Learning Maximin Strategies with Best Arm Identification Techniques*

Alberto Marchesi
Politecnico di Milano
Milan, Italy
alberto.marchesi@polimi.it

Francesco Trovò
Politecnico di Milano
Milan, Italy
francesco1.trovo@polimi.it

Nicola Gatti
Politecnico di Milano
Milan, Italy
nicola.gatti@polimi.it

ABSTRACT

We tackle the problem of learning equilibria in *simulation-based* games. In such games, the players' utility functions cannot be described analytically, as they are given through a black-box simulator that can be queried to obtain noisy estimates of the utilities. This is the case in many real-world games in which a complete description of the elements involved is not available upfront, such as complex military settings and online auctions. In these situations, one usually needs to run costly simulation processes to get an accurate estimate of the game outcome. As a result, solving these games begets the challenge of designing learning algorithms that can find (approximate) equilibria with high confidence, using as few simulator queries as possible. Moreover, since running the simulator during the game is unfeasible, the algorithms must first perform a pure exploration learning phase and, then, use the (approximate) equilibrium learned this way to play the game. In this work, we focus on two-player zero-sum games with *infinite strategy spaces*. Drawing from the best arm identification literature, we design two algorithms with theoretical guarantees to learn *maximin* strategies in these games. The first one works in the *fixed-confidence* setting, guaranteeing the desired confidence level while minimizing the number of queries. Instead, the second algorithm fits the *fixed-budget* setting, maximizing the confidence without exceeding the given maximum number of queries. First, we formally prove δ-PAC theoretical guarantees for our algorithms under some regularity assumptions, which are encoded by letting the utility functions be drawn from a Gaussian process. Then, we experimentally evaluate our techniques on a testbed made of randomly generated games and instances representing simple real-world security settings.

1 INTRODUCTION

Over the last two decades, game-theoretic models have received a growing interest from the AI community, as they allow to design artificial agents endowed with the ability of reasoning strategically in complex multi-agent settings. This surge of interest was driven by many successful applications of game theory to challenging real-world problems, such as building robust protection strategies in security domains [23], designing truthful auctions for web advertising [10], and solving (*i.e.*, finding the equilibria of) large zero-sum recreational games, *e.g.*, Go [19], different variants of Poker [5, 6], and Bridge [17].

Most of the game-theoretic studies in AI focus on models where a complete description of the game is available, *i.e.*, the players' utilities can be expressed analytically. This is the case of recreational games, which are commonly used as benchmarks for evaluating algorithms to compute equilibria in games [4]. However, in many real-world problems, the players' utilities may *not* be readily available, as they are the outcome of a complex process governed by unknown parameters. This is the case, *e.g.*, in complex military settings where a comprehensive description of the environment and the units involved is not available, and online auctions in which the platform owner does not have complete knowledge of the parties involved. These scenarios can be addressed with *simulation-based games* (SBGs) [28], where the players' utilities are expressed by means of a black-box simulator that, given some players' strategies, can be queried to obtain a noisy estimate of the utilities obtained when playing such strategies. These models beget new challenges in the design of algorithms to solve games: *(i)* they have to learn (approximate) equilibria by using only noisy observations of the utilities, and *(ii)* they should use as few queries as possible, since running the simulator is usually a costly operation. Additionally, using the simulator while playing the game is unfeasible, since the simulation process might be prohibitively time consuming, as it is the case, *e.g.*, in military settings where the units have to take prompt decisions when on the battlefield. Thus, the algorithms must first perform a pure exploration learning phase and, then, use the (approximate) equilibrium learned this way to play the game.

Despite the modeling power of SBGs, recent works studying such games are only sporadic, addressing specific settings such as, *e.g.*, symmetric games with a large number of players [21, 30], empirical mechanism design [25], and two-player zero-sum finite games [9] (see Section 2 for more details and additional related works). To the best of our knowledge, the majority of these works focus on the case in which each player has a finite number of strategies available. However, in most of the game settings in which simulations are involved, the players have an infinite number of choices available, *e.g.*, physical quantities, such as angle of movement and velocity of units on a military field, bids in auctions, and trajectories in robot planning. Dealing with infinite strategies leads to further challenges in the design of learning algorithms, since, being a complete exploration of the strategy space unfeasible, providing strong theoretical guarantees is, in general, a non-trivial task.

1.1 Original Contributions

We study the problem of learning equilibria in *two-player zero-sum* SBGs with *infinite strategy spaces*, providing theoretical guarantees. Specifically, we focus on *maximin* strategies for the first player, *i.e.*, those maximizing her utility under the assumption that the second

*This work has been accepted for publication at AAMAS 2020.

GAIW: Games, Agents and Incentives Workshops at AAMAS 2020, May 2020, Auckland, New Zealand.

player acts so as to minimize it, after observing the first player's course of play. For instance, this is the case in security games where a terrestrial counter-air defensive unit has to shoot an heat-seeking missile to a moving target that represents an approaching enemy airplane, which, after the attack has started, can respond to it by deploying an obfuscating flare with the intent of deflecting the missile trajectory. When dealing with infinite strategy spaces, some regularity assumptions on the players' utilities are in order, since, otherwise, one cannot design learning algorithms with provable theoretical guarantees. In this work, we encode our regularity assumptions on the utility function by modeling it as a sample from a *Gaussian process* (GP) [32]. We design two algorithms able to learn (approximate) maximin strategies in two-player zero-sum SBGs with infinite strategy spaces, drawing from techniques used in the best arm identification literature. The first algorithm we propose, called M-GP-LUCB, is for the *fixed-confidence* setting, where the objective is to find an (approximate) maximin strategy with a given (high) confidence, using as few simulator queries as possible. Instead, the second algorithm, called SE-GP, is for the *fixed-budget* setting, in which a maximum number of queries is given in advance, and the task is to return an (approximate) maximin strategy with confidence as high as possible. First, we prove δ-PAC (*i.e.*, *probably approximately correct*) theoretical guarantees for our algorithms in the easiest setting in which the strategy spaces are finite. Then, we show how these results can be generalized to SBGs with infinite strategy spaces by leveraging the GP assumption. Finally, we experimentally eventuate our algorithms on a testbed made of randomly generated games and instances based on the missile-airplane security game described above. For SBGs with finite strategy spaces, we also compare our algorithms with the M-LUCB algorithm introduced by [9] (the current state-of-the-art method for learning maximin strategies in two-player zero-sum finite games), showing that our methods dramatically outperform it. [1]

2 RELATED WORKS

Over the last years, the problem of learning approximate equilibria in SBGs received considerable attention from the AI community. In this section, we survey the main state-of-the-art works on the problem of learning equilibria in SBGs, highlighting which are the crucial differences with our work. Let us remark that the majority of these works focus on SBGs with finite strategy spaces, while, to the best of our knowledge, ours provides the first learning algorithms with theoretical guarantees for SBGs with infinite strategy spaces.

The first computational studies on SBGs date back to the work of Vorobeychik et al. [29], who focus on n-player general-sum games, experimentally evaluating standard regression techniques to learn *Nash equilibria* (NEs) in such games. Their approach is to first learn the players' payoff functions and then compute an NE in the game learned this way. Gatti and Restelli [11] extend this work to sequential games. Given the nature of regression techniques, this approach also works for SBGs with infinite strategy spaces. However, the proposed methodology does not allow us to derive theoretical guarantees on the approximation quality of the obtained solutions, and it does not adopt any principled rule for choosing the next query to be performed. In contrast, our algorithms are δ-PAC, and, by exploiting techniques from the best arm identification literature, they also perform queries intelligently, allowing for a great reduction in the used number of queries.

A similar approach, which is still based on learning payoff functions using regression, is adopted by some recent works studying finite SBGs with many symmetric players [21, 30]. Their goal is to exploit the symmetries so as to learn symmetric NEs in large games efficiently. Wiedenbeck et al. [30] focus on GP regression, since, as they show experimentally, it leads to better performances compared to other techniques. Subsequently, Sokota et al. [21] provide an advancement over the previous work, using neural networks to approximate the utility function (instead of GPs) and providing a way to guide sampling so as to focus it on the neighborhood of candidate equilibrium points. These works significantly depart from ours, since *(i)* they aim at finding symmetric NEs in large SBGs with many symmetric players, *(ii)* they are restricted to games with finite strategy spaces, and (*iii)* they do not provide any theoretical guarantees on the quality of the obtained solutions.

Recently, some works proposed learning algorithms for finite SBGs, relying on the PAC framework to prove theoretical guarantees [25, 26, 26, 33]. Specifically, Viqueira et al. [26] and Wright and Wellman [33] focus on learning NEs in n-player general-sum games. However, their results are limited to the case of finite strategy spaces and cannot be easily generalized to settings involving infinite strategy spaces, as they do not introduce any regularity assumption on the players' utility functions. Moreover, the querying algorithms they propose are based on a global exploration of the strategy profiles of the game, without relying on specific selection rules, except for the elimination of sub-optimal strategies. In contrast, our algorithms exploit best arm identification techniques, and, thus, they employ principled selection rules that allow to focus queries on the most promising strategy profiles.

It is also worth pointing out some works that, while being not directly concerned with SBGs, address related problems. Recently, a growing attention has been devoted to no-regret learning algorithms in games with bandit feedback [3, 8, 12]. The methods developed in this framework are significantly different from ours, as they fit the classical multi-armed bandit scenario where the objective is to minimize the cumulative regret. Instead, we adopt the best arm identification perspective, where the goal is to identify an optimal arm with high confidence. Thus, our querying algorithms might achieve large regret during the learning process, since they are focused on a pure exploration task in which exploitation is not a concern. Moreover, no-regret learning algorithms require strong assumptions to converge to equilibria in games with bandit feedback (such as, *e.g.*, concavity of the players' utility functions [3]). In contrast, our theoretical guarantees do not need any explicit requirement on the utilities (except for a reasonable degree of smoothness, encoded by the GP assumption), and, thus, they also hold when the players' utility functions exhibit a complex (*e.g.*, non-concave) dependence on the players' strategies.

There are also other related problems not directly connected with SBGs that are worth citing, such as, *e.g.*, meta-game analysis [24], learning unknown game parameters or players' rationality models

[1] The complete proofs of our theoretical results are available in the extended version [15].

by observing played actions [13, 14], combining supervised learning techniques with decision-making in uncertain optimization models [31], and online learning in games [2].

3 PRELIMINARIES

A *two-player zero-sum game with infinite strategy spaces* is a tuple $\Gamma = (\mathcal{X}, \mathcal{Y}, u)$, where $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}^d$ are compact and convex sets of strategies available to the first and the second player, respectively, while $u : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ is a function defining the utility for the first player. [2] Since the game is zero-sum, the second player's utility is given by $-u$. A *two-player zero-sum game with finite strategy spaces* is defined analogously, with $\mathcal{X}$ and $\mathcal{Y}$ being finite sets, *i.e.*, $\mathcal{X} := \{x^1, \dots, x^n\}$ and $\mathcal{Y} := \{y^1, \dots, y^m\}$, with $n > 1$ and $m > 1$ denoting the finite numbers of strategies available to the first and the second player, respectively. For the ease of notation, letting $\Pi := \mathcal{X} \times \mathcal{Y}$, we denote with $\boldsymbol{\pi} := (x, y) \in \Pi$ a *strategy profile*, *i.e.*, a tuple specifying a strategy $x \in \mathcal{X}$ for the first player and a strategy $y \in \mathcal{Y}$ for the second player.

In this work, we are concerned with the computation of *maximin* strategies, adopting the perspective of the first player. In words, we seek for a first player's strategy that maximizes her utility, assuming a worst-case opponent that acts so as to minimize it. Since the game is zero sum, we can assume that the second player decides how to play after observing the first player's move, and, thus, playing a maximin strategy is the best choice for the first player. [3] Formally, given a first player's strategy $x \in \mathcal{X}$, we denote with $y^*(x) \in \arg\min_{y \in \mathcal{Y}} u(x, y)$ a second player's best response to x. Then, $x^* \in \mathcal{X}$ is a maximin strategy for the first player if $x^* \in \arg\max_{x \in \mathcal{X}} u(x, y^*(x))$, with $\boldsymbol{\pi}^* := (x^*, y^*(x^*))$ denoting its corresponding maximin strategy profile. [4]

3.1 Simulation-Based Games

In SBGs, the utility function u is not readily available, but it is rather specified by an exogenous simulator that provides noisy point estimates of it. As a result, in SBGs, one cannot explicitly compute a maximin strategy, and, thus, the problem is to learn one by sequentially querying the simulator. At each round t, the simulator is given a strategy profile $\boldsymbol{\pi}_t \in \Pi$ and returns an estimated utility $\tilde{u}_t := u(\boldsymbol{\pi}_t) + e_t$, where $e_t \sim \mathcal{N}(0, \lambda)$ is i.i.d. Gaussian noise. The goal is to find a good approximation (see Equation (1)) of a maximin strategy $x^* \in \mathcal{X}$ as rapidly as possible, *i.e.*, limiting the number of queries to the simulator. To achieve this, we follow the approach of Garivier et al. [9] and propose some *dynamic querying algorithms* (see Algorithm 1, where $\textsc{Sim}(\boldsymbol{\pi})$ represents a simulator query for $\boldsymbol{\pi} \in \Pi$), which are generally characterized by the following components:

- a querying rule that indicates which strategy profile $\boldsymbol{\pi}_t \in \Pi$ is sent as input to the simulator at each round t;
- a stopping rule that determines the round T after which the algorithm terminates its execution;
- a final guess $\overline{\boldsymbol{\pi}} := (\bar{x}, \bar{y}) \in \Pi$ for the (true) maximin strategy profile $\boldsymbol{\pi}^*$ of the game.

Algorithm 1 Dynamic Querying Algorithm

```
t ← 1
do
    Select a strategy profile π_t ∈ Π according
        to the querying rule
    Get estimated utility ũ_t ← Sim(π_t)
    Update the algorithm parameters using ũ_t
    t ← t + 1
while stopping condition is not met
return final guess π̄ = (x̄, ȳ) for the maximin profile
```

Given a desired approximation $\epsilon \geq 0$, the objective of the algorithm is to find an ϵ-maximin strategy with high accuracy, using as few queries as possible to the simulator. Formally, given $\delta \in (0, 1)$, our goal is to design algorithms that are δ-PAC, *i.e.*, they satisfy the following condition:

$$\forall u \quad \mathbb{P}\Big(\big| u(\boldsymbol{\pi}^*) - u(\bar{x}, y^*(\bar{x})) \big| \leq \epsilon \Big) \geq 1 - \delta, \tag{1}$$

while keeping the number of rounds T as small as possible. This is known as the *fixed-confidence* setting (see Section 4). An alternative is to consider the *fixed-budget* case, where the maximum number of rounds T is given in advance, and the goal is to minimize the probability δ that $\bar{x}$ is not an ϵ-maximin strategy (see Section 5). Notice that, for SBGs with finite strategy spaces, the δ-PAC property in Equation (1) can only require $u(\boldsymbol{\pi}^*) - u(\bar{x}, y^*(\bar{x})) \leq \epsilon$, since it is always the case that $u(\boldsymbol{\pi}^*) > u(\bar{x}, y^*(\bar{x}))$. [5]

3.2 Gaussian Processes

To design δ-PAC algorithms working with SBGs having infinite strategy spaces, we first need to introduce some regularity assumptions on the utility functions u. In this work, we model the utility as a sample from a GP, which is a collection of dependent random variables, one for each action profile $\boldsymbol{\pi} \in \Pi$, every finite subset of which is multivariate Gaussian distributed [32]. A $\text{GP}(\mu(\boldsymbol{\pi}), k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ is fully specified by its *mean* function $\mu : \Pi \mapsto \mathbb{R}$, with $\mu(\boldsymbol{\pi}) := \mathbb{E}[u(\boldsymbol{\pi})]$, and its *covariance* (or *kernel*) function $k : \Pi \times \Pi \mapsto \mathbb{R}$, with $k(\boldsymbol{\pi}, \boldsymbol{\pi}') := \mathbb{E}[(u(\boldsymbol{\pi}) - \mu(\boldsymbol{\pi}))(u(\boldsymbol{\pi}') - \mu(\boldsymbol{\pi}'))]$. W.l.o.g., we assume that $\mu \equiv \mathbf{0}$ and the variance is bounded, *i.e.*, $k(\boldsymbol{\pi}, \boldsymbol{\pi}) := \sigma^2 \leq 1$ for every $\boldsymbol{\pi} \in \Pi$. Note that the GP assumption guarantees that the utility function u has a certain degree of smoothness, without relying on rigid parametric assumptions, such as linearity. Intuitively, the kernel function k determines the correlation of the utility values across the space of strategy profiles Π, thus encoding the smoothness properties of the utility functions u sampled from $\text{GP}(\mu(\boldsymbol{\pi}), k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ (for some examples of commonly used kernels, see Section 7).

We also need GPs in our algorithms, as they use $\text{GP}(\mathbf{0}, k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ as prior distribution over u. The major advantage of working with GPs is that they admit simple analytical formulas for the mean and covariance of the posterior distribution. These relations can

[2] For the ease of presentation, in the following we focus on the case in which $\mathcal{X} \subset [0, 1]$ and $\mathcal{Y} \subset [0, 1]$ are closed intervals. The generalization of our results to the case in which the strategy spaces are compact and convex subsets of $\mathbb{R}^d$ is straightforward.
[3] This assumption is in line with the classical Stackelberg model in which the second player (follower) gets to play after observing the strategy of the first one (leader) [27].
[4] Even though playing a maximin strategy may not be the optimal choice for the first player if the players are assumed to play simultaneously, this is the case if we require additional (mild) technical assumptions guaranteeing that $\boldsymbol{\pi}^*$ is an equilibrium point of the game; see [20] for additional details.
[5] This is in line with the definition provided by Garivier et al. [9].

be easily expressed using matrix notation, as follows. Let $\tilde{\mathbf{u}}_t \coloneqq [\tilde{u}_1, \dots, \tilde{u}_t]^\top$ be the vector of utility values observed up to round t, obtained by querying the simulator on the strategy profiles $\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_t$. Then, the posterior distribution over u is still a GP, with mean $\mu_t(\boldsymbol{\pi})$, covariance $k_t(\boldsymbol{\pi}, \boldsymbol{\pi}')$, and variance $\sigma_t^2(\boldsymbol{\pi})$, which are defined as follows:

$$\mu_t(\boldsymbol{\pi}) \coloneqq \mathbf{k}_t(\boldsymbol{\pi})^\top (K_t + \lambda I)^{-1} \tilde{\mathbf{u}}_t, \tag{2}$$

$$k_t(\boldsymbol{\pi}, \boldsymbol{\pi}') \coloneqq k(\boldsymbol{\pi}, \boldsymbol{\pi}') - \mathbf{k}_t(\boldsymbol{\pi})^\top (K_t + \lambda I)^{-1} \mathbf{k}_t(\boldsymbol{\pi}'), \tag{3}$$

$$\sigma_t^2(\boldsymbol{\pi}) \coloneqq k_t(\boldsymbol{\pi}, \boldsymbol{\pi}), \tag{4}$$

where $\mathbf{k}_t(\boldsymbol{\pi}) \coloneqq [k(\boldsymbol{\pi}, \boldsymbol{\pi}_1), \dots, k(\boldsymbol{\pi}, \boldsymbol{\pi}_t)]^\top$ and K_t is the positive definite $t \times t$ kernel matrix, whose (i, j)-th entry is $k(\boldsymbol{\pi}_i, \boldsymbol{\pi}_j)$. The posterior parameters update formulas can also be expressed recursively, thus avoiding costly matrix inversions, as shown in [7]. Letting $\boldsymbol{\pi}_t$ and $\tilde{u}_t$ be, respectively, the queried strategy profile and the observed utility at round t, we can write:

$$\mu_t(\boldsymbol{\pi}) \leftarrow \mu_{t-1}(\boldsymbol{\pi}) + \frac{k_{t-1}(\boldsymbol{\pi}, \boldsymbol{\pi}_t)}{\lambda + \sigma_{t-1}^2(\boldsymbol{\pi}_t)} (\tilde{u}_t - \mu_{t-1}(\boldsymbol{\pi}_t)), \tag{5}$$

$$k_t(\boldsymbol{\pi}, \boldsymbol{\pi}') \leftarrow k_t(\boldsymbol{\pi}, \boldsymbol{\pi}') - \frac{k_{t-1}(\boldsymbol{\pi}, \boldsymbol{\pi}_t) k_{t-1}(\boldsymbol{\pi}_t, \boldsymbol{\pi}')}{\lambda + \sigma_{t-1}^2(\boldsymbol{\pi}_t)}, \tag{6}$$

$$\sigma_t^2(\boldsymbol{\pi}) \leftarrow \sigma_{t-1}^2(\boldsymbol{\pi}) - \frac{k_{t-1}^2(\boldsymbol{\pi}, \boldsymbol{\pi}_t)}{\lambda + \sigma_{t-1}^2(\boldsymbol{\pi}_t)}. \tag{7}$$

Clearly, at the beginning of the algorithms, the estimates are given by the GP prior $\mathrm{GP}(\mu(\boldsymbol{\pi}), k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$, *i.e.*, formally, $\mu_0(\boldsymbol{\pi}) \coloneqq \mu(\boldsymbol{\pi})$, $k_0(\boldsymbol{\pi}, \boldsymbol{\pi}') \coloneqq k(\boldsymbol{\pi}, \boldsymbol{\pi}')$, and $\sigma_0^2(\boldsymbol{\pi}) \coloneqq k(\boldsymbol{\pi}, \boldsymbol{\pi}) = \sigma^2$.

4 FIXED-CONFIDENCE SETTING

In this section and the following one (Section 5), we present our learning algorithms for the easiest setting of SBGs with finite strategy spaces. Then, in Section 6, we show how they can be extended to SBGs with infinite strategy spaces.

For the fixed-confidence setting, we propose a δ-PAC dynamic querying algorithm (called M-GP-LUCB, see Algorithm 2) based on the M-LUCB approach introduced by Garivier et al. [9] and provide a bound on the number of rounds T_δ it requires, as a function of the confidence level δ. While our algorithm shares the same structure as M-LUCB, it uses confidence bounds relying on the GP assumption, and, thus, different proofs are needed to show its δ-PAC properties. As shown in Section 6, our algorithm and its theoretical guarantees have the crucial advantage of being easily generalizable to SBGs with infinite strategy spaces.

For every strategy profile $\boldsymbol{\pi} \in \Pi$, the algorithm keeps track of a confidence interval $[L_t(\boldsymbol{\pi}), U_t(\boldsymbol{\pi})]$ on $u(\boldsymbol{\pi})$ built using the utility values $\tilde{u}_t$ observed from the simulator up to round t. Using $\mathrm{GP}(\mathbf{0}, k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ as prior distribution over the utility function u, the lower bounds of the intervals are defined as $L_t(\boldsymbol{\pi}) \coloneqq \mu_t(\boldsymbol{\pi}) - \sqrt{b_t}\sigma_t(\boldsymbol{\pi})$ and the upper bounds as $U_t(\boldsymbol{\pi}) \coloneqq \mu_t(\boldsymbol{\pi}) + \sqrt{b_t}\sigma_t(\boldsymbol{\pi})$, where μ_t and σ_t^2 are the mean and the variance of the posterior distribution computed with observations up to round t (see Equations (2)–(4)), while b_t is an exploration term that depends from the context (see Theorem 4.1).

At the end of every even round t, the algorithm selects the strategy profiles to give as inputs to the simulator during the next two rounds $t+1$ and $t+2$. For every $x \in \mathcal{X}$, let

$$\gamma_t(x) \coloneqq \operatorname*{argmin}_{y \in \mathcal{Y}} L_t(x, y)$$

be the second player's best response to x computed using the lower bounds L_t. Moreover, let

$$\bar{x}_t \coloneqq \operatorname*{argmax}_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \mu_t(x, y)$$

be the maximin strategy computed using the posterior mean μ_t. Then, in the following two rounds, the algorithm selects the strategy profiles $\boldsymbol{\pi}_{t+1}$ and $\boldsymbol{\pi}_{t+2}$, defined as follows:

$$\boldsymbol{\pi}_{t+1} \coloneqq (\bar{x}_t, \gamma_t(\bar{x}_t)) \tag{8}$$

$$\boldsymbol{\pi}_{t+2} \coloneqq \operatorname*{argmax}_{\boldsymbol{\pi} \in \{(x, \gamma_t(x))\}_{x \neq \bar{x}_t}} U_t(\boldsymbol{\pi}). \tag{9}$$

This choice is made so as to advance the algorithm towards its termination. In particular, the M-GP-LUCB algorithm stops when, according to the confidence intervals, the strategy $\bar{x}_t$ is probably approximately better than all the others, *i.e.*, when it holds $L_t(\boldsymbol{\pi}_{t+1}) > U_t(\boldsymbol{\pi}_{t+2}) - \epsilon$. Intuitively, $\boldsymbol{\pi}_{t+1}$ represents the best candidate for being a maximin strategy profile, while $\boldsymbol{\pi}_{t+2}$ is the second-best candidate. Thus, the algorithm stops if $\boldsymbol{\pi}_{t+1}$ is better than $\boldsymbol{\pi}_{t+2}$ with sufficiently high confidence, *i.e.*, whenever the lower bound for the former is larger than the upper bound for the latter (up to an approximation of ϵ). The final strategy profile recommended by the algorithm is $\overline{\boldsymbol{\pi}} \coloneqq (\bar{x}_t, \gamma_t(\bar{x}_t))$.

Algorithm 2 M-GP-LUCB(ϵ, δ)

Initialize $t \leftarrow 0$, $\mu_0(\boldsymbol{\pi}) \leftarrow \mathbf{0}$, $k_0(\boldsymbol{\pi}, \boldsymbol{\pi}') \leftarrow k(\boldsymbol{\pi}, \boldsymbol{\pi}')$
do
  Select $\boldsymbol{\pi}_{t+1}$ and $\boldsymbol{\pi}_{t+2}$ using Eqs. (8)–(9)
  $\tilde{u}_{t+1} \leftarrow \textsc{Sim}(\boldsymbol{\pi}_{t+1})$, $\tilde{u}_{t+2} \leftarrow \textsc{Sim}(\boldsymbol{\pi}_{t+2})$
  Compute $\mu_{t+2}(\boldsymbol{\pi})$ and $k_{t+2}(\boldsymbol{\pi}, \boldsymbol{\pi}')$ using observations $\tilde{u}_{t+1}$, $\tilde{u}_{t+2}$ and Eqs. (5)–(7)
  $t \leftarrow t + 2$
while $L_t(\boldsymbol{\pi}_{t+1}) \leq U_t(\boldsymbol{\pi}_{t+2}) - \epsilon$
return $\overline{\boldsymbol{\pi}} = (\bar{x}_t, \gamma_t(\bar{x}_t))$

The following theorem shows that M-GP-LUCB is δ-PAC and provides an upper bound on the number of rounds T_δ it requires. The analysis is performed for $\epsilon = 0$, *i.e.*, when $\overline{\boldsymbol{\pi}}$ is evaluated with respect to an exact maximin profile. [6] Note that the upper bound for T_δ depends on the utility-dependent term $H^*(u) \coloneqq \sum_{\boldsymbol{\pi} \in \Pi} c(\boldsymbol{\pi})$, where, for $\boldsymbol{\pi} = (x, y) \in \Pi$, $c(\boldsymbol{\pi})$ is defined as follows:

$$c(\boldsymbol{\pi}) \coloneqq \frac{1}{\max\left\{(\Delta^*)^2, \left(\frac{u(x^*, y^*(x^*)) + u(x^{**}, y^*(x^{**}))}{2} - u(x, y^*(x))\right)^2\right\}},$$

where, for the ease of writing, we let $\Delta^* \coloneqq u(\boldsymbol{\pi}) - u(x, y^*(x))$ and $x^{**} \in \operatorname{argmax}_{x \in \mathcal{X} \setminus \{x^*\}} u(x, y^*(x))$, *i.e.*, x^{**} is a first player's maximin strategy when x^* is removed from the available ones. This term has the same role as $H_1 \coloneqq \sum_{i \in \{1, \dots, P\}} \frac{1}{\Delta_{(i)}^2}$ used by Audibert et al. [1] in the best arm identification setting, where, denoting as $\boldsymbol{\pi}^i$ the i-th strategy profile in Π, we let $\Delta_{(i)} \coloneqq |u(\boldsymbol{\pi}^*) - u(\boldsymbol{\pi}^i)|$, with

[6] Assuming $\epsilon = 0$ also requires the additional w.l.o.g. assumption that the utility value of an exact maximin strategy and that one of a second-best maximin strategy are different.

$\Delta_{(1)} \leq \Delta_{(2)} \leq \ldots \leq \Delta_{(P)}$. Intuitively, $H^*(u)$ and H_1 characterize the hardness of the problem instances by determining the amount of rounds required to identify the maximin profile and the best arm, respectively.

Theorem 4.1. *Using a generic nondecreasing exploration term $b_t > 0$, the M-GP-LUCB algorithm stops its execution after at most T_δ rounds, where:*

$$T_\delta \leq \inf \left\{ t \in \mathbb{N} \ : \ 8\, H^*(u)\, b_t\, \lambda - \frac{\lambda\, n\, m}{\sigma^2} < t \right\}. \tag{10}$$

Specifically, letting $b_t := 2 \log\left(\frac{n\, m\, \pi^2\, t^2}{6\delta}\right)$, the algorithm returns a maximin profile with confidence at least $1-\delta$, and:

$$T_\delta \leq 64\, H^*(u)\, \lambda \left(\log\left(64\, H^*(u)\, \lambda\, \pi \sqrt{\frac{n\, m}{6\delta}} \right) + \right.$$
$$\left. + 2 \log\left(\log\left(64\, H^*(u) \lambda \pi \sqrt{\frac{n\, m}{6\delta}} \right) \right) \right), \tag{11}$$

where we require that $64\, \lambda\, \pi \sqrt{\frac{n\, m}{6\delta}} > 4.85$.

Intuitively, from the result in Theorem 4.1, we can infer that the most influential terms on the number of rounds required to get a specific confidence level δ are $H^*(u)$ and the noise variance λ, which impact as multiplicative constants on T_δ. On the other hand, T_δ scales only logarithmically with the number of strategy profiles $|\Pi| = n\, m$, thus allowing the execution of the M-GP-LUCB algorithm also in settings where the players have a large number of strategies available.

5 FIXED-BUDGET SETTING

In the fixed-budget setting, the goal is to design δ-PAC algorithms that, given the maximum number of available rounds T (*i.e.*, the budget), find an ϵ-maximin strategy with confidence $1-\delta_T$ as large as possible. We propose a successive elimination algorithm (called GP-SE, see Algorithm 3), which is based on an analogous method proposed by Audibert et al. [1] for the best arm identification problem. The fundamental idea behind our GP-SE algorithm is a novel elimination rule, which is suitably defined for the problem of identifying maximin strategies.

The algorithm works by splitting the number of available rounds T into $P-1$ phases, where, for the ease of notation, we let $P := |\Pi| = n\, m$ be the number of players' strategy profiles. At the end of each phase, the algorithm excludes from the set of candidate solutions the strategy profile that has the lowest chance of being maximin. Specifically, letting Π_p be the set of the remaining strategy profiles during phase p, at the end of p, the algorithm dismisses the strategy profile $\boldsymbol{\pi}_p := (x_p, y_p) \in \Pi_p$, defined as follows:

$$(x_p, \cdot) := \operatorname*{argmin}_{\boldsymbol{\pi} \in \Pi_p} \mu_p(\boldsymbol{\pi}), \tag{12}$$

$$y_p := \operatorname*{argmax}_{y \in \mathcal{Y} : (x_p, y) \in \Pi_p} \mu_p(x_p, y), \tag{13}$$

where μ_p represents the mean of the posterior distribution computed at the end of phase p (see Equations (2)-(4)). Intuitively, the algorithm selects the first player's strategy x_p that is less likely to be a maximin one, together with the second player's strategy y_p that is the worst for her given x_p. At the end of the last phase, the (unique) remaining strategy profile $\overline{\boldsymbol{\pi}} = (\bar{x}, \bar{y})$ is recommended by the algorithm.

Algorithm 3 GP-SE(T)

1: Initialize $\Pi_1 \leftarrow \Pi$, $\mu_0(\boldsymbol{\pi}) \leftarrow \mathbf{0}$
2: **for** $p = 1, 2, \ldots, P-1$ **do**
3: For each $\boldsymbol{\pi} \in \Pi_p$, query Sim($\boldsymbol{\pi}$) for $T_p - T_{p-1}$ rounds
4: Compute $\mu_p(\boldsymbol{\pi})$ using observations
5: Select $\boldsymbol{\pi}_p$ according to Eqs. (12)–(13)
6: $\Pi_{p+1} \leftarrow \Pi_p \setminus \{\boldsymbol{\pi}_p\}$
7: **return** the unique element $\overline{\boldsymbol{\pi}}$ of Π_P

Following [1], the length of the phases have been carefully chosen so as to obtain an optimal (up to a logarithmic factor) convergence rate. Specifically, letting $\overline{\log}(P) := \frac{1}{2} + \sum_{i=2}^{P} \frac{1}{i}$, let us define $T_0 := 0$ and, for every phase $p \in \{1, \ldots, P-1\}$, let:

$$T_p := \left\lceil \frac{T - P}{\overline{\log}(P)(P + 1 - p)} \right\rceil. \tag{14}$$

Then, during each phase p, the algorithm selects every remaining strategy profile in Π_p for exactly $T_p - T_{p-1}$ rounds. Let us remark that the algorithm is guaranteed to do not exceed the number of available rounds T. Indeed, each $\boldsymbol{\pi}_p$ is selected for T_p rounds, while $\overline{\boldsymbol{\pi}}$ is chosen T_{P-1} times, and $\sum_{p=1}^{P-1} T_p + T_{P-1} \leq T$ holds by definition.

The following theorem provides an upper bound on the probability δ_T that the strategy profile $\overline{\boldsymbol{\pi}}$ recommended by the GP-SE algorithm is *not* ϵ-maximin, as a function of the number of rounds T. As for the fixed-confidence setting, our result holds for the case in which $\epsilon = 0$.

Theorem 5.1. *Letting T be the number of available rounds, the GP-SE algorithm returns a maximin strategy profile $\boldsymbol{\pi}^*$ with confidence at least $1-\delta_T$, where:*

$$\delta_T := 2P(n + m - 2) e^{-\frac{T-P}{8\lambda \overline{\log}(P) H_2}}, \tag{15}$$

and $H_2 := \max_{i \in \{1, \ldots, P\}} i\, \Delta_{(i)}^{-2}$.

As also argued by Audibert et al. [1], a successive elimination method provides two main advantages over a simple round robin querying strategy in which every strategy profile is queried for the same number of rounds. First, it provides a similar bound on δ_T with a better dependency on the parameters, and, second, it queries the maximin strategy profile a larger number of times, thus returning a better estimate of its expected utility.

6 SIMULATION-BASED GAMES WITH INFINITE STRATEGY SPACES

We are now ready to provide our main results on SBGs with infinite strategy spaces. In the first part of the section, we show how the δ-PAC algorithms proposed in Sections 4 and 5 for finite SBGs can be adapted to work with infinite strategy spaces while retaining some theoretical guarantees on the returned ϵ-maximin profiles. This requires to work with a (finite) discretized version of the original (infinite) SBGs, where the players' strategy spaces are approximated with grids made of equally spaced points. Then, in the second part

of the section, we provide some results for the situations in which one cannot work with this kind of discretization, and, instead, only a limited number of points is sampled from the players' strategy spaces. This might be the case when, *e.g.*, the dimensionality d of the players' strategy spaces is too high, or there are some constraints on the strategy profiles that can be queried. Clearly, in this setting, we cannot prove δ-PAC results, as the quality of the ϵ-maximin strategy profiles inevitable depends on how the points are selected.

Let us remark that our main results rely on our assumption that the utility function u is drawn from a GP, provided some mild technical requirements are satisfied (see Assumption 1).

6.1 δ-PAC Results for Evenly-Spaced Grids

The idea is to work with a discretization of the players' strategy spaces, each made of at least K_ϵ equally spaced points, where $\epsilon \geq 0$ is the desired approximation level. This induces a new (restricted) SBGs with finite strategy spaces, where techniques presented in the previous sections can be applied. In the following, for the ease of presentation, given an SBG with infinite strategy spaces Γ, we denote with $\Gamma(K)$ the finite SBG obtained when approximating the players' strategy spaces with K equally spaced points, *i.e.*, a game in which the players have $n = m = K$ strategies available and the utility value of each of the $n\,m$ strategy profiles is the same as that one of the corresponding strategy profile in Γ.

First, let us introduce the main technical requirement that we need for our results to hold.

Assumption 1 (Kernel Smoothness)**.** *A kernel $k(\boldsymbol{\pi}, \boldsymbol{\pi}')$ is said to be* smooth *over Π if, for each $L > 0$ and for some constants $a, b > 0$, the functions u drawn from $\mathrm{GP}(\mathbf{0}, k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ satisfy:*

$$\mathbb{P}\left(\sup_{\boldsymbol{\pi}\in\Pi}\left|\frac{\partial u}{\partial x}\right| > L\right) \leq a e^{-\frac{L^2}{b^2}}, \tag{16}$$

$$\mathbb{P}\left(\sup_{\boldsymbol{\pi}\in\Pi}\left|\frac{\partial u}{\partial y}\right| > L\right) \leq a e^{-\frac{L^2}{b^2}}. \tag{17}$$

This assumption is standard when using GPs in online optimization settings [22], and it is satisfied by many common kernel functions for specific values of a and b, such as the squared exponential kernel and the Matérn one with smoothness parameter $\nu > 2$ (see Section 7 for details on the definition of these kernel functions).

We are now ready to state our main result:

Theorem 6.1. *Assume that u is drawn from a $\mathrm{GP}(\mathbf{0}, k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ satisfying Assumption 1. Given $\epsilon > 0$ and $\delta \in (0, 2)$, let $\overline{\boldsymbol{\pi}} := (\bar{x}, \bar{y}) \in \Pi$ be a maximin strategy profile for a finite game $\Gamma(K)$ where K is at least $K_\epsilon := \left\lceil \frac{b}{2\epsilon}\sqrt{\log\left(\frac{4a}{\delta}\right)}\right\rceil + 1$. Then, the following holds:*

$$\mathbb{P}\Big(\left|u(\boldsymbol{\pi}^*) - u(\overline{\boldsymbol{\pi}})\right| \leq \epsilon\Big) \geq 1 - \frac{\delta}{2}. \tag{18}$$

The following two results rely on Theorem 6.1 to show that the M-GP-LUCB (Algorithm 2) and the GP-SE (Algorithm 3) algorithms can be employed to find, with high confidence, ϵ-maximin strategy profiles in SBGs with infinite strategy spaces. Let us remark that, while for SBGs with finite strategy spaces our theoretical analysis is performed for $\epsilon = 0$, in the case of infinite strategy spaces it is necessary to assume a nonzero approximation level ϵ.

Corollary 6.2. *Assume that u is drawn from a $\mathrm{GP}(\mathbf{0}, k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ satisfying Assumption 1. Given $\epsilon > 0$ and $\delta \in (0, 1)$, letting $b_t := 2\log\left(\frac{nm\pi^2 t^2}{3\delta}\right)$, the M-GP-LUCB algorithm applied to $\Gamma(K)$ with K at least $K_\epsilon := \left\lceil \frac{b}{2\epsilon}\sqrt{\log\left(\frac{4a}{\delta}\right)}\right\rceil + 1$ returns a strategy profile $\overline{\boldsymbol{\pi}} := (\bar{x}, \bar{y})$ such that $\mathbb{P}\left(|u(\boldsymbol{\pi}^*) - u(\bar{x}, y^*(\bar{x}))| \leq \epsilon\right) \geq 1 - \delta$. Moreover, the algorithm stops its execution after at most:*

$$T_{\delta,\epsilon} \leq 64\, H^*(u)\, \lambda \left[\log\left(64\, H^*(u)\, \lambda\, \pi\, K_\epsilon \sqrt{\frac{1}{3\delta}}\right) + \right.$$
$$\left. +2\, \log\left(\log\left(64\, H^*(u) \lambda\, \pi\, K_\epsilon \sqrt{\frac{1}{3\delta}}\right)\right)\right], \tag{19}$$

where we require that $64\,\lambda\,\pi\,K_\epsilon\sqrt{\frac{1}{3\delta}} > 4.85$.

Corollary 6.3. *Assume that u is drawn from a $GP(\mathbf{0}, k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ satisfying Assumption 1. Given $\epsilon > 0$ and $\delta \in (0, 1)$, letting T be the number of available rounds, the GP-SE algorithm applied to $\Gamma(K)$ with K at least $K_\epsilon := \left\lceil \frac{b}{2\epsilon}\sqrt{\log\left(\frac{4a}{\delta}\right)}\right\rceil + 1$ returns a profile $\overline{\boldsymbol{\pi}} := (\bar{x}, \bar{y})$ such that $\mathbb{P}\left(|u(\boldsymbol{\pi}^*) - u(\bar{x}, y^*(\bar{x}))| > \epsilon\right) < \delta_{T,\epsilon}$, where:*

$$\delta_{T,\epsilon} := 4K_\epsilon^2(K_\epsilon - 1)e^{-\frac{T-K_\epsilon^2}{8K_\epsilon^2\log(K_\epsilon^2)H_2}} + 2ae^{-\frac{b^2}{4\epsilon^2(K_\epsilon-1)^2}}. \tag{20}$$

In the result of Corollary 6.3, the discretization parameter K_ϵ depends on a confidence level δ that has to be chosen in advance. Another possibility is to try to minimize the overall confidence $\delta_{T,\epsilon}$ by appropriately tuning the parameter δ. Formally, a valid confidence level can be defined as follows:

$$\delta_{\text{opt}} := \inf\left\{\delta \in (0, 1) \;:\; \delta_{T,\epsilon}\right\}, \tag{21}$$

noticing that $\delta_{T,\epsilon}$ depends on δ also through the term K_ϵ. Unfortunately, this minimization problem does not admit a closed-form optimal solution. Nevertheless, we can compute an (approximate) optimal value for δ by employing numerical optimization methods [16].

6.2 Arbitrary Discretization

Whenever using an equally-spaced grid as a discretization scheme is unfeasible, the theoretical results based on Theorem 6.1 do not hold anymore. Nevertheless, given any finite sets of players' strategies, we can bound with high probability the distance of a maximin profile $\boldsymbol{\pi}^*$ from the strategy profile learned in the resulting (finite) discretized SBG. Formally, let $\mathcal{X}_n \subseteq \mathcal{X}$ be a finite set of n first player's strategies and, similarly, let $\mathcal{Y}_m \subseteq \mathcal{Y}$ be a finite set of m second player's strategies. Thus, the resulting finite SBG $\Gamma := (\mathcal{X}_n, \mathcal{Y}_m, u)$ has $n\,m$ strategy profiles. Let

$$d_x^{\max} = \max_{x\in\mathcal{X}} \min_{x_i\in\mathcal{X}_n} |x - x_i|,$$
$$d_y^{\max} = \max_{y\in\mathcal{Y}} \min_{y_i\in\mathcal{Y}_m} |y - y_i|,$$

then, we can show the following result.

Theorem 6.4. *Assume that u is drawn from a $\mathrm{GP}(\mathbf{0}, k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$ satisfying Assumption 1. Given $\delta \in (0, 2)$, let $\overline{\boldsymbol{\pi}} := (\bar{x}, \bar{y}) \in \mathcal{X}_n \times \mathcal{Y}_m$*

Table 1: Experimental results of algorithms M-LUCB, M-G-LUCB, and M-GP-LUCB on SBGs with finite strategy spaces.

			M-LUCB			M-G-LUCB			M-GP-LUCB			GP-SE
		T	T_δ	*%end*	*%opt*	T_δ	*%end*	*%opt*	T_δ	*%end*	*%opt*	*%opt*
SQE	$l = 0.1$	30k	10673.86	53.33	87.13	227.23	96.70	86.73	229.19	93.33	86.66	100.00
	$l = 2.0$	30k	23788.96	13.73	84.80	2460.19	89.23	77.73	2020.89	76.66	93.36	93.23
		100k	42103.86	46.66	88.63	3535.00	91.86	78.56	5656.77	76.77	93.30	96.60
$M_{1.5}$	$l = 0.1$	30k	13869.59	56.06	66.66	222.06	100.00	66.83	224.87	100.00	66.66	100.00
	$l = 2.0$	30k	18978.75	33.33	76.26	2532.98	91.30	76.90	3775.65	88.03	78.86	98.30
		100k	28798.42	50.00	80.70	3662.00	95.36	77.00	4618.27	93.33	79.53	98.76
$M_{2.5}$	$l = 0.1$	30k	13335.26	79.80	86.66	168.61	97.90	86.06	171.62	96.66	86.66	99.83
	$l = 2.0$	30k	20404.41	24.66	89.26	1984.55	92.10	86.63	2435.84	88.24	95.27	95.60
		100k	49198.82	62.06	92.86	2617.31	93.70	87.13	2626.89	86.66	94.93	96.46

be a maximin strategy profile for a finite game $\Gamma := (\mathcal{X}_n, \mathcal{Y}_m, u)$. *Then, the following holds:*

$$\mathbb{P}\left(\left|u(\boldsymbol{\pi}^*) - u(\overline{\boldsymbol{\pi}})\right| \leq b\sqrt{\log\left(\frac{4a}{\delta}\right)}\max\left\{d_x^{\max}, d_y^{\max}\right\}\right) \geq 1 - \frac{\delta}{2}.$$

Let us remark that the result in Theorem 6.4 can be applied any time using an equally-spaced grid as a discretization scheme is unfeasible, as it is the case, *e.g.*, when the dimensionality d of the players' strategy spaces is too large.

7 EXPERIMENTAL RESULTS

We experimentally evaluate our algorithms on both finite and infinite SBGs. As for the finite case, we compare the performances (with different metrics) of our M-GP-LUCB and GP-SE algorithms against two baselines. The first one is the M-LUCB algorithm proposed by Garivier et al. [9], which is the state of the art for learning maximin strategies in finite SBGs and can be easily adapted to our setting by using a different exploration term b_t. [7] We introduce a second baseline to empirically evaluate how our algorithms speed up their convergence by leveraging correlation of the utilities. Specifically, it is a variation of our M-GP-LUCB algorithm (called M-G-LUCB) where utility values are assumed drawn from independent Gaussian random variables, instead of a GP. [8] As for SBGs with infinite strategy spaces, there are no state-of-the-art techniques that we can use as a baseline for comparison. Thus, we show the quality (in terms of ϵ) of the strategy profiles returned by our algorithms using different values of K_ϵ for the discretized games. The average ϵ values obtained empirically (called $\hat{\epsilon}$ thereafter) are compared against the theoretical values prescribed by Theorem 6.1 (for the given K_ϵ), so as to evaluate whether our bounds are strict or not.

7.1 Random Game Instances

As for finite SBGs, we test the algorithms on random instances generated by sampling from $\mathrm{GP}(\mathbf{0}, k(\boldsymbol{\pi}, \boldsymbol{\pi}'))$, using the following two commonly used kernel functions (see [32] for more details):

- *squared exponential*: $k(\boldsymbol{\pi}, \boldsymbol{\pi}') := e^{-\frac{1}{2l^2}||\boldsymbol{\pi}-\boldsymbol{\pi}'||^2}$, where l is a length-scale parameter;
- *Matérn*: $k(\boldsymbol{\pi}, \boldsymbol{\pi}') := \frac{2^{1-\nu}}{G(\nu)} r^\nu B_\nu(r)$, where $r := \frac{\sqrt{2\nu}}{l}||\boldsymbol{\pi} - \boldsymbol{\pi}'||$, ν controls the smoothness of the functions, l is a length-scale parameter, B_ν is the second-kind Bessel function, and G is the Gamma function.

We set the kernel parameters to $l \in \{0.1, 2\}$ and $\nu \in \{1.5, 2.5\}$, generating 30 instances for each possible combination of kernel function and parameter values. As for SBGs with infinite strategy spaces, we test on instances generated from distributions with $l = 0.1$ and, with the Matérn kernel, $\nu \in \{1.5, 2.5\}$. The infinite strategy spaces are approximated with a discretization scheme based on a grid made of 100 equally-spaced points.

In the fixed-confidence setting, we let $\delta = 0.1$ and stop the algorithms after $T \in \{30\text{k}, 100\text{k}\}$ rounds. Similarly, the GP-SE algorithm is run with a budget $T \in \{30\text{k}, 100\text{k}\}$. For each possible combination of algorithm, game instance, and round-limit T, we average the results over 100 runs.

Results on Finite SBGs. The results are reported in Table 1, where T_δ is the average number of queries used by the algorithm in the runs not exceeding the round-limit T, *%end* is the percentage of runs the algorithm terminates before T rounds, and *%opt* is the percentage of runs the algorithm is able to correctly identify the maximin profile $\boldsymbol{\pi}^*$. Notice that M-GP-LUCB and M-G-LUCB clearly outperform M-LUCB, as the latter requires a number of rounds T_δ an order of magnitude larger. M-GP-LUCB and M-G-LUCB provide similar performances in terms of T_δ, but the former identifies the maximin profile more frequently than the latter. While always using the maximum number of rounds T, GP-SE is the best algorithm in identifying the maximin profile.

Results on Infinite SBGs. Figure 1 provides the values of ϵ and $\hat{\epsilon}$ for an instance generated from a Matérn kernel with $\nu = 2.5$. In all the instances, $\hat{\epsilon}$ is lower than ϵ, empirically proving the correctness of the theoretical guarantees provided in Section 6. Moreover, as expected, $\hat{\epsilon}$ decreases as the number of discretization points K_ϵ increases.

7.2 Security Game Instances

We also test on a SBG instance with infinite strategy spaces inspired by the real-world security game setting described in Section 1. This game models a military scenario in which a terrestrial counter-air defensive unit has to fire a heat-seeking missile to an approaching

[7] Since our utilities are not in [0, 1] (as they are drawn from a Gaussian instead of a Bernoulli), we multiply the b_t provided in [9] by the utility range.

[8] The formulas for updating the mean μ_t and the variance σ_t^2 of the posterior distribution are changed accordingly.

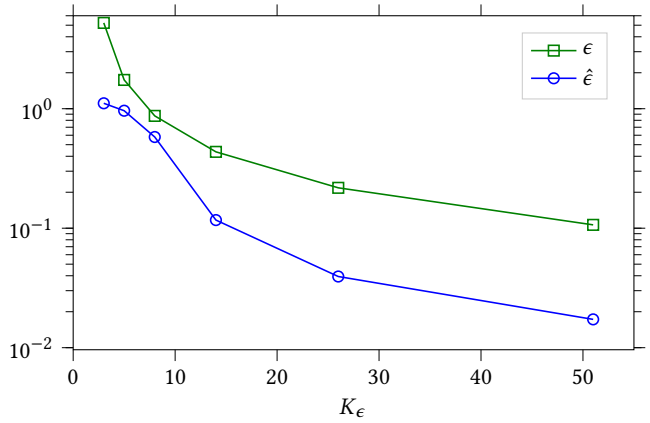


Figure 1: ϵ vs. $\hat{\epsilon}$ for different values of K_ϵ (Matérn kernel with smoothness parameter $\nu = 2.5$).

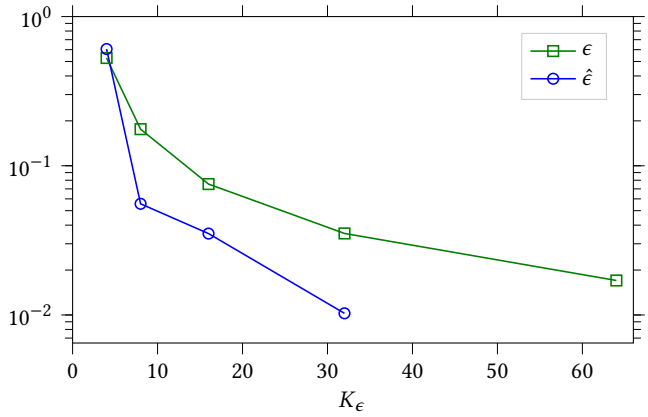
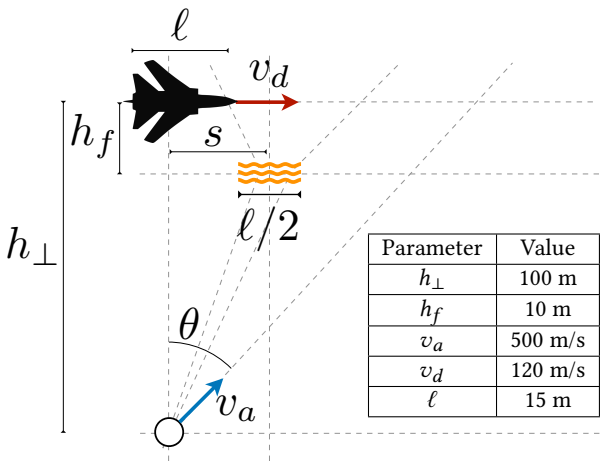


Figure 2: ϵ vs. $\hat{\epsilon}$ for different values of K_ϵ (*Hit-the-Spitfire* security game).

enemy airplane, which, after the missile has been launched, can deploy an obfuscating flare so as to try to deflect it. We call this game *Hit-the-Spitfire*. The model underlying such game and the parameters used in the experiment are depicted in Figure 3, where $h_\perp$ is the distance between the airplane and the terrestrial unit, h_f is the distance of the flare from the plane, v_a and v_d are the speed of the missile and the plane, respectively, while ℓ is the length of the plane, with the flare covering half of this space ($\frac{\ell}{2}$). The first player (the counter-air defensive unit) can determine the angle $\theta \in [0, 1]$ (radians) at which the missile is launched, while the second player (the airplane) has to decide the position $s \in [0, s_{\max}]$ where to release the flare. If the missile hits the plane, then it incurs damage $d \in \mathbb{R}^+$ that depends on the hitting point (the nearer to the center of the plane, the higher). If the missile hits the flare, then there is some probability that it is deflected away from the airplane, otherwise, the missile still hits the target. The probability of deflection is large when the distance of the airplane from the deployed flare is larger. We run the M-GP-LUCB with $\delta = 0.1$.

Results. Figure 2 reports the results of running the M-GP-LUCB algorithm with $\delta = 0.1$ on the *Hit-the-Spitfire* game (performing 100 runs for each K_ϵ). Notice that, in most of the cases, $\hat{\epsilon}$ is lower than the theoretical value ϵ. This is unexpected, since, in this setting, the assumption that the utility function u is drawn from a GP does not hold. We remark that, in all the runs, M-GP-LUCB is able to identify the maximin strategy profile over the given grid.



Parameter	Value
$h_\perp$	100 m
h_f	10 m
v_a	500 m/s
v_d	120 m/s
ℓ	15 m

Figure 3: *Hit-the-Spitfire* security game instance and values for its parameters used in the experiments.

8 DISCUSSION AND FUTURE WORKS

We addressed the problem of learning *maximin* strategies in two-player zero-sum SBGs with *infinite strategy spaces*, providing algorithms with theoretical guarantees. To the best of our knowledge, we provided the first learning algorithms for infinite SBGs enjoying δ-PAC theoretical guarantees on the quality of the returned solutions. This significantly advances the current state of the art for SBGs, as dealing with infinite strategies paves the way to the application of such models in complex real-world settings. The fundamental ingredient of our results is the assumption that the utility functions are drawn from a GP, which allows us to encode function regularities without relying on specific parametric assumptions, such as, *e.g.*, linearity.

In future, we will extend our work along different directions. For instance, we may address the case of general (*i.e.*, non-zero-sum and with more than two players) SBGs with finite (or even infinite) strategy spaces, where one seeks for an (approximate) Nash equilibrium. Along this line, an interesting question is how to generalize our learning algorithms based on best arm identification techniques to deal with Nash-equilibrium conditions instead of maximin ones. This would pave the way to the application of our techniques to other interesting problems, such as multi-agent evaluation by means of meta-games [18, 24]. Another interesting direction for future works is to study how to apply our techniques in empirical mechanism design problems [25].

ACKNOWLEDGMENTS

This work has been partially supported by the Italian MIUR PRIN 2017 Project ALGADIMAR "Algorithms, Games, and Digital Market".

REFERENCES

[1] J. Audibert, S. Bubeck, and R. Munos. 2010. Best Arm Identification in Multi-Armed Bandits. In *Proceedings of the Conference On Learning Theory (COLT)*. 41–53.

[2] L. Bisi, G. De Nittis, F. Trovò, M. Restelli, and N. Gatti. 2017. Regret Minimization Algorithms for the Followers Behaviour Identification in Leadership Games. In *Proceedings of the Conference on Uncertainty in Artificial (UAI)*. 1–10.

[3] Mario Bravo, David Leslie, and Panayotis Mertikopoulos. 2018. Bandit learning in concave N-person games. In *Proceeding of the conference on Neural Information Processing Systems (NIPS)*. 5661–5671.

[4] N. Brown and T. Sandholm. 2017. Safe and nested subgame solving for imperfect-information games. In *Proceeding of the conference on Neural Information Processing Systems (NIPS)*. 689–699.

[5] Noam Brown and Tuomas Sandholm. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359, 6374 (2018), 418–424.

[6] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890.

[7] S.R. Chowdhury and A. Gopalan. 2017. On kernelized multi-armed bandits. In *Proceedings of the International Conference on Machine Learning (ICML)*. 844–853.

[8] Abraham D Flaxman, Adam Tauman Kalai, Adam Tauman Kalai, and H Brendan McMahan. 2005. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the ACM-SIAM Symposium On Discrete Algorithms (SODA)*. 385–394.

[9] A. Garivier, E. Kaufmann, and W.M. Koolen. 2016. Maximin action identification: A new bandit framework for games. In *Proceedings of the Conference On Learning Theory (COLT)*. 1028–1050.

[10] Nicola Gatti, Alessandro Lazaric, Marco Rocco, and Francesco Trovò. 2015. Truthful learning mechanisms for multi-slot sponsored search auctions with externalities. *ARTIF INTELL* 227 (2015), 93–139.

[11] Nicola Gatti and Marcello Restelli. 2011. Equilibrium approximation in simulation-based extensive-form games. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 199–206.

[12] Robert D Kleinberg. 2005. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems*. 697–704.

[13] C.K. Ling, F. Fang, and J.Z. Kolter. 2018. What game are we playing? end-to-end learning in normal and extensive form games. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 396–402.

[14] C.K. Ling, F. Fang, and J.Z. Kolter. 2019. Large Scale Learning of Agent Rationality in Two-Player Zero-Sum Games. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 6104–6111.

[15] Alberto Marchesi, Francesco Trovò, and Nicola Gatti. 2019. Learning Probably Approximately Correct Maximin Strategies in Simulation-Based Games with Infinite Strategy Spaces. *CoRR* abs/1911.07755 (2019). arXiv:1911.07755 http://arxiv.org/abs/1911.07755

[16] J. Nocedal and S. Wright. 2006. *Numerical optimization*. Springer Science & Business Media.

[17] J. Rong, T. Qin, and B. An. 2019. Competitive Bridge Bidding with Deep Neural Networks. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 16–24.

[18] Mark Rowland, Shayegan Omidshafiei, Karl Tuyls, Julien Perolat, Michal Valko, Georgios Piliouras, and Remi Munos. 2019. Multiagent Evaluation under Incomplete Information. In *Advances in Neural Information Processing Systems*. 12270–12282.

[19] 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. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[20] M. Sion. 1958. On general minimax theorems. *PAC J MATH* 8, 1 (1958), 171–176.

[21] S. Sokota, C. Ho, and B. Wiedenbeck. 2019. Learning Deviation Payoffs in Simulation-Based Games. In *AAAI Technical Track: Game Theory and Economic Paradigms*. 1–8.

[22] N. Srinivas, A. Krause, S.M. Kakade, and M. Seeger. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*. 1015–1022.

[23] M. Tambe. 2011. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge university press.

[24] K. Tuyls, J. Perolat, M. Lanctot, J.Z. Leibo, and T. Graepel. 2018. A generalised method for empirical game theoretic analysis. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 77–85.

[25] E.A. Viqueira, C. Cousins, Y. Mohammad, and Greenwald. A. 2019. Empirical Mechanism Design: Designing Mechanisms from Data. In *Proceedings of the Conference on Uncertainty in Artificial (UAI)*. 1–11.

[26] E.A. Viqueira, A. Greenwald, C. Cousins, and E. Upfal. 2019. Learning Simulation-Based Games from Data. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 1778–1780.

[27] H. von Stackelberg. 1934. *Marktform und Gleichgewicht*. Springer, Vienna.

[28] Y. Vorobeychik and M.P. Wellman. 2009. Strategic analysis with simulation-based games. In *Proceedings of the IEEE Winter Simulation Conference (WSC)*. 359–372.

[29] Y. Vorobeychik, M.P. Wellman, and S. Singh. 2007. Learning payoff functions in infinite games. *MACH LEARN* 67, 1-2 (2007), 145–168.

[30] B. Wiedenbeck, F. Yang, and M.P. Wellman. 2018. A regression approach for modeling games with many symmetric players. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 1266–1273.

[31] B. Wilder, B. Dilkina, and M. Tambe. 2019. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, Vol. 33. 1658–1665.

[32] C.K.I. Williams and C.E. Rasmussen. 2006. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA.

[33] M. Wright and M.P. Wellman. 2019. Probably Almost Stable Strategy Profiles in Simulation-Based Games. In *Proceedings of the Games, Agents, and Incentives Workshops (GAIW)*. 1–9.