Cooperative Multi-player Bandit Optimization

Ilai Bistritz, Nicholas Bambos
Stanford University
{bistritz,bambos}@stanford.edu

Abstract

Consider a team of cooperative players that take actions in a networked-environment. At each turn, each player chooses an action and receives a reward that is an unknown function of all the players' actions. The goal of the team of players is to learn to play together the action profile that maximizes the sum of their rewards. However, players cannot observe the actions or rewards of other players, and can only get this information by communicating with their neighbors. We design a distributed learning algorithm that overcomes the informational bias players have towards maximizing the rewards of nearby players they got more information about. We assume twice continuously differentiable reward functions and constrained convex and compact action sets. Our communication graph is a random time-varying graph that follows an ergodic Markov chain. We prove that even if at every turn players take actions based only on the small random subset of the players' rewards that they know, our algorithm converges with probability 1 to the set of stationary points of (projected) gradient ascent on the sum of rewards function. Hence, if the sum of rewards is concave, then the algorithm converges with probability 1 to an optimal action profile.

1 Introduction

Bandit optimization is concerned with a single player that at each turn takes an action and receives a reward ([1,2]). Based on the past rewards, the player gradually learns to play better actions that yield higher rewards. In a networked environment, many players take simultaneous actions. Examples are smart cities, robotic networks, autonomous vehicles, the smart grid and communication networks. In wireless communication networks, one player's transmission creates interference for all others. With autonomous vehicles, every vehicle creates congestion for all the others. Therefore, the actions of one player affect the rewards of all others, and by that their future actions. This coupled learning gives rise to a game-theoretic formulation, with conflicting individual reward functions.

Conflicting interests between players are often unavoidable due to limited resources. However, conflicts do not have to result in a competition. In many applications, the players are willing or simply programmed to cooperate. By cooperating, the outcome can turn out to be better for everyone than with competition. Cooperation is naturally the case when players are deployed by the same operator or that their protocol is standardized. These players operate as a team to solve their conflicts in order to maximize the social welfare, as was done in [3–11]. In wireless networks, each user typically wants to maximize its throughput, which reduces the throughput of other devices. However, all devices must run a protocol that is programmed in their chip (e.g., LTE or WiFi), to the benefit of all. With autonomous vehicles, each user wants to minimize its trip duration. However, the vehicles are programmed to comply with the traffic rules, and are likely to follow a standardized communication protocol to increase efficiency and safety [12].

If a player knew all the rewards of all other players at all times, the distributed learning problem boils down to a single-player optimization problem with bandit feedback [1, 13]. The common approach is then to estimate the gradient-based on samples of the unknown reward function.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

In large networks, players do not know the rewards of all of their peers. Collecting all the rewards is not realistic, since it incurs huge delays if it is even feasible given the communication infrastructure. Cooperating players that try each to estimate the gradient of the sum of their rewards based on bandit feedback will find that at every turn, most feedback (rewards in the sum) is missing. While the players are willing to maximize the greater good, they just do not how, and naturally they know more about their local objective and environment. Therefore, with cooperative players, the optimization of the global objective is limited by the information players have rather than by their selfishness.

In this paper, we show that convergence to an optimal action profile (or to the set of stationary points for the non-concave case) is possible even when each player takes stochastic gradient steps based only on few reward values of its peers. The idea is that if each player has some probability to receive any term in the sum of rewards, then even if the sum is far from being fully known at any given turn, the resulting noise is averaged over time and an optimal action profile can still be reached. However, the estimation of the sum of rewards is biased since players have different probabilities of getting the reward value of different players, and different accuracies for estimating the gradients of their rewards. Our algorithm learns to correct this bias and our analysis shows how to tune the step size and the gradient estimation to allow this correction. This is the first multi-player bandit algorithm for continuous optimization that converges to an optimal action profile (in the concave case).

1.1 Previous Work

Multi-player bandits, where multiple players learn and interact with each other, have received a surge of interest recently [14–25]. The current literature focuses on the discrete multi-armed bandit problem under a collision model where if two or more players pick the same arm they all receive zero reward. We consider here the continuous optimization case, under a more general game structure that allows modeling richer interactions between the players than the collision model.

At first glance, our scenario resembles distributed optimization with bandit feedback, studied in [13, 26–30]. However, it is the interaction, or the game structure between the players, that makes these two scenarios essentially different. First of all, [13,26–30] all query the sum of rewards at least twice. When a game is considered, a player cannot perturb the same point (action profile) twice to estimate the gradient, since the action profile changes every query (i.e., turn). To overcome that, we must estimate the gradient from a single sample as was done in [1,31,32]. Yet a larger obstacle is that in consensus-based algorithms like [13,26–30], each player holds a local version of the optimization variable, which is the action profile in our case. Hence, to implement [13, 26–30], each player has to know the effect of other players on its reward in order to query the objective at its local variable. Under this assumption of a known game structure, [5] employed distributed optimization to learn an optimal action profile. This is simply infeasible in our multi-player bandit scenario where players do not even know their local reward function. Furthermore, this approach requires each player in each turn to send entire action profiles, which is at least N-dimensional, and much bigger than that for large action spaces such as the set of weights of a neural network, as in multi-agent reinforcement learning [4]. Our approach does not assume that players can query the sum of rewards function. Instead, the bandit feedback available to each player is the reward they receive as a function of the actual action profile, and rewards communicated to them by other players. Hence, our approach also bypasses the need to transmit any vectors, as only scalars are communicated between the players.

The literature on learning in games [3, 6, 7, 33–37] studies the outcome of an interaction between conflicting players that each runs a learning algorithm designed to maximize its own accumulated reward. This outcome is often the Nash equilibrium (NE) of the game [31, 38, 39]. From an engineering point of view, one can design the game such that the NE has good global performance [8], or design the algorithm to learn an efficient NE among the existing ones [6, 7]. In the NE-based works, typically no communication between players is required, but the performance is suboptimal while our goal is to reach an optimal action profile. Finally, [11, 40] solves what can be considered the discrete optimization equivalent of our problem for the class of interdependent games.

Multiagent	Bandit Feedback	Known Objective
No Interaction	0th-order Distributed Optimization [26–30]	Distributed Optimization [41,42]
Interaction	**Cooperative Multi-player Bandits**	[5, 10, 43]

Table 1: Multiagent continuous optimization - high level taxonomy

2 Problem Formulation

We consider the following general multi-player bandit scenario, which is repeated over discrete turns indexed by t. We use capital letters to denote random variables, lowercase letters to denote their realizations, and bold letters to denote vectors. We also use the standard game-theoretic notation where $\boldsymbol{x}_{-n}$ is the vector of actions of all players except player n.

Definition 1. Let $\mathcal{N} = \{1, ..., N\}$ be a set of players. Each player $n \in \mathcal{N}$ picks an action $\boldsymbol{x}_n \in \mathcal{A}_n$ and receives a reward as a function of the actions of all players $u_n : \mathcal{A}_1 \times ... \times \mathcal{A}_N \to \mathbb{R}$, such that:

- $\mathcal{A}_n$ is closed and bounded and of the following form, for some positive integers s_n and d_n:

$$\mathcal{A}_n = \left\{ \boldsymbol{x} \in \mathbb{R}^{d_n} \mid q_i^n(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, s_n \right\} \tag{1}$$

  where for each i and n, $q_i^n(\boldsymbol{x})$ is continuously differentiable and convex and for each n and $\boldsymbol{x}$, the gradients $\nabla q_i^n(\boldsymbol{x})$ for all i such that $q_i^n(\boldsymbol{x}) = 0$ (active constraints) are linearly independent. This is a standard general constraint set in functional form. For example, $\mathcal{A}_n$ can be a d_n-dimensional ball or any hyper-rectangle.

- $u_n(\boldsymbol{x}_m, \boldsymbol{x}_{-m})$ is twice continuously differentiable in $\boldsymbol{x}_m$, for every $m, n = 1, ..., N$. Then, since $\mathcal{A}_n$ is compact and convex, it follows from the mean value theorem and the extreme value theorem [44, Page 89, Page 127] that both $u_n(\boldsymbol{x}_m, \boldsymbol{x}_{-m})$ and $\nabla u_n(\boldsymbol{x}_m, \boldsymbol{x}_{-m})$ are Lipschitz continuous.We refer to L as the maximal Lipschitz constant of $u_n(\boldsymbol{x}_m, \boldsymbol{x}_{-m})$, $\nabla u_n(\boldsymbol{x}_m, \boldsymbol{x}_{-m})$ over all m and n.

We emphasize that the function $u_n(\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$ is unknown even to player n, who only receives its reward value at each turn. Mathematically, the structure described above is a non-cooperative game. However, in our scenario the players are cooperative and their goal is to learn the action profile that maximizes $\sum_{n=1}^{N} u_n(\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$.

Players only know the action they played and the received reward (i.e., the bandit feedback). In particular, players cannot observe the actions of other players. For example, this is the case in wireless networks where the action might be the transmission power of each device, but only the total interference can be measured by (and affects) other players. This is also the case in distributed learning (in the spirit of Example 1 below) where the action is the parameters of the machine learning model of each player, that dictate the decision each player makes in real-time (e.g., autonomous vehicles).

As a result, information about the reward values of other players can only be obtained via communication. In practice, every player can only communicate with physically nearby players or with players that the available infrastructure allows it to. The number of available links and their stability may be time-varying based on the physical conditions of the environment or the mobility of the players. Therefore, we model the available communication links between the players using the following discrete-time graph process:

Definition 2. Let $G(t) = (\mathcal{V}, \mathcal{E}(t))$ be the (undirected) communication graph process, such that $t \in \mathbb{N}$, $\mathcal{V} = \{1, ..., N\}$ and player n and player m can communicate at turn t if and only if $(n, m) \in \mathcal{E}(t)$. Let $\mathcal{N}_n(t)$ be the set of neighbors of player n in $G(t)$ (including itself). We assume that $G(t)$ is an ergodic Markov chain on a state space $\mathcal{G}$ such that the graph union $\bigcup_{G \in \mathcal{G}} G$ is connected.

The assumption that $\bigcup_{G \in \mathcal{G}} G$ is connected means that for any players m, n, the stationary probability that there is a path between them is positive. However, this does not mean that $G(t)$ is likely to be connected and it is possible that $G(t)$ is never connected. A similar model was studied in [45].

Our goal is to distributedly solve the following team problem, where players only receive bandit feedback of their rewards as a result of their interaction:

$$\max_{\boldsymbol{x}_1, ..., \boldsymbol{x}_N \in \mathcal{A}_1 \times ... \times \mathcal{A}_N} \sum_{n=1}^{N} u_n(\boldsymbol{x}_1, ..., \boldsymbol{x}_N). \tag{2}$$

As with gradient ascent, the simplest convergence guarantee is obtained for a concave objective. However, we do not assume a concave objective, and our algorithm converges to the set of KKT stationary points in the non-concave case. We now give examples for cases of interest where the sum of rewards function is concave. More examples can be found in [10, 31]

Example 1. Multi-player Linear Regression - Each player n chooses $\boldsymbol{x}_n \in \mathcal{A}_n \subset \mathbb{R}^d$ where $\mathcal{A}_n$ is of the form in (1). Let $u_n(\boldsymbol{x}) = -\|A_n\boldsymbol{x} - \boldsymbol{y}_n\|^2$ where A_n is $m \times dN$ and $\boldsymbol{y}_n$ is $m \times 1$. Hence

$$u_n(\boldsymbol{x}) = -\left\|\tilde{A}_n\boldsymbol{x}_n + \boldsymbol{y}_n + B_n\boldsymbol{x}_{-n}\right\|^2 \triangleq -\left\|\tilde{A}_n\boldsymbol{x}_n - \tilde{\boldsymbol{y}}_n(\boldsymbol{x}_{-n})\right\|^2$$

where $\tilde{A}_n$ is $m \times d$ and B_n is $m \times d(N-1)$ such that $A_n = \begin{bmatrix} \tilde{A}_n & B_n \end{bmatrix}$. This represents a multi-player learning case where each player is trying to solve a linear regression to learn a linear model with parameters $\boldsymbol{x}_n$. The player then makes decisions based on this model, so we can treat $\boldsymbol{x}_n$ as the effective action of the player. However, the measurements collected by player n, $\tilde{\boldsymbol{y}}_n(\boldsymbol{x}_{-n})$, are a linear mixture of the other players' actions $\boldsymbol{x}_{-n}$. Since $u_n(\boldsymbol{x})$ is concave in $\boldsymbol{x}$ for each n, then $\sum_{n=1}^N u_n(\boldsymbol{x})$ is concave and all stationary points attain the global maximum. Using our algorithm, the players can learn a joint-model that minimizes their total regression error.

Example 2. Congestion Game - Let the set of resources be $\mathcal{R}$ and the congestion function be a convex non-decreasing $f : \mathbb{R}_+ \to \mathbb{R}$. The action of player n that has weight $w_n > 0$ is an allocation over $\mathcal{R}$, so it can play all non-negative $\boldsymbol{x}_n$ such that $\sum_{r\in\mathcal{R}} x_{n,r} = w_n$. Any other convex action space that meets Definition 1 is also possible, introducing more (or less) constraints on the allocation. The load on resource r is $l_r(\boldsymbol{x}) = \sum_{n=1}^N x_{n,r}$. The cost function of player n is its average congestion $c_n(\boldsymbol{x}) = \sum_{r\in\mathcal{R}} x_{n,r} f(l_r(\boldsymbol{x}))$. Hence, our global cost is

$$\sum_n c_n(\boldsymbol{x}) = \sum_{n=1}^N \sum_{r\in\mathcal{R}} x_{n,r} f(l_r(\boldsymbol{x})) = \sum_{r\in\mathcal{R}} f(l_r(\boldsymbol{x})) \sum_{n=1}^N x_{n,r} = \sum_{r\in\mathcal{R}} l_r(\boldsymbol{x}) f(l_r(\boldsymbol{x}))$$

which is a convex function of $\boldsymbol{x}$ since $f(l_r)$ is convex: $\frac{d^2(l_r f(l_r))}{dl_r^2} = 2f'(l_r) + l_r f''(l_r) \geq 0$, and $l_r(\boldsymbol{x})$ is affine. Hence, our algorithm will converge to an action profile that minimizes $\sum_n c_n(\boldsymbol{x})$, which can be the total delay in the system since it is an increasing function of the load. This is useful to model autonomous vehicles or any other team of players that share resources.

Example 3. Public Goods in Networks ([46]): Each player puts an effort $x_n \in [0, x_{\max}]$ for some maximal possible effort $x_{\max} > 0$. The reward of player $\boldsymbol{n}$ is its benefit from the total neighboring produced goods, minus a linear production cost:

$$u_n(\boldsymbol{x}) = v\left(x_n + \sum_{m\neq n} g_{m,n}x_m\right) - cx_n$$

where v is a twice differentiable, increasing and strictly concave function and $c > 0$. The coefficients $g_{m,n} \in \{0, 1\}$ determine which player is affected by whom. Since v is concave and $f_n(\boldsymbol{x}) = x_n + \sum_{m\neq n} g_{m,n}x_m$ is affine, then $u_n(\boldsymbol{x})$ is concave in $\boldsymbol{x}$ for each n, so $\sum_{n=1}^N u_n(\boldsymbol{x})$ is concave and all stationary points attain the global maximum. Using our algorithm, the players can find an optimal effort profile that maximizes the total benefit in the network. One can also consider negative externalizes, where $g_{m,n} = -1$, since $\sum_{n=1}^N u_n(\boldsymbol{x})$ remains concave in this case.

3 Multi-player Bandit Learning Algorithm

The distributed learning algorithm we propose to solve (2) is described in Algorithm 1. In Algorithm 1, each player takes stochastic gradient steps on $\sum_{m=1}^N u_m(\boldsymbol{x})$, using only the information available to them. The algorithm consists of three steps: playing perturbed actions, communicating on the graph $G(t)$ and learning the next action $\boldsymbol{X}_n(t+1)$.

3.1 Playing a Perturbed Action

To allow for gradient estimation from bandit (zero-order) feedback, players perturb the action $\boldsymbol{X}_n(t)$ they learned to play using a random perturbation $\boldsymbol{Z}_n(t)$, as specified in (4). To get intuition about how the perturbation works, consider the one-dimensional case where $Z_n(t) = \pm 1$ with probability $\frac{1}{2}$. Then we have $\mathbb{E}\left\{\frac{Z_n(t)}{\delta(t)} u_m(x + \delta(t)\boldsymbol{Z}_n(t))\right\} = \frac{u_m(x+\delta(t)) - u_m(x-\delta(t))}{2\delta(t)}$, which converges to the gradient as the sampling radius $\delta(t)$ decreases. Hence, $\frac{\boldsymbol{Z}_n(t)}{\delta(t)} u_m(\boldsymbol{x} + \delta(t)\boldsymbol{Z}_n(t))$ is a noisy estimation of $\nabla_{\boldsymbol{x}_n} u_m(\boldsymbol{x})$, with a vanishing bias.

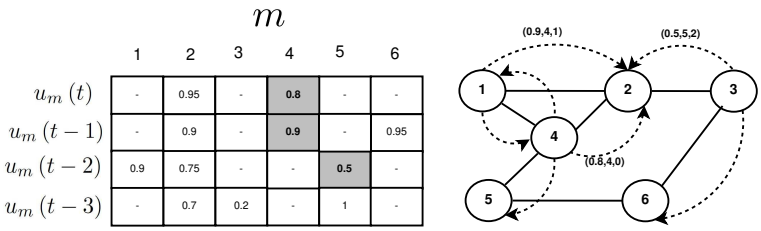


Figure 1: Left: reward values known to Player 2. Right: messages received by Player 2 at turn t.

To ensure the perturbation is in $\mathcal{A}_n$ (i.e., feasible), we use instead $\tilde{\boldsymbol{X}}_n(t) = \boldsymbol{X}_n(t) + \delta(t)\boldsymbol{W}_n(t)$ for $\boldsymbol{W}_n(t) = \boldsymbol{Z}_n(t) - \frac{\boldsymbol{X}_n(t)-\theta}{r}$, as was suggested in [1]. The parameters r, θ are chosen (by player n locally) such that a d_n-dimensional ball centered at θ with radius r is strictly inside $\mathcal{A}_n$. Then $\theta + r\boldsymbol{Z}_n(t) \in \mathbb{B}_r(\theta) \subseteq \mathcal{A}_n$. Hence if $\frac{\delta(t)}{r} < 1$ then feasibility follows from the convexity of $\mathcal{A}_n$:

$$\tilde{\boldsymbol{X}}_n(t) = \left(1 - \frac{\delta(t)}{r}\right)\boldsymbol{X}_n(t) + \frac{\delta(t)}{r}\left(\theta + r\boldsymbol{Z}_n(t)\right) \in \mathcal{A}_n. \tag{3}$$

3.2 Communication

With no communication, each player n can only estimate the gradient of its own reward function u_n. The stable point in this case is a point $\boldsymbol{x}$ where $\nabla_{\boldsymbol{x}_n} u_n(\boldsymbol{x}) = 0$ for all n (if $\boldsymbol{x}$ is not on the boundary). Every (local) Nash equilibrium is such a stable point, and under certain conditions such a stable point is a (local) Nash equilibrium [47, 48]. In any case, from an optimization perspective, this stable point does not solve (2) and may lead to very poor performance.

On the other extreme, if players knew all the rewards of their peers, then the problem becomes a single-player bandit optimization with $\sum u_n(\boldsymbol{x})$ as the reward function [1, 13], where each player n updates the coordinates of $\boldsymbol{x}_n$ in $\boldsymbol{x}$. However, in a large network with many players in different locations, this centralized architecture is infeasible. In a real-time scenario, players do not have enough time to gather much information from all over the network before making the next decision. Furthermore, players may be mobile and the available communication links may change with time.

Therefore, player n's estimation of the gradient of $\sum u_m$ is based on the information available to it at the moment. This information consists of a random and partial list of delayed rewards that player n has collected so far by communicating with other players on the time-varying graph $G(t)$, and its own reward. Surprisingly, this is enough to optimize $\sum u_m$, as we show in Theorem 1. Moreover, the communication does not need to be synchronized or very structured, as we describe next.

Definition 3. Define the set of players that, at time t, player n knows their τ-delayed reward by $\mathcal{U}_n^\tau(t)$, for $\tau \geq 0$. Naturally, $n \in \mathcal{U}_n^\tau(t)$ for all τ and t. Also define the set $\mathcal{U}_n(t)$ such that $(m,\tau) \in \mathcal{U}_n(t)$ if and only if $m \in \mathcal{U}_n^\tau(t)$ for some $\tau \leq M$.

In our communication protocol, at each turn t, each player n chooses to broadcast to all of its neighbors in $G(t)$ a random subset $\mathcal{B}_n$ of the recent rewards it knows, listed in $\mathcal{U}_n(t)$ (steps 2(a) and 2(b) of Algorithm 1). As a result, each player gets to know more recent reward values, and updates the lists $\mathcal{U}_n^\tau(t)$ for $\tau = 0, ..., M$, shifting old entries one step back and discarding $\mathcal{U}_n^M(t-1)$.

Each message that a player sends is a triplet of scalars, $(u_m(t-\tau), m, \tau)$, consisting of a reward, its delay τ and the index m of the player that earned this reward. A player sends one message for each element in $\mathcal{B}_n$. Note that the exchanged reward values are those that were actually earned by playing the actual perturbed actions $\tilde{\boldsymbol{X}}(t)$ (see (4)), as opposed to $\boldsymbol{X}(t)$. We use $u_n(t)$ to denote the reward that player n receives at time t, which is $u_n\left(\tilde{\boldsymbol{X}}(t)\right)$ and not $u_n(\boldsymbol{X}(t))$.

For a general communication protocol, it is natural to expect that a player will be more likely to broadcast some reward values than others. For example, players are likely to broadcast their own rewards since they always know them. This creates a "communication bias" that steers the algorithm away from maximizing $\sum_{m=1}^N u_m$. To correct the bias, each player n needs to estimate the probability p_m^n to receive the reward value of player m, and compensate for that in its estimation of the gradient of the sum $\sum_{m=1}^N u_m(\boldsymbol{x})$. This is described in Step 2(e) and Step 3(a) of Algorithm 1.

Our general communication protocol leaves the designer the freedom to choose the distribution of the broadcast set $\mathcal{B}_n(t)$ (given $\mathcal{U}_n(t)$). We only assume that no player's messages are being ignored. A simple example for such a "proper communication protocol" is that in each turn, each player picks uniformly at random one reward to broadcast from the set of all its known rewards $\mathcal{U}_n(t)$.

Definition 4. The communication in Algorithm 1 is proper if for all n and $(m,\tau) \in \mathcal{U}_n(t)$, player n broadcasts $(u_m(t-\tau), m, \tau)$ with positive probability, i.e., $\mathbb{P}((m,\tau) \in \mathcal{B}_n(t)) > 0$ for all t.

A toy example for the communication network is depicted in Fig. 1, for $N = 6$ players and $M = 3$. The table of rewards known to Player 2 is listed on the left. In this case, $\mathcal{U}_2^0(t) = \{2,4\}$, $\mathcal{U}_2^1(t-1) = \{2,4,6\}$, $\mathcal{U}_2^2(t-2) = \{1,2,5\}$ and $\mathcal{U}_2^3(t-3) = \{2,3,5\}$. At time t, Player 1, Player 3 and Player 4 broadcast each a triplet (u,n,τ). Player 2 inserts these values into its table, as can be seen in the grey entries. Note that it is possible for Player 3 to know the reward of Player 5 from two turns ago since the distance between them is 2. Fig. 1 does not depict the values concurrently received by all other players, and the value that Player 2 broadcasts.

Algorithm 1 Cooperative Multi-player Bandit Optimization

Initialization: Let $\delta(t) = \frac{\delta_0}{t^q}$, $\eta(t) = \frac{\eta_0}{t^p}$, $\forall t$, for some $\delta_0, \eta_0, p, q > 0$. Let the memory M be a non-negative integer. Let $\theta, r > 0$ be s.t. $\mathbb{B}_r(\theta) \subseteq \mathcal{A}_n$. Let $\varepsilon_0 > 0$ be small enough (as a function of the distribution of $G(t)$ and the communication protocol, see Lemma 1). Initialize $\boldsymbol{X}_n(1) \in \mathcal{A}_n$.
For each $t \geq 1$**, each player** n **runs:**

1. **Playing a perturbed action**:
   (a) Generate $\boldsymbol{Z}_n(t)$ uniformly at random on the d_n-dimensional unit sphere $\mathbb{S}^{d_n}$.
   (b) Play the perturbed action
   $$\tilde{\boldsymbol{X}}_n(t) = \boldsymbol{X}_n(t) + \delta(t)\left(\boldsymbol{Z}_n(t) - \frac{\boldsymbol{X}_n(t) - \theta}{r}\right) \tag{4}$$
   (c) Receive the reward $u_n(t) = u_n\left(\tilde{\boldsymbol{X}}_1(t), ..., \tilde{\boldsymbol{X}}_N(t)\right)$.
2. **Communication:**
   (a) Generate a random subset $\mathcal{B}_n(t) \subseteq \mathcal{U}_n(t)$.
   (b) Broadcast $\{u_l(t-\tau), l, \tau\}_{(l,\tau)\in\mathcal{B}_n(t)}$ for all neighbors $m \in \mathcal{N}_n(t)$.
   (c) Receive $\{u_l(t-\tau), l, \tau\}_{(l,\tau)\in\mathcal{B}_m(t)}$ from all neighbors $m \in \mathcal{N}_n(t)$.
   (d) Update $\mathcal{U}_n^{\tau+1}(t+1) = \mathcal{U}_n^{\tau}(t) \cup \{l\}$ for every (l,τ) received.
   (e) Update $p_m^n(t) = \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{1}_{\{m\in\mathcal{U}_n^M(\tau)\}}$ for all m.
3. **Learning the next action**:
   (a) Define $a_m^n(t) = \frac{1}{\max\{p_m^n(t),\varepsilon_0\}}$.
   (b) Compute the sum of gradients estimation
   $$\boldsymbol{Y}_n(t) = \frac{d_n \boldsymbol{Z}_n(t-M)}{\delta(t-M)} \sum_{m\in\mathcal{U}_n^M(t)} a_m^n(t)\, u_m(t-M). \tag{5}$$
   (c) Update the learned action and project it onto $\mathcal{A}_n$ (using $\prod_{\mathcal{A}_n}$)
   $$\boldsymbol{X}_n(t+1) = \prod_{\mathcal{A}_n}\left(\boldsymbol{X}_n(t) + \eta(t)\,\boldsymbol{Y}_n(t)\right). \tag{6}$$

End

3.3 Learning the Next Action

In the third step, players use the bandit feedback they gathered to approximate the gradient of $\sum u_n$ with respect to their action and take this stochastic gradient step to update their action $\boldsymbol{X}_n(t)$. Our approximation of the gradient is based on simultaneous perturbation [32], which is also known as the FKM algorithm [1]. The perturbation is described in Step 1 of Algorithm 1.

At turn t, each player uses the rewards from turn $t-M$ it was able to gather (listed in $\mathcal{U}_n^M(t)$) to estimate $\sum_{m=1}^N \nabla_{\boldsymbol{x}_n} u_m(\boldsymbol{x}(t-M))$. Typically, the set $\mathcal{U}_n^M(t)$ is much smaller than $\mathcal{N}$, so this estimate is almost always poor. Players have M turns to collect rewards that were received M turns ago (see Fig. 1). Rewards from different turns cannot be combined since they approximate the gradients at different action profiles. Hence, players cannot and do not wait to gather all information from all over the graph. Instead, each gradient step uses the last row in the table in Fig.1, and for large N the reward table is always sparse and lacking. Nevertheless, our algorithm is still able to converge to the set of KKT stationary point of $\sum_n u_n(\boldsymbol{x})$ since $\mathcal{U}_n^M(t)$ has a positive probability to include any of the players, and the algorithm learns over time to correct the associated communication bias.

Algorithm 1 converges to the same set of KKT stationary points that projected gradient ascent on $\sum_n u_n(\boldsymbol{x})$ would converge to. Being an especially noisy case of stochastic gradient descent, it is likely that Algorithm 1 avoids the convergence to saddle points (see [49, 50]) and only converges to local maxima even with a non-concave objective. In machine learning applications, and in particular with neural networks, local maxima often have surprisingly good performance (e.g., [51]).

4 Convergence Analysis

We now state our main result. The proof is involved and requires a series of lemmas which we postpone to the appendix. Here we discuss the proof idea and explain the main technical challenges.

Theorem 1 (Main Theorem)**.** *Let the N players each independently run Algorithm 1 to play the game in Definition 1. Let $\eta(t)=\frac{\eta_0}{t^p}$, $\delta(t)=\frac{\delta_0}{t^q}$ for some $\delta_0,\eta_0>0$. Assume that $G(t)$ is an ergodic Markov chain on $\mathcal{G}$ such that the graph union $\bigcup_{G\in\mathcal{G}} G$ is connected, and that the communication protocol is proper (Definition 4). Assume that the memory M is large enough, as a function of the distribution of $G(t)$ and the communication protocol* [1]*. If $0<p,q\leq 1$, $p-q>\frac{1}{2}$ and $p>\frac{q+3}{4}$ (e.g., $p=0.8, q=0.1$) then, as $t\to\infty$, the action profile $(\boldsymbol{X}_1(t),...,\boldsymbol{X}_N(t))$ converges with probability 1 to the set of KKT stationary points:*

$$\mathcal{KKT}=\left\{\boldsymbol{x}\,|\forall n,\,\exists\boldsymbol{\lambda}^n\geq 0\ s.t.\ -\nabla_{\boldsymbol{x}_n}\sum_{m=1}^N u_m(\boldsymbol{x})+\sum_{i:q_i^n(\boldsymbol{x}_n)=0}\lambda_i^n\nabla q_i^n(\boldsymbol{x}_n)=0\right\}. \tag{7}$$

Hence, if $\sum_{n=1}^N u_n(\boldsymbol{x})$ is concave and for all n, there exists an $\boldsymbol{x}_n$ s.t. $q_i^n(\boldsymbol{x}_n)<0$ for all i, then $(\boldsymbol{X}_1(t),...,\boldsymbol{X}_N(t))$ converges to the set $\arg\max_{\boldsymbol{x}\in\mathcal{A}}\sum_{n=1}^N u_n(\boldsymbol{x})$ with probability 1 as $t\to\infty$.

Algorithm 1 belongs to the family of stochastic approximation algorithms of the form:

$$\boldsymbol{X}(t+1)=\prod_{\mathcal{A}_1\times...\times\mathcal{A}_N}(\boldsymbol{X}(t)+\eta(t)\boldsymbol{Y}(t)). \tag{8}$$

Note that (8) occurs when each player runs (6) distributedly, since $\prod_{\mathcal{A}_1\times...\times\mathcal{A}_N}=\prod_{\mathcal{A}_1}\times...\times\prod_{\mathcal{A}_N}$. In our case, the gradient sample is $\boldsymbol{Y}(t)=(\boldsymbol{Y}_1(t),...,\boldsymbol{Y}_N(t))$ where $\boldsymbol{Y}_n(t)$ is given by (5). The sample $\boldsymbol{Y}(t)$ is a delayed and a noisy version of $g(\boldsymbol{x})=(g_1(\boldsymbol{x}),...,g_N(\boldsymbol{x}))$, where

$$g_n(\boldsymbol{x})=\sum_{m=1}^N\nabla_{\boldsymbol{x}_n}u_m(\boldsymbol{x}). \tag{9}$$

The sample in (5) is not identical to the exact sum of gradients (9) for two main reasons: the gradient bias and noise and the communication bias and noise. The third reason is the M-samples delay in (5), for which the bias can be bounded as we do in Theorem 1. However, the delay does not create any actual bias but only shifts the trajectory of $\boldsymbol{Y}_n(t)$ by M, without affecting the convergence.

Our proof strategy is to show that under the conditions of Theorem 1, the biases and noises that give (5) instead of (9) are well behaved over time, therefore leading the stochastic optimization in the right direction. Specifically, the main challenge in designing Algorithm 1 is to control the bias each player has towards better estimating the gradients of the players it knows more about, which naturally are

[1] For a constant G, $M=N$ is sufficient. See Subsection 8.1 in the appendix for details

the nearby players. Most of our analysis deals with overcoming this issue. However, even after we tamed the biases and noises by a smart tuning of our algorithm, standard stochastic gradient descent results cannot be applied to prove convergence. The reason is that our noise has memory and is state-dependent (i.e., $\boldsymbol{x}_n(t)$-dependent) through $\mathcal{U}_n^M(t)$ and $a_m^n(t)$ (and the projection onto $\mathcal{A}_n$). Instead, we have to employ the more general Kushner-Clark Theorem to complete our proof ([52, Theorem 5.3.1, Page 191], [53, Pages 297-300] or Theorem 2 in the appendix).

We denote by $(\Omega, \mathcal{F}, \mathbb{P})$ the probability space that includes the random graph $G(t)$, the random communication protocol and the random perturbations $\boldsymbol{Z}_n(t)$. We also define the following process:

Definition 5. Define the memory process of Algorithm 1 by $\Gamma(t) = \left(\{\mathcal{U}_n(t)\}_{n=1}^N, G(t)\right)$.

We emphasize that no player has access to $\Gamma(t)$. In particular, players do not know the rewards of other players or the structure of the graph. The importance of $\Gamma(t)$ lies in it being an ergodic Markov chain, when defined on the set of "feasible states", which are the feasible $\{\mathcal{U}_n(t)\}_{n=1}^N, G(t)$ pairs.

The first step in controlling the communication bias is to employ a proper communication protocol (see Definition 4) and a large enough memory M. Then, the unique stationary distribution $\{\pi_\gamma\}$ of this ergodic chain assigns a positive probability that a message from a given player arrives at a given destination in less than M steps. We emphasize that it is impossible (for any algorithm) to optimize a sum of functions if samples of some of the functions are never received. Given a specific communication protocol and transition probabilities for $G(t)$, one can easily deduce what M is large enough. Even with a well designed M, the rewards of some players are only rarely received by certain other players, and players must estimate their gradients based only on a very partial subset of received rewards. This is formalized and further discussed in Lemma 1 in Section 8 of the appendix.

Next, we analyze the gradient noise and bias. We show that the gradient bias, that results from the positive sampling radius $\delta(t)$, decreases at rate $O(\delta(t))$. We then show that the noise, which is the component of the error after subtracting the (conditional) expectation, is a martingale sequence that averages out to zero over time. This is formalized in Lemma 2 in Section 9 of the appendix.

Based on Lemma 1 and Lemma 2, we can now address the main technical challenge of controlling the communication bias and noise. In a naive implementation of our algorithm, even with large enough M, a player is biased towards optimizing the rewards of nearby players over players that are far away in the network $G(t)$. This communication bias occurs since a player is more likely to receive the rewards of nearby players. To compensate for that, players in our algorithm learn on the fly the probabilities $p_m^n(t)$ of receiving different reward values and use them to correct the bias. We show that the communication bias vanishes as the estimations $p_m^n(t)$ converge to their stationary values despite the effect of all the wrong decisions made in the past. This holds since the mixing of the associated Markov chain is faster than the gradient steps, governed by the step size sequence $\eta(t)$. We then use Markov chain concentration inequalities [54] to show that the communication noise averages out over time, fluctuating around the true mean value with respect to $\{\pi_\gamma\}$. However, the noise never vanishes and is always significant, so only the average across time gets accurate.

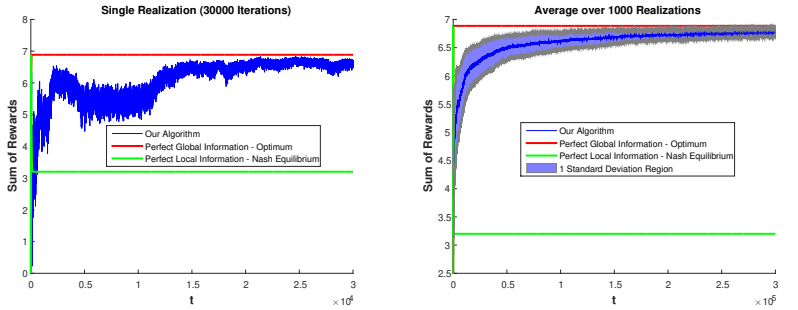
The crux of the analysis is the interplay between the gradient bias and noise and the communication bias and noise. Due to the decreasing $\delta(t)$, gradients estimated later on are more accurate than earlier gradients. Hence, player n will be biased towards improving the reward functions for which the gradients have been estimated later on. These are typically the gradients of nearby players in the network $G(t)$, and is still a problem even when $\{p_m^n(t)\}$ are estimated with high accuracy. To prevent that, we want the sampling radius $\delta(t) = O(t^{-q})$ to vanish slowly enough compared to the step size $\eta(t) = O(t^{-p})$. However, this leads to slower convergence due to smaller step sizes. Our analysis reveals that $p = \frac{q+3}{4}$ is the threshold after which the communication bias vanishes with time and the algorithm converges. This is formalized in Lemma 3 in Section 10 of the appendix.

Finally, equipped with Lemma 1, Lemma 2 and Lemma 3, the application of the Kushner-Clark Theorem is made possible, which is done in Section 11 of the appendix.

5 Numerical Example

We demonstrate the numerical behavior of Algorithm 1 using a congestion game with $N = 8$ players and $R = 4$ resources. The action $\boldsymbol{x}_n \in [0,1]^R$ of player n is the amount of usage of each of the

R resources. The utility function is $u_n(\boldsymbol{x}) = \sum_{r=1}^{R} x_{n,r} - \frac{1}{20}\sum_{r=1}^{R} x_{n,r}\left(\sum_{m=1}^{N} x_{m,r}\right)^2$, where the first term models the value player n has for the resources (which we did not include in Example 2). We used the step size sequence $\eta(t) = \frac{0.2}{t^{0.9}}$ and the sampling radius sequence $\delta(t) = \frac{0.2}{t^{0.1}}$. The memory was $M = 6$. Players broadcasted one reward value per turn, uniformly at random. The communication graph G was constant and random for each realization, with an average degree of 4.25. It can be seen that Algorithm 1 converges to $\arg\max_{\boldsymbol{x}\in\mathcal{A}} \sum_{n=1}^{N} u_n(\boldsymbol{x})$. From early on, the performance is already much better than with no communication but with perfect local information. We also show a typical realization to demonstrate that convergence is stochastic and not monotone.



(a) Typical realization (b) Performance over 1000 realizations

Figure 2: Congestion game with $N = 8$ players and $R = 4$ resources

6 Conclusions

We have developed a novel algorithm for cooperative multi-player bandit optimization where the reward of each player is a function of all players' actions. In our algorithm, each player approximates the gradient of the sum of rewards with respect to its local action based only on bandit feedback. However, our algorithm does not require players to collect all the rewards in the sum before approximating the gradient, which would be infeasible. Instead, players approximate the gradient of the sum of rewards based only on a random small subset of rewards they obtained in the last M turns ("recently") by communicating with their neighbors on a time-varying graph. This introduces a new type of bias and noise to our stochastic gradients that was yet to be analyzed in the literature. Players running our algorithm learn to correct this communication bias with time by estimating the probabilities of getting the reward values of different players, and by correctly tuning the step size and sampling radius of the gradient approximation.

Our communication protocol is general and leaves the designer a significant degree of freedom to optimize over. From an engineering point of view, it is desirable to design a communication protocol that maximizes the "weakest link", which is $\min_{n,m} \pi_m^n$. This involves equalizing the frequencies at which messages of different players are relayed, helping more remote players to be well represented. Note that in Algorithm 1, one can choose $\varepsilon_0 = \min_{n,m} \pi_m^n$, and $\frac{1}{\varepsilon_0}$ appears in many of our bounds, implying that maximizing ε_0 will minimize the convergence time.

Our algorithm is robust to imperfections such as a time-varying communication graph and delays, but also to asynchronous players and noisy rewards. To keep the presentation simple, we included only some of these imperfections and left the rest for future work. Moreover, since our algorithm is a very noisy instance of stochastic gradient ascent, it is likely that it can be shown to avoid saddle points of the sum of rewards function even if the objective is non-concave (see [49, 50]).

Our work shows that converging to an optimal action profile (or to the set of KKT stationary points in the non-concave case) is possible even in our general multi-player bandit optimization scenario by only communicating few reward values each turn. This paves the road to study convergence time or communication overhead guarantees, possibly inspiring a new line of multi-player bandit optimization algorithms designed to optimize these measures. This can be formalized using regret or the finite sample complexity of "best-action profile identification".

7 Broader Impact

This work is mainly theoretical in nature, introducing a new framework for distributed learning algorithms where each agent affects all of its peers through its actions. Therefore there are no specific ethical considerations relevant to this work. From a broader perspective, the work contributes to the growing field of multi-agent learning. Multi-agent learning can be thought of as the next wave of machine learning, where the isolated black box machines will start interacting and learning from each other to form a large machine learning network. This shift involves automating more decision making processes, that will interact between themselves for our benefit. This, however, does not come without some application-specific ethical issues and concerns, as is already being discussed today for autonomous vehicles, that is a special case of multi-agent learning.

Acknowledgments and Disclosure of Funding

This research was supported by the Koret Foundation grant for Smart Cities and Digital Living.

References

[1] A. D. Flaxman, A. T. Kalai, A. T. Kalai, and H. B. McMahan, "Online convex optimization in the bandit setting: gradient descent without a gradient," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 385–394, Society for Industrial and Applied Mathematics, 2005.

[2] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 928–936, 2003.

[3] Y.-H. Chang, T. Ho, and L. P. Kaelbling, "All learning is local: Multi-agent learning in global reward games," in *Advances in neural information processing systems*, pp. 807–814, 2004.

[4] C. C. Moallemi and B. V. Roy, "Distributed optimization in adaptive networks," in *Advances in Neural Information Processing Systems*, pp. 887–894, 2004.

[5] T. Tatarenko, "Stochastic learning in multi-agent optimization: Communication and payoff-based approaches," *Automatica*, vol. 99, pp. 1–12, 2019.

[6] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team markov games," in *Advances in neural information processing systems*, pp. 1603–1610, 2003.

[7] X. Wang and T. Sandholm, "Learning near-pareto-optimal conventions in polynomial time," in *Advances in Neural Information Processing Systems*, pp. 863–870, 2004.

[8] J. R. Marden and A. Wierman, "Distributed welfare games," *Operations Research*, vol. 61, no. 1, pp. 155–168, 2013.

[9] N. Li and J. R. Marden, "Designing games for distributed optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 230–242, 2013.

[10] T. Alpcan and L. Pavel, "Nash equilibrium design and optimization," in *2009 International Conference on Game Theory for Networks*, pp. 164–170, IEEE, 2009.

[11] J. R. Marden, H. P. Young, and L. Y. Pao, "Achieving pareto optimality through distributed learning," *SIAM Journal on Control and Optimization*, vol. 52, no. 5, pp. 2753–2770, 2014.

[12] S. Kato, S. Tsugawa, K. Tokuda, T. Matsui, and H. Fujii, "Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications," *IEEE Transactions on intelligent transportation systems*, vol. 3, no. 3, pp. 155–161, 2002.

[13] A. Agarwal, D. P. Foster, D. J. Hsu, S. M. Kakade, and A. Rakhlin, "Stochastic convex optimization with bandit feedback," in *Advances in Neural Information Processing Systems*, pp. 1035–1043, 2011.

[14] P. Alatur, K. Y. Levy, and A. Krause, "Multi-player bandits: The adversarial case," *Journal of Machine Learning Research*, vol. 21, no. 77, pp. 1–23, 2020.

[15] I. Bistritz and A. Leshem, "Distributed multi-player bandits-a game of thrones approach," in *Advances in Neural Information Processing Systems*, pp. 7222–7232, 2018.

[16] I. Bistritz and A. Leshem, "Game of thrones: Fully distributed learning for multi-player bandits," *arXiv preprint arXiv:1810.11162*, 2018.

[17] I. Bistritz, T. Z. Baharav, A. Leshem, and N. Bambos, "My fair bandit: Distributed learning of max-min fairness with multi-player bandits," *arXiv preprint arXiv:2002.09808, to appear in International Conference on Machine Learning*, 2020.

[18] E. Boursier and V. Perchet, "SIC-MMAB: synchronisation involves communication in multiplayer multi-armed bandits," in *Advances in Neural Information Processing Systems*, pp. 12048–12057, 2019.

[19] J. Rosenski, O. Shamir, and L. Szlak, "Multi-player bandits–a musical chairs approach," in *International Conference on Machine Learning*, pp. 155–163, 2016.

[20] Q. Zhao, "Multi-armed bandits: Theory and applications to online learning in networks," *Synthesis Lectures on Communication Networks*, vol. 12, no. 1, pp. 1–165, 2019.

[21] D. Kalathil, N. Nayyar, and R. Jain, "Decentralized learning for multiplayer multiarmed bandits," *IEEE Transactions on Information Theory*, vol. 60, no. 4, pp. 2331–2345, 2014.

[22] K. Liu and Q. Zhao, "Distributed learning in multi-armed bandit with multiple players," *IEEE Transactions on Signal Processing*, vol. 58, no. 11, pp. 5667–5681, 2010.

[23] A. Magesh and V. V. Veeravalli, "Multi-user mabs with user dependent rewards for uncoordinated spectrum access," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 969–972, IEEE, 2019.

[24] A. Mehrabian, E. Boursier, E. Kaufmann, and V. Perchet, "A practical algorithm for multiplayer bandits when arm means vary among players," in *International Conference on Artificial Intelligence and Statistics*, pp. 1211–1221, PMLR, 2020.

[25] P.-A. Wang, A. Proutiere, K. Ariu, Y. Jedra, and A. Russo, "Optimal algorithms for multiplayer multi-armed bandits," in *International Conference on Artificial Intelligence and Statistics*, pp. 4120–4129, PMLR, 2020.

[26] D. Hajinezhad, M. Hong, and A. Garcia, "Zone: Zeroth-order nonconvex multiagent optimization over networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 3995–4010, 2019.

[27] Y. Tang and N. Li, "Distributed zero-order algorithms for nonconvex multi-agent optimization," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 781–786, IEEE, 2019.

[28] D. Yuan and D. W. Ho, "Randomized gradient-free method for multiagent optimization over time-varying networks," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 6, pp. 1342–1347, 2015.

[29] A. K. Sahu, D. Jakovetic, D. Bajovic, and S. Kar, "Distributed zeroth order optimization over random networks: A kiefer-wolfowitz stochastic approximation approach," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 4951–4958, IEEE, 2018.

[30] Z. Yu, D. W. Ho, and D. Yuan, "Distributed randomized gradient-free mirror descent algorithm for constrained optimization," *arXiv preprint arXiv:1903.04157*, 2019.

[31] M. Bravo, D. Leslie, and P. Mertikopoulos, "Bandit learning in concave n-person games," in *Advances in Neural Information Processing Systems*, pp. 5666–5676, 2018.

[32] J. C. Spall *et al.*, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE transactions on automatic control*, vol. 37, no. 3, pp. 332–341, 1992.

[33] Z. Zhou, P. Mertikopoulos, A. L. Moustakas, N. Bambos, and P. Glynn, "Mirror descent learning in continuous games," in *CDC'17: Proceedings of the 55th IEEE Annual Conference on Decision and Control*, 2017.

[34] Y.-H. Chang and L. P. Kaelbling, "Playing is believing: The role of beliefs in multi-agent learning," in *Advances in Neural Information Processing Systems*, pp. 1483–1490, 2002.

[35] J. Hartline, V. Syrgkanis, and E. Tardos, "No-regret learning in bayesian games," in *Advances in Neural Information Processing Systems*, pp. 3061–3069, 2015.

[36] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Advances in neural information processing systems*, pp. 1729–1736, 2008.

[37] X. Xu and Q. Zhao, "Distributed no-regret learning in multiagent systems: Challenges and recent developments," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 84–91, 2020.

[38] M. Bowling, "Convergence and no-regret in multiagent learning," in *Advances in neural information processing systems*, pp. 209–216, 2005.

[39] P. Mertikopoulos and Z. Zhou, "Learning in games with continuous action sets and unknown payoff functions," *Mathematical Programming*, pp. 1–43, 2016.

[40] A. Menon and J. S. Baras, "A distributed learning algorithm with bit-valued communications for multi-agent welfare optimization," in *52nd IEEE Conference on Decision and Control*, pp. 2406–2411, IEEE, 2013.

[41] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[42] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic control*, vol. 57, no. 3, pp. 592–606, 2012.

[43] T. Alpcan, L. Pavel, and N. Stefanovic, "A control theoretic approach to noncooperative game design," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 8575–8580, IEEE, 2009.

[44] P. D. Lax and M. S. Terrell, *Multivariable Calculus with Applications*. Springer, 2017.

[45] B. Touri, *Product of random stochastic matrices and distributed averaging*. Springer Science & Business Media, 2012.

[46] Y. Bramoullé and R. Kranton, "Public goods in networks," *Journal of Economic Theory*, vol. 135, no. 1, pp. 478–494, 2007.

[47] E. Mazumdar, L. J. Ratliff, and S. S. Sastry, "On gradient-based learning in continuous games," *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 1, pp. 103–131, 2020.

[48] L. J. Ratliff, S. A. Burden, and S. S. Sastry, "On the characterization of local nash equilibria in continuous games," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2301–2307, 2016.

[49] R. Pemantle *et al.*, "Nonconvergence to unstable points in urn models and stochastic approximations," *The Annals of Probability*, vol. 18, no. 2, pp. 698–712, 1990.

[50] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent only converges to minimizers," in *Conference on learning theory*, pp. 1246–1257, 2016.

[51] D. Soudry and Y. Carmon, "No bad local minima: Data independent training error guarantees for multilayer neural networks," *arXiv preprint arXiv:1605.08361*, 2016.

[52] H. J. Kushner and D. S. Clark, *Stochastic approximation methods for constrained and unconstrained systems*, vol. 26. Springer Science & Business Media, 2012.

[53] S. Bhatnagar, H. Prasad, and L. Prashanth, *Stochastic recursive algorithms for optimization: simultaneous perturbation methods*, vol. 434. Springer, 2012.

[54] K.-M. Chung, H. Lam, Z. Liu, and M. Mitzenmacher, "Chernoff-hoeffding bounds for markov chains: Generalized and simplified," *arXiv preprint arXiv:1201.0559*, 2012.

[55] F. Garin, D. Varagnolo, and K. H. Johansson, "Distributed estimation of diameter, radius and eccentricities in anonymous networks," *IFAC Proceedings Volumes*, vol. 45, no. 26, pp. 13–18, 2012.

[56] F. Chung and L. Lu, "The diameter of sparse random graphs," *Advances in Applied Mathematics*, vol. 26, no. 4, pp. 257–279, 2001.

[57] P. Billingsley, *Probability and measure*. John Wiley & Sons, 2008.

[58] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.