text{int}}, t)$ *is an environment state;*
- $A_i$ *is a finite set of accelerations* $(\ddot{h}_i)$*;*
- $\Delta_i(s_i)$ *returns a set of available accelerations;*
- *observation function* $obs_i$ *is implemented via* $F$*;*
- *the local transition function* $\delta_i$ *updates its trust level according to its current trust level, its updated advisory and its executed action;*
- *the environment transition function* $\delta_E(s_E, \alpha)$ *is defined as:* $h' = h - \Delta t(\dot{h}_{\text{own}} - \dot{h}_{\text{int}}) - 0.5\Delta t^2(\ddot{h}_{\text{own}} - \ddot{h}_{\text{int}})$*,* $\dot{h}'_{\text{own}} = \dot{h}_{\text{own}} + \ddot{h}_{\text{own}}\Delta t$*,* $\dot{h}'_{\text{int}} = \dot{h}_{\text{int}} + \ddot{h}_{\text{int}}\Delta t$ *and* $t' = t - \Delta t$*, where* $\Delta t = 1$ *is the time step.*

**Game Tree Unfolding.** The finite-horizon evolution of an NS-CSG $\mathsf{C}$ from a given global state $s$ can be unfolded into a finite tree in the usual way by applying *strategies* to select

actions. We distinguish between (past) *histories* of a given state and its (future) *paths*.

We assume that the duration of the game is finite with $K$ stages. A history $h$ of $\mathsf{C}$ in stage $\ell \in [0, K]$ is a sequence $h = s^0 \xrightarrow{\alpha^0} s^1 \xrightarrow{\alpha^1} \cdots \xrightarrow{\alpha^{\ell-1}} s^\ell$ where $s^k \in S$, $\alpha^k \in A$ and $\delta(s^k, \alpha^k)(s^{k+1}) > 0$. The prefix of $h$ ending in stage $\bar{\ell}$ is denoted by $h_{\leq \bar{\ell}}$ for any $\bar{\ell} \leq \ell$. The set of all histories in stage $\ell$ for all initial states (for an initial state $s$) is denoted as $H^\ell$ ($H^\ell_s$), the set of all histories before stage $K$ is $H^{<K} = \cup_{0 \leq \ell < K} H^\ell$ ($H^{<K}_s = \cup_{0 \leq \ell < K} H^\ell_s$) and the set of all histories from $s$ is $H_s = H^{<K}_s \cup H^K_s$. We denote by $last(h)$ the last state of the history $h \in H_s$. If $h \in H^{<K}$, we denote by $\mathrm{Succ}(h)$ the set of one-stage successors of $h$.

For a state $s = (s_1, \ldots, s_n, s_E)$, the available actions of $\mathsf{Ag}_i$ are denoted by $A_i(s)$, i.e., $A_i(s)$ equals $\Delta_i(s_i)$ if $\Delta_i(s_i) \neq \varnothing$ and equals $\{\perp\}$ otherwise, and we denote by $A(s)$ the possible joint actions in a state, i.e. $A(s) = A_1(s) \times \cdots A_n(s)$.

We can now define strategies, strategy profiles and correlated profiles. In each case, we follow [Yan et al., 2022] in assuming a *fully observable* setting, i.e., where decisions are made based on the full state of the NS-CSG, not just the parts of it revealed by the agents' observation functions. An extension to partial observability is left for future work.

**Definition 2** (Strategy)**.** *A* strategy *for* $\mathsf{Ag}_i$ *is a function* $\sigma_i : H^{<K} \to \mathbb{P}(A_i \cup \{\perp\})$ *such that, if* $\sigma_i(h)(a_i) > 0$*, then* $a_i \in A_i(last(h))$*. A* strategy profile $\sigma = (\sigma_1, \ldots, \sigma_n)$ *comprises a strategy for each agent. We denote by* $\Sigma^{\mathrm{N}}_i$ *the set of all strategies for* $\mathsf{Ag}_i$ *and by* $\Sigma^{\mathrm{N}} = \Sigma^{\mathrm{N}}_1 \times \cdots \times \Sigma^{\mathrm{N}}_n$ *the set of all strategy profiles.*

Alternatively, we can use a *correlated profile*, in which agent choices are correlated. For brevity, we refrain from formally defining a correlation mechanism (such as public signals) and map directly to joint actions.

**Definition 3** (Correlated profile)**.** *A* correlated profile *is a function* $\tau : H^{<K} \to \mathbb{P}(A)$ *such that if* $\tau(h)(\alpha) > 0$*, then* $\alpha = (a_1, \ldots, a_n)$ *and* $a_i \in A_i(last(h))$ *for all* $i \in N$*. We denote by* $\Sigma^{\mathrm{C}}$ *the set of correlated profiles.*

A (future) path $\pi$ of $\mathsf{C}$ starting from a history $h \in H^\ell$ in stage $\ell$ until the game ends in stage $K$ is a sequence $\pi = s^\ell \xrightarrow{\alpha^\ell} \cdots \xrightarrow{\alpha^{K-1}} s^K$ where $s^\ell = last(h)$, $s^k \in S$, $\alpha^k \in A$ and $\delta(s^k, \alpha^k)(s^{k+1}) > 0$. For path $\pi$, $\pi(k)$ is the $(k+1)$th state, $\pi[k]$ the action associated with the $(k+1)$th transition from $\pi(k)$ to $\pi(k+1)$, and $last(\pi)$ the final state.

**Rewards.** We endow NS-CSGs with *rewards*, that define agents' objectives. We use $r = (r_i)_{i \in N}$ where each agent $\mathsf{Ag}_i$ has a reward structure $r_i = (r^A_i, r^S_i)$ comprising action reward function $r^A_i : S \times A \to \mathbb{R}$ and state reward function $r^S_i : S \to \mathbb{R}$. An *objective profile* is $Y = (Y_1, \ldots, Y_n)$ where $Y_i(\pi)$ is the accumulated reward of $\mathsf{Ag}_i$ along a path $\pi$ starting in stage $\ell$, is defined as:

$$Y_i(\pi) = \sum_{k=0}^{K-\ell-1} \Big( r^A_i(\pi(k), \pi[k]) + r^S_i(\pi(k)) \Big) + r^S_i(last(\pi)).$$

Given a strategy profile $\sigma \in \Sigma^{\mathrm{N}}$, we denote by $\mathbb{E}^\sigma_{\ell,h}[Y_i]$ the expected value of $Y_i$ when starting from $h \in H^\ell$ and ending at the $K$th stage. Given a correlated profile $\tau \in \Sigma^{\mathrm{C}}$, we denote by $\mathbb{E}^\tau_\ell[Y_i, a'_i | a_i, h]$ the expected value of $Y_i$ when starting from $h \in H^\ell$ and ending at the $K$th stage, under the strategy that $\mathsf{Ag}_i$ takes the actual action $a'_i$ instead of the recommended action $a_i$ at $h$, and otherwise the recommendation by $\tau$ is followed by all agents.

An NS-CSG is *zero-sum* if $\sum_{i=1}^n \big( r^A_i(s, \alpha) + r^S_i(s) \big) = 0$ for all $s \in S$ and all $\alpha \in A$; otherwise, it is *nonzero-sum*.

**Social Welfare Subgame-Perfect Equilibria.** A *Nash equilibrium* (NE) ensures that no agent has an incentive to deviate unilaterally from their strategy. Here we work with *subgame-perfect Nash equilibria* (SPNEs) [Osborne et al., 2004], which are NEs in every state of the game. Since an SPNE is therefore an NE of every subgame of the original game, the agents' behaviour from any point in the game onward forms an NE of the continuation game, regardless of what happened before. We also consider the less well studied notion of *subgame-perfect correlated equilibria* (SPCEs) [Murray and Gordon, 2007]. For an SPCE, no agent can expect to gain by disobeying the recommendation of the correlated profile after any history of play.

The formal definitions of both types of subgame-perfect equilibria (SPE) follow, where we denote by $\mu = \mu_{-i}[\mu_i] = (\mu_1, \ldots, \mu_n)$ $(i \in N)$ the strategy profile, where $\mu_{-i}$ refers to the strategy profile except $\mu_i$. For SPCEs, we again omit a correlation mechanism and abuse notation by expressing it as individual deviations from the recommended actions associated to a correlated profile $\tau$.

**Definition 4** (Subgame-perfect equilibrium)**.** *For an initial state* $s \in S$*, a strategy profile* $\sigma^* = (\sigma^*_1, \ldots, \sigma^*_n) \in \Sigma^{\mathrm{N}}$ *is a* subgame-perfect Nash equilibrium *(SPNE) if* $\mathbb{E}^{\sigma^*}_{\ell,h}[Y_i] \geq \mathbb{E}^{\sigma^*_{-i}[\sigma_i]}_{\ell,h}[Y_i]$ *for all* $\sigma_i \in \Sigma^{\mathrm{N}}_i$*, all* $i \in N$ *and all* $h \in H^{<K}_s$*. A correlated profile* $\tau^* \in \Sigma^{\mathrm{C}}$ *is a* subgame-perfect correlated equilibrium *(SPCE) if* $\mathbb{E}^{\tau^*}_\ell[Y_i, a_i | a_i, h] \geq \mathbb{E}^{\tau^*}_\ell[Y_i, a'_i | a_i, h]$ *for all* $a_i, a'_i \in A_i(last(h))$*, all* $i \in N$ *and all* $h \in H^{<K}_s$*.*

We emphasize that the SPE is defined here for a given initial state. Since multiple SPEs can exist, we introduce additional optimality constraints. First, we define the *social welfare* $W^\sigma_{\ell,h}$ ($W^\tau_{\ell,h}$, resp.) of a history $h \in H^\ell$ $(\ell < K)$ under a strategy profile $\sigma$ (a correlated profile $\tau$, resp.) as the sum of expected values of objective profiles $Y_i$ starting in $h$ for all agents, that is, $W^\sigma_{\ell,h} = \mathbb{E}^\sigma_{\ell,h}[\sum_{i=1}^n Y_i]$ ($W^\tau_{\ell,h} = \mathbb{E}^\tau_{\ell,h}[\sum_{i=1}^n Y_i]$, resp.). Social-welfare optimal SPNE and and SPCE are then defined as follows.

**Definition 5** (Social welfare SPE). *For an initial state $s \in S$, an SPNE $\sigma^*$ is a social welfare optimal SPNE (SW-SPNE) of $\mathsf{C}$ if $W^{\sigma^*}_{0,s} \geq W^{\sigma}_{0,s}$ for all SPNEs $\sigma$ of $\mathsf{C}$. An SPCE $\tau^*$ is a social welfare optimal SPCE (SW-SPCE) of $\mathsf{C}$ if $W^{\tau^*}_{0,s} \geq W^{\tau}_{0,s}$ for all SPCEs $\tau$ of $\mathsf{C}$.*

Notice that, starting from a fixed initial state, SW-SPNE and SW-SPCE are *globally optimal*, i.e. over the social welfare achieved over a finite horizon from that start state.

Our approach of defining optimality in terms of the value from a fixed initial state is further motivated by the following result, which reveals that SW-SPNEs and SW-SPCEs do not possess the property of subgame perfection on social welfare, i.e., an SPNE or SPCE with optimal social welfare at one state might induce a non-optimal social welfare at another state as the game moves forward.

**Lemma 6** (No optimal subgame perfection). *For an initial state $s \in S$, an NS-CSG may have no SPNE (resp., SPCE) that is an SW-SPNE (resp., SW-SPCE) for all its subgames.*

A proof of this, and all other results in the paper can be found in Appendix A. Note also that this and the following results are stated in the context of NS-CSGs, but they also apply to general CSGs with discrete states and actions.

## 3 GENERALIZED BI

We now consider how to *compute* equilibria for NS-CSGs. For a fixed initial state, finite-horizon NS-CSGs are finite games, obtained by unfolding the game tree while invoking the NN perception function. In principle, this allows us to employ established game-theoretic solution such as backward induction. We next prove that the classical *generalized backward induction* (GBI) [Shoham and Leyton-Brown, 2009] can be used to find a finite-horizon SPNE or SPCE through local optimisation, but that this equilibrium might have an arbitrarily bad social welfare.

Algorithm 1 shows a version of the classical GBI method, for concurrent extensive-form games over a finite horizon, which aims to find an SPNE or SPCE that maximises social welfare, by computing an NE or CE which is *locally* maximal social welfare optimal at each history. In Algorithm 1, $\mathsf{HISTORY}(\mathsf{C}, s, \ell)$ computes a set of all histories in stage $\ell$ given an initial state $s \in S$. $\mathsf{SUCCESSOR}(\mathsf{C}, H^{\ell+1}_s, h)$ extracts a set of all successors of a history $h$ in stage $\ell$ from $H^{\ell+1}_s$. $\mathsf{SWE_SOLVER}\big(\mathsf{C}, r, \mathsf{T}, , h, \{V^{h'} \mid h' \in \mathrm{Succ}(h)\}\big)$ computes an SWNE or SWCE $\mu^h$ (depending on the equilibrium type $\mathsf{T} \in \{\mathrm{CE}, \mathrm{NE}\}$) of an induced normal-form game with actions available at $last(h)$ and utilities from the equilibrium payoffs $V^{h'}$ of all successors $h'$ of $h$, and then assigns the equilibrium payoff associated with this equilibrium to $V^h$. This procedure is iterated from the bottom up until $\ell = 0$, i.e., $h = s$, where the equilibrium payoffs of

**Algorithm 1** Generalized b/w induction (GBI) via SWE

**Input:** NS-CSG $\mathsf{C}$, rewards $r$, equ. type $\mathsf{T}$, initial state $s$
**Output:** an equilibrium $\mu$, equilibrium payoff vector $V$

$H^{\ell}_s \leftarrow \mathsf{HISTORY}(\mathsf{C}, s, \ell)$ for all $\ell \leq K$
**for** $\ell = K, K-1, \ldots, 0; h \in H^{\ell}_s$ **do**
  **if** $\ell = K$ **then**
    $V^h \leftarrow (r^S_1(last(h)), \ldots, r^S_n(last(h)))$
  **else**
    $\mathrm{Succ}(h) \leftarrow \mathsf{SUCCESSOR}(\mathsf{C}, H^{\ell+1}_s, h)$
    $(\mu^h, V^h) \leftarrow \mathsf{SWE_SOLVER}\big(\mathsf{C}, r, \mathsf{T}, , h, \{V^{h'} \mid h' \in \mathrm{Succ}(h)\}\big)$
$\mu \leftarrow \{\mu^h\}_{h \in H^{<K}_s}, V \leftarrow \{V^h\}_{h \in H_s}$
**return** $\mu, V$

histories at stage $K$ (i.e., where the game ends) are equal to final states' rewards. For this algorithm, we have the following proposition.

**Proposition 7.** *Given an initial state $s \in S$, GBI finds an SPNE $\sigma$ (SPCE $\tau$, resp.) with social welfare $W^{\sigma}_{0,s} = \sum_{i \in N} V^s_i$ ($W^{\tau}_{0,s} = \sum_{i \in N} V^s_i$, resp.).*

Although GBI can find an SPNE or SPCE, unfortunately it may return one with an arbitrarily bad social welfare with respect to the optimum.

**Lemma 8** (Bad social welfare). *The SPNE (SPCE, resp.) obtained by GBI SWE can be arbitrarily bad on social welfare with respect to an SW-SPNE $\sigma^*$ (SW-SPCE $\tau^*$, resp.) for some state $s \in S$, i.e., $W^{\sigma^*}_{0,s} - W^{\sigma}_{0,s}$ ($W^{\tau^*}_{0,s} - W^{\tau}_{0,s}$, resp.) is positive and unbounded.*

## 4 FROZEN SUBGAME IMPROVEMENT

Lemma 8 indicates that a GBI-based approach does not guarantee optimal social welfare. Motivated by this, we now consider further techniques to synthesize SW-SPNE and SW-SPCE for NS-CSGs. We first present an *exact* approach based on an unfolding of the game tree and the solution of a *nonlinear program*. However, this does not scale to large games. So we then propose an iterative *approximation* method called *frozen subgame improvement*. This works by first finding an arbitrary initial SPNE or SPCE and then iteratively freezing a set of variables and computing a new SPNE or SPCE with an increasing social welfare.

In this section, we focus initially on the case of *two-agent* NS-CSGs and then later discuss how to generalise this.

**Exact Computation of SW-SPNE and SW-SPCE.** Given an initial state $s \in S$, the game unfolds by considering all paths, thus generating a game tree which can be fully characterized by $H_s$. During the game tree construction, $last(h)$ can be computed for any $h \in H_s$, and if $h'$ is a successor of $h$, the joint action(s) that leads to $h'$ from $h$

can be determined. In contrast to [Akintunde et al., 2020a], where perception functions are assumed to be piecewise linear and encoded as constraints, unfolding the game tree allows us to treat NNs outside the optimisation problem.

We encode subgame perfection as a nonlinear program. An SPNE of the original game is an NE of every subgame, i.e., for each history $h \in H_s^{<K}$, it can be encoded as follows [1]:

$$\begin{aligned} V_i^h - \textstyle\sum_{(a_i,a_j)\in A_i\times A_j} \mu_i^h(a_i)\cdot\mu_j^h(a_j)\cdot \mathsf{Z}_i^{h,(a_i,a_j)} &= 0 \\ V_i^h - \textstyle\sum_{a_j\in A_j} \mu_j^h(a_j)\cdot \mathsf{Z}_i^{h,(a_i,a_j)} &\geq 0,\ \forall a_i \in A_i \\ \textstyle\sum_{a_i\in A_i} \mu_i^h(a_i) = 1,\ \mu_i^h(a_i) &\geq 0 \end{aligned} \tag{1}$$

for $i, j \in \{1, 2\}, i \neq j$, where $\mu_i^h \in \mathbb{P}(A_i(last(h)))$, $V^h = (V_1^h, V_2^h) \in \mathbb{R}^2$ denotes the expected accumulated reward vector from $h$ to the end of the game, and $\mathsf{Z}_i^{h,\alpha}$ denotes the expected accumulated reward to be received by $\mathsf{Ag}_i$ after executing the joint action $\alpha$ at $h$. In an SPCE, no agent can gain by deviating from the recommendation in any given history, and thus we have:

$$\begin{aligned} V_i^h - \textstyle\sum_{\alpha\in A} \mu_\alpha^h \cdot \mathsf{Z}_i^{h,\alpha} &= 0 \\ \textstyle\sum_{a_j\in A_j} (\mathsf{Z}_i^{h,(a_i,a_j)} - \mathsf{Z}_i^{h,(a_i',a_j)}) \cdot \mu_{(a_i,a_j)}^h &\geq 0 \\ \textstyle\sum_{\alpha\in A} \mu_\alpha^h = 1, \quad \mu_\alpha^h &\geq 0 \end{aligned} \tag{2}$$

where $i, j \in \{1, 2\}$, $i \neq j$, $a_i, a_i' \in A_i$, $\mu^h = \{\mu_\alpha^h\}_{\alpha\in A}$ and $\mu_\alpha^h$ represents the probability of the joint action $\alpha$ being recommended at $h$.

The SPNE and SPCE imply that, for each $h \in H_s^{<K}$ and $\alpha \in A(last(h))$, the reward for $\mathsf{Ag}_i$ satisfies:

$$\begin{aligned} \mathsf{Z}_i^{h,\alpha} = {} & r_i^A(last(h), \alpha) + r_i^S(last(h)) \\ & + \textstyle\sum_{h'\in \mathrm{Succ}(h)} \delta(last(h), \alpha)(last(h'))V_i^{h'} \end{aligned} \tag{3}$$

where, for each history $h \in H_s^K$, we take the reward vector $V^h = (r_1^S(last(h)), r_2^S(last(h)))$. For each $h \in H_s^{<K}$, let $C^{\mathrm{N},h}(\mu_1^h, \mu_2^h, V^h, \{V^{h'}\}_{h'\in\mathrm{Succ}(h)})$ be the union of constraints (1) and (3) (for Nash equilibria), and $C^{\mathrm{C},h}(\mu^h, V^h, \{V^{h'}\}_{h'\in\mathrm{Succ}(h)})$ be the union of constraints (2) and (3) (for correlated). The union of $C^{\mathrm{N},h}$ for all such histories is denoted by $C^{\mathrm{N}}(\mu^{\mathrm{N}}, V)$ and the union of $C^{\mathrm{C},h}$ by $C^{\mathrm{C}}(\mu^{\mathrm{C}}, V)$, where $\mu^{\mathrm{N}} := \{\mu_1^h, \mu_2^h\}_{h\in H_s^{<K}}$, $\mu^{\mathrm{C}} := \{\mu^h\}_{h\in H_s^{<K}}$ and $V := \{V^h\}_{h\in H_s^{<K}}$. Note that $C^{\mathrm{N}}(\mu^{\mathrm{N}}, V)$ ($C^{\mathrm{C}}(\mu^{\mathrm{C}}, V)$, resp.) is polynomial in $\mu^{\mathrm{N}}$ ($\mu^{\mathrm{C}}$, resp.) and $V$, and is nonlinear as $\mathsf{Z}_i^{h,\alpha}$ is related to variables $V_i^{h'}$ for $h' \in \mathrm{Succ}(h)$.

**Theorem 9** (Computation of SW-SPNE and SW-SPCE)**.** *For a two-agent NS-CSG* $\mathsf{C}$ *with an initial state* $s \in S$*,*

*(i) a strategy profile $\sigma$ is an SPNE iff there is a solution of the constraints $C^{\mathrm{N}}(\mu^{\mathrm{N}}, V)$ such that $\sigma_1(h) = \mu_1^h$ and $\sigma_2(h) = \mu_2^h$ for each $h \in H_s^{<K}$;*

*(ii) a correlated profile $\tau$ is an SPCE iff there is a solution of the constraints $C^{\mathrm{C}}(\mu^{\mathrm{C}}, V)$ such that $\tau(h) = \mu^h$ for each $h \in H_s^{<K}$;*

*(iii) a strategy profile $\sigma$ is an SW-SPNE iff there is an optimal solution $(\mu^*, V^*)$ of the nonlinear program:*

$$\max_{\mu^{\mathrm{N}}, V} \ \textstyle\sum_{i\in N} V_i^s \qquad \text{subject to} \quad C^{\mathrm{N}}(\mu^{\mathrm{N}}, V) \tag{4}$$

*such that $\sigma_1(h) = \mu_1^{*,h}$ and $\sigma_2(h) = \mu_2^{*,h}$ for each $h \in H_s^{<K}$, and the social welfare $W_{0,s}^\sigma$ is equal to the optimal value $\sum_{i\in N} V_i^{*,s}$;*

*(iv) a correlated profile $\tau$ is an SW-SPCE iff there is an optimal solution $(\mu^*, V^*)$ of the nonlinear program:*

$$\max_{\mu^{\mathrm{C}}, V} \ \textstyle\sum_{i\in N} V_i^s \qquad \text{subject to} \quad C^{\mathrm{C}}(\mu^{\mathrm{C}}, V) \tag{5}$$

*such that $\tau(h) = \mu^{*,h}$ for each $h \in H_s^{<K}$, and the social welfare $W_{0,s}^\tau$ is equal to the optimal value $\sum_{i\in N} V_i^{*,s}$.*

Problem (4) has at most $(|A_1| + |A_2| + 2)v$ variables and $(2|A_1||A_2| + 2|A_1| + 2|A_2| + 4)v$ constraints, and (5) has at most $(|A_1||A_2| + 2)v$ variables and $(|A_1||A_2| + |A_1|^2 + |A_2|^2 - |A_1| - |A_2| + 3)v$ constraints, where $v$ is the number of non-leaf nodes in the generated game tree and $v = \big((|A_1||A_2||S_1||S_2|)^K - 1\big)/(|A_1||A_2||S_1||S_2| - 1)$ in the worst case. The computation of SW-SPNE and SW-SPCE in Theorem 9 also applies to conventional stochastic games, because the game tree construction can work for general transition functions with finite branching.

**Frozen Subgame Improvement.** Nonlinear programs in Theorem 9 can be used to find an SW-SPNE or SW-SPCE efficiently for a small joint action profile and a short horizon. For larger problems, scalability is an issue because the numbers of variables and constraints are both exponential. To deal with this, we propose an approximation algorithm called *Frozen Subgame Improvement (FSI)* (Algorithm 2) that trades optimality for scalability.

The main idea of FSI is as follows. First, GBI is used to find a feasible solution to (4) or (5) depending on the equilibrium type $\mathsf{T} \in \{\mathrm{CE}, \mathrm{NE}\}$, i.e., an SPNE or SPCE. Then, a history $h \in H_s^{<K}$ is selected, for example by sampling uniformly. We freeze the distributions over (joint) actions and equilibrium payoffs corresponding to the histories that are not prefixes of $h$. Thus, (4), and similarly (5), can be simplified into a nonlinear program with a smaller number of variables and constraints. Finally, a new solution is computed by merging the frozen part of the current solution and an optimal solution of the simpler nonlinear program. The process performs a predefined number $m_{\max}$ of iterations.

In Algorithm 2, GENERALIZED_BI$(\cdot)$ computes an SPNE or SPCE $\mu$ and the associated equilibrium payoff vector $V$ by adopting a simpler version of Algorithm 1, in which an

[1] To simplify notation, $a_i \in A_i$ refers to $a_i \in A_i(last(h))$ in (1) and (2), and similarly for $a_j$ and $a_i'$.

**Algorithm 2** Frozen Subgame Improvement (FSI)

**Input:** NS-CSG $\mathsf{C}$, reward $r$, equ. type $\mathsf{T}$, init. state $s$, $m_{\text{max}}$
**Output:** an equilibrium $\mu$, equilibrium payoff vector $V$

1: $(\mu, V) \leftarrow \mathsf{GENERALIZED_BI}(\mathsf{C}, r, \mathsf{T}, , s)$
2: $m \leftarrow 0$
3: **repeat**
4: $\quad h \leftarrow \mathsf{A_HISTORY}(H_s^{<K}, \mu, V)$
5: $\quad P \leftarrow$ (4) or (5) (depending on $\mathsf{T}$) after freezing $\mu^{h'}, V^{h'}$ for each history $h' \in H_s^{<K}$ that is not a prefix of $h$ (say $h \in H_s^{\ell}$ for some $\ell < K$);
6: $\quad \{\mu^{*,h_{\leq \bar{\ell}}}, V^{*,h_{\leq \bar{\ell}}}\}_{\bar{\ell} \leq \ell} \leftarrow \mathsf{NP_SOLVER}(P)$
7: $\quad \mu \leftarrow \{\mu^{*,h_{\leq \bar{\ell}}}\}_{\bar{\ell} \leq \ell} \cup \{\text{the frozen } \mu^{h'}\}$
8: $\quad V \leftarrow \{V^{*,h_{\leq \bar{\ell}}}\}_{\bar{\ell} \leq \ell} \cup \{\text{the frozen } V^{h'}\}$
9: $\quad m \leftarrow m + 1$
10: **until** $m = m_{\text{max}}$
11: **return** $\mu, V$

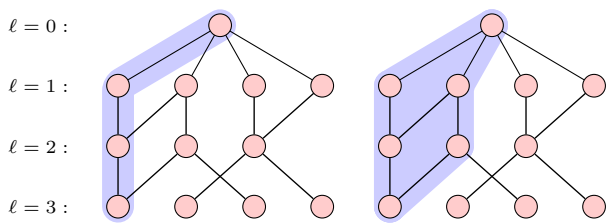


Figure 2: FSI over regions. Sampled history (left) and the corresponding region (right).

NE or CE is computed at step 7 instead of an SWNE or SWCE. $\mathsf{A_HISTORY}(\cdot)$ returns a history. Here, we sample a history from $H_s^{K-1}$ uniformly; an alternative is presented in Appendix B. $\mathsf{NP_SOLVER}(\cdot)$ computes an optimal solution to a given nonlinear program.

For FSI, we have the following results:

**Theorem 10** (FSI). *If FSI is adopted to solve* (4) *(*(5)*, resp.) approximately, then:*

*(i) the pair $(\mu, V)$ is a feasible solution to* (4) *(*(5)*, resp.) at the end of each iteration $m$, that is, $\mu$ is an SPNE (SPCE, resp.) and $V$ is the equilibrium payoff vector;*

*(ii) the social welfare $\sum_{i \in N} V_i^s$ is monotonically increasing in $m$, and also monotonically increasing in $m_{max}$.*

**FSI over Regions.** If each agent has a limited memory and takes actions conditioned on the current state and stage, we can unfold the game into a graph where each node in a stage represents one reachable state exactly in that stage, as in Fig. 2. With respect to the game tree, the number of nodes in this graph is greatly decreased if many states are frequently visited in a stage. The FSI can be directly adapted to this graph by first sampling a history (Fig. 2: left) and then optimising over a region of states, which contain all histories that reach its last state (Fig. 2: right).

**Multi-agent.** SW-SPNE and SW-SPCE computation for *multi-agent* ($n$>2) NS-CSGs can be performed by replacing (1) or (2) with the encoding of NE/CE computation for the induced multi-agent normal-form game at each $h \in H_s^{<K}$.

## 5 EXPERIMENTAL EVALUATION

We have implemented a prototype version of our FSI method (Algorithm 2). This uses components from PRISM-games 3.0 [Kwiatkowska et al., 2020], which supports discrete CSGs without perception. In particular, we use its SMT-based/linear programming method for synthesising CSG SWNE/SWCE to initialise the vector of equilibria values in line 1 of Algorithm 2. Its support for two-player finite-horizon equilibria [Kwiatkowska et al., 2019] also gives an equivalent version of the GBI algorithm (Algorithm 1).

The optimisation problems for computing SW-SPNE and SW-SPCE values for states are solved using Gurobi. In order to improve the scalability of FSI, our implementation considers a reduced set of histories by: (i) limiting the information that the players have access to at each state to be the values of the variables in that state plus *time*, i.e., how many transitions have been made up until that point; and (ii) constructing histories not over states, but *regions* of states which are independent from a decision-making standpoint.

Our evaluation employs two case studies: the first is used to show the applicability of our equilibria improvement algorithm, and the second to demonstrate the usefulness of equilibria properties for analysing NS-CSGs. An overview is provided below, with Appendix C including more details.

**Automated Parking.** We first formulate a dynamic vehicle parking problem as an NS-CSG (a static assignment game is considered in, e.g., [Ayala et al., 2011]). There are 2 players (vehicles) targeting 2 parking slots in a $5 \times 4$ grid, shown in Fig. 3 (target cells are green, forbidden cells are red, black arrows show traffic rules). We consider two reward structures. One minimises time, while the other extends the first by giving a bonus to player 2 for visiting a designated cell (in yellow). This is a discrete-state model in which percepts identify agent locations precisely. We use it to compare the equilibria algorithms for two different time horizons $K$. For this model, both vehicles get a reward of -1 for each move, vehicle 2 gets a reward of 5.5 when visiting the bonus cell and the speeds of vehicle 1 and 2 are of two and one grid per move, respectively.

We first consider Nash equilibria. For the first reward structure, our FSI algorithm and the GBI algorithm, which only considers local SWNE values, both return the SW-SPNE strategy with reward sum $-5.0$ in Fig. 3(top-left). For the second reward structure, FSI finds a new SW-SPNE strategy with reward sum $-4.5$ in Fig. 3(top-right) giving a higher social welfare, while GBI still returns the strategy on the left, which is not an SW-SPNE in this case.

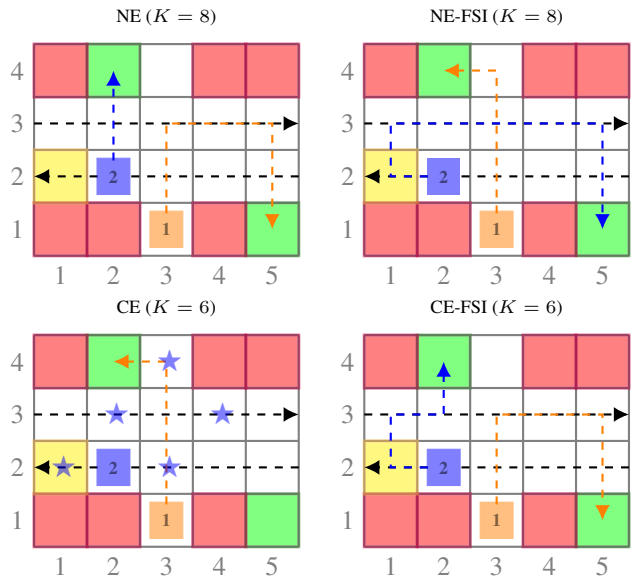


Figure 3: Strategies for the automated parking example.

For correlated equilibria, for $K = 8$ both algorithms produce the same strategy as in Fig. 3(bottom-right), for which the reward sum is -1.5. We then reduce the time horizon to $K = 6$. For this case, in the strategy constructed by the GBI algorithm in Fig. 3(bottom-left), vehicle 2 is instructed to move left in order to get the bonus, while vehicle 1 is instructed to park in the closest spot. However, given the shorter horizon, vehicle 2 does not have enough time to park in the remaining spot and the overall reward sum is -2.5. The possible final positions for vehicle 2 are indicated by the blue stars. In the strategy synthesised by the FSI algorithm, however, both cars park and the sum of rewards is higher.

**Two-Agent Aircraft Collision Avoidance Scenario.** Secondly, we consider an NS-CSG model of the VCAS[2] system, as described earlier in Example 1. We study its equilibria strategies, in contrast to the zero-sum (reachability) properties analysed in [Akintunde et al., 2020a]. Fig. 4 plots the altitude $h$ for equilibria and zero-sum strategies when maximising $h$ for a given instant $k$. It can be seen that, with respect to the safety criterion established by [Akintunde et al., 2020a, Julian and Kochenderfer, 2019], i.e., avoiding a near mid-air collision, equilibria strategies allow the two aircraft to reach a safe configuration within a shorter horizon, which would be missed by a zero-sum analysis.

Finally, we consider a second reward structure incorporating the trust level and fuel consumption, and vary the agent uncertainty parameters $\epsilon_i$ (see Appendix C for details). We also fix a different safety limit of $h = 200$. Table 1 shows the altitude and number of violations (times that no advisory is taken) for the generated equilibria. To give an indication of scalability and performance, we also include the total number of states in the game unfolding and the time for model construction and algorithm execution for both NE and CE. For this example, both types of equilibria yield the

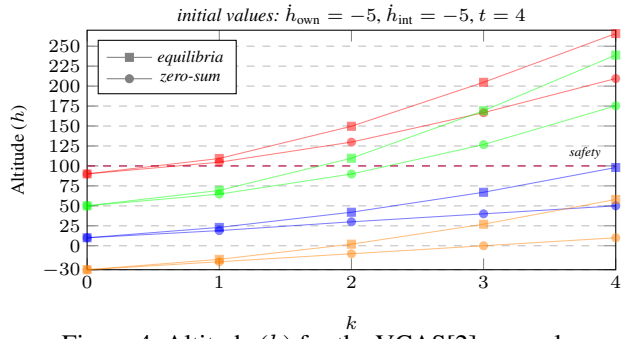


Figure 4: Altitude ($h$) for the VCAS[2] example.

| $\epsilon_{own}, \epsilon_{int}$ | $t$ | States | Constr. time (s) | B.I. time (s) NE | B.I. time (s) CE | $h$ | Viol. |
|---|---|---|---|---|---|---|---|
| 0, 0 | 2 | 100 | 0.06 | 0.1 | 0.05 | 82 | 0 |
| | 3 | 836 | 0.6 | 0.7 | 0.3 | 123 | 0 |
| | 4 | 6997 | 36.6 | 8.0 | 1.8 | 199 | 25% |
| 0.1, 0 | 2 | 157 | 0.1 | 0.2 | 0.1 | 82 | 0 |
| | 3 | 1622 | 1.4 | 1.0 | 0.3 | 123 | 0 |
| | 4 | 16028 | 273.8 | 14.2 | 3.3 | 199 | 20% |
| 0.1, 0.2 | 2 | 251 | 0.1 | 0.2 | 0.07 | 82 | 0 |
| | 3 | 3174 | 4.4 | 1.5 | 0.6 | 123 | 0 |
| | 4 | 36639 | 1497.2 | 26.7 | 5.8 | 199 | 20% |

Table 1: Statistics for the VCAS[2] example.

same values for the properties considered.

We see that the agents always comply with the advisory system for smaller values of $t$, given that reaching safety would be of a higher priority. This changes, however, when both aircraft have a larger horizon to consider. By examining the generated equilibria strategies (see Appendix C), we see that small deviations in order to reduce consumption would not prevent them from approaching the safety limit.

**Efficiency and scalability.** For equilibria computation using GBI, which computes locally optimal equilibria, CE are generally considerably faster to compute than NE. This is due to the fact that finding an optimal CE in a state can be reduced to solving a *linear program*, while computing an optimal NE requires finding all solutions of a *linear complementarity problem*. The same, however, is not observed when comparing the performance of FSI on the two types of equilibria. This is because a path-based encoding requires a greater number of constraints and variables for CE, and we need to solve nonlinear programs.

## 6 CONCLUSIONS

We have considered finite-horizon equilibria computation for CSGs whose agents are equipped with NN-based perception mechanisms. We developed an approximate algorithm that improves on social welfare equilibria values and strategies, for both SPNE and SPCE, compared to backward induction, which can only reason about local optimality. A prototype implementation showcased its applicability and advantages on two case studies. Future work will focus on infinite-horizon properties (incorporating finite-horizon equilibria with receding horizon synthesis [Raman et al., 2015]) and temporal logic specifications.

## References


Dilip Abreu, Benjamin Brooks, and Yuliy Sannikov. Algorithms for stochastic games with perfect monitoring. *Econometrica*, 88(4):1661–1695, 2020.

Michael E. Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. Verifying Strategic Abilities of Neural-symbolic Multi-agent Systems. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, pages 22–32. IJCAI Organization, 9 2020a.

Michael E Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. Formal verification of neural agents in non-deterministic environments. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, pages 25–33. Springer, 2020b.

Robert J Aumann. Subjectivity and correlation in randomized strategies. *Journal of mathematical Economics*, 1(1): 67–96, 1974.

Daniel Ayala, Ouri Wolfson, Bo Xu, Bhaskar Dasgupta, and Jie Lin. Parking slot assignment games. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 299–308. Association for Computing Machinery, 2011.

Patricia Bouyer, Nicolas Markey, and Daniel Stan. Mixed Nash equilibria in concurrent terminal-reward games. In *FSTTCS 2014*, pages 1–12, 2014.

Thomas Brihaye, Véronique Bruyère, Aline Goeminne, Jean-François Raskin, and Marie van den Bogaard. The complexity of subgame perfect equilibria in quantitative reachability games. *Logical Methods in Computer Science*, 16(4):1–43, 2020.

Andriy Burkov and Brahim Chaib-draa. An approximate subgame-perfect equilibrium computation technique for repeated games. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 10)*, page 729–736. AAAI Press, 2010.

Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Strategy improvement for concurrent reachability and turn-based stochastic safety games. *Journal of Computer and System Sciences*, 79(5):640–657, 2013.

Luca de Alfaro and Rupak Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68(2):374–397, 2004.

Luca De Alfaro, Thomas A Henzinger, and Orna Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188–217, 2007.

Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neural-symbolic artificial intelligence. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, pages 4943–4950. IJCAI Organization, 07 2020.

Dileepa Fernando, Naipeng Dong, Cyrille Jegourel, and Jin Song Dong. Verification of strong Nash-equilibrium for probabilistic bar systems. In *International Conference on Formal Engineering Methods (ICFEM 2018)*, pages 106–123. Springer, 2018.

Drew Fudenberg and David Levine. Subgame–perfect equilibria of finite–and infinite–horizon games. In *A Long-Run Collaboration On Long-Run Games*, pages 3–20. World Scientific, 2009.

Karel Horák and Branislav Bošanskỳ. Solving partially observable stochastic games with public observations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 19)*, volume 33, pages 2029–2036. AAAI Press, 2019.

Kyle D. Julian and Mykel J. Kochenderfer. A reachability method for verifying dynamical systems with deep neural network controllers. *CoRR*, abs/1903.00520, 2019.

Kyle D. Julian, Shivam Sharma, Jean-Baptiste Jeannin, and Mykel J. Kochenderfer. Verifying aircraft collision avoidance neural networks through linear approximations of safe regions. *CoRR*, abs/1903.00762, 2019.

Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

Mitri Kitti. Subgame perfect equilibria in discounted stochastic games. *Journal of Mathematical Analysis and Applications*, 435(1):253–266, 2016. ISSN 0022-247X.

Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. Equilibria-based probabilistic model checking for concurrent stochastic games. In *Proc. 23rd International Symposium on Formal Methods (FM'19)*, volume 11800 of *LNCS*, pages 298–315. Springer, 2019.

Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. Prism-games 3.0: Stochastic game verification with concurrency, equilibria and time. In *International Conference on Computer Aided Verification (CAV 2020)*, pages 475–487. Springer, 2020.

Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. Automatic verification of concurrent stochastic systems. *Formal Methods in System Design*, pages 1–63, 2021.

Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. Correlated equilibria and fairness in

concurrent stochastic games. In *Proc. 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'22)*, LNCS. Springer, 2022.

Luís Lamb, Artur Garcez, Marco Gori, Marcelo Prates, Pedro Avelar, and Moshe Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, pages 4810–4817. IJCAI Organization, 07 2020.

Peixuan Li and Chuangyin Dang. An arbitrary starting tracing procedure for computing subgame perfect equilibria. *J. Optim. Theory Appl.*, 186(2):667–687, 2020.

Michael L. Littman, Nishkam Ravi, Arjun Talwar, and Martin Zinkevich. An efficient optimal-equilibrium algorithm for two-player game trees. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI 06)*, page 298–305. AUAI Press, 2006.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 17)*, page 6382–6393. Curran Associates Inc., 2017.

Federico Mari, Igor Melatti, Ivano Salvo, Enrico Tronci, Lorenzo Alvisi, Allen Clement, and Harry Li. Model checking coalition Nash equilibria in MAD distributed systems. In *Symposium on Self-Stabilizing Systems (SSS 2009)*, pages 531–546. Springer, 2009.

Panagiotis Kouvaros Michael E Akintunde, Elena Botoeva and Alessio Lomuscio. Venmas: Verification of neural-symbolic multi-agent systems, 2020. https://vas.doc.ic.ac.uk/software/neural/.

Chris Murray and Geoff Gordon. *Finding correlated equilibria in general sum stochastic games*. Carnegie Mellon University, School of Computer Science, Machine Learning, 2007.

John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning (ICML 2017)*, pages 2681–2690. JMLR.org, 2017.

Martin J Osborne et al. *An introduction to game theory*, volume 3. Oxford university press, New York, 2004.

Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *proc. 35th Conference on Neural Information Processing Systems (NeurIPS'21)*, 2021.

Vasumathi Raman, Mattias Fält, Tichakorn Wongpiromsarn, and Richard M. Murray. Online horizon selection in receding horizon temporal logic planning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3493–3499, 2015.

R. Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory*, 4:25–55, 1975.

Lloyd S Shapley. Stochastic games. *Proc. National Academy of Sciences*, 39(10):1095–1100, 1953.

Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Rui Yan, Xiaoming Duan, Zongying Shi, Yisheng Zhong, Jason R. Marden, and Francesco Bullo. Policy evaluation and seeking for multi-agent reinforcement learning via best response. *IEEE Transactions on Automatic Control*, pages 1–16, 2021.

Rui Yan, Gabriel Santos, Gethin Norman, David Parker, and Marta Kwiatkowska. Strategy synthesis for zero-sum neuro-symbolic concurrent stochastic games. *arXiv:2202.06255*, 2022.

Sevin Yeltekin, Yongyang Cai, and Kenneth L. Judd. Computing equilibria of dynamic games. *Operations Research*, 65(2):337–356, 2017.

# Finite-horizon Equilibria for Neuro-symbolic Concurrent Stochastic Games

# Technical Appendix

## A PROOFS OF MAIN RESULTS

To prove Lemmas 6 and 8, we introduce the following example.

**Example 2.** *Consider a two-stage two-agent game with deterministic transitions in Fig. 5, in which each agent has two actions: $\{U, D\}$ for agent 1 and $\{L, R\}$ for agent 2. Non-leaf and leaf nodes, containing the node numbers, are marked with circles and rectangles, respectively. For clarity, several histories reaching stage 2 are not displayed here. Edges are labelled with the associated joint actions. The payoff vectors below leaf nodes are the terminal rewards, while the payoff vectors below non-leaf nodes denote the unique equilibrium payoffs (expected accumulated rewards) from these nodes to the leaf nodes, where $\phi$ is negative. The immediate rewards along the edges are assumed to be zero.*

*By GBI, there are three NEs at node 4: $\mu^{4(1)} = \{(1,0),(1,0)\}$, $\mu^{4(2)} = \{(1/5,4/5),(1,0)\}$ and $\mu^{4(3)} = \{(0,1),(0,1)\}$, and the respective equilibrium payoffs are $V^{4(1)} = (0,8)$, $V^{4(2)} = (0,8/5)$ and $V^{4(3)} = (5,2)$. The NE and the equilibrium payoff at the initial node 1 depend on which NE is considered at node 4. If $V^{4(1)}$ or $V^{4(2)}$ is selected, then there is a unique NE at node 1: $\mu^{1(1)} = \{(1,0),(1,0)\}$ with equilibrium payoff $(1, 1+\phi)$. If $V^{4(3)}$ is chosen, then there is a unique NE at node 1: $\mu^{1(2)} = \{(0,1),(1,0)\}$ with equilibrium payoff $(5,2)$.*

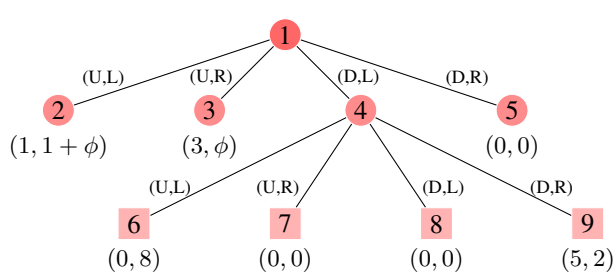


Figure 5: A two-stage game tree with two agents with $\phi < 0$.

**Proof of Lemma 6.** We consider the game in Example 2. Given $\phi < 0$, the SW-SPNE and SW-SPCE starting at node 1 are the same and unique with social welfare $5 + 2 = 7$, in which the strategy at node 4 is $\mu^{4(3)}$. However, the SW-SPNE and SW-SPCE for the subgame starting at node 4 are both $\mu^{4(1)}$ instead of $\mu^{4(3)}$, which completes the proof.

**Proof of Proposition 7.** It is well known in game theory that, for a normal-formal game, (mixed-strategy) NEs always exist [Nash, 1951] and all NEs are fully characterized by the set of feasible solutions of a nonlinear program with compact constraints [Osborne et al., 2004]. This implies that the SWNEs, which are NEs maximising social welfare, always exist as well. Since every NE is a CE and all CEs are fully characterized by the set of feasible solutions of a linear program with compact constraints [Aumann, 1974], then SWCEs always exist, which completes the proof.

**Proof of Lemma 8.** We consider Example 2 again. Since $\mu^{4(1)}$ has the maximum social welfare, then Generalized BI via SWE feeds $V^{4(1)}$ to node 1 for both the case of SWNE and SWCE, thus leading to node 1's social welfare $W^{\mu}_{0,s} = 2+\phi$. However, node 1's social welfare $W^{\mu^*}_{0,s}$ under both SW-SPNE and SW-SPCE $\mu^*$ is 7. Thus, if $\phi$ is negative enough, the difference $W^{\mu^*}_{0,s} - W^{\mu}_{0,s} = 5 - \phi$ is positive and unbounded.

**Proof of Theorem 9.** The conclusions (i) and (ii) are straightforward by the encoding procedure. The sets of feasible solutions to (4) and (5) are not empty, as (mixed-strategy) NEs of a normal-form game always exist [Nash, 1951], and thus so are CEs. Additionally, they are compact by noting the constraints (1), (2) and (3). Then, the conclusions (iii) and (iv) follow from the continuity of the objective function.

**Proof of Theorem 10.** In Algorithm 2, step 1 returns a feasible solution to the nonlinear program (4) or (5) (depending on the equilibrium type $\mathsf{T}$). Since the variables of the nonlinear program $P$ (step 5) are independent of the frozen variables due to the game tree structure and the his-

tory selection (or region construction), the pair $(\mu, V)$ in steps 7 and 8 is still a feasible solution to (4) or (5). The conclusions follow from the coordinate descent optimization with constraints [Wright, 2015].

# B FURTHER DETAILS FOR ALGORITHMS

## B.1 APPROXIMATION ALGORITHMS

FSI is described in Sec. 4 and, in particular, Algorithm 2. In Fig. 6, we give an illustration of the approach: FSI freezes all variables related to the red histories and optimizes over the blue history, where each node contains the current equilibrium payoff.

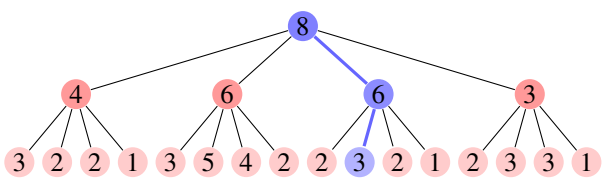


Figure 6: An example for Frozen Subgame Improvement.

We also suggest an alternative approach for the selection of histories in FSI, shown in Algorithm 3. It returns a history by starting from the initial state $s$, moving to the successor with the maximum social welfare indicated by the current equilibrium payoff $V$ and perturbed by $\epsilon$ (if there are multiple such successors, we select one randomly), and iterating until the stage $K-1$, where $\mathsf{UNIFORM}(\cdot)$ is a uniform sampling function.

**Algorithm 3** Finding a History by Maximum Social Welfare

**Input:** histories $H_s$, distribution $\mu$, equilibrium payoff $V$, exploration rate $\epsilon \in [0,1]$
**Output:** a history $h \in H_s^{K-1}$

1: $h \leftarrow s$
2: **repeat**
3: $\quad h' \leftarrow \arg\max_{h'' \in \mathrm{Succ}(h)} \sum_{i \in N} V_i^{h''}$
4: $\quad$ **if** $\mathsf{UNIFORM}([0,1]) > \epsilon$ **then**
5: $\quad\quad h \leftarrow h'$
6: $\quad$ **else**
7: $\quad\quad h \leftarrow \mathsf{UNIFORM}(\mathrm{Succ}(h))$
8: $\quad$ **end if**
9: **until** $h \in H_s^{K-1}$
10: **return** $h$

# C FURTHER DETAILS FOR CASE STUDIES

## C.1 AUTOMATED PARKING

The formal details of the NS-CSG model for the automated parking case study are as follows. There are two players (vehicles)$\{\mathsf{Ag}_i\}_{i \in N}$ for $N = \{1,2\}$ and two parking slots $M = \{1,2\}$ in a $5 \times 4$ grid $C$. The coordinate of the cell in the $i$th row and $j$th column is denoted by $(i,j)$. Thus, $C = \{(i,j) \mid i \in [5], j \in [4]\}$. The coordinates of two parking slots are $y_1 = (2,4)$ and $y_2 = (5,1)$. Fig. 3 shows the grid. Vehicles are forbidden to enter the red cells and have to follow the traffic rules indicated by black arrows.

The environment state is $s_E = (x_1, x_2)$, where $x_i \in C$ is vehicle $i$'s coordinate. Each agent $i \in N$ is as follows:

- a state of agent $\mathsf{Ag}_i$ is $s_i = (prv_i, (x_1, x_2))$, where the local state $prv_i$ is dummy, and the coordinates $x_k \in C$ ($k \in N$) of two vehicles constitute the percept;
- actions include four directions $\mathrm{U} = (0,1)$, $\mathrm{D} = (0,-1)$, $\mathrm{L} = (-1,0)$, and $\mathrm{R} = (1,0)$. We assume that $\mathsf{Ag}_1$ is twice as fast as $\mathsf{Ag}_2$, i.e., $A_2 = \{\mathrm{U}, \mathrm{D}, \mathrm{L}, \mathrm{R}\}$ and $A_1 = A_2 \times A_2 \setminus \{\mathrm{UD}, \mathrm{DU}, \mathrm{LR}, \mathrm{RL}\}$;
- the available action function is such that $a_i \in \Delta_i(s_i)$ iff taking action $a_i$ at $s_i$ does not break the traffic rules or enter a red cell;
- observation function $obs_i$ computes the cells where two vehicles are, i.e., $obs_i(s_1, s_2, s_E) = (x_1, x_2)$;
- the local transition function $\delta_i$ is dummy.

For $\alpha = (a_1, a_2) \in A_1 \times A_2$, $\delta_E(s_E, \alpha) = (x_1', x_2')$ where $x_i' = x_i + a_i$ for all $i \in N$. The two vehicles start from $x_1^0 = (3,1)$ and $x_2^0 = (2,2)$.

There are two reward structures. The first one is plain time minimizing: $r_i^A(s, \alpha) = 0$; if $x_1 = x_2$, then $r_i^S(s) = -20$; if $x_1 \neq x_2$ and $x_i = y_j$ for some $j \in M$, then $r_i^S(s) = 0$; $r_i^S(s) = -1$ otherwise. The second one is time minimizing with bonus, in which we add a bonus of 5.5 to agent 2 at a designated cell (in yellow): $r_2^S(s) = 5.5 - 1 = 4.5$ if $x_2 = (1,2)$ when $k \leq 1$.

This example was modelled using the PRISM-games modelling language, since the simplicity of the perception mechanism lets it be reduced to a discrete-state CSG. Table 2 shows statistics for the models constructed and the time for their verification.

## C.2 TWO-AGENT AIRCRAFT COLLISION AVOIDANCE SCENARIO

In the VCAS[2] system (Figure 1) there are two aircraft (ownship and intruder, denoted by $\mathsf{Ag}_i$ for $i \in \{\mathrm{own}, \mathrm{int}\}$),

| $K$ | States | Trans. | Constr. time (s) | B.I. time(s) | | Region size | | Verif. time(s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | NE | CE | NE | CE | NE | CE |
| 6 | 258 | 1080 | 0.01 | 0.6 | 2.1 | 24.0%<br>19.4%<br>17.8% | 22.5%<br>20.2%<br>16.3% | 0.4<br>0.4<br>0.2 | 1.5<br>1.0<br>0.3 |
| 8 | 386 | 1689 | 0.2 | 1.4 | 4.9 | 37.3%<br>32.4%<br>25.9% | 32.4%<br>27.5%<br>25.9% | 3.8<br>1.8<br>1.1 | 2.5<br>2.6<br>1.8 |

Table 2: Statistics for the automated parking example.

each of which is equipped with an NN-controlled collision avoidance system called VCAS. Each second, VCAS issues an advisory ($ad_i$) from which, together with the current trust in the previous advisory ($tr_i$), the pilot needs to make a decision about accelerations, aiming at avoiding a near mid-air collision (NMAC), a region where two aircraft are separated by less than 100 ft vertically and 500 ft horizontally.

The environment state $s_E = (h, \dot{h}_{\text{own}}, \dot{h}_{\text{int}}, t)$ records the altitude $h$ of the intruder relative to the ownship (ft), the vertical climb rate $\dot{h}_{\text{own}}$ of the ownship (ft/sec), the vertical climb rate $\dot{h}_{\text{int}}$ of the intruder (ft/sec), and the time $t$ until loss of horizontal separation of the two aircraft (sec).

Each aircraft is endowed with a perception function implemented via a feed-forward NN $f_{ad_i} : \mathbb{R}^4 \to \mathbb{R}^9$ with four inputs, seven hidden layers of 45 nodes and nine outputs representing the score of each possible advisory. There are nine NNs $F = \{f_i : \mathbb{R}^4 \to \mathbb{R}^9 \mid i \in [9]\}$, each of which corresponds to an advisory.

Each advisory will provide two non-zero acceleration actions for the agent to select from, except that the agent is also allowed to adopt zero acceleration. The trust in the previous advisory and previous advisory (percept) are stored in a state of the agent $s_i = (tr_i, ad_i)$. There are four trust levels $\{4, 3, 2, 1\}$ and nine possible advisories [Akintunde et al., 2020b]. The current advisory is computed from the previous advisory $ad_i$ and environment state $s_E$ using the observation function $obs_i$. The trust level is increased probabilistically if the current advisory is compliant with the executed action, and decreased otherwise.

Formally, each agent $\mathsf{Ag}_i$ for $i \in \{\text{own}, \text{int}\}$ and the environment $E$ are defined as follows:

- $s_i = (tr_i, ad_i)$ is a state of the agent $\mathsf{Ag}_i$ with local state $tr_i \in [4]$ and percept $ad_i \in [9]$;
- the set of environment states is $S_E = [-3000, 3000] \times [-2500, 2500] \times [-2500, 2500] \times [0, 40]$, with $s_E = (h, \dot{h}_{\text{own}}, \dot{h}_{\text{int}}, t)$ as above;
- $A_i = \{0, \pm 3.0, \pm 7.33, \pm 9.33, \pm 9.7, \pm 11.7\}$, where $a_i \in A_i$ is an acceleration $\ddot{h}_i$;
- the available action function $\Delta_i$ returns two non-zero acceleration actions [Akintunde et al., 2020a] shown in Table 3 given a state of the agent, plus zero acceleration;
- observation function $obs_i$, implemented via $F$, is given by $ad'_i = obs_i(ad_i, s_E)$, where $obs_{\text{own}}(ad_{\text{own}}, s_E) = \text{argmax}(f_{ad_{\text{own}}}(h, \dot{h}_{\text{own}}, \dot{h}_{\text{int}}, t))$ and $obs_{\text{int}}(ad_{\text{int}}, s_E) = \text{argmax}(f_{ad_{\text{int}}}(-h, \dot{h}_{\text{int}}, \dot{h}_{\text{own}}, t))$;
- the local transition function $\delta_i$ computes a trust level according to the current trust level $tr_i$, the updated advisory $ad'_i$ and the executed action $a_i$: if $a_i$ is compliant with $ad'_i$ (i.e., $a_i$ is non-zero), when $tr_i \leq 3$, then $tr'_i = tr_i + 1$ with probability $1 - \epsilon_i$ and $tr'_i = tr_i$ with probability $\epsilon_i$, and when $tr_i = 4$, then $tr'_i = tr_i$; otherwise, when $tr_i \geq 2$, then $tr'_i = tr_i - 1$ with probability $1 - \epsilon_i$ and $tr'_i = tr_i$ with probability $\epsilon_i$, and when $tr_i = 1$, then $tr'_i = tr_i$, where $\epsilon_i \in [0, 1]$.
- the environment transition function $\delta_E(s_E, \alpha)$ is defined as: $h' = h - \Delta t(\dot{h}_{\text{own}} - \dot{h}_{\text{int}}) - 0.5\Delta t^2(\ddot{h}_{\text{own}} - \ddot{h}_{\text{int}})$, $\dot{h}'_{\text{own}} = \dot{h}_{\text{own}} + \ddot{h}_{\text{own}}\Delta t$, $\dot{h}'_{\text{int}} = \dot{h}_{\text{int}} + \ddot{h}_{\text{int}}\Delta t$ and $t' = t - \Delta t$, where $\Delta t = 1$ is the time step.

When computing the equilibria presented in Fig. 4, we use two reward structures, with the first given by $r^S_{\text{own}}(s) = r^S_{\text{int}}(s) = h$ if $k = t_{\text{init}} - t$, and 0 otherwise. For the zero-sum case, the reward for the intruder is negated. In both cases, action rewards are set to 0 for all state-action pairs, i. e., $r^A_{\text{own}}(s, \alpha) = r^A_{\text{int}}(s, \alpha) = 0, \forall s \in S, \alpha \in A$.

This case study was developed by extending the implementation available in [Michael E Akintunde and Lomuscio, 2020]. We first modified the original code in order to consider all actions recommended by the advisory system plus the action corresponding to zero acceleration. We later develop this model further by adding trust values to the states of the agents and the corresponding probabilistic updates as described in Section 2. In both cases, we build a game tree by considering all states the system could be in and translate that into a PRISM-games model.

We also consider another reward structure with additional preferences: (i) not only safety but also trust matters; and (ii) reducing fuel consumption is desired in addition to maintaining safety. More specifically, if $|h| \leq 200$, then $r^A_i(s, \alpha) = 0$ and $r^S_i(s) = |h|/h_{\max} + tr_i/4$; if $|h| > 200$, then $r^A_i(s, \alpha) = -|\ddot{h}_i|/\ddot{h}_{\max}$ and $r^S_i(s) = 0$ for $i \in \{\text{own}, \text{int}\}$, where $h_{\max}$ and $\dot{h}_{\max}$ are the maximal absolute values of all altitudes and accelerations in the generated game tree, respectively. The initial values are $h = 50$, $\dot{h}_{\text{own}} = -5$, $tr_{\text{own}} = 4$, $\dot{h}_{\text{int}} = 5$ and $tr_{\text{int}} = 4$.

In Section 5 we discussed equilibria strategies for different values of the uncertainty parameter $\epsilon_{\text{own}}$. Fig. 7 (left) illustrates that following the advisories is the best strategy when safety and trust are the priority. The strategy in Fig. 7 (right) shows that violations may occur (i.e., zero action is adopted after $s^2$) when the altitude is expected to reach a safer level in the future.

| Label ($ad_i$) | Advisory | Description | Vertical Range (Min, Max) ft/min | Available Actions ft/s$^2$ |
|---|---|---|---|---|
| 1 | COC | Clear of Conflict | $(-\infty, +\infty)$ | -3, +3 |
| 2 | DNC | Do Not Climb | $(-\infty, 0]$ | -9.33, -7.33 |
| 3 | DND | Do Not Descend | $[0, +\infty)$ | +7.33, +9.33 |
| 4 | DES1500 | Descend at least 1500 ft/min | $(-\infty, -1500]$ | -9.33, -7.33 |
| 5 | CL1500 | Climb at least 1500 ft/min | $[+1500, +\infty)$ | +7.33, +9.33 |
| 6 | SDES1500 | Strengthen Descend to at least 1500 ft/min | $(-\infty, -1500]$ | -11.7, -9.7 |
| 7 | SCL1500 | Strengthen Climb to at least 1500 ft/min | $[+1500, +\infty)$ | +9.7, +11.7 |
| 8 | SDES2500 | Strengthen Descend to at least 2500 ft/min | $(-\infty, -2500]$ | -11.7, -9.7 |
| 9 | SCL2500 | Strengthen Climb to at least 2500 ft/min | $[+2500, +\infty)$ | +9.7, +11.7 |

Table 3: Two non-zero available actions given an advisory.

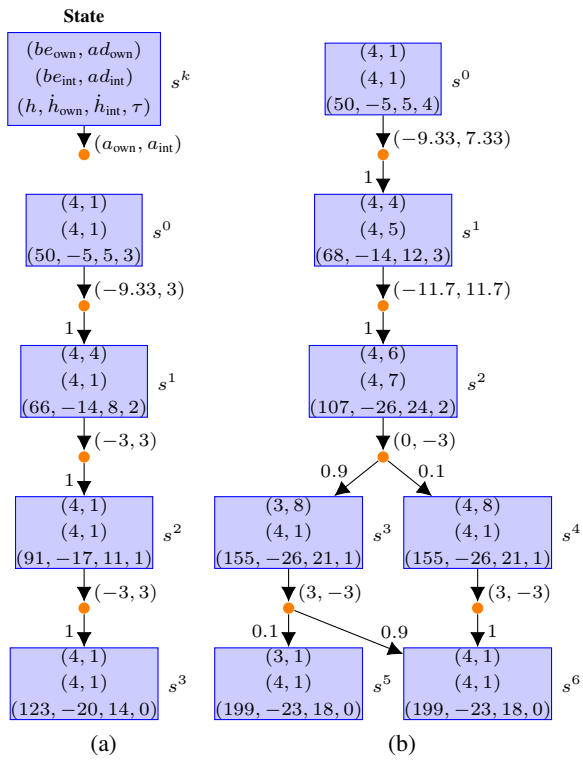


Figure 7: Strategies for the VCAS[2] example: (a) $\epsilon_{\text{own}} = 0$, $\epsilon_{\text{int}} = 0$ and $\tau = 3$; (b) $\epsilon_{\text{own}} = 0.1$, $\epsilon_{\text{int}} = 0$ and $\tau = 4$.