Safe Multi-agent Learning via Trapping Regions

Czechowski, Aleksander; Oliehoek, Frans A.

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI 2023

Citation (APA)
Czechowski, A., & Oliehoek, F. A. (2023). Safe Multi-agent Learning via Trapping Regions. In E. Elkind (Ed.), *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI 2023* (pp. 82-90). (IJCAI International Joint Conference on Artificial Intelligence; Vol. 2023-August). International Joint Conferences on Artificial Intelligence (IJCAI).

Safe Multi-agent Learning via Trapping Regions

Aleksander Czechowski , **Frans A. Oliehoek**
Delft University of Technology

Abstract

One of the main challenges of multi-agent learning lies in establishing convergence of the algorithms, as, in general, a collection of individual, self-serving agents is not guaranteed to converge with their joint policy, when learning concurrently. This is in stark contrast to most single-agent environments, and sets a prohibitive barrier for deployment in practical applications, as it induces uncertainty in long term behavior of the system. In this work, we apply the concept of trapping regions, known from qualitative theory of dynamical systems, to create safety sets in the joint strategy space for decentralized learning. We propose a binary partitioning algorithm for verification that candidate sets form trapping regions in systems with known learning dynamics, and a heuristic sampling algorithm for scenarios where learning dynamics are not known. We demonstrate the applications to a regularized version of Dirac Generative Adversarial Network, a four-intersection traffic control scenario run in a state of the art open-source microscopic traffic simulator SUMO, and a mathematical model of economic competition.

1 Introduction

In the recent years, enormous progress has been made for single agent planning and learning algorithms, with agents matching or exceeding human performance in various tasks and games [Mnih *et al.*, 2013; Silver *et al.*, 2016]. The vast success of single agent learning can be partially explained by robustness and strong convergence properties of the underlying algorithms in their basic form, such as Q-learning [Watkins and Dayan, 1992] or policy gradients [Sutton *et al.*, 1999]. Despite wide interest, the same cannot be however said for multi-agent learning. Even most basic models, e.g. replicator learning for normal form games, exhibit nonconvergence, cyclic or even chaotic behavior [Sato *et al.*, 2002]. Even worse, it has been shown that in decoupled learning systems, there can be no learning rule that guarantees convergence to a Nash equilibrium [Hart and Mas-Colell, 2003]. These nonconvergent examples have also been found in more practical learning problems, such as Generative Adversarial Networks [Mescheder *et al.*, 2018]. Due to the above limitations, many successful multi-agent learning methods resorted to using a centralized component, such as centralized critic in actor-critic learning [Lowe *et al.*, 2017] or meta-solvers in double-oracle type algorithms like Parallel Nash Memory [Oliehoek *et al.*, 2006] and Policy Space Response Oracles [Lanctot *et al.*, 2017]. Alternatives in form of decentralized algorithms usually rely on specific assumptions on the reward structure for convergence, for instance fictitious play or exploitability descent in zero-sum games [Brown, 1951; Lockhart *et al.*, 2019].

Despite this undisputed progress in designing convergent multi-agent algorithms, it can be argued that in practical, real-world multi-agent scenarios there will be plenty of situations, where convergence cannot be enforced. One can easily envisage a situation, e.g. in automated traffic control and driving scenarios, or in automated trading, where multiple entities (be it traffic lights, vehicles, or brokers) follow own learning protocols for individual reward maximization. Such learning rules, even though designed to be convergent in static, single-agent environments, would invariably interfere with one another in a multi-agent setting, sometimes resulting in cyclic, or divergent outcomes. The lack of convergence guarantees in such general settings forms a major obstacle for introduction of online learning systems in practical applications, as it introduces a lot of uncertainty over what will be the state of the system, if learning is left unsupervised. Can we nevertheless still establish a type of safety certificates, that would allow us to conclude that simultaneous learning will not spin out of control?

In this paper, we suggest a novel approach to address issue. We start from the realization, that convergence is often not absolutely necessary for reliability. From systems designers perspective, it is often enough to know that learning has *rough* stability guarantees – that is, that agents will not leave a predetermined region of the strategy space during learning.

Due to space restrictions, the proofs of our theorems and lemmas have been postponed to the Supplementary Material, and can be accessed in the extended version of the paper [Czechowski and Oliehoek, 2023]. There, we also provide an additional example, where we find trapping regions in a model of economic competition, which ensures that none of the competing companies will reduce their production to zero.

For a conceptual application, let us consider a traffic light control network as in Figure 3, where individual traffic light controllers learn the best balance of green time between the phases to minimize the waiting time for approaching vehicles. It is not absolutely necessary that by learning each intersection reaches a final, static setting, but would be essential that at all times it gives minimal green time of at least few seconds to each phase, to make sure that no vehicle gets stuck indefinitely on a red traffic light, and also serve the lingering pedestrian flows.

We propose a method of *a priori* verifying these constraints, by establishing *trapping regions*; regions of strategy space, which learning trajectories will never escape. The idea behind this concept is simple: a candidate set for a trapping region is formed by the constraints imposed by practical, problem-dependent safety considerations. By verifying whether such set is forward-invariant for the joint learning operator, we obtain a yes–or–no answer on whether it is safe to allow multi-agent learning (possibly in a decentralized manner), without breaking these constraints. This method can be seen an alternative solution concept in systems, where Nash equilibria are difficult or impossible to reach by learning dynamics. Trapping regions are intended to be used as a safety prerequisite. For instance a road authority could pre-approve the algorithms of automated road users, by checking whether their joint policy forms suitable trapping regions – before they are deployed in real life.

This paper is organized as follows. Section 2 introduces the setting and necessary preliminaries. In Section 3 we present the definition of a trapping region, prove several useful theorems and lemmas that are useful for their verification for Lipschitz-continuous learning, and present two algorithms, for verifying whether given hyperrectangular sets forms a trapping region, only from the knowledge of learning operator on the set boundary. The first algorithm, is based on binary partitioning, and is applicable when learning dynamics are known analytically and Lipschitz, and we would like to have a mathematically rigorous guarantee. The second one is a heuristic algorithm, applicable in scenarios where learning dynamics can only be sampled, its dependability is directly correlated to the number of boundary samples taken.

Finally, in Section 4 we introduce two examples, that illustrate the applications of trapping regions. The first of our first of our examples is a toy problem, a simple GAN-like learning scenario, where gradient learning starting from almost all points never converges, whereas trapping regions are abundant and easy to find. In our second example we deal with a practical traffic control problem, where four intersection in a traffic network adjust their strategies to dispatch traffic in an optimal manner. For this problem, we construct and verify a trapping region, which ensures all traffic directions will be given enough time, when traffic controllers are left to learn unsupervised.

1.1 Related Work

Trapping regions are well known and standard tool in qualitative theory of dynamical systems, e.g. [Meiss, 2007; Bonatti, 2006], but to the best of our knowledge have not been directly applied in learning and control scenarios. The majority of work on safety guarantees in control theory focuses on so-called constrained optimization [Altman, 1999]. In the context of safe reinforcement learning, the focus has been on designing algorithms that satisfy particular safety constraints, c.f. [Garcıa and Fernández, 2015] and references therein. In the multi-agent case, research has been directed towards methods where an orchestrator [ElSayed-Aly *et al.*, 2021] or agents individually [Lu *et al.*, 2021] are adapting their behavior to respect the constraints; this has also been the underlying philosophy in the method of *barrier functions* [Wills and Heath, 2004; Yang *et al.*, 2020]. There are also strong connections to methods of formal verification methods, in particular ones based on reachability analysis [Ruan *et al.*, 2018; Wang *et al.*, 2021].

Relation to Lyapunov Control. Our method shares most similarities with the ones based on Lyapunov functions, such as Neural Lyapunov Control [Chang *et al.*, 2019], see also [Berkenkamp *et al.*, 2017]. There are however several key differences that we would like to highlight here. Most importantly, Lyapunov-based methods are only applicable to (locally) convergent scenarios, as the existence of a Lyapunov function implies the existence of a locally attracting equilibrium of the system. On the other hand, trapping regions are very well suited to deal with problems, where learning never converges to a stationary solution. Even if the learning trajectory does not converge, the bounds provided by the trapping region will ensure that they never diverge into unsafe regions of the joint strategy space. This has strong practical consequences: a non-convergent learning process without safety guarantees would have to be indefinitely supervised, whereas trapping regions ensure that it only explores safe parts of the strategy space, and does not require supervision. Dealing with non-stationarity is particularly important for multi-agent systems, as it was shown e.g. in [Kleinberg *et al.*, 2011] outcomes of a non-convergent cyclic learning process can lead to higher social welfare than these of a stationary Nash; and even worse, often stationarity cannot be ensured, as some learners can be outside of our control (e.g. in adversarial scenarios). In Section 4.1, we provide a low-dimensional example where none of the learning trajectories converge to the equilibrium point, but trapping regions are easy to find, c.f. Figure 2.

The second difference comes in computational complexity. Neural Lyapunov Control requires evaluation of learning directions over a whole domain, while we only require it on a boundary of a domain, effectively reducing the dimension of the verification problem by one (c.f. Lemma 1).

2 Preliminaries

We consider decentralized learning schemes for groups of n agents that can be represented compactly by discrete adaptive dynamics of the form:

$$\begin{aligned} x^1_{t+1} &:= x^1_t + \gamma F_1(x^1_t, \ldots, x^n_t), \\ &\ldots, \\ x^n_{t+1} &:= x^n_t + \gamma F_n(x^1_t, \ldots, x^n_t), \end{aligned} \tag{1}$$

where $x_i \in X_i \subset \mathbb{R}^{k_i}$ represents a point in the strategy space of a given agent i (e.g. weights in a neural network or ratios of playing a mixed strategy), and the parameter $\gamma \in \mathbb{R}^+$ denotes the adaptation rate. Throughout this paper, we assume that the learning operators are continuous, and we denote by $N = \sum_i k_i$ the dimensionality of the joint learning space. The maps $F_i : X_i \to \mathbb{R}^{k_i}$ represent the *learning operators*, i.e. the outputs of the algorithms of each agent based on the inputs. For instance, for individual gradient-ascent type of algorithms we have

$$F_i(x^1, \dots, x^n) = \nabla_{x_i} \mathbb{E}(R_i | x^1, \dots, x^n). \tag{2}$$

with $R_i : \mathbb{R}^N \to \mathbb{R}$ being the individual reward/payoff for agent i. To simplify the exposition, we will sometimes represent the learning system (1) in a vectorized notation

$$x_{t+1} := x_t + \gamma F(x_t) \tag{3}$$

with $F = [F_1, \dots, F_n]^T$ and $x = [x^1, \dots, x^n]^T$. Joint strategy sequences $\{x_t\}_t$ which satisfy (3) will be referred to as the *learning trajectories*.

An *equilibrium* for the system (3) is a point in the joint strategy space $x_* \in \mathbb{R}^N$ such that $F(x_*) = 0$. In gradient learning, it is also a necessary condition for a strategy profile to be a *local optimum*, with the sufficient condition being that the Hessian of the learning operator F is negative definite. We remark that an equilibrium for the learning system (3) does not necessarily need to be a Nash equilibrium; however, a local optimum is a *local Nash point*, i.e. no agent is able to increase their reward unilaterally from a such point by performing a small deviation in its strategy.

For single agent learning, gradient descent in (2) does converge to a local optimum under mild assumptions of regularity of R_i, and suitable choices of γ (i.e. γ can be constant, but needs to be suitably small). In general multi-agent setting, learning schemes given by systems of form (1) can have complicated, even chaotic dynamics, and might not converge to equilibria at all, as for instance in relatively simple two-player games [Sato *et al.*, 2002].

3 Trapping Regions

Convergence in multi-agent learning cannot be always guaranteed; however the key aspect for security / reliability is often enough to ensure, that learning agents do not diverge into regions of policy space, which can yield dangerous combinations of strategies. To this end, one needs to contain the learning trajectories within a prescribed safety region. Motivated by this rationale, in this section we formally define the *trapping region* – a subset of the joint strategy space, characterized by the property that learning trajectories that begin within such region can never leave it.

We remark that the definitions and theorems below could have been framed in a continuous learning setting by working with the ordinary differential equation $\dot{x} = F(x)$. but we opted for a discrete point of view, as more commonly encountered in literature on learning systems. The discrete system (3) does in fact emerge as the Euler numerical solution of the ODE, with step size γ.

3.1 Formal Definition and Forward Invariance

In what follows, we will denote by $\operatorname{int} X$ and ∂X respectively the topological interior and boundary of a set X, by $\operatorname{dist}(x, X)$ the Euclidean distance between a point x and a set X, and by $\operatorname{diam}(X)$ the diameter (in Euclidean distance) of a set X. By X^l we will denote the Cartesian product of l copies of X. We also recall that a compact set is a set which is bounded, and which contains limit points for all convergent sequences of its elements. We first recall the classical definition of a trapping region in context of learning dynamics (1).

Definition 1. *c.f. [Bonatti, 2006]. Let $\mathbf{T} \subset \mathbb{R}^N$ be a compact subset of the joint strategy space, and let $\gamma > 0$. If*

$$x + \gamma F(x) \subset \operatorname{int} \mathbf{T}, \quad \forall x \in \mathbf{T}, \tag{4}$$

then we call $\mathbf{T}$ a trapping region (for the system (3)*, with learning rate γ).*

The following theorem, a folklore in dynamical systems community, highlights the advantage of establishing trapping regions; a trapping region not only guarantees that the learning curves starting inside can never escape it, but also that there is a learning equilibrium (possibly a Nash equilibrium) inside of it.

Theorem 1. *Let $\mathbf{T}$ be a trapping region. Then*

1. *Any learning trajectory* (3) *that starts in $\mathbf{T}$ never leaves $\mathbf{T}$,*
2. *If $\mathbf{T}$ is convex, then there exists a learning equilibrium $x^* \in \operatorname{int} \mathbf{T}$.*

3.2 Algorithmic Verification: Explicit Learning Dynamics

In practice, verification of condition (4) can be troublesome, as the volume of the trapping region usually requires a prohibitively high amount of samples. For small learning rates and continuous learning dynamics, it is however enough to verify this assumption on the boundary, as any trajectory that could leave the region would have to pass through the boundary area. This is a standard argument, more commonly known in the continous case (e.g. [Meiss, 2007]), and is formalized for our discrete setting in Lemma 1.

Lemma 1. *Given a compact set $\mathbf{T}$, if $\gamma > 0$ is sufficiently small, and for all $x \in \partial \mathbf{T}$ we have*

$$x + \gamma F(x) \in \operatorname{int} \mathbf{T}, \tag{5}$$

then $\mathbf{T}$ is a trapping region.

Lemma 1 can be used to derive exact inequalities needed to be satisfied by the learning operators, which are sufficient to establish a trapping region. In what follows, we denote by $x^i = [x^{i1}, \dots, x^{ik_i}]^T$ and $F_i = [F_{i1}, \dots, F_{ik_i}]^T$ the components of strategies and learning operators for each agent $i \in 1, \dots, n$. In the examples we will sometimes omit the second subscript, when the strategy space of each agent is one-dimensional.

Definition 2. *Let $\mathbf{T}$ be a set of the form of a product of intervals*

$$\mathbf{T} := [x_-^{11}, x_+^{11}] \times \dots \times [x_-^{nk_n}, x_+^{nk_n}] \subset \mathbb{R}^N. \tag{6}$$

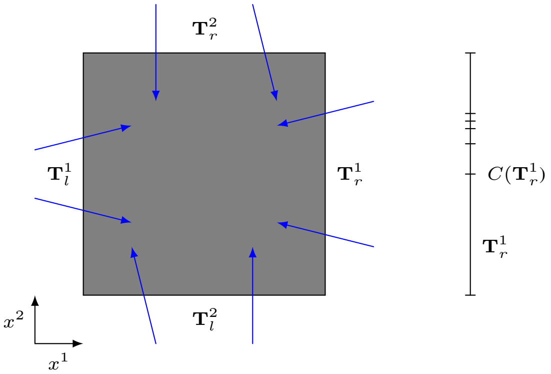

Figure 1: A schematic illustration of a trapping region $\mathbf{T}$, arrows indicate the possible directions of learning dynamics on the boundary (left), and a binary partitioning of the first right face of $\mathbf{T}$ (right).

For $i \in 1,\ldots,n$, $j \in 1,\ldots k_i$, we denote by $\mathbf{T}_l^{ij}$ the set of all points $x \in \mathbf{T}$, such that π_{ij} – the projection onto i-th agents j-th component satisfies $\pi_{ij}x = x_-^{ij}$. We call this set the (ijth) left face *of* $\mathbf{T}$. *Similarly, we denote by $\mathbf{T}_r^{ij}$ the set of all points $x \in \mathbf{T}$, such that $\pi_{ij}x = x_+^{ij}$, and call it* the (ijth) right face *of* $\mathbf{T}$.

Our next Lemma follows follows directly from Lemma 1 applied to a trapping region of form of a product of intervals.

Lemma 2. *Given a set $\mathbf{T} \in \mathbf{R}^N$ which is a product of intervals, assume that the following* isolation inequalities *are satisfied:*

$$\begin{aligned} F_{ij}(x) > 0,\ \forall x \in \mathbf{T}_l^{ij}, \\ F_{ij}(x) < 0,\ \forall x \in \mathbf{T}_r^{ij}. \end{aligned} \tag{7}$$

Then, the set $\mathbf{T}$ is a trapping region for $\gamma > 0$ sufficiently small.

For Lipschitz-continuous learning dynamics, and trapping regions of form of a product of intervals, explicit bound on the range of γ can be given.

Theorem 2. *Let $\mathbf{T}$ be as in Lemma 2 and F be Lipschitz-continuous with Lipschitz constant over $\mathbf{T}$ bounded from above by L. The upper bound on step size γ for which $\mathbf{T}$ forms a trapping region in the learning system (1) can be given explicitly by*

$$\gamma < \frac{\min_{p\in\{l,r\}}\min_{ij}\min_{x\in\mathbf{T}_p^{ij}}|F_{ij}(x)|}{L\max_{x\in\mathbf{T}}||F(x)||_{max}}. \tag{8}$$

Remark 1. *For sufficiently regular boundaries $\partial\mathbf{T}$, conditions (7) can be generalized to situations, where $\mathbf{T}$ is not a product of intervals. Namely, for $\mathbf{T}$ to be a trapping region it is enough that*

$$\langle F(x), n_{\partial\mathbf{T}}(x)\rangle < 0, \forall x \in \partial\mathbf{T}, \tag{9}$$

where $n_{\partial\mathbf{T}}(x)$ is the normal vector to $\partial\mathbf{T}$, pointing in direction outwards of $\mathbf{T}$, c.f. [Meiss, 2007].

The visualization of assumption (7) from Lemma 2 is presented in Figure 1; the intuition behind it is that the learning operators F_{ij} have to point inwards, into the trapping region, so their values have to be positive on left faces and negative on right faces. When the adaptive dynamics are not given explicitly (e.g. they depend on a reward from environment simulator), one may need to resort to verifying condition (7) approximately, by evaluating the learning dynamics F_{ij} on a finite subset of points, which provide good enough coverage of faces of $\mathbf{T}$. If some analytical knowledge on learning dynamics is available, we can verify (7) rigorously (with sufficient numerical precision). For instance, assume that we know the upper bound for the Lipschitz constant of F over $\mathbf{T}$, given by L. Our verification is based on the following observation. We will check whether

$$\pm F_{ij}(x) > 0,\ x \in S, \tag{10}$$

where S denotes either of the faces $\mathbf{T}_l^{ij}$, $\mathbf{T}_r^{ij}$, respectively, or their hyperrectangular subsets. Then it is enough to verify that either

$$\mp F_{ij}(C(S)) + L\,\mathrm{diam}(S)/2 < 0, \tag{11}$$

where $C(S)$ is the baricenter (i.e. the centroid / intersection of diagonals) of S. Alternatively, we can show that

$$\pm F_{ij}(C(S)) \leq 0, \tag{12}$$

which will prove that the candidate $\mathbf{T}$ is not a trapping region.

If S is the whole face of $\mathbf{T}$ (i.e. $\mathbf{T}_l^{ij}$ or $\mathbf{T}_r^{ij}$ for some i, j), then the verification of inequality (11) can fail, even despite that $\mathbf{T}$ is a trapping region. Therefore, we propose to adopt *binary space partitioning* mechanism [Fuchs *et al.*, 1980] to iteratively subdivide faces of $\mathbf{T}$ into smaller hyperrectangles, until inequalities fail, or all hyperrectangles have been verified. For details, we refer to the pseudocode in Algorithm 1. The function SPLIT in Algorithm 1 splits a hyperrectangle S into two disjoint non-empty hyperrectangles S_1, S_2, such that $S = S_1 \cup S_2$ in half, along the longest dimension of the hyperrectangle.

Theorem 3. *If $\mathbf{T}$ is a trapping region, Algorithm 1 is guaranteed to terminate in finite steps. Without loss of generality, assume that $\mathbf{T}$ is a unit hypercube. Then, the computational complexity of the algorithm is $O\left(log(L/2m^*)^{\sum_{i=1}^n k_i - 1}\sum_{i=1}^n k_i\right)$, where*

$$m^* = \min_{i,j,x\in T_{l,r}^{ij}} |F_{ij}(x)|. \tag{13}$$

Conversely, if Algorithm 1 terminates and returns true, $\mathbf{T}$ is a trapping region for learning rates as in Theorem 2.

3.3 Algorithmic Verification: Sampled Learning Dynamics

In some situations, the exact learning dynamics are not available – e.g. they depend on a reward, which can only be obtained from a real world or an experiment. Then, one has to resort to heuristic verification of trapping regions, by sampling points from the faces of the interval set. We provide the pseudocode for this situation in Algorithm 2, and apply it in practice in traffic management example in Section 4.2.

Algorithm 1 Rigorous trapping region verification via binary space partitioning.

```
Inputs: Learning dynamics $F$,
$\mathbf{T} = [x_-^{11}, x_+^{11}] \times \cdots \times [x_-^{nk_n}, x_+^{nk_n}]$ – a candidate for the trapping region,
$L$ – upper bound for Lipschitz constant of $F$ over $\mathbf{T}$.
Returns: Is $\mathbf{T}$ a trapping region?
Start:
for agent i in 1:$n$ in parallel do
    for coordinate j in 1:$k_n$ in parallel do
        for direction in {left,right} in parallel do
            if direction is left then
                SETS_TO_CHECK = $\{\mathbf{T}_l^{ij}\}$, $\delta = -1$
            else
                SETS_TO_CHECK = $\{\mathbf{T}_r^{ij}\}$, $\delta = 1$
            while SETS_TO_CHECK $\neq \emptyset$ do
                $S$ = SETS_TO_CHECK.POP()
                $C(S) = \mathrm{baricenter}(S)$
                if $\delta F_{ij}(C(S)) \geq 0$ then
                    return false // no isolation
                else if $\delta F_{ij}(C(S)) + L\,\mathrm{diam(S)}/2 \geq 0$ then
                    // need subdivision to check isolation
                    $S_1, S_2$=SPLIT($S$) // binary partitioning
                    SETS_TO_CHECK.PUSH($S_1$, $S_2$)
return true
```

Algorithm 2 Non-rigorous trapping region verification via sampling.

```
Inputs: Learning dynamics $F$,
$F$ – learning dynamics, can be only sampled (e.g. from simulator),
$\mathbf{T} = [x_-^{11}, x_+^{11}] \times \cdots \times [x_-^{nk_n}, x_+^{nk_n}]$ – a candidate for the trapping region,
$M$ – sample size per face
Returns: Is $\mathbf{T}$ a trapping region?
Start:
for agent i in 1:$n$ in parallel do
    for coordinate j in 1:$k_n$ in parallel do
        for direction in {left,right} in parallel do
            if direction is left then
                SET = $\mathbf{T}_l^{ij}$, $\delta = -1$
            else
                SET= $\mathbf{T}_r^{ij}$, $\delta = 1$
            // a uniformly spaced sample of $M$ points
            S = SAMPLE_POINTS(SET, $M$)
            for $x \in S$ in parallel do
                // $F$ evaluated on sample points
                if $\delta F_{ij}(x) \geq 0$ then
                    return false // no isolation
return true
```

Proposition 1. *Algorithm 2 always terminates in finite steps, regardless of whether* $\mathbf{T}$ *is a trapping region or not. The computational complexity of the algorithm is* $O(2M\sum_{i=1}^{n} k_i)$.

We remark that Algorithm 2 contains a naive, uniform sampling strategy, and one can envision a more sophisticated tree-like partitioning, like in Algorithm 1, where we resample the regions in which we are closest to failing the isolation inequalities. However, due to computational demands of our illustrative example, the traffic experiment in Section 4.2, we have opted for uniform sampling, as it offers highest parallel execution potential, and was most suited for execution in a computational cluster environment – every sample evaluation can be executed as a separate process.

Theorem 4. *Let* S^* *be the set of all sampled points,* D *be the size of mesh generated by the sample, i.e.*

$$D = \sup_{i,j,x \in T_{l,r}^{ij}} \min_{x^* \in S^*} ||x - x^*||. \tag{14}$$

Also let

$$m^* = \min_{i,j,x^* \in S^*} ||F_{ij}(x^*)|| \tag{15}$$

quantify how close we were to fail verifying isolation over S^*. *If Algorithm 2 returns true and* F *is Lipschitz-continuous with Lipschitz constant* $L < m^*/D$, *then* $\mathbf{T}$ *is a trapping region for learning rates as in Theorem 2.*

4 Examples

In this Section we will provide examples of application of Algorithm 1 to two systems with known dynamics – a Generative Adversarial Network in Subsection 4.1 and of application of Algorithm 2 to a traffic learning system with dynamics provided by the system simulator in Subsection 4.2. Additional example in a model of economic competition is provided in the Supplementary Material.

4.1 Generative Adversarial Learning

Our first example is a system, which exemplifies the issue of non-convergence of multi-agent learning, but where trapping regions can be readily constructed. Since the learning is non-convergent, methods based on Lyapunov functions, and regions of attractions of equilibria would not be applicable to this scenario. We consider a parameterized family of learning systems, where the parameter controls the coupling between learner rewards – from completely decoupled, to strongly coupled. More concretely, agent one is in control of a continuous variable $\psi \in \mathbb{R}$, and agent two controls $\theta \in \mathbb{R}$. The rewards of each agent are the negative of the loss functions for each, and these are given by

$$L_1(\psi, \theta) = \psi^4 + \epsilon\psi\theta, \tag{16}$$

and

$$L_2(\psi, \theta) = \theta^4 - \epsilon\psi\theta, \tag{17}$$

for some positive, small ϵ.

Both agents use gradient descent on their respective loss functions, with a fixed step γ, which leads to following update rules

$$\begin{aligned} \psi_{t+1} &:= \psi_t - \gamma(4\psi_t^3 + \epsilon\theta_t), \\ \theta_{t+1} &:= \theta_t - \gamma(4\theta_t^3 - \epsilon\psi_t), \end{aligned} \tag{18}$$

which, for short, we shall denote by $(\psi_{t+1}, \theta_{t+1}) =: G(\psi_t, \theta_t)$.

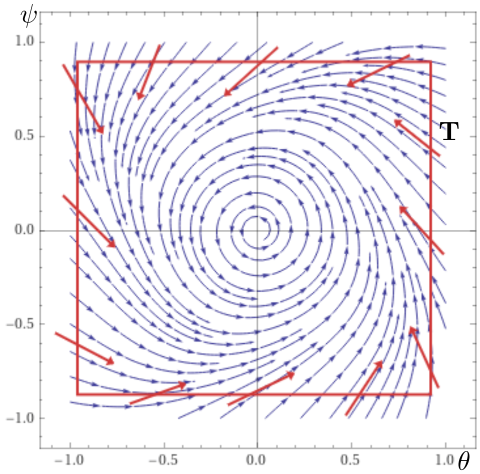

Figure 2: Learning trajectories of regularized Dirac-GAN diverge from the center equilibrium, but can be proven to be contained by a trapping region. In such non-convergent scenario a construction of a Lyapunov function is impossible.

Although a system with such prescribed loss functions is nothing more than a toy example, it serves to accentuate the problems of non-convergence. Similar learning systems have been thoroughly analyzed in literature; this system in fact has the same update rules as the famously non-convergent Dirac-GAN example in [Mescheder *et al.*, 2018] with the Wasserstein loss function, where both the generator and the discriminator apply an L^4 regularization term weighted by factor inversely proportional to ϵ.

The dynamics of (18) are surprisingly complicated for such low dimensional system. The system possesses a single Nash equilibrium $(\psi, \theta) = (0, 0)$ (also the only learning equilibrium), regardless of the value of ϵ. For joint optimization, the equilibrium is always locally unstable (regardless of how small the system coupling parameter ϵ is), and the learning trajectories starting from its near proximity diverge from it until they enter a cyclic regime. For initial conditions of larger norm, they converge towards the cyclic attractor, and never reach the equilibrium; in fact none of the other trajectories does. The divergence from the Nash equilibrium is formalized via the following proposition below (with $\|\cdot\|$ denoting L^2 norm):

Proposition 2. *For any $\gamma > 0$ and any $\epsilon > 0$ there exist a value $R_0 > 0$, such that for any (ψ_0, θ_0) with $0 < \|(\psi_0, \theta_0)\| < R_0$ we have $\|G(\psi_0, \theta_0)\| > \|(\psi_0, \theta_0)\|$. As a consequence, $(0, 0)$ is a repelling equilibrium.*

On the other hand, it is easy to find trapping regions. We report that by executing Algorithm 1 we have successfully established existence of various trapping regions for different values of ϵ, and computed γ bounds via formula (8) (by making use of quantities obtained in the algorithm):

- $\mathbf{T} = [-0.1, 0.1]^2$ and $\epsilon \in \{0.01, 0.02, 0.03, 0.04\}$ (we report failure, i.e. it is not a trapping region for $\epsilon = 0.05$) and the upper bounds on γ are given by $\{4.6 \times 10^{-3}, 7.9 \times 10^{-3}, 1.7 \times 10^{-3}, 10^{-17}\}$, respectively;
- $\mathbf{T} = [-0.2, 0.2]^2$ and $\epsilon \in \{0.05, 0.1, 0.15\}$ (failure for $\epsilon = 0.2$), and the upper bounds on γ are given by $\{1.9 \times 10^{-2}, 4.6 \times 10^{-4}, 2.0 \times 10^{-4}\}$, respectively.

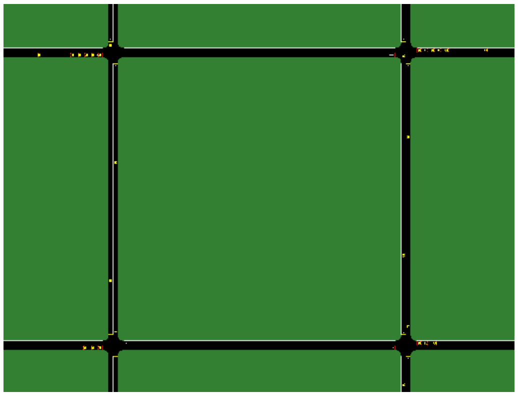

Figure 3: Multi-intersection traffic management problem map. Individual intersections optimize their own traffic demand by assigning green time to competing traffic streams. The safe set is defined by having minimally at least 20 seconds of uninterrupted green time for each traffic direction.

The Lipschitz constants in both examples were found analytically, by maximizing the L^1 norm of the total derivative $||D_{(\psi,\theta)}G||$ over $(\psi, \theta) \in \mathbf{T}$. We remark that the closer we got to the point of failure, the more subdivisions were needed in the partitioning algorithm, however the execution time was near immediate – a few seconds at most on a modern laptop, without leveraging parallelization. To contrast, a brute force optimization without verifying the trapping region would yield endless execution without convergence, and without any guarantees that learning will not diverge.

For this particular system, we can also prove the existence of an ϵ-parameterized family of trapping regions theoretically, by the following proposition:

Proposition 3. *The square given by $[-\sqrt{\epsilon}, \sqrt{\epsilon}]^2$ is a trapping region for step size $\gamma > 0$ small enough. As a consequence, trajectories never leave $[-\sqrt{\epsilon}, \sqrt{\epsilon}]^2$, and there is an equilibrium inside $[-\sqrt{\epsilon}, \sqrt{\epsilon}]^2$ (it is in fact the global Nash equilibrium $(0, 0)$).*

4.2 Multi-Agent Traffic Management

Our second example is of a more practical nature. We analyze a rectangular network of four signalized intersections, each situated 200 meters from its two nearest neighbors, as depicted in Figure 3. Each of the intersections controls traffic by alternating between one of two phases – giving green to either the vertical or the horizontal stream of vehicles. The *cycle time*, i.e. the total time for serving the horizontal and, subsequently, the vertical movement is set to 60 seconds. For each episode of simulation, of length of two hours, each intersection can select a strategy from the continuous set $A = [0, 60]$, which determines the amount of green seconds to be assigned to the first phase (*the offset*). The remainder of the cycle is assigned to the second phase. The vehicle streams are generated on all roads in all directions (i.e. east ↔ west, and north ↔ south), and, for simplicity, we excluded left and right turning movements on intersections. The simulation is controlled by

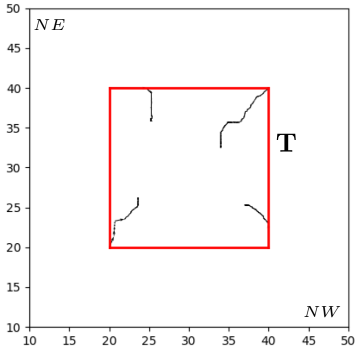

Figure 4: The trapping region $\mathbf{T}$ (in red) and learning trajectories in the traffic control scenario (in black) – a projection onto first two dimensions of joint strategy space (corresponding to offsets of northeastern and northwestern intersections). Learning curves, which begin on the boundary of the trapping region do not escape it, and evolve in the interior of the set.

an open-source microscopic traffic simulator SUMO [Lopez *et al.*, 2018], version 1.8.0. The episodical payoff for each intersection is the negative of the aggregate number of vehicle-seconds on all road lanes incoming to the intersection, therefore the goal of each intersection is to dispatch the incoming traffic as efficiently as possible. To ensure that the learning dynamics do not exhibit trivial symmetries, the simulations were performed for a simple instance of asymmetrical demand. One vehicle would be spawned each ten seconds on the beginning of each of the outmost lanes of the network, with the exception of the northeast ↔ northwest stream, where vehicles are spawned every five seconds. Analogous computations could have been performed for other traffic patterns.

For our experiment, the selection of the strategy by the learners is performed via decoupled gradient descent, as in Equations (1) and (2). Each intersection controller estimates the gradient of own reward by difference quotients:

$$\delta\nabla_{x_i}R_i(x) \approx R_i(x_i, x_{-i}) - R_i(x_i + \delta, x_{-i}) \tag{19}$$

for some small δ (in our experiments $\delta = 0.1$). The adaptation rate γ is set to 10^{-6}. Such settings were chosen as they would give satisfactory results for learning on one intersection, while keeping other intersections fixed.

As discussed previously, non-convergence is an undesirable learning effect in such traffic management scenario, as one would like to ensure that learning always stays within some predetermined bounds, so minimal green time can be given to vehicle flows and pedestrians within each cycle. As a reasonable prerequisite we assume that each phase should be given at least 20 seconds of green time, which translates to the candidate for a trapping region given by $\mathbf{T} = [20, 40]^4$. From the nature of the problem, we expect the reward function to be continuous, but we do not have an analytical formula for it. Therefore, we apply Algorithm 2 and sample faces of $\mathbf{T}$ with a uniform rectangular grid of five points in each direction ($M = 125$). This part of computation is parallelized over multiple threads, and so it does not significantly increase verification time.

The numerical evaluation of isolation inequalities (7) is successful; and, according to Theorem 1 we conclude that each learning trajectory that starts in $\mathbf{T}$ stays in $\mathbf{T}$, regardless of whether it converges, and that there is a learning equilibrium in $\operatorname{int}\mathbf{T}$. We illustrate this by plotting projections of the trapping region and four learning trajectories in Figure 4. We remark that due to parallelization, verifying the region was much more efficient computationally, than computing learning trajectories. It took about half an hour on 32 CPUs, whereas computation of depicted 500 steps of each learning trajectory took about eight hours, and could utilize only one CPU per trajectory. The experiment used AMD 7452 and AMD 7502P CPUs, 2.35 and 2.5 Ghz respectively. It would have been technically possible to execute the algorithm on a GPU, however in this example it would bring no advantage, as the most time consuming part was obtaining reward difference quotients from the system simulator, which can only run on CPUs.

5 Applicability, Limitations and Future Work

In this paper we have introduced the method of trapping regions, which can be used to circumvent safety problems caused by non-convergence in multi-agent learning. We have presented algorithms for verification of trapping regions, and theoretical results on the implications for safety and provided examples in GAN learning, in an applied traffic management scenario, and (in the Supplementary Material) in a standard mathematical model of economic competition.

Our examples are relatively low-dimensional. We however remark, that even low-dimensional learning can be non-convergent and highly unpredictable, and therefore pose safety concerns (e.g. the chaotic example of chaos in a simple two player game [Sato *et al.*, 2002]). Moreover, low-dimensionality of strategy spaces does not mean that the learning systems need to be trivial; for instance agents can be controlled by high-dimensional pre-trained neural networks with the last layer being retrained online. Our algorithms are well suited to deal with such scenarios. The extension of the method to high-dimensional settings is a challenge for future research, due to exponential complexity of verification algorithms w.r.to the joint action space. We see possibilities for exploiting symmetries of the action space as a method for dimensionality reduction: e.g., in mean field games where an infinite amount of identical agents share same learning dynamics [Yang *et al.*, 2018], or by employing coordination graphs [Kuyer *et al.*, 2008].

Acknowledgements

This project was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 —INFLUENCE).

erc
European Research Council
Established by the European Commission

References

[Altman, 1999] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[Berkenkamp *et al.*, 2017] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 908–919, 2017.

[Bonatti, 2006] Christian Bonatti. Generic properties of dynamical systems. *Encyclopedia of Mathematical Physics*, pages 494–502, 2006.

[Brown, 1951] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[Chang *et al.*, 2019] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *NeurIPS*, volume 32, 2019.

[Czechowski and Oliehoek, 2023] Aleksander Czechowski and Frans A. Oliehoek. Safety guarantees in multi-agent learning via trapping regions. *arXiv*, https://arxiv.org/abs/2302.13844, 2023.

[ElSayed-Aly *et al.*, 2021] Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. Safe multi-agent reinforcement learning via shielding. *AAMAS*, 2021.

[Fuchs *et al.*, 1980] Henry Fuchs, Zvi M Kedem, and Bruce F Naylor. On visible surface generation by a priori tree structures. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 124–133, 1980.

[Garcıa and Fernández, 2015] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *JMLR*, 16(1):1437–1480, 2015.

[Hart and Mas-Colell, 2003] Sergiu Hart and Andreu Mas-Colell. Uncoupled dynamics do not lead to nash equilibrium. *American Economic Review*, 93(5):1830–1836, December 2003.

[Kleinberg *et al.*, 2011] Robert D Kleinberg, Katrina Ligett, Georgios Piliouras, and Éva Tardos. Beyond the Nash equilibrium barrier. In *ICS*, pages 125–140, 2011.

[Kuyer *et al.*, 2008] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer, 2008.

[Lanctot *et al.*, 2017] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *NIPS*, page 4193–4206, 2017.

[Lockhart *et al.*, 2019] Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr Timbers, and Karl Tuyls. Computing approximate equilibria in sequential adversarial games by exploitability descent. In *IJCAI*, page 464–470, 2019.

[Lopez *et al.*, 2018] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems*, pages 2575–2582, 2018.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *NIPS*, page 6382–6393, 2017.

[Lu *et al.*, 2021] Songtao Lu, Kaiqing Zhang, Tianyi Chen, Tamer Basar, Lior Horesh, Ria Vinod, Pin Yu Chen, et al. Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning. In *AAAI*, volume 35, pages 8767–8775, 2021.

[Meiss, 2007] James D Meiss. *Differential dynamical systems*. SIAM, 2007.

[Mescheder *et al.*, 2018] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, pages 3481–3490, 2018.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *NIPS Deep Learning Workshop*, 2013.

[Oliehoek *et al.*, 2006] Frans A Oliehoek, Edwin D De Jong, and Nikos Vlassis. The parallel nash memory for asymmetric games. In *Proceedings of the 8th annual conference on genetic and evolutionary computation*, pages 337–344, 2006.

[Ruan *et al.*, 2018] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. *IJCAI*, page 2651–2659, 2018.

[Sato *et al.*, 2002] Yuzuru Sato, Eizo Akiyama, and J Doyne Farmer. Chaos in learning a simple two-person game. *PNAS*, 99(7):4748–4751, 2002.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Sutton *et al.*, 1999] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.

[Wang *et al.*, 2021] Xiaoyan Wang, Jun Peng, Shuqiu Li, and Bing Li. Formal reachability analysis for multi-agent reinforcement learning systems. *IEEE Access*, 9:45812–45821, 2021.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[Wills and Heath, 2004] Adrian G. Wills and William P. Heath. Barrier function based model predictive control. *Automatica*, 40(8):1415–1422, 2004.

[Yang *et al.*, 2018] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *ICML*, pages 5571–5580, 2018.

[Yang *et al.*, 2020] Yongliang Yang, Kyriakos G Vamvoudakis, and Hamidreza Modares. Safe reinforcement learning for dynamical games. *International Journal of Robust and Nonlinear Control*, 30(9):3706–3726, 2020.