

Collision Avoidance Verification of Multiagent Systems with Learned Policies

Zihao Dong¹, Shayegan Omidshafiei², Michael Everett¹

Abstract—For many multiagent control problems, neural networks (NNs) have enabled promising new capabilities. However, many of these systems lack formal guarantees (e.g., collision avoidance, robustness), which prevents leveraging these advances in safety-critical settings. While there is recent work on formal verification of NN-controlled systems, most existing techniques cannot handle scenarios with more than one agent. To address this research gap, this paper presents a backward reachability-based approach for verifying the collision avoidance properties of Multi-Agent Neural Feedback Loops (MA-NFLs). Given the dynamics models and trained control policies of each agent, the proposed algorithm computes *relative backprojection sets* by (simultaneously) solving a series of Mixed Integer Linear Programs (MILPs) offline for each pair of agents. We account for state measurement uncertainties, making it well aligned with real-world scenarios. Using those results, the agents can quickly check for collision avoidance online by solving low-dimensional Linear Programs (LPs). We demonstrate the proposed algorithm can verify collision-free properties of a MA-NFL with agents trained to imitate a collision avoidance algorithm (Reciprocal Velocity Obstacles). We further demonstrate the computational scalability of the approach on systems with up to 10 agents.

I. INTRODUCTION

Verifying the properties of multiagent systems has been an important research area for several decades [1]–[3]. Meanwhile, neural network (NN) control policies are becoming a key component of many state-of-the-art multiagent systems, such as for swarming [4] and autonomous driving [5], yet the above verification algorithms cannot deal with these NNs. Obtaining formal guarantees (e.g., for collision avoidance, robustness) for closed-loop systems with NN controllers, i.e., neural feedback loops (NFLs), remains challenging primarily due to the high dimensionality and nonlinearities of NNs. Recent literature aims to provide these formal guarantees, typically by formulating reachability analysis problems and using convex relaxations to obtain tractable verification algorithms [6]–[11]. However, existing work focuses primarily on verification of scenarios with a single agent.

As shown in Fig. 1, extending these ideas to multiagent systems raises additional challenges, both in analyzing the complex multiagent interactions and in handling the increase in dimensionality. Ref. [12], [13] considers the verification of a two agent system with neuro-symbolic agents. Ref. [14] is the first approach for formally verifying multiagent NFLs (MA-NFLs). In particular, [14] extends Reach-SDP [8] to compute forward reachable sets over a single representation that contains all agents’ controllers and dynamics.

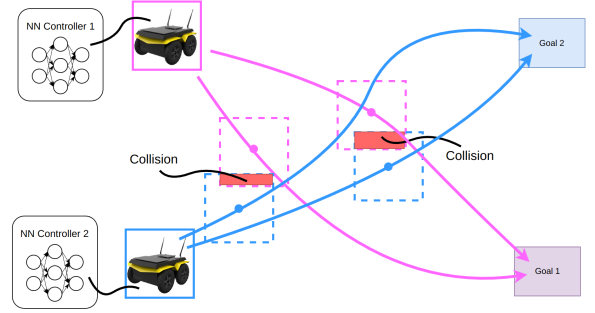


Fig. 1: Complex interactions between agents present challenges in formal safety verification. This analysis is further complicated when the agents are using NN control policies.

While this approach provides meaningful bounds for several multiagent systems, our experiments suggest that systems with agents trained to avoid collisions are not well-suited for forward reachability-based techniques, especially when the given controllers are safe. In Fig. 2, one agent uses the vector field policy from [10], and the other agent is static at the origin (i.e., using a policy that always commands zero velocity). Since the (convex) forward reachable sets (computed by merging both controllers as in [14], then solving mixed integer linear programs (MILPs)) contain both ways the agent could avoid the static agent, the reachable sets intersect with the obstacle, meaning the algorithm is unable to verify collision avoidance. Meanwhile, backward reachability analysis algorithms, such as BReach-LP [10] (which can handle this scenario as a single-agent problem), prove the agents will not collide because although multiple disjoint paths might lead to the target set, these paths must remain within the target set. However, BReach-LP (and its recent extensions) do not handle more general multiagent systems (e.g., with multiple non-static agents).

Therefore, this paper aims to extend backward reachability analysis to MA-NFLs. Backward reachability-based algorithms typically compute backprojection sets of the avoid sets [10]. However, computing backprojection sets in the global frame is ineffective in the context of collision avoidance for MA-NFLs. Unlike in single agent setting where the avoid set is static, our avoid set is a moving region controlled by another neural network. Computing its backprojection set in the global frame will be thus equivalent to computing the backprojection set of the whole state space. In this work, we propose **Relative Backward Reachability (ReBAR)** and **ReBAR-MA**, computing *relative backprojection sets* in a relative coordinate frame using MILPs. This approach

¹Northeastern University, Boston, MA, USA. e-mail: {dong.zih, m.everett}@northeastern.edu. ²Work done while at Google. Code: <https://github.com/neu-autonomy/ReBAR>

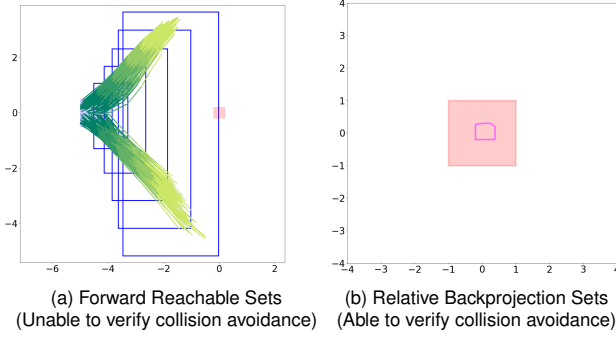


Fig. 2: (a): Forward reachable sets of agent 1 (blue) intersect with agent 2 (red). Unable to verify system safety despite no simulated rollouts (green lines) enters agent 2. (b): Result of ReBAR (magenta) is within agent 2, system is verified safe.

enables successful verification of the scenario in Fig. 2. In particular, Fig. 2(b) shows that the relative backprojection set computed by ReBAR (magenta) lies within the collision set (red), meaning the agents can only collide if they start in a collision. Furthermore, for systems that cannot be verified safe offline, we propose an efficient online safety check using LPs. In summary, the contributions of this work include:

- ReBAR: the first algorithm for formally verifying the collision avoidance properties of MA-NFLs, by computing relative backprojection sets with state uncertainty offline using MILPs and performing online safety checks online using low-dimensional LPs,
- ReBAR-MA: an extension of ReBAR to handle scenarios with arbitrary numbers of agents by performing parallelizable pair-wise verification,
- an extension of them to provide safety guarantee over multiple timesteps in an iterative manner, and
- demonstrations of ReBAR and ReBAR-MA on relevant multiagent systems, including a 2-agent NFL trained to imitate the Reciprocal Velocity Obstacle (RVO) algorithm [15] and systems with up to 10 agents.

II. PRELIMINARIES

A. Multiagent System Dynamics

For a system with n agents, let the state of agent $i \in [n]$, where $[n]$ denotes the set $\{1, \dots, n\}$, be $\mathbf{x}_t^{(i)} \in \mathbb{R}^{n_x}$ and the joint system state, $\mathbf{X}_t \in \mathcal{X} \subseteq \mathbb{R}^{n \times n_x}$, be

$$\mathbf{X}_t = [\mathbf{x}_t^{(1)\top}, \mathbf{x}_t^{(2)\top}, \dots, \mathbf{x}_t^{(n)\top}]^\top. \quad (1)$$

Throughout the paper, superscripts (i) denote correspondence to agent i (not derivatives). Each agent's dynamics can be described by the discrete-time, linear, time-invariant system,

$$\begin{aligned} \mathbf{x}_{t+1}^{(i)} &= \mathbf{A}^{(i)} \mathbf{x}_t^{(i)} + \mathbf{B}^{(i)} \mathbf{u}_t^{(i)} + \mathbf{c}^{(i)} \\ \mathbf{Y}_t &= \mathbf{C} \mathbf{X}_t + \mathcal{E} \end{aligned} \quad (2)$$

where $\mathbf{A}^{(i)} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{B}^{(i)} \in \mathbb{R}^{n_x \times n_u}$, $\mathbf{C} \in \mathbb{R}^{n_x}$ are known system matrices, $\mathbf{c}^{(i)}$ is a known exogenous input, and $\mathcal{E} = [\epsilon^{(0)\top}, \dots, \epsilon^{(n)\top}]^\top \in \mathbb{R}^{n \times n_x}$ is measurement uncertainty, resulting in uncertain state measurement $\mathbf{Y}_t \in \mathcal{Y} \subseteq \mathbb{R}^{n \times n_x}$,

$$\mathbf{Y}_t = [\mathbf{y}_t^{(1)\top}, \mathbf{y}_t^{(2)\top}, \dots, \mathbf{y}_t^{(n)\top}]^\top. \quad (3)$$

We assume uncertain state measurement is within a L_∞ ball centered at $\mathbf{x}_t^{(i)}$ with radius $\epsilon^{(i)}$, i.e., $\mathbf{y}_t^{(i)} \in \mathcal{B}_\infty(\mathbf{x}_t^{(i)}, \epsilon^{(i)})$. Given control limits $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ and state feedback policy $\pi^{(i)} : \mathcal{X} \rightarrow \mathcal{U}$, the control input for each agent $\mathbf{u}_t^{(i)} = \pi^{(i)}(\mathbf{Y}_t)$. The closed-loop dynamics for a single agent is denoted as:

$$\mathbf{x}_{t+1}^{(i)} = f^{(i)}(\mathbf{X}_t; \mathcal{U}, \mathbf{A}^{(i)}, \mathbf{B}^{(i)}, \mathbf{C}, \mathbf{c}^{(i)}, \pi^{(i)}, \epsilon^{(i)}) \quad (4)$$

Let $\mathbf{X}^{(i,j)} = [\mathbf{x}^{(i)\top}, \mathbf{x}^{(j)\top}]^\top$, the closed-loop dynamics for agents i and j is denoted

$$\mathbf{X}_{t+1}^{(i,j)} = f^{(i,j)}(\mathbf{X}_t; f^{(i)}, f^{(j)}). \quad (5)$$

B. Neural Network Controller Architecture

This work assumes the control policies are feedforward NNs with L hidden layers. Each layer $\ell \in [L+1]$ has n_ℓ neurons, weight $\mathbf{W}^{(i),\ell} \in \mathbb{R}^{n_{\ell+1} \times n_\ell}$ and bias $\mathbf{b}^{(i),\ell} \in \mathbb{R}^{n_{\ell+1}}$. The piecewise linear nonlinearity (e.g. ReLU) is denoted $\sigma^{(i),\ell}$. The control of agent i is computed as:

$$\begin{aligned} \mathbf{x}^{(i),0} &= \mathbf{Y}_t \\ \mathbf{z}^{(i),\ell} &= \mathbf{W}^{(i),\ell} \mathbf{x}^{(i),\ell} + \mathbf{b}^{(i),\ell}, \forall \ell \in [L] \\ \mathbf{x}^{(i),\ell+1} &= \sigma^{(i),\ell}(\mathbf{z}^{(i),\ell}), \forall \ell \in [L-1] \\ \pi^{(i)}(\mathbf{Y}_t) &= \mathbf{z}^{(i),L}. \end{aligned} \quad (6)$$

Control limit is incorporated as in [14]. Accounting for general activations (tanh) or other architectures (convolution) using convex relaxations is left for future work [11].

C. Collision Sets

For two agents $i, j \in [n]$, $i \neq j$, assume their positions $(x^{(i)}, y^{(i)})$, $(x^{(j)}, y^{(j)})$ are elements of state vector \mathbf{X} . The collision set for agents i, j is defined as a convex polytope,

$$\mathcal{C}^{(i,j)} \triangleq \{\mathbf{X} \mid \mathbf{H} \mathbf{X}^{(i,j)} \leq \mathbf{d}\}, \quad (7)$$

where $\mathbf{H} \in \mathbb{R}^{n_f \times 2n_x}$ and $\mathbf{d} \in \mathbb{R}^{n_f}$, and n_f denotes the number of facets bounding this convex polytope.

D. Backreachable & Backprojection Set

We extend the backprojection set definition in [10] to multiagent systems with state uncertainty, to handle avoid sets parameterized by coordinates of moving agents (instead of static obstacles). For agents $i, j \in [n]$, the backreachable set contains all states such that $\exists \mathbf{u}_t^{(i)}, \mathbf{u}_t^{(j)} \in \mathcal{U}$ that drive the system into collision set on the next timestep.

$$\mathcal{R}_{-1}(\mathcal{C}^{(i,j)}) \triangleq \left\{ \mathbf{X}_t \left| \begin{array}{l} \mathbf{x}_{t+1}^{(i)} = \mathbf{A}^{(i)} \mathbf{x}_t^{(i)} + \mathbf{B}^{(i)} \mathbf{u}_t^{(i)} + \mathbf{c}^{(i)} \\ \mathbf{x}_{t+1}^{(j)} = \mathbf{A}^{(j)} \mathbf{x}_t^{(j)} + \mathbf{B}^{(j)} \mathbf{u}_t^{(j)} + \mathbf{c}^{(j)} \\ \mathbf{u}_t^{(i)}, \mathbf{u}_t^{(j)} \in \mathcal{U} \\ \mathbf{X}_{t+1}^{(i,j)} \in \mathcal{C}^{(i,j)} \end{array} \right. \right\}. \quad (8)$$

And similarly, the backprojection set contains all states such that the policies $\pi^{(i)}, \pi^{(j)}$ will drive the system into the collision set on the next timestep,

$$\mathcal{P}_{-1}(\mathcal{C}^{(i,j)}) \triangleq \left\{ \mathbf{X}_t \left| \begin{array}{l} \mathbf{X}_{t+1}^{(i,j)} = f^{(i,j)}(\mathbf{X}_t; f^{(i)}, f^{(j)}) \\ \mathbf{X}_{t+1}^{(i,j)} \in \mathcal{C}^{(i,j)} \end{array} \right. \right\}. \quad (9)$$

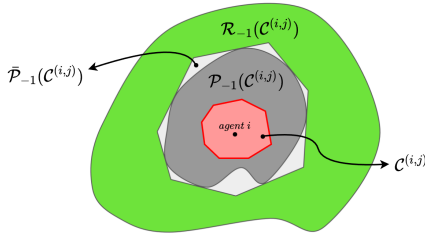


Fig. 3: $\mathcal{C}^{(i,j)}$ (red) is a convex polytope around agent i ; $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)})$ (light grey) is the tightest convex polytope of $\mathcal{P}_{-1}(\mathcal{C}^{(i,j)})$ (grey) using the given facets; $\mathcal{R}_{-1}(\mathcal{C}^{(i,j)})$ (green) contains all states that $\exists u \in \mathcal{U}$ s.t. agent i, j collides

III. APPROACH

A. Relative Backreachable & Backprojection Sets

To compute safety certificates offline and use them online for safety verification, we compute backprojection sets of the collision set using the agents' relative coordinates. Let $\mathbf{p}_t^{(i)} = (x_t^{(i)}, y_t^{(i)})$ denote the position of agent i at timestep t , and let $\mathbf{p}_t^{(j \rightarrow i)} = (x_t^{(j)} - x_t^{(i)}, y_t^{(j)} - y_t^{(i)})$ denote the relative position of agent j w.r.t. agent i . The collision set can be expressed using the relative coordinates as:

$$\mathcal{C}^{(i,j)} \triangleq \{\mathbf{p}^{(j \rightarrow i)} \mid \mathbf{H}\mathbf{X}^{(i,j)} \leq \mathbf{d}\}. \quad (10)$$

Eq. (10) is the failure set we consider in this work. The *relative backreachable & backprojection set* are defined as:

$$\mathcal{R}_{-1}(\mathcal{C}^{(i,j)}) \triangleq \left\{ \mathbf{p}_t^{(j \rightarrow i)} \left| \begin{array}{l} \mathbf{x}_{t+1}^{(i)} = \mathbf{A}^{(i)}\mathbf{x}_t^{(i)} + \mathbf{B}^{(i)}\mathbf{u}_t^{(i)} + \mathbf{c}^{(i)} \\ \mathbf{x}_{t+1}^{(j)} = \mathbf{A}^{(j)}\mathbf{x}_t^{(j)} + \mathbf{B}^{(j)}\mathbf{u}_t^{(j)} + \mathbf{c}^{(j)} \\ \mathbf{u}_t^{(i)}, \mathbf{u}_t^{(j)} \in \mathcal{U} \\ \mathbf{p}_{t+1}^{(j \rightarrow i)} \in \mathcal{C}^{(i,j)} \end{array} \right. \right\} \quad (11)$$

$$\mathcal{P}_{-1}(\mathcal{C}^{(i,j)}) \triangleq \left\{ \mathbf{p}_t^{(j \rightarrow i)} \left| \begin{array}{l} \mathbf{X}_{t+1}^{(i,j)} = f^{(i,j)}(\mathbf{X}_t; f^{(i)}, f^{(j)}) \\ \mathbf{p}_{t+1}^{(j \rightarrow i)} \in \mathcal{C}^{(i,j)} \end{array} \right. \right\} \quad (12)$$

Fig. 3 illustrates the relationship between collision set, relative backprojection set, relative backprojection set over approximation (RBPOA), and relative backreachable set. Unlike prior work [10], these sets are defined relative to one agent to be more suitable for analyzing collision avoidance of MA-NFLs.

B. Two Agent Verification

Because the relative backprojection sets can be non-convex and involve potentially high-dimensional NNs, computing these sets exactly is computationally intractable. Instead, we will compute a convex over approximation of these sets using finitely many facets $\mathbf{a} \in \mathbb{R}^2$. We begin by considering a system with two agents. For each facet, the following optimization problem finds a half-space containing the relative backprojection set:

$$\begin{aligned} \min_{\mathbf{x}_t} \quad & \mathbf{a}^\top \mathbf{p}_t^{(j \rightarrow i)} \\ \text{s.t.} \quad & \mathbf{X}_t \in \mathcal{X} \\ & \mathbf{X}_{t+1}^{(i,j)} = f^{(i,j)}(\mathbf{X}_t; f^{(i)}, f^{(j)}) \\ & \mathbf{p}_{t+1}^{(j \rightarrow i)} \in \mathcal{C}^{(i,j)}, \end{aligned} \quad (13)$$

and the intersection of the resulting half-spaces is a convex over-approximation of the relative backprojection set. Because we have assumed the neural network controllers have piecewise linear activation functions, the optimization problem Eq. (13) is a MILP, with the same number of binary variables as the number of (uncertain) ReLU neurons in the neural network controller. The solutions to the MILPs provide the tightest convex over approximation using the given facets (subjected to numerical tolerance). In Eq. (13), t is an arbitrary timestep, meaning that the resulting convex polytope $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)})$ is an over approximation of the relative backprojection set at any arbitrary timestep $t \geq 0$. By utilizing the relative coordinate frame, we make implicit where the collision occurs globally, allowing to move the expensive RBPOA computation offline. Algorithm 1 summarizes the approach for verifying a 2-agent NFL. Computing the RBPOA requires solving n_f MILPs in total.

Algorithm 1 ReBAR

Input: collision set $\mathcal{C}^{(i,j)}$, closed-loop dynamics function $f^{(i,j)}$, number of facets n_f
Output: RBPOA $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)})$
1: $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \leftarrow \mathbb{R}^2$
2: **for** $\mathbf{a} \in \text{getFacets}(n_f)$ **do**
3: $b \leftarrow \text{solveForFacet}(\mathbf{a}, f^{(i,j)})$ (Eq. (13))
4: $\mathcal{A} \leftarrow \{\mathbf{p}^{(j \rightarrow i)} \mid \mathbf{a}^\top \mathbf{p}^{(j \rightarrow i)} \geq b\}$
5: $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \leftarrow \bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \cap \mathcal{A}$
6: **end for**
7: **return** $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)})$

Similar to [10], the 2-agent system is verified safe, i.e. the agents will not collide at any future timestep if they do not collide at $t = 0$, if the resulting RBPOA is a subset of the collision set, as formalized next.

Definition III.1 (Verified Safe). *A system with 2 agents i, j is verified safe w.r.t. $\mathcal{C}^{(i,j)}$ if $\mathbf{p}_t^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)} \implies \forall \tau \geq 0, \mathbf{p}_{t+\tau}^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)}$.*

Lemma III.1. *A system with 2 agents i, j with closed-loop dynamics function $f^{(i,j)}$ and collision set $\mathcal{C}^{(i,j)}$ is verified safe if $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \subseteq \mathcal{C}^{(i,j)}$.*

Proof. Consider a 2-agent system satisfying $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \subseteq \mathcal{C}^{(i,j)}$. Let $\mathbf{X}_0 \in \mathcal{X}, \mathbf{p}_0^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)}$ be the starting state of the system, then $\mathbf{p}_0^{(j \rightarrow i)} \notin \bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)})$, and thus $\mathbf{p}_1^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)}$. At timestep $k \geq 0$, assume $\mathbf{X}_k \in \mathcal{X}$ and $\mathbf{p}_k^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)}$, thus $\mathbf{p}_k^{(j \rightarrow i)} \notin \bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)})$, and $\mathbf{p}_{k+1}^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)}$. By induction, this analysis can be extended to any arbitrary timestep $t \geq 0$, hence a two agent system is safe (from collision) if $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \subseteq \mathcal{C}^{(i,j)}$. \square

When a 2-agent NFL is verified safe by Lemma III.1, the agents will not collide as long as they start from a non-colliding state, even if they have noisy state measurement. However, a system that is not verified safe by Lemma III.1 may still provide collision-free behavior in a subset of \mathcal{X} .

In practice, if only a noisy state estimate is available (e.g., uncertainty bounds over \mathbf{x}_t), it is not immediately obvious whether $\mathbf{p}^{(j \rightarrow i)}$ lies in the RBPOA. Let the convex polytope $\mathbf{A}_t^{(i,j)} \mathbf{p}^{(j \rightarrow i)} \leq \mathbf{b}_t^{(i,j)}$ denote the uncertain state, we solve the following LP:

$$\begin{aligned} \min \quad & 0 \\ \text{s.t.} \quad & \mathbf{X}_t \in \mathcal{X}, \\ & \mathbf{A}_t^{(i,j)} \mathbf{p}^{(j \rightarrow i)} \leq \mathbf{b}_t^{(i,j)} \\ & \mathbf{p}^{(j \rightarrow i)} \in \bar{\mathcal{P}}_{-1}(\mathcal{C}_t^{(i,j)}) \end{aligned} \quad (14)$$

If Eq. (14) is infeasible, then, even with measurement uncertainty, the system is guaranteed collision-free on the next timestep because ReBAR computes an over approximation of the true unsafe region. Note Eq. (14) is a LP that does not involve NN controllers, as thus can be solved efficiently, enabling our method to provide a safety guarantee online.

ReBAR can be extended to provide safety guarantees over an extended time horizon τ by computing RBPOAs at multiple timesteps $\bar{\mathcal{P}}_{-\tau,0}(\mathcal{C}^{(i,j)})$. We initialize the zeroth RBPOA as the collision set and step backward in time to compute RBPOA recursively using Algorithm 1 and the previous timestep RBPOA as collision set. Computing $\bar{\mathcal{P}}_{-k}(\mathcal{C}^{(i,j)})$ using Algorithm 1 only involves a change of variable (replacing $\mathcal{C}^{(i,j)}$ with $\bar{\mathcal{P}}_{-k+1}(\mathcal{C}^{(i,j)})$), so every step backward in time incurs similar time complexity.

C. Scaling to More Agents

For systems with more than 2 agents, considering all agents in one optimization problem (as in [14]) becomes computationally intractable because the dimensionality of the problem will be prohibitive for verification as the number of agents increases. Instead, we propose to split the task into $\mathcal{O}(n^2)$ pair-wise verification sub-problems, each of which is the same as Eq. (13) and can be solved using Algorithm 1 in similar runtime as the two agent verification problem. Furthermore, the sub-problems only involve controllers of agent i and j , and thus can be solved simultaneously across available resources, allowing our approach to scale to systems with larger numbers of agents. The approach is summarized in Algorithm 2. Note this algorithm can be extended to n -step case similar as Algorithm 1. When every pair of agents is verified safe by Lemma III.1, then the system is verified safe, i.e., no agents will collide if they start from non-colliding states, as we formalize next.

Definition III.2 (Multi-Agent Verified Safe). *A system with n agents is verified safe if $\forall i, j \in [n], i \neq j, \mathbf{p}_t^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)} \implies \forall \tau \geq 0, \mathbf{p}_{t+\tau}^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)}$*

Lemma III.2. *For a system with n agents, closed-loop dynamics functions $f^{(0,1)}, f^{(0,2)}, \dots, f^{(n-1,n)}$, and collision sets $\mathcal{C}^{(0,1)}, \mathcal{C}^{(0,2)}, \dots, \mathcal{C}^{(n-1,n)}$. If $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \subseteq \mathcal{C}^{(i,j)} \forall i, j \in [n]$, the multi-agent system is verified safe.*

Proof. For a n -agent system that is pair-wise verified safe by Lemma III.1, let $\mathbf{X}_0 \in \mathcal{X}, \mathbf{p}_0^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)} \forall i, j \in [n], i \neq j$, be the starting state. Assume $\forall i, j \in [n], i \neq j$, at

timestep $k \geq 0, \mathbf{X}_k \in \mathcal{X}$ and $\mathbf{p}_k^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)}$. Because $\forall i, j \in [n], i \neq j$, the relative backprojection set over approximation between agent i, j is a subset of their collision set, i.e. $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \subseteq \mathcal{C}^{(i,j)}$, we must have $\forall i, j \in [n], i \neq j, \mathbf{p}_k^{(j \rightarrow i)} \notin \bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)})$, and $\mathbf{p}_{k+1}^{(j \rightarrow i)} \notin \mathcal{C}^{(i,j)}$, i.e., no collision at timestep $k+1$. Proof completed by induction. \square

Algorithm 2 ReBAR-MA

Input: collision sets $\mathcal{C}^{(1,2)}, \dots, \mathcal{C}^{(n-1,n)}$
closed-loop dynamic functions $f^{(1,2)}, \dots, f^{(n-1,n)}$
number of facets n_f
Output: RBPOAs $\{\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(1,2)}), \dots, \bar{\mathcal{P}}_{-1}(\mathcal{C}^{(n-1,n)})\}$
1: $\bar{\mathcal{P}} = \{\}$
 # In Parallel
2: **for** $i, j \in [n]$ **do**
3: $\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}) \leftarrow \text{ReBAR}(\cdot)$
4: $\bar{\mathcal{P}}.append(\bar{\mathcal{P}}_{-1}(\mathcal{C}^{(i,j)}))$
5: **end for**
6: **return** $\bar{\mathcal{P}}$

In case we cannot verify the safety of the system, we can use the RBPOAs online to verify safety of states similar as the two agent case. If Eq. (14) is infeasible for all pairs of agents, then the state of the system is safe and the proof is similar to our proof of Lemma III.2.

IV. EXPERIMENT

All results are obtained using the optimization solver Gurobi [16] on a PC running Ubuntu 22.04 with a Intel i9-13900K CPU. ReBAR verifies a pair of agents mimicing RVO [15] offline in 11s and run safety check online in 1.4ms on average. We computed RBPOAs up to 5 steps, and visualize them with the RBPUAs we generated by exhaustive sampling. For systems with up to 10 agents trained to mimic the vector field in [10] with [20,20] hidden neurons, ReBAR-MA verifies each pair of agents in 200s to 320s, and the online check takes less than 1.6ms for each pair of agents.

A. Two Agent Verification

In this section, we demonstrate ReBAR by verifying multi-agent collision avoidance NN controllers trained to mimic the RVO algorithm [15]. We collect 1M state-action pairs for agent positions randomly sampled from $[0, 0] \times [10, 10]$, with agent goals (9.0, 5.0) and (5.0, 9.0), respectively. The state vector of every agent contains its world frame coordinate $x^{(i)}, y^{(i)}$, and velocity $v_{x^{(i)}}, v_{y^{(i)}}$. The agents' radius is set to 0.5, and thus safety radius $r = 1.0$. The input to each agent's NN control policy is the joint state of the agents, and each agent has its own, potentially different, policy. The collision set is represented as the L_∞ norm ball, i.e. $\mathcal{C}^{(1,2)} = \mathcal{B}_\infty(0, 1)$. With $\mathcal{U} = [-1, -1] \times [1, 1]$, we use a single integrator dynamics model,

$$\mathbf{x}_{t+1}^{(i)} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_2 \end{bmatrix} \mathbf{x}_t^{(i)} + \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{I}_2 \end{bmatrix} \mathbf{u}_t^{(i)}. \quad (15)$$

The MSE loss (between NN output and RVO output) of training converged to 0.001 after 20 epochs, indicating

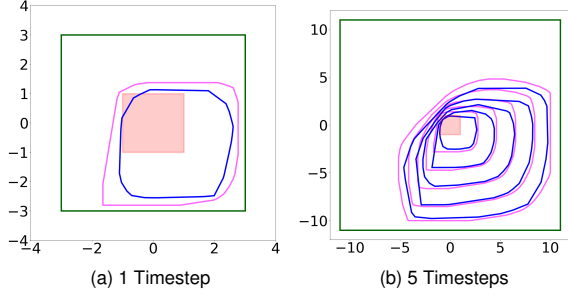


Fig. 4: (a): RBPOA (magenta), Collision Set (red), Relative Backreachable Set (darkgreen), and RBPUA (blue) obtained by monte-carlo simulation are also shown. (b) Up to 5 step RBPOA, confirmed by sampled RBPUAs.

the NN policy encodes the RVO algorithm reasonably well. However, ReBAR is able to identify unsafe regions in this multiagent system, despite the low testing loss. For example, the RBPOA generated by Algorithm 1, with measurement uncertainty $\epsilon^{(1)} = \epsilon^{(2)} = [0.5, 0.5, 0, 0]$, is shown in Fig. 4. The RBPOA shows the two agent system is unsafe, in particular when agent 2 is to the bottom right of agent 1, which matches our intuition because the goal of agent 2 is on the top left of the goal of agent 1. The RBPOA we computed encloses the ground truth relative backprojection set under approximation (RBPUA) from 2M Monte Carlo simulations, successfully identifying unsafe cases that exhaustive sampling fails to capture and significantly shrinking the relative backreachable set, making it a good reference to be used online for safety verification. Fig. 4(b) shows RBPOAs for 5 timesteps (magenta) and their respective RBPUAs (blue).

Even though this system could not be verified safe everywhere ahead of time (i.e., the RBPOA is not inside the collision set), the system could still be safe in many parts of the state space. Therefore, we can still provide an online safety check using the RBPOA that was just computed. In particular, for a given set of initial states (e.g., the current pose estimates of each agent with some uncertainty bounds), we can check whether a collision may be possible collision at the next timestep. For example, if the two agents' true coordinates are only known to be within the hyper-rectangle $[4, 4.5, 7.5, 3.5] \times [6, 5.5, 9, 4.5]$, ReBAR correctly alerts that a collision could occur in the next timestep. In Fig. 5(a), the unsafe region from Monte Carlo simulation is shown in blue, and is enclosed by the unsafe region (red) generated by solving Eq. (14) with coordinates of the agents as the objective function, demonstrating the desired ability of ReBAR to identify unsafe behaviors of a faulty controller. Alternatively, in Fig. 5(b), where the initial states are known to be within $[3, 3, 0.5, 4] \times [4, 4, 1.5, 5]$, Eq. (14) is infeasible, and therefore the online check returns that there is no collision possible on the next timestep, which is empirically confirmed by Monte Carlo simulations. Computing the RBPOAs takes 11.2 ± 0.16 s on average (of 10 experiments), and the online LP takes 1.4 ± 0.40 ms.

We further provide experiment on a double integrator

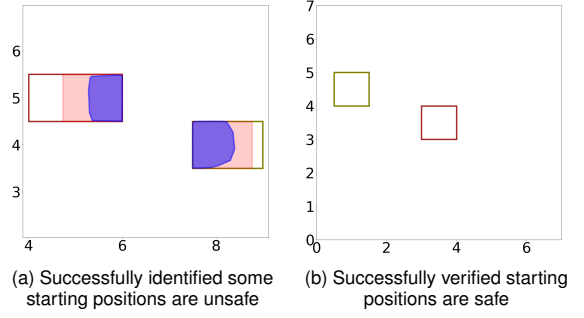


Fig. 5: agent 1 (brown) and agent 2 (olive) state with uncertainty. (a): Unsafe set over approximation (red) encloses unsafe set under approximation (blue). (b): state safe by online safety check, all simulations do not result in collision.

model. The setting remains the same as the previous experiment, except the neural network controller learns to predict the acceleration that matches the agents' velocity with the optimal velocity produced by RVO. The new control signal vector is $[a_{x^{(i)}}, a_{y^{(i)}}] \in [-2, -2] \times [2, 2]$. The double integrator dynamic model we consider is given in Eq. (16). Verifying this system takes 9.7 ± 0.39 s, demonstrating our approach's capability of handling more complex dynamics.

$$\mathbf{x}_{t+1}^{(i)} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{I}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0} & \mathbf{I}_2 & \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix} \mathbf{x}_t^{(i)} + \begin{bmatrix} 0.5\mathbf{I}_2 \\ \mathbf{I}_2 \end{bmatrix} \mathbf{u}_t^{(i)}. \quad (16)$$

B. Runtime and Scalability

# agents	RBPOA [s/pair]	online [ms/pair]	# agents	RBPOA [s/pair]	online [ms/pair]
2	201.50	1.49	7	260.43	1.38
3	239.46	1.39	8	251.47	1.52
4	217.77	1.40	9	300.60	1.35
5	201.97	1.31	10	321.68	1.60
6	265.23	1.59			

TABLE I: Pair-wise untimed computing RBPOA/online safety check of systems with up to 10 agents controlled by vector field policy with [20, 20] hidden neurons.

To demonstrate the scalability of ReBAR-MA, we verify systems with up to 10 agents and report the pair-wise offline and online runtime in Table I. To demonstrate our method's ability to handle larger and more complex NN controllers, we trained the agents with [20,20] hidden neurons, that mimics the vector field policy in [10]. Note we do not train the agents to avoid each other. We set $\epsilon = 0.5$ for all experiments. From the table we can see that pair-wise verification runtime does not increase dramatically w.r.t. number of agents and fluctuates between 201.50s to 321.68s, and the online safety check takes 1.31ms to 1.60ms per pair on average. As our approach is parallelizable, we are able to verify a system up to 10 agent in minutes in the ideal case where we compute all RBPOAs in parallel. In the worst case, if we compute all RBPOAs and online safety check sequentially, the total verification runtime will be $\mathcal{O}(n^2)$ the respective average pair-wise runtime. The resulting RBPOA and online safety check for the 10-agent system is shown in Fig. 6. For a

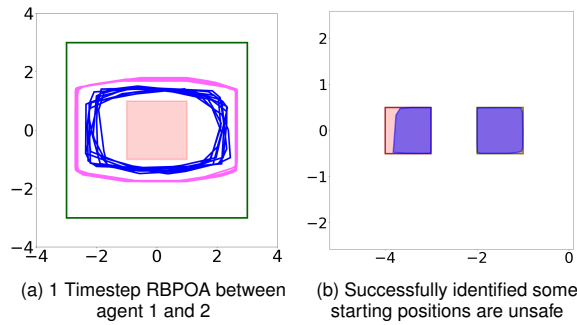


Fig. 6: 10 vector field agent system. (a): Overlaid RBPOA of agent 1 and 2-10. (b): unsafe region (red) marked by online safety check identifies sampled unsafe region (blue)

10-agent system, sampling-based methods are inefficient for computing the relative backprojection set, due to the high dimensionality of the input joint state vector. Instead, our RBPOA provides an over approximation to the sampled RBPUA, and identifies unsafe cases online, demonstrating our pair-wise approach’s scalability to increasing number of neural network controlled agents.

V. CONCLUSION

This paper presented ReBAR and ReBAR-MA for verifying the collision avoidance safety of Multi-Agent Neural Feedback Loops (MA-NFLs) by computing the relative backprojection set over approximation with state uncertainty offline, and using the result to provide a real-time safety guarantee. We demonstrate them by verifying agents trained mimicing RVO [15], and system with up to 10 agents. Future work will consider algorithmic extensions to handle general activation functions and dynamics through linear relaxation and verification in decentralized observation space.

REFERENCES

- [1] L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi, “Probabilistic verification of a decentralized policy for conflict resolution in multi-agent systems,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006, pp. 2448–2453.
- [2] T. T. Doan, Y. Yao, N. Alechina, and B. Logan, “Verifying heterogeneous multi-agent programs,” in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 2014, pp. 149–156.
- [3] P. Kouvaros, A. Lomuscio, E. Pirovano, and H. Punchihewa, “Formal verification of open multi-agent systems,” in *Proceedings of the 18th international conference on autonomous agents and multiagent systems*, 2019, pp. 179–187.
- [4] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, “Learning decentralized controllers for robot swarms with graph neural networks,” in *Conference on robot learning*, PMLR, 2020, pp. 671–682.
- [5] P. Palanisamy, “Multi-agent connected autonomous driving using deep reinforcement learning,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020, pp. 1–7.
- [6] S. Dutta, X. Chen, and S. Sankaranarayanan, “Reachability analysis for neural feedback systems using regressive polynomial rule inference,” in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 157–168.
- [7] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, “Reachnn: Reachability analysis of neural-network controlled systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–22, 2019.
- [8] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas, “Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming,” in *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 5929–5934.
- [9] M. Everett, G. Habibi, C. Sun, and J. P. How, “Reachability analysis of neural feedback loops,” *IEEE Access*, vol. 9, pp. 163 938–163 953, 2021.
- [10] N. Rober, M. Everett, and J. P. How, “Backward reachability analysis for neural feedback loops,” in *2022 IEEE 61st Conference on Decision and Control (CDC)*, IEEE, 2022, pp. 2897–2904.
- [11] K. Xu, Z. Shi, H. Zhang, *et al.*, “Automatic perturbation analysis for scalable certified robustness and beyond,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1129–1141, 2020.
- [12] R. Yan, G. Santos, G. Norman, D. Parker, and M. Kwiatkowska, “Strategy synthesis for zero-sum neuro-symbolic concurrent stochastic games,” *arXiv preprint arXiv:2202.06255*, 2022.
- [13] R. Yan, G. Santos, G. Norman, D. Parker, and M. Kwiatkowska, “Partially observable stochastic games with neural perception mechanisms,” *arXiv preprint arXiv:2310.11566*, 2023.
- [14] O. Gates, M. Newton, and K. Gatsis, “Scalable forward reachability analysis of multi-agent systems with neural network controllers,” in *2023 62nd IEEE Conference on Decision and Control (CDC)*, IEEE, 2023, pp. 67–72.
- [15] J. Van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *2008 IEEE international conference on robotics and automation*, Ieee, 2008, pp. 1928–1935.
- [16] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2023. [Online]. Available: <https://www.gurobi.com>.