Randomized Sampling for Large Zero-Sum Games ⋆,⋆⋆

Shaunak D. Bopardikar [a], Alessandro Borri [b], João P. Hespanha [c], Maria Prandini [d], Maria D. Di Benedetto [e]

[a] *United Technologies Research Center Inc., Berkeley, CA, USA*

[b] *Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", Consiglio Nazionale delle Ricerche (IASI-CNR), Rome, Italy.*

[c] *Center for Control, Dynamical Systems and Computation, University of California at Santa Barbara, CA 93106, USA*

[d] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy*

[e] *Department of Electrical and Information Engineering, University of L'Aquila, Italy*

Abstract

This paper addresses the solution of large zero-sum matrix games using randomized methods. We formalize a procedure, termed as the *sampled security policy (SSP) algorithm*, by which a player can compute policies that, with a high confidence, are security policies against an adversary using randomized methods to explore the possible outcomes of the game. The SSP algorithm essentially consists of solving a stochastically sampled subgame that is much smaller than the original game. We also propose a randomized algorithm, termed as the *sampled security value (SSV) algorithm*, which computes a high-confidence security-level (i.e., worst-case outcome) for a given policy, which may or may not have been obtained using the SSP algorithm. For both the SSP and the SSV algorithms we provide results to determine how many samples are needed to guarantee a desired level of confidence. We start by providing results when the two players sample policies with the same distribution and subsequently extend these results to the case of mismatched distributions. We demonstrate the usefulness of these results in a hide-and-seek game that exhibits exponential complexity.

Key words: Game theory, Randomized algorithms, Zero-Sum Games, Optimization

1 Introduction

This paper addresses zero-sum games in which one or both players are faced with a large number of choices, possibly infinitely many. For such games, the computation of security levels (i.e., worst-case outcomes) and the corresponding security policies requires the exploration of very large decision trees.

Games where players are faced with deciding among a very large number of options arise in combinatorial problems, where the number of possible options grows exponentially with the size of the problem. This situation is common to many domains: In *path planning*, the number of possible decisions typically increases exponentially with the number of points to visit (cf., e.g., [4]). In *network security*, system administrators need to consider multi-stage, multi-host attacks that may consist of long sequences of actions by an attacker in their attempt to circumvent the system defenses (cf., e.g., [27]). In practice, this leads to policy spaces that grow exponentially with the number of stages involved in an attack. More generally, in *partial-information feedback games* players must choose feedback policies that assign an action to each possible observation and therefore the number of

⋆ This material is based upon work supported in part by ARO MURI Grant number W911NF0910553, in part by the Center of Excellence for Research DEWS, University of L'Aquila, Italy, and in part by the European Commission under the MoVeS project, FP7-ICT-2009-257005. Corresponding author – Shaunak D. Bopardikar. This work was performed while this author was at the University of California Santa Barbara, CA, USA.

⋆⋆A preliminary version of this work entitled "Randomized Sampling for Large Zero-Sum Games" was presented at the 2010 IEEE Conference on Decision and Control, USA.

Email addresses: bshaunak@gmail.com (Shaunak D. Bopardikar), alessandro.borri@iasi.cnr.it (Alessandro Borri), hespanha@ece.ucsb.edu (João P. Hespanha), prandini@elet.polimi.it (Maria Prandini), mariadomenica.dibenedetto@univaq.it (Maria D. Di Benedetto).

feedback policies grows exponentially with the size of the players' observation spaces (cf., e.g., [22,7]).

The exploration of large policy spaces is generally a hard task that can become computationally intractable when addressing partial information games (NP-complete in the size of the game tree, see [19]). A method to face this issue is Monte Carlo sampling. The key idea is to confine the search to a decision tree of reduced size by guessing or sampling the other player's moves, and then use conventional minimax search to determine the strategy to play with. Techniques based on this idea has been successfully applied to several partial information games such as, e.g., Scrabble (cf. [18]), Bridge (cf. [21]), and Kriegspiel chess (cf. [29]). The survey paper [9] shows how Monte Carlo sampling has becoming increasing popular and has been extensively adopted, not only in the game context, but also in other domains such as e.g. path planning. Indeed, when the underlying system is stochastic but it is difficult to derive an analytic description of the probabilistic distribution characterizing its evolution, solutions based on simulation are typically adopted, which entails the use of Monte Carlo sampling.

The recent successes in using randomized methods to explore large decision trees (e.g., in [25,23,9]) motivates the question that is behind the results in this paper: *Suppose that my opponent is using a randomized algorithm to explore the game decision tree, can I produce a security value and an associated security policy that are correct with high probability?* The answer to this question is affirmative and we show that such security values/policies can be constructed using randomized algorithms. What is somewhat surprising about the results reported is that one can obtain high-confidence security policies by restricting ones attention to a subset of policies that can be much smaller than the total set of policies available to the players. Moreover, this restricted set of policies may be quite different from the set of policies that the opponent considered in her randomized exploration of the game decision tree.

We call *sampled security policy* (SSP) the randomized algorithm proposed to obtain high-confidence security values/policies. The SSP algorithm can be described as follows: Suppose that player P_2 selected a policy based on a random exploration of the policies available to both players. The precise algorithm used by P_2 to select her policy based on this extraction is typically unknown. Player P_1 should then proceed as follows: randomly select a subset of the total set of available policies to both players; construct the zero-sum matrix game corresponding to the selected subset of policies, *ignoring all other policies available to the two players*; and compute the security value/policies associated with the matrix. Player P_1 can select either a mixed or a pure value/policy. In both cases, since a large number of policies have been ignored, the security policies obtained by this process will generally not be security policies for the whole game and therefore player P_1 may obtain an outcome that is strictly worse than the value computed based on her submatrix. However, we show that this happens with low probability as long as the size of the submatrix is sufficiently large. Moreover, this result holds regardless of the algorithm used by P_2 to compute her policy based on the random tree exploration. In fact, P_2 could also be using the SSP algorithm to compute her own policy.

Related Work

Two-player zero-sum matrix games have been studied extensively over the past decades (cf. the textbook by [3]). The classical Mini-Max theorem (cf. [35]) guarantees the existence of an optimal pair of strategies for the two players, each of which is a security policy for the corresponding player. However, when the matrix is of large size, the computation of the optimal strategies involves solving optimization problems with a large number of variables and constraints.

A probabilistic approach has proven to be computationally efficient in evaluating large sized games. Using probabilistic analysis, the existence of simple, near-optimal strategies over a subset with logarithmically smaller size of the original matrix game was established in [26]. A popular method to solve win-lose type of multi-stage or dynamic games is to evaluate the root of a game tree, in which every node is alternately an *AND* and an *OR* operation, while the leaves have a value of either 0 or 1. [28] presents randomized algorithms to evaluate such game trees more efficiently than deterministic algorithms.

Randomized methods have been successful in providing efficient solutions to complex control design problems with probabilistic guarantees. [24] adopts a probabilistic approach to show the existence of randomized algorithms with polynomial complexity to solve complex robust stability analysis problems. [30] proposes a randomized method for a probabilistic analysis of the worst-case controller performance, and determine sample size bounds. More recently, [31] discusses the application of randomized methods to several control design problems in the presence of uncertainty. Randomized methods have also been used to provide a probabilistic approximation to the minimax value of a cost function in robust control design problems (cf. [20]). Their sample complexity requirement is, in general, much higher than for the notion of security that we propose in this paper, since they are concerned with the sampled minimax value being close to the global minimax value with high confidence. A randomized approach is used in the linear programming reformulation of approximate dynamic programming in [16]. [33,34] demonstrate the use of randomized algorithms to solve control design problems and a number of well known complex problems in matrix theory through a statistical learning approach. Statistical learning theory [32] provides a framework for

probabilistic robust control synthesis. Using these tools, [1] considers semi-infinite optimization problems under uncertainty with a possibly non-convex objective function.

In [12,14,13], the authors introduce the so-called *scenario approach* to solve convex optimization problems with an infinite number of constraints. Applications of this approach to systems and control are discussed in [12] and in [15]. [10] and [11] study the sample complexity of randomized approaches to system analysis and design, and provide, in particular, an explicit expression of the sample-size for the scenario approach to convex optimization based on an approximation of the implicit expression given in [14]. These bounds were further refined in [2]. The results in these papers are instrumental to establish several of the results in the present paper.

Contributions

Throughout the paper, we explain the results from the perspective of the player P_1 — the minimizer, — who finds herself playing against an opponent P_2 — the maximizer, — who computes her policy based on a random exploration of the game decision tree.

The contributions of this paper are four-fold. First, we show that the SSP algorithm provides a security policy for P_1 with probability $1-\delta$, provided that the size of the subgame solved by P_1 is sufficiently large. We provide two bounds on the size of the subgame, one that is valid when P_1 uses general mixed policies and the other when P_1 is restricted to consider only pure policies. The latter may require much smaller submatrix sizes when the entries of the matrix A take values in a finite set. The bounds are *game independent* and can be computable a-priori for any desired confidence level $1-\delta$, $\delta > 0$. While the size of the subgame grows with the desired confidence level $1-\delta$, it is *completely independent of the size of the original matrix game*, which could, in fact, be even infinite and not even have a value. Moreover, this bound is also independent of the precise algorithm that P_2 uses to construct her policy based on the portion of the tree that she explored.

The results outlined above assume that, while P_1 does not know the precise subtree that P_2 explored to compute her policy, P_1 does know the distribution that P_2 used to construct her random subtree. When this is not the case, there will be a mismatch between the distribution that P_1 uses in the SSP algorithm and the distribution that P_2 uses for her random exploration. The second contribution of the paper addresses this issue via two approaches. The first approach adopts sample complexity bounds obtained in [17], which deals with the so-called ambiguous chance constrained problems. More precisely, we determine bounds on the sizes of the submatrices in the SSP algorithm when the mismatch between the distributions used by the two players remains below a specified distance $\rho < 1$, measured in the Prohorov metric. This approach requires no knowledge of the matrix game, but the bounds hold only when the confidence parameter satisfies the condition $\rho < \delta$, for a desired confidence level of $1-\delta$. The second approach is based on a novel characterization of the distance between the sampling distributions, which we call the *mismatch factor*, and is applicable to any confidence level $1-\delta$ and any mismatch factor between the distributions. However, as one would expect, for a given confidence lever, a large mismatch factor requires a large number of samples. These results take advantage of the game structure and, in fact, when the mismatch is associated with policy domination, we show that the bounds with mismatch are exactly the same as the ones without mismatch. Essentially, this means that if P_1 knows that a particular subset $S_{\text{worse-for-}\mathsf{P}_2}$ of P_2's policies is dominated by another subset of P_2's policies $S_{\text{better-for-}\mathsf{P}_2}$ (in the sense that $S_{\text{worse-for-}\mathsf{P}_2}$ is worse than $S_{\text{better-for-}\mathsf{P}_2}$ from P_2's perspective), then P_1 need not sample policies from $S_{\text{worse-for-}\mathsf{P}_2}$. The benefit of the second approach goes beyond investigating the confidence of the SSP algorithm, as it extends the bounds of the scenario approach derived in [13] and [2] to mismatched distributions.

Third, we propose a randomized algorithm, which we call *sampled security-value* (SSV) that P_1 can use to obtain a high-probability security level for a given policy. The bound on the size of the subgame that P_1 extracts to determine her high-probability security level holds for any policy available to P_1, regardless of whether or not this policy was obtained from the SSP algorithm. As for the SSP algorithm, the computation required by the SSV algorithm is independent of the size of the original matrix game and also of the precise algorithm that P_2 uses to construct her policy. When applied to a policy obtained using the SSP algorithm for a confidence level δ_{SSP}, the SSV algorithm can be used to study the security of the policy for different (perhaps tighter) confidence levels δ.

Fourth and finally, we apply the SSP and SSV algorithms to solve a hide-and-seek game, in which one player hides a treasure in one of N points and the other player searches for the treasure by visiting each of the points. This is formalized as a zero-sum game in which the player that hides the treasure wants to maximize the distance that the other player needs to travel until the treasure is found. To determine the optimal strategy for this game, one would need to solve a matrix game whose size is $N \times N!$. Thus, exact solutions to this problem require computation that scales exponentially with the number of points N. Our approach is *independent of the size of the game* and therefore the size of the matrix plays no role in the amount of computation required.

As compared to the preliminary conference version [5], this paper presents new results that include the version of the SSP algorithm for pure policies and its analysis, the mismatch in the distributions used by the players to

construct the subgames, and improves upon the *explicit* sample size bounds using the results in [2]. The more recent paper [7] formalizes the sampling procedure to dynamic or multi-stage, partial information games.

Organization

This paper is organized as follows. The problem formulation and the SSP algorithm are presented in Section 2. Bounds on the subgame size to provide high-confidence SSP solutions are established in Section 3 for the case when the two players use identical distributions to sample the matrix. These bounds are extended in Section 4 to allow mismatch between the distributions. Section 5 presents the SSV algorithm and the related bounds on the size of the subgame. Finally, we demonstrate the procedure applied to the search game in Section 6.

2 Problem Formulation

Consider a zero-sum matrix game defined by an arbitrary $M \times N$ real-valued matrix A, in which player P_1 is the minimizer and selects rows and player P_2 is the maximizer and selects columns. We are interested in problems where the number N of (pure) policies available to P_2 is very large, typically due to combinatorial explosion, forcing P_2 to explore a random subset of her own (pure) policy space with only $n_2 \ll N$ policies, and perhaps also only a random subset of the possibles responses by P_1. Based on this, P_2 selects a policy z^* that she will use to play against P_1.

Denoting by $\mathcal{B}^{k\times\ell}$ the set of $k \times \ell$ left-stochastic $(0,1)$-valued matrices (i.e., matrices whose entries belong to the set $\{0,1\}$ with exactly one 1 per column), we can express the process by which P_2 samples her own policy space by selecting a random matrix Π_2 from the set $\mathcal{B}^{N\times n_2}$. The matrix $\Pi_2 \in \mathcal{B}^{N\times n_2}$ has one row for each of the possible policies of P_2 in the original game defined by A and one column for each policy that was actually explored by P_2. A one in row i, column j of Π_2 signifies that the jth policy explored by P_2 corresponds to the column i of A. P_2's random exploration results in a mixed policy z^* that can be written as

$$z^* = \Pi_2 z_2^* \in \mathcal{S}_N, \quad z_2^* \in \mathcal{S}_{n_2}, \tag{1}$$

where, for a given integer k, $\mathcal{S}_k$ denotes the probability simplex of size k. P_1 may know the distribution used to extract Π_2, but *will not know the matrix* Π_2 *that was actually extracted nor which algorithm was used to determine* z_2^* and therefore will not know the policy z^* obtained by P_2.

For P_1 to compute a high-confidence response against P_2's policy z^* in (1), we introduce the *sampled security policy (SSP) Algorithm* 1.

Algorithm 1 [SSP Algorithm]

1: P_1 randomly selects m_1 rows and n_1 columns of A, which she uses to construct an $m_1 \times n_1$ submatrix A_1 of A. This can expressed by the selection of two random matrices $\Gamma_1 \in \mathcal{B}^{M\times m_1}$ and $\Pi_1 \in \mathcal{B}^{N\times n_1}$ and then computing the product $A_1 = \Gamma_1' A \Pi_1$.

2: P_1 computes the mixed security value $\bar{V}(A_1)$ and the corresponding security policy y_1^* for A_1:

$$\bar{V}(A_1) = \max_{z\in\mathcal{S}_{n_1}} y_1^{*\prime} A_1 z = \min_{y\in\mathcal{S}_{m_1}} \max_{z\in\mathcal{S}_{n_1}} y' A_1 z$$

We call $\bar{V}(A_1)$ P_1*'s sampled security value*. When multiple security policies y_1^* exist, P_1 selects for y_1^* the one with the minimum Euclidean norm (since the set of security policies is convex, it contains a unique element with minimum norm).

3: P_1 computes her mixed policy for the original game:

$$y^* := \Gamma_1 y_1^*,$$

resulting in the outcome $y^{*\prime} A z^* = y_1^{*\prime}\Gamma_1' A \Pi_2 z_2^*$. We call y^* P_1*'s sampled security policy.*

For the SSP algorithm to be useful, it needs to provide appropriate guarantees of correctness, which are formalized by the following definitions. We say that *the SSP algorithm is* ϵ*-secure for player* P_1 *with confidence* $1-\delta$ if

$$\mathrm{P}_{\Gamma_1,\Pi_1,\Pi_2}\left(y^{*\prime} A z^* \le \bar{V}(A_1) + \epsilon\right) \ge 1 - \delta. \tag{2}$$

Here and in the sequel, we use a subscript in the probability measure P to remind the reader which random variables define the event that is being measured. In essence, condition (2) states that the probability that the outcome of the game will violate P_1's sampled security value $\bar{V}(A_1)$ by more than ϵ is smaller than δ. As stated, this definition requires the bound to hold regardless of the algorithm used by P_2 to select her policy z^*. In fact, we even allow z^* to be obtained using an algorithm that randomly explores P_1's policy space [1]. While our results do not depend on it, P_2 could have obtained z^* also using the SSP algorithm.

The previous definition guarantees that P_1 will be surprised with (low) probability δ when playing with policies obtained from a one-shot solution to the SSP algorithm. However, no specific guarantee is given regarding the inherent safety of the specific policy y^* obtained using the SSP algorithm. So, e.g., suppose that player P_1 computes y^* once using the SSP algorithm and then plays this policy multiple times against a sequence of policies z^* for P_2, each obtained by a distinct random

[1] In this case, the probability measure in (2) depends on additional random variables that we do not explicitly include in the subscript.

explorations of her policy space. Then P_1 could conceivably be surprised many more times than one would expect for a low value of δ. This would happen if she was "unlucky" and got a (low probability) y^* that is particularly bad or a value $\bar{V}(A_1)$ that is particularly optimistic. To avoid this scenario, we introduce an additional notion of security that refers to the security of a specific policy/value: we say that *a policy y^* with value $\bar{V}(A_1)$ is ϵ-secure for player P_1 with confidence $1-\delta$* if

$$\mathrm{P}_{\Pi_2}\left({y^*}'Az^* \leq \bar{V}(A_1)+\epsilon \mid y^*,\ \bar{V}(A_1)\right) \geq 1-\delta. \quad (3)$$

Note that the subscript in the probability measure now only includes the matrix corresponding to randomized exploration of the policy space by P_2 since the probability guarantees are given for a specific security policy and value of P_1.

So far, we have not specified the joint distribution of the row/column extraction matrices Γ_1 and Π_1 for P_1 in the SSP Algorithm 1, but these distributions, jointly with that of matrix Π_2 for P_2, clearly affect the outcome of the algorithm. In the context of noncooperative games, one should presume the extractions of the two players to be independent of each other. For simplicity, we further assume that players extract rows and columns independently, as stated in the following assumption:

Assumption 2.1 (Independence) *The random matrices Γ_1 and Π_1 in the SSP Algorithm 1 and the matrix Π_2 corresponding to player P_2's randomized exploration are statistically independent and each of them has independent and identically distributed columns.* □

Under Assumption 2.1, we shall determine in Section 3 bounds on the size of the random matrices extracted by P_1 in the SSP Algorithm 1 that guarantee high-confidence ϵ-security results with $\epsilon = 0$. These results are valid when P_1 knows precisely the distribution used by P_2 to explore her game decision tree, i.e., to extract columns of the game matrix A. If we allow for mismatched distributions, we can then prove ϵ-security results with a value for $\epsilon > 0$ that depends on the distance between the distributions used by P_1 and P_2 to extract columns of A (see Section 4).

Remark 2.1 (General games) The results in this paper do not depend on the fact that the original game is a finite matrix game. They extend trivially to any cost-function $J(u,d)$, $u \in \mathcal{U}$, $d \in \mathcal{D}$ where $\mathcal{U}$ and $\mathcal{D}$ denote the sets of policies for the minimizer and maximizer, respectively. In fact, it is not even necessary that the original game has saddle-point policies since all that the results use is the fact that, when we take finite samples of the sets of policies, we obtain finite matrix games. In fact, these results also apply to dynamic games, as is discussed in [7]. □

3 Bounds for Probabilistic Guarantees

In this section, we present theoretical bounds on the number of policies that player P_1 needs to consider for the SSP Algorithm to guarantee desired confidence levels. The results in this section refer to the case where the players sample policies for P_2 (i.e., *columns* of A) using identical distributions. This assumption is subsequently relaxed in Section 4.

3.1 Mixed Sampled Security Policies

The main result of this section provides a bound on the size of the submatrix A_1 in the SSP Algorithm that guarantees ϵ-security with $\epsilon = 0$ for the mixed policy y^*. We recall that P_2 is assumed to use a policy z^* of the form (1), where Π_2 is a column-selection matrix and z_2^* some vector in $\mathcal{S}_{n_2}$ that is obtained using a deterministic or stochastic algorithm. The case of P_2 using a sample of P_1's policies to determine her policy z^* also gets included as none of the results in this paper require P_2 to use the same distribution as that used by P_1 for extracting rows of A.

Theorem 3.1 (SSP Algorithm) *Suppose that Assumption 2.1 holds and that $\Pi_1 \in \mathcal{B}^{N\times n_1}$ and $\Pi_2 \in \mathcal{B}^{N\times n_2}$ have identically distributed columns. The SSP Algorithm is $(\epsilon=0)$-secure for P_1 with confidence $1-\delta$, $\delta \in (0,1)$ as long as*[2]

$$n_1 = \left\lceil \frac{m_1+1}{\delta} - 1 \right\rceil \bar{n}_2, \quad (4)$$

with $\bar{n}_2 \geq n_2$. Additionally, suppose that we increase n_1 to satisfy

$$n_1 = \left\lceil \frac{1}{\delta}\Big(m_1 + \sqrt{2m_1 \ln\frac{1}{\beta}} + \ln\frac{1}{\beta}\Big) \right\rceil \bar{n}_2, \quad (5)$$

for some $\beta \in (0,1)$. Then, with probability larger than $1-\beta$, the SSP Algorithm generates a sampled security policy y^* with value $\bar{V}(A_1)$ that is $(\epsilon=0)$-secure for P_1 with confidence $1-\delta$. □

In words, this result states that it is always possible to guarantee $(\epsilon = 0)$-security for P_1, if she constructs her submatrix A_1 utilizing a sufficiently large number of columns n_1. In particular, she always needs to choose a number of columns n_1 larger than the number of columns n_2 that P_2 is considering for her mixed policies [cf. (4) and (5)]. The additional number of columns that P_1 needs to consider is a function of the number m_1 of rows that P_1 wants to consider for her mixed policy and the desired confidence level.

[2] Given a scalar $x \in \mathbb{R}$, we denote by $\lceil x \rceil$ the smallest integer that is larger than or equal to x.

The probability $1-\beta$ associated with y^*'s security probabilistic guarantee accounts for the possibility that the confidence bound (3) fails altogether due to an "unfortunate" sample used by P_1 to compute y^*. However, note that only the logarithm of the confidence level β appears (5) and therefore a relatively small value for the number of columns n_1 suffices to make β extremely small (and, hence, $1-\beta \simeq 1$).

Remark 3.2 (P_1's knowledge of n_2) According to Theorem 3.1, for player P_1 to enjoy guaranteed ($\epsilon = 0$)-security with confidence $1-\delta$, she must know an upper bound $\bar{n}_2$ on the number of columns that P_2 used to construct the policy z^* in (1). However, if P_1 does not know $\bar{n}_2$ precisely and, e.g., underestimates $\bar{n}_2$ by a certain percentage, then (4) and (5) are still useful in the sense that they predict that the performance degradation in the actual confidence level δ grows linearly with $\bar{n}_2$. This is because the bounds in (4) and (5) scale with $\frac{\bar{n}_2}{\delta}$. □

Proof of Theorem 3.1: By definition of the security value $\bar{V}(A_1)$, we have that

$$\begin{aligned}\bar{V}(A_1) &= \min_{y\in\mathcal{S}_{m_1}} \max_{z\in\mathcal{S}_{n_1}} y'\Gamma_1'A\Pi_1 z \\ &= \min_{y\in\mathcal{S}_{m_1}} \max_{j\in\{1,\ldots,n_1\}} y'\Gamma_1'A\Pi_1 e_j(n_1) \\ &= \min_{\theta\in\Theta}\Big\{v : y'\Gamma_1'A\Pi_1 e_j(n_1) \le v,\ \forall j\in\{1,\ldots,n_1\}\Big\},\end{aligned} \tag{6}$$

where $\theta := (y_1, v)$, $\Theta := \mathcal{S}_{m_1}\times\mathbb{R}$, and we use $e_j(n)$ to denote the jth element of the canonical basis of $\mathbb{R}^n$.

Since n_1 is an integer multiple of $\bar{n}_2$, i.e., $n_1 = K\bar{n}_2$ with $K = \left\lceil \frac{m_1+1}{\delta} - 1 \right\rceil$, we can use the $K\bar{n}_2$ columns of $\Pi_1 \in \mathcal{B}^{N\times K\bar{n}_2}$ to construct K independent and identically distributed (i.i.d.) matrices $\Delta_1, \Delta_2, \ldots, \Delta_K \in \mathcal{B}^{N\times\bar{n}_2}$. For an arbitrary realization of the matrix $\Gamma_1 \in \mathcal{B}^{M\times m_1}$, which is independent of the Δ_i by Assumption 2.1, let us define the function $f_{\Gamma_1} : \Theta\times\mathcal{B}^{N\times\bar{n}_2}\to\mathbb{R}$ by

$$f_{\Gamma_1}(\theta,\Delta) = \max_{j\in\{1,\ldots,\bar{n}_2\}} y_1'\Gamma_1'A\Delta e_j(\bar{n}_2) - v. \tag{7}$$

We can then rewrite (6) as

$$\bar{V}(A_1) = \min_{\theta\in\Theta}\big\{v : f_{\Gamma_1}(\theta,\Delta_i)\le 0,\ \forall i\in\{1,\ldots,K\}\big\} \tag{8}$$

and conclude from [13, Proposition 3] that the (conditional) probability that another matrix Δ sampled independently from the same distribution as the Δ_i satisfies the constraint $f_{\Gamma_1}(\theta^*,\Delta)\le 0$ satisfies:

$$\mathrm{P}_{\Pi_1,\Delta}\left(f_{\Gamma_1}(\theta^*,\Delta)\le 0 \mid \Gamma_1\right) \ge \frac{K-m_1}{K+1} \ge 1-\delta, \tag{9}$$

where θ^* denotes the value in Θ that achieves the minimum in (8) and the second inequality is a consequence of (4). Since the minimum in (8) is achieved for the sampled security policy/value $\theta^* = (y_1^*, \bar{V}(A_1))$, we can use the definition (7) of f_{Γ_1} to re-write (9) as

$$\mathrm{P}_{\Pi_1,\Delta}\left(y_1^{*\prime}\Gamma_1'A\Delta e_j(\bar{n}_2)\le \bar{V}(A_1), \quad \forall j\in\{1,\ldots,\bar{n}_2\} \mid \Gamma_1\right)\ge 1-\delta.$$

Since $n_2\le\bar{n}_2$, we further conclude that

$$\mathrm{P}_{\Pi_1,\Delta}\left(y_1^{*\prime}\Gamma_1'A\Delta e_j(n_2)\le \bar{V}(A_1), \quad \forall j\in\{1,\ldots,n_2\} \mid \Gamma_1\right)\ge 1-\delta.$$

Under Assumption 2.1, when the columns of Π_1 and Π_2 are identically distributed, the matrix consisting of the first n_2 columns of Δ can be viewed as the matrix Π_2 and the inequality above implies that

$$\mathrm{P}_{\Pi_1,\Pi_2}\left(y_1^{*\prime}\Gamma_1'A\Pi_2 e_j(n_2)\le \bar{V}(A_1), \quad \forall j\in\{1,\ldots,n_2\} \mid \Gamma_1\right)\ge 1-\delta.$$

Since $y_1^{*\prime}\Gamma_1'A\Pi_2 e_j(n_2)\le\bar{V}(A_1),\ \forall j\in\{1,\ldots,n_2\}$, implies $y_1^{*\prime}\Gamma_1'A\Pi_2 z\le\bar{V}(A_1),\ \forall z\in\mathcal{S}^{n_2}$, we conclude that

$$\mathrm{P}_{\Pi_1,\Pi_2}\left(y_1^{*\prime}\Gamma_1'A\Pi_2 z_2^*\le\bar{V}(A_1)\mid\Gamma_1\right)\ge 1-\delta.$$

We have shown that this bound holds for an arbitrary realization of Γ_1, therefore it also holds for the unconditional probability, which shows that the SSP Algorithm is ($\epsilon=0$)-secure for P_1 with confidence $1-\delta$ as per (2).

If instead of using [13, Proposition 3] and (4) to obtain (9), we use [2, Theorem 4] and (5), we obtain that

$$\mathrm{P}_{\Delta}\left(f_{\Gamma_1}(\theta^*,\Delta)\le 0 \mid \Gamma_1,\theta^*\right)\ge 1-\delta, \tag{10}$$

with probability at least $1-\beta$, where the confidence level $1-\beta$ refers to the extraction of $\Pi_1 = [\Delta_1,\ldots,\Delta_K]$ that defines θ^* (given Γ_1). The proof can now proceed exactly as before, but with (9) replaced by (10), which now involves a probability conditioned to $\theta^* = (y^*, \bar{V}(A_1))$. Thus if n_1 satisfies (5), then with probability at least $1-\beta$, the policy y^* with value $\bar{V}(A_1)$ is ($\epsilon=0$)-secure for P_1 with confidence $1-\delta$. ■

3.2 Pure Sampled Security Policy

Suppose that P_1 restricts herself to use pure policies in Step 2 of the SSP Algorithm 1. If we let $e_i(m_1)$ denote the ith element of the canonical basis of $\mathbb{R}^{m_1}$, then Step 2 becomes:

2: P_1 computes the pure security value $\bar{V}_{\text{pure}}(A_1)$:

$$\begin{aligned}\bar{V}_{\text{pure}}(A_1) &= \max_{z\in\mathcal{S}_{n_1}} e'_{i^*}(m_1)A_1 z \\ &= \min_{i\in\{1,\dots,m_1\}} \max_{z\in\mathcal{S}_{n_1}} e'_i(m_1)A_1 z,\end{aligned}$$

and the corresponding pure security policy y_1^* for A_1:

$$y_1^* = e_{i^*}(m_1).$$

We call $\bar{V}_{\text{pure}}(A_1)$ P_1*'s pure sampled security value.* When multiple pure security policies y_1^* exist, P_1 can pick any of them.

A bound similar to (5) can be established for the resulting *pure SSP Algorithm*, but the number of columns n_1 that P_1 needs to sample can often be much smaller. Note that the bound still holds for any policy z^* for player P_2 of the form (1), pure or mixed.

Theorem 3.3 (Pure SSP Algorithm) *Suppose that Assumption 2.1 holds and that $\Pi_1 \in \mathcal{B}^{N\times n_1}$ and $\Pi_2 \in \mathcal{B}^{N\times n_2}$ have identically distributed columns. Suppose that we select*

$$n_1 = \left\lceil \frac{1}{\delta}\Big(\ln\big(m_1 \cdot \#(\Gamma'_1 A)\big) + \ln\frac{1}{\beta}\Big)\right\rceil \bar{n}_2, \qquad (11)$$

for $\beta,\delta \in (0,1)$ and $\bar{n}_2 \geq n_2$, where $\#(\Gamma'_1 A)$ denotes the total number of distinct values that the entries of $\Gamma'_1 A$ can take. Then, with probability $1-\beta$, the pure SSP Algorithm yields a pure sampled security policy y^* with value $\bar{V}_{\text{pure}}(A_1)$ that is $(\epsilon = 0)$-secure for P_1 with confidence $1-\delta$. □

In several matrix games the number of distinct values that entries of A can take is small and therefore $\#(\Gamma'_1 A)$ is small. This occurs, e.g., in win-lose-tie games. For such games, (11) provides significant computational gains with respect to (5) because the bound in (11) grows with the *logarithm* of m_1, whereas the one in (5) grows *linearly* with m_1. Even if the matrix A can have many distinct values, significant computational savings are possible since $\#(\Gamma'_1 A) \leq m_1 N$ and hence, at worse, (11) still only grows with the *logarithm* of $m_1^2 N$. The price to pay for these computational savings is that the pure security level $\bar{V}_{\text{pure}}(A_1)$ could be higher than the value $\bar{V}(A_1)$ obtained with mixed policies.

The proof of Theorem 3.3 – reported in the technical report [6] – is conceptually similar to the second part of the proof of Theorem 3.1, with the main difference being that the policy selection involves optimizing over a finite set of cardinality $\#(\Gamma'_1 A)$, and, hence, we can use the bounds in [2, Theorem 3] instead of those in [2, Theorem 4].

Remark 3.4 The bound for the pure SSP Algorithm corresponding to (4) in Theorem 3.1 would be

$$n_1 = \left\lceil \frac{m_1 \cdot \#(\Gamma'_1 A)}{\delta} - 1 \right\rceil \bar{n}_2,$$

which can be obtained by first deriving the single level of probability version of the bounds in [2, Theorem 3], following similar steps as in [13, Proposition 3]. Given that this bound is worse than (4), there is no computational advantage for player P_1 in considering pure rather than mixed policies. Consequently, this result is not included in Theorem 3.3. □

4 Mismatch in the Sampling Distributions

We now investigate the effect of a mismatch between the distribution that P_1 uses to select the columns of the policy-selection matrix Π_1 used in the SSP Algorithm of Section 3 and the distribution that P_2 uses to select the columns of the policy-selection matrix Π_2 that she uses to determine the policy z^* in (1).

We pursue two approaches: The first one is based on a characterization of the mismatch between distributions using the Prohorov metric and provides bounds that are independent of the game matrix A. The second approach provides a novel characterization of the mismatch between the two distributions that can take the matrix A into consideration.

Due to space limitations, we provide only some intuition on the proof of the results in this section. The reader is referred to [6] for details.

4.1 Prohorov Metric-Based Approach

For a given integer k, the distributions used to select the columns of Π_1 and Π_2 can be used to construct two measures m^k and $\tilde{m}^k$, respectively, for the column-selection matrices taking values in $\mathcal{B}^{N\times k}$ with i.i.d. columns. Using the discrete metric

$$d(x_1,x_2) = \begin{cases}1, & x_1 \neq x_2,\\ 0, & x_1 = x_2,\end{cases} \qquad \forall x_1,x_2 \in \mathcal{B}^{N\times k}, \quad (12)$$

we can regard $\mathcal{B}^{N\times k}$ as a metric space, for which the Prohorov metric between m^k and $\tilde{m}^k$ is simply given by the total variation metric

$$\pi(m^k,\tilde{m}^k) = \sup_{B\in\mathcal{F}} |m^k(B) - \tilde{m}^k(B)|, \qquad (13)$$

where $\mathcal{F}$ denotes the Borel sigma-algebra on $\mathcal{B}^{N\times k}$. The following theorem is based on the results for the ambiguous chance constrained problems in [17] and should be viewed as a generalization of the bound (5) in Theorem 3.1 for the case of mismatched distributions.

Theorem 4.1 (Matrix-independent mismatch) *Suppose that Assumption 2.1 holds, that*

$$\pi\big(m^{\bar{n}_2}, \tilde{m}^{\bar{n}_2}\big) \leq \rho < 1$$

for some $\bar{n}_2 \geq n_2$, and that we select

$$n_1 = \Big\lceil 2(m_1+1) + \frac{2}{\delta-\rho}\ln\frac{1}{\beta} + \frac{2(m_1+1)}{\delta-\rho}\ln\frac{2}{\delta-\rho}\Big\rceil \bar{n}_2 \tag{14}$$

for $\beta \in (0,1)$, $\delta \in (\rho, 1)$. Then, with probability larger than $1-\beta$, the SSP Algorithm generates a sampled security policy y^* with value $\bar{V}(A_1)$ that is $(\epsilon = 0)$-secure for P_1 with confidence $1-\delta$. □

The first step in proving this result, consists of expressing (6) as

$$\bar{V}(A_1) = \min_{\theta\in\Theta}\big\{v : f_{\Gamma_1}(\theta,\Delta) \leq 0,\ \forall\,\Delta \text{ such that } d(\Delta,\Delta_i) \leq \rho \text{ for some } i \in \{1,\ldots,K\}\big\},$$

where f_{Γ_1} is defined in (7), d denotes the discrete metric (12), and K is the number multiplying $\bar{n}_2$ on the right hand side of (14). Next, we apply [17, Theorem 6] to conclude that a random variable Δ with probability distribution $\tilde{m}^{\bar{n}_2}$ satisfies

$$\tilde{m}^{\bar{n}_2}\big(f_{\Gamma_1}(\theta^*,\Delta) \leq 0 \mid \Gamma_1, \theta^*\big) \geq 1-\delta,$$

with probability at least

$$1 - (\mathrm{e}\,K/(m_1+1))^{m_1+1}\,\mathrm{e}^{-(\delta-\rho)(K-(m_1+1))}. \tag{15}$$

Finally we show that (15) exceeds $1-\beta$ via routine algebraic simplifications.

In the spirit of Theorem 3.1, the bound (14) is completely independent of the matrix game A. However, this result has the limitation that it is applicable only for confidence levels $\delta > \rho$ and therefore does not permit confidence levels larger than $1-\rho$.

4.2 Mismatch Factor-Based Approach

Our second approach to characterize the impact of a mismatch between the sampling distributions relies on generalizations of the bounds of the scenario approach to convex optimization in [13] and [2] that were instrumental to the proof of Theorem 3.1. We start by presenting these generalizations, which are useful beyond the context of the problem considered here.

4.2.1 Scenario Optimization

Consider a sequence of K i.i.d. random variables Δ_1, Δ_2, …, Δ_K taking values in a set D. These random variables are used to specify a set of K constraints in the following convex optimization problem: [3]

$$\theta^* = \arg\min_{\theta\in\Theta}\Big\{c'\theta : f(\theta,\Delta_i) \leq 0,\ \forall i \in \{1,\ldots,K\}\Big\}, \tag{16}$$

where $c \in \mathbb{R}^{n_\theta}$ and the constraint-defining function $f : \Theta \times D \to \mathbb{R}$ is convex with respect to θ, for each fixed value of D, and Θ is a convex subset of $\mathbb{R}^{n_\theta}$.

The results that follow provide bounds on the probability that an additional independent random variable $\bar{\Delta}$, also taking values in D but with a different distribution, satisfies the following (somewhat relaxed) version of the constraint that appears in (16) for the optimal θ^*:

$$f(\theta^*, \bar{\Delta}) \leq \epsilon,$$

for some $\epsilon \geq 0$. [13, Proposition 3] and [2, Theorem 4] provide such bounds when Δ_i and $\bar{\Delta}$ have the same distribution and $\epsilon = 0$. Denoting by m and $\tilde{m}$ the measures associated with the distributions of Δ_i and $\bar{\Delta}$, respectively, define the *mismatch factor between m and $\tilde{m}$* by

$$\mu_f(\epsilon) := \inf_{\mu\in\mathbb{R}}\Big\{\mu :\ \tilde{m}\big(f(\theta,\bar{\Delta}) > \epsilon\big) \leq \mu\, m\big(f(\theta,\Delta) > 0\big),\ \forall\theta\in\Theta\Big\}. \tag{17}$$

When $m = \tilde{m}$, we have $\mu_f(\epsilon) \leq 1$, with equality when $\epsilon = 0$. However, when the distributions do not match, $\mu_f(\epsilon)$ can be arbitrarily large. As the name indicates, the mismatch factor $\mu_f(\epsilon)$ can be viewed as a measure of how much the distributions of Δ and $\bar{\Delta}$ differ. Aside from not being a metric, it differs more fundamentally from the Prohorov metric (13) in that (i) (17) only regards the discrepancy between values of the measures for "violation" events of the type $f(\theta,\bar{\Delta}) > \epsilon$; (ii) it considers a kind of multiplicative uncertainty in the probabilities (instead of differences); and (iii) it allows for the "relaxation" parameter $\epsilon > 0$ that can bring $\mu_f(\epsilon)$ down if we are willing to allow $f(\theta,\bar{\Delta})$ to grow as large as $\epsilon > 0$. As we shall see shortly, smaller values of $\mu_f(\epsilon)$ lead to smaller probabilities of violation.

The following two results generalize [13, Proposition 3] and [2, Theorem 4], respectively, for mismatched distributions and $\epsilon > 0$.

[3] In case of several possible multiple minima, the one with the smallest Euclidean norm should be selected.

Lemma 4.2 *For every $\epsilon \geq 0$,*

$$\mathrm{P}_{\bar{\Delta},\Delta_1,\ldots,\Delta_K}\Big(f\big(\theta^*,\bar{\Delta}\big) \leq \epsilon\Big) \geq 1 - \frac{\mu_f(\epsilon)\, n_\theta}{K+1}.$$

□

Lemma 4.3 *Given $\epsilon > 0$, $\delta \in (0,1)$, $\beta \in (0,1)$, and*

$$K \geq \Big\lceil \frac{\mu_f(\epsilon)}{\delta}\Big((n_\theta - 1) + \sqrt{2(n_\theta - 1)\ln\frac{1}{\beta}} + \ln\frac{1}{\beta}\Big)\Big\rceil,$$

we have that

$$\mathrm{P}_{\bar{\Delta}}\left(f(\theta^*,\bar{\Delta}) \leq \epsilon \,|\, \theta^*\right) \geq 1 - \delta,$$

with probability[4] *larger than $1-\beta$.* □

4.2.2 Probabilistic Guarantees

Lemmas 4.2 and 4.3 allow us to generalize Theorem 3.1 for the case of mismatched distributions. This generalization involves a family of functions $f_\Gamma : \Theta \times B^{N\times\bar{n}_2} \to \mathbb{R}$, with $\Theta := (\mathcal{S}_{m_1}, \mathbb{R})$ and $\bar{n}_2$ an integer larger than n_2, parameterized by the matrix $\Gamma \in \mathcal{B}^{M\times m_1}$ and defined by

$$f_\Gamma(\theta,\Delta) = \max_{j\in\{1,\ldots,\bar{n}_2\}} y_1'\Gamma' A\Delta e_j(\bar{n}_2) - v.$$

We shall use these functions to compute the mismatch factor between the measures m^k and $\tilde{m}^k$ for column-selection matrices taking values in $\mathcal{B}^{N\times k}$ with i.i.d. columns, constructed using the distributions used to select the columns of Π_1 and Π_2, respectively.

Theorem 4.4 (Matrix-dependent mismatch) *Suppose that Assumption 2.1 holds, and that*

$$\tilde{m}^{\bar{n}_2}\big(f(\theta,\bar{\Delta}) > \epsilon\big) \leq \mu\, m^{\bar{n}_2}\big(f(\theta,\Delta) > 0\big), \qquad (18)$$

$\forall\theta \in \Theta, \forall\Gamma \in \mathcal{B}^{M\times m_1}$, for some $\mu \in (0,\infty)$, $\bar{n}_2 \geq n_2$ and $\epsilon \geq 0$. The SSP Algorithm is ϵ-secure for P_1 with confidence $1-\delta$, $\delta \in (0,1)$ as long as

$$n_1 = \Big\lceil \frac{\mu}{\delta}(m_1+1) - 1\Big\rceil \bar{n}_2.$$

Additionally, suppose that we increase n_1 to satisfy

$$n_1 = \Big\lceil \frac{\mu}{\delta}\Big(m_1 + \sqrt{2m_1\ln\frac{1}{\beta}} + \ln\frac{1}{\beta}\Big)\Big\rceil \bar{n}_2,$$

for some $\beta \in (0,1)$. Then, with probability larger than $1-\beta$, the SSP Algorithm generates a sampled security policy y^* with value $\bar{V}(A_1)$ that is ϵ-secure for P_1 with confidence $1-\delta$. □

[4] The confidence level $1-\beta$ refers to the extraction of $\Delta_1,\ldots\Delta_K$ that defines θ^*.

The proof of this result is conceptually similar to the proof of Theorem 3.1, with the main difference being that we use Lemmas 4.2 and 4.3 instead of [13, Proposition 3] and [2, Theorem 4], respectively.

Theorem 4.4 shows that, even when there is a mismatch in the distributions, it is still possible to achieve high-confidence security policies. However, the number of samples required by the SSP algorithm essentially needs to be multiplied by μ. Alternatively, if one uses the number of samples dictated by Theorem 3.1 and there is mismatch in the distributions, then one obtains security with confidence $1-\delta/\mu$ (instead of $1-\delta$) since one can go from the formulas in Theorem 3.1 to the ones in Theorem 4.4 by simply replacing $1/\delta$ by μ/δ.

Remark 4.5 ($\mu < 1$) For values of $\epsilon > 0$ and matched (or closely matched distributions), the mismatch factors may actually be smaller than 1. Theorem 4.4 is still applicable and states that if one is willing to accept some $\epsilon > 0$, one may get $1-\delta$ confidence with a smaller number of samples than those required by Theorem 3.1. □

Remark 4.6 (Matrix-independent results) If we choose μ to satisfy

$$\tilde{m}^{\bar{n}_2}(B) \leq \mu\, m^{\bar{n}_2}(B), \quad \forall B \in \mathcal{F},$$

where $\mathcal{F}$ denotes the Borel sigma-algebra on $\mathcal{B}^{N\times\bar{n}_2}$, then (18) holds with $\epsilon = 0$ for every matrix game and we obtain a game independent result. The price is, of course, that such μ does not explore the structure of the particular game and may therefore be much larger than what is needed. In fact, we shall see in the next section that the structure of the matrix A may dictate that some mismatch should not lead to a degradation in the confidence levels. This is the case when A exhibits some form of policy domination. □

4.2.3 Matrix games with dominated policies

Consider a situation when P_1 knows of some particularly good policies that P_2 may apply to play the game. For example, suppose that the entries in some column $c_{\text{better-for-}\mathsf{P}_2}$ of A are all element-wise larger than those in some other column $c_{\text{worse-for-}\mathsf{P}_2}$. In this case, it turns out that P_1 can increase the probability of sampling the column $c_{\text{better-for-}\mathsf{P}_2}$ at the expense of decreasing the probability of selecting $c_{\text{worse-for-}\mathsf{P}_2}$ and this mismatch does not require a larger bound on the number of columns to sample. This observation is formalized in the remaining of this section.

We begin with the following notion of dominance.

Definition 1 (ϵ-Dominance) *Given an $M\times N$ matrix A, the vector $d^* \in \mathcal{B}^{N\times 1}$ is said to be* ϵ-dominated *by the*

vector $d \in \mathcal{B}^{N\times 1}$ for some $\epsilon \geq 0$ if

$$e_i(M)'Ad^* \leq e_i(M)'Ad + \epsilon, \quad \forall i \in \{1,\ldots,M\},$$

where $e_i(M)$ is the ith canonical basis element of $\mathbb{R}^M$.

With $\epsilon = 0$, the above definition becomes identical to that of domination between pure policies in matrix games (cf. [3]). Next, we introduce the notion of two sampling distributions being perturbed.

Definition 2 (Perturbed sampling) *Given two distinct vectors $d, d^* \in \mathcal{B}^{N\times 1}$ and two probability measures $m, \tilde{m}$ on $\mathcal{B}^{N\times 1}$, we say that m is a* perturbation *of $\tilde{m}$ with respect to the pair (d, d^*) if*

(1) m differs from $\tilde{m}$ only over $\{d, d^*\} \subseteq \mathcal{B}^{N\times 1}$, i.e.,

$$\tilde{m}(e_j(N)) = m(e_j(N)),$$

for all j such that $e_j(N) \notin \{d^*, d\}$, where $e_j(N)$ denotes the jth element of the canonical basis of $\mathbb{R}^N$;

(2) the probability of extracting d^* according to m is smaller than according to $\tilde{m}$, i.e.,

$$m(d^*) \leq \tilde{m}(d^*).$$

We now present the main result of this subsection.

Theorem 4.7 (Domination) *Given the game matrix A, suppose that for some $\epsilon \geq 0$, there exist vectors $d^*, d \in \mathcal{B}^{N\times 1}$ such that d^* is ϵ-dominated by d. Suppose that Assumption 2.1 holds and that the columns of the matrices Π_1 and Π_2 are sampled according to distributions m and $\tilde{m}$, respectively. If m is a perturbation of $\tilde{m}$ with respect to (d, d^*), then Theorem 4.4 holds with $\mu = 1$.*

This result shows that even when P_1 extracts with low probability (possibly equal to zero) the column d^*, the bounds of Section 3 hold.

To establish Theorem 4.7, we need to show that

$$\mathrm{P}_{\bar{\Delta}}\left(f_\Gamma(\theta, \bar{\Delta}) > \epsilon\right) \leq \mathrm{P}_{\Delta}\left(f_\Gamma(\theta, \Delta) > 0\right), \qquad (19)$$

for any $\theta \in \Theta$, $\Gamma \in \mathcal{B}^{M\times m_1}$, since from this condition we have that $\mu = 1$ satisfies (18). We achieve this by individually considering the following two cases:

$$y_1'\Gamma' Ad^* - v > \epsilon, \quad \text{or} \quad y_1'\Gamma' Ad^* - v \leq \epsilon.$$

For each one of these cases, we prove (19) by using the definition of ϵ-dominance and perturbed sampling, along with the independence of column extraction.

5 A-posteriori assessment of a given policy

Suppose now that P_1 obtained a policy y^* using either a randomized or a deterministic algorithm. In this section, we are interested in computing a high-confidence security level value $\bar{V}(y^*)$ for this policy, when y^* is played against P_2's policy z^* in (1). The *sampled security-value* (SSV) Algorithm 2 addresses this question.

Algorithm 2 [SSV Algorithm]

1: P_1 randomly selects k_1 columns of A, which corresponds to the selection of a random matrix $\bar{\Pi}_1 \in \mathcal{B}^{N\times k_1}$.

2: P_1 computes

$$\bar{V}(y^*) = \max_{j\in\{1,\ldots,k_1\}} y^{*\prime} A\bar{\Pi}_1 e_j(k_1), \qquad (20)$$

where $e_j(k_1)$ denotes the jth element of the canonical basis of $\mathbb{R}^{k_1}$. We call $\bar{V}(y^*)$ *P_1's a-posteriori sampled security value.*

We use the qualifier "a-posteriori" for the sampled security value $\bar{V}(y^*)$ to emphasize that this value is computed *after* a particular security policy y^* has been obtained. We say that *the SSV algorithm is ϵ-secure for player P_1's policy y^* with confidence* $1-\delta$ if

$$\mathrm{P}_{\bar{\Pi}_1,\Pi_2}\left(y^{*\prime} Az^* \leq \bar{V}(y^*) + \epsilon \mid y^*\right) \geq 1 - \delta.$$

This condition states that the probability that the outcome of the game will violate P_1's a-posteriori sampled security value $\bar{V}(y^*)$ by more than ϵ is smaller than δ. As stated, this definition requires the bound to hold regardless of the algorithms used to generate y^* and z^*. In particular, we leave open the possibility that both policies could have been computed using the SSP algorithm, perhaps with confidence levels different than δ. In fact, one could imagine P_1 computing y^* using the SSP algorithm for a confidence level δ_{SSP} and then studying the security of such policy for tighter confidence levels δ using the SSV algorithm. Thus, the SSV algorithm combined with the SSP algorithm can be viewed as a heuristic for designing high-confidence security policies.

Also for the SSV, we can define a stronger notion of security that guarantees the inherent security of the a-posteriori sampled security value $\bar{V}(y^*)$, when P_1 plays y^* repeatedly against a sequence of policies z^* for P_2, each obtained by a distinct random exploration of her policy space. We say that *the a-posteriori sampled security value $\bar{V}(y^*)$ is ϵ-secure for player P_1's policy y^* with confidence* $1-\delta$ if

$$\mathrm{P}_{\Pi_2}\left(y^{*\prime} Az^* \leq \bar{V}(y^*) + \epsilon \mid y^*,\ \bar{V}(y^*)\right).$$

As for the SSP Algorithm 1, we assume that the column extraction matrices, $\bar{\Pi}_1$ used by player P_1 in the SSV Algorithm 2 and Π_2 used by player P_2 in her randomized policy exploration, are independent.

Assumption 5.1 (Independence) *The random matrix $\bar{\Pi}_1$ in the SSV Algorithm 2 and matrix Π_2 involved in player P_2 randomized exploration are statistically independent and each of them has independent and identically distributed columns.* □

We now provide a bound on the number of samples k_1 used in the SSV Algorithm to guarantee ϵ-security. Akin to Section 3, we restrict our attention to the case when player P_1 uses the same distribution as player P_2 to extract columns of A. Results along the same lines as those shown in Section 4 for the SSP Algorithm could be obtain for the SSV Algorithm in the case of mismatched distributions, but we omit them because they are fundamentally similar.

Theorem 5.1 (SSV Algorithm) *Suppose that Assumption 5.1 holds and that $\bar{\Pi}_1 \in \mathcal{B}^{N\times k_1}$ and $\Pi_2 \in \mathcal{B}^{N\times n_2}$ have identically distributed columns. The SSV algorithm is $(\epsilon = 0)$-secure for player P_1's policy y^* with confidence $1-\delta$, $\delta \in (0,1)$ as long as*

$$k_1 = \left\lceil \frac{1}{\delta} - 1 \right\rceil \bar{n}_2, \tag{21}$$

with $\bar{n}_2 \geq n_2$. Additionally, suppose that we increase k_1 to satisfy

$$k_1 = \left\lceil \frac{1}{\delta} \ln \frac{1}{\beta} \right\rceil \bar{n}_2, \tag{22}$$

for some $\beta \in (0,1)$. Then, with probability $1-\beta$, the SSV Algorithm generates an a-posteriori sampled security value $\bar{V}(y^*)$ that is $(\epsilon = 0)$-secure for player P_1's policy y^* with confidence $1-\delta$. □

Proof of Theorem 5.1: Defining $K := \left\lceil \frac{1}{\delta} - 1 \right\rceil$ and the function $\bar{f} : \mathcal{S}_M \times \mathcal{B}^{N\times \bar{n}_2} \to \mathbb{R}$ by

$$\bar{f}(y, \Delta) = \max_{j\in\{1,\ldots,\bar{n}_2\}} y' A \Delta e_j(\bar{n}_2),$$

we can re-write (20) as

$$\bar{V}(y^*) = \max_{i\in\{1,\ldots,K\}} \bar{f}(y^*, \Delta_i),$$

where the matrices $\Delta_1, \Delta_2, \ldots, \Delta_K \in \mathcal{B}^{N\times\bar{n}_2}$ are obtained by partitioning the $K\bar{n}_2$ columns of $\bar{\Pi}_1 \in \mathcal{B}^{N\times K\bar{n}_2}$ into K i.i.d. matrices.

For any given y^* (independent of the Δ_i's), we conclude from [13, Proposition 4] that the (conditional) probability that another matrix Δ, sampled independently from the same distribution as the Δ_i, satisfies the constraint

$$\bar{f}(y^*, \Delta) \leq \bar{V}(y^*) := \max_{i\in\{1,\ldots,K\}} \bar{f}(y^*, \Delta_i),$$

can be lower-bounded as follows:

$$\mathrm{P}_{\bar{\Pi}_1,\Delta}\left(\bar{f}(y^*,\Delta) \leq \bar{V}(y^*) \mid y^*\right) \geq \frac{K}{K+1} \geq 1-\delta, \tag{23}$$

where the second inequality is a consequence of (21). From the definition of $\bar{f}$, we conclude from (23) that

$$\mathrm{P}_{\bar{\Pi}_1,\Delta}(y^{*\prime} A \Delta e_j(\bar{n}_2) \leq \bar{V}(y^*), \forall j \in \{1,\ldots,\bar{n}_2\} \mid y^*) \geq 1-\delta,$$

and, since $n_2 \leq \bar{n}_2$, we also have that

$$\mathrm{P}_{\bar{\Pi}_1,\Delta}(y^{*\prime} A \Delta e_j(n_2) \leq \bar{V}(y^*), \forall j \in \{1,\ldots,n_2\} \mid y^*) \geq 1-\delta.$$

Under Assumption 5.1, when the columns of $\bar{\Pi}_1$ and Π_2 are identically distributed, the matrix consisting of the first n_2 columns of Δ can be viewed as the matrix Π_2 and the inequality above implies that

$$\mathrm{P}_{\bar{\Pi}_1,\Pi_2}(y^{*\prime} A \Pi_2 e_j(n_2) \leq \bar{V}(y^*), \forall j \in \{1,\ldots,n_2\} \mid y^*) \geq 1-\delta.$$

Since $y^{*\prime} A \Pi_2 e_j(n_2) \leq \bar{V}(y^*),\ \forall j \in \{1,\ldots,n_2\}$, implies $y^{*\prime} A \Pi_2 z \leq \bar{V}(y^*),\ \forall z \in \mathcal{S}_{n_2}$, we conclude that

$$\mathrm{P}_{\Pi_2,\bar{\Pi}_1}\left(y^{*\prime} A \Pi_2 z_2^* \leq \bar{V}(y^*) \mid y^*\right) \geq 1-\delta,$$

which shows that SSV Algorithm is $(\epsilon = 0)$-secure with confidence $1-\delta$.

If, instead of using [13, Proposition 4] and (21) to obtain (23), we use [14, Theorem 1] and (22), we obtain

$$\mathrm{P}_{\Delta}\left(\bar{f}(y^*,\Delta) \leq \bar{V}(y^*) \mid y^*, \bar{V}(y^*)\right) \geq 1-\delta, \tag{24}$$

with probability larger than $1-\beta$, where the confidence level $1-\beta$ refers to the extraction of $\bar{\Pi}_1 = [\Delta_1, \ldots, \Delta_K]$ that defines $\bar{V}(y^*)$. The proof can now proceed exactly as before, but with (23) replaced by (24), which now involves a probability conditioned to y^*, and $\bar{V}(y^*)$. This shows that if k_1 satisfies (22), then with probability larger than $1-\beta$, the security value $\bar{V}(y^*)$ is $(\epsilon = 0)$-secure with confidence $1-\delta$. ■

6 Example: Hide-and-seek matrix game

In this section, we apply the SSP and SSV Algorithms to a classic search problem: Consider a zero-sum game where P_1 hides a non-moving object (treasure) in one of N points $\{p_1, \ldots, p_N\} \subset \mathbb{R}^2$ on the plane and P_2 wants

to find the treasure with minimum cost, by traveling from point to point until she finds it.

The game is played over the set of mixed policies:

- P_1 chooses a probability distribution $z \in \mathcal{S}_N$ for the treasure over the N points, and
- P_2 chooses a probability distribution $y \in \mathcal{S}_M$ over the set $\mathcal{R} := \{r_j : j = 1, \ldots, M\}$ of $M := N!$ routes that start at P_1's initial position $p_0 \in \mathbb{R}^2$ and go through all possible permutations of the points.

When P_1 chooses to hide the treasure at point p_i and P_2 selects route r_j, the outcome of the game is equal to the length of route r_j from P_1's initial position p_0 to the point p_i where the treasure lies. Namely,

$$A_{ij} = -\sum_{k=1}^{k^*_{ij}} \|r_j(k) - r_j(k-1)\|, \tag{25}$$

where $r_j(k) \in \mathbb{R}^2$, $k \in \{1, \ldots, N\}$ denotes the kth point in the route r_j with $r_j(0) = p_0$, and the summation ends at the index k^*_{ij} for which $r_j(k^*_{ij}) = p_i$ is the point where the treasure is hidden. The minus sign in (25) is needed to maintain consistency with the formulation in the first part of the paper, where P_1 is the minimizer. Indeed, P_1 hides the treasure to maximize the distance and therefore to minimize the entries of A.

For a large N, the exact computation of the optimal mixed strategies is intractable because the size of the matrix A is $N \times N!$. However, the results in this paper lead to a computational complexity that is *independent of the size of the game,* which means that we can provide probabilistic guarantees for games with an arbitrarily large number of points.

In this game, only the player P_2 that chooses paths has a large number of options ($M = N!$) so we can assume that both players consider all possible N locations where P_1 can hide the treasure (all rows of A), but randomly select only a small number of paths (columns of A) to construct their submatrices. However, the player P_1 that hides the treasure should respect the bounds provided by Theorems 3.1 and 5.1 to avoid unpleasant surprises.

In our numerical experiments, we considered $N = 10$ points distributed uniformly randomly in a square region. To illustrate the use of the SSP and the SSV Algorithms, we fixed $m_1 = N$, $\beta = 10^{-5}$, and $\bar{n}_2 = 10$ (Figure 1) or $\bar{n}_2 = 1000$ (Figure 2). To achieve a confidence level of $\delta = .01$, two approaches are possible:

SSP only: Execute the SSP Algorithm 1 with n_1 satisfying (5) to obtain a sampled security value and a sampled security policy with confidence $1 - \delta = 99\%$.

SSP+SSV: Execute the SSP Algorithm 1 with a value for n_1 smaller than the one indicated by (5) to obtain a sampled security policy, and then run the SSV Algorithm 2 with a value of k_1 satisfying (22) to obtain an a-posteriori sampled security value with confidence $1 - \delta = 99\%$.

While the SSP+SSV option requires solving a smaller subgame in the SSP algorithm, and is therefore computationally more attractive, it typically results in a worst sampled security policy and therefore the corresponding security value is typically worst. However, one can see that the curves corresponding to the SSP+SSV option are relatively flat, which indicates that significant computational savings are possible without a significant degradation in the sampled security level. Note that the security levels computed using either of the approaches above are random variables since they depend on the randomly selected columns of the matrix A. The plots in Figures 1 and 2 show Monte Carlo estimates of the mean and standard deviation of these random variables.

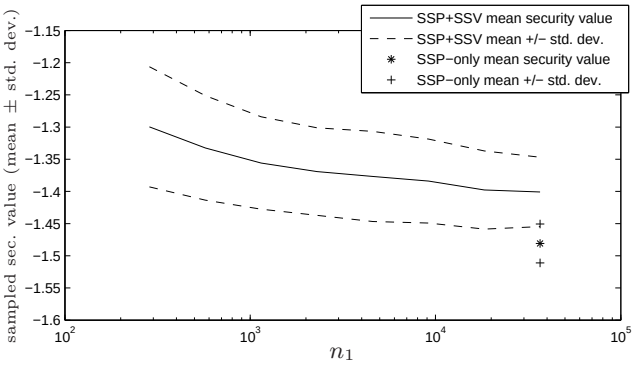

Fig. 1. Numerically determined values for the 99% confidence sampled security value ($\delta = .1$). The solid line (mean) and dashed lines (plus/minus one standard deviation) were obtained using the SSP+SSV approach, using different values of n_1 in the SSP Algorithm (with n_1 in the x-axis) and a value for k_1 in the SSV Algorithm satisfying (22). The star '*' was obtained using the SSP-only approach, using the value for n_1 satisfying (5). The remaining parameters used are as follows: the number of points is $N = 10$, the side length of the square region is 1 unit, $m_1 = \bar{n}_2 = 10$, $\beta = 10^{-5}$, and the columns were drawn uniformly randomly. Each mean and standard deviation was estimated using 300 Monte Carlo samples.

7 Conclusions and Future Directions

We addressed the solution of large zero-sum matrix games using randomized techniques. We provided a procedure based on randomized sampling by which a player can construct policies that are security with high-probability against an adversary engaged in a randomized exploration of the games characterized by large decision trees. We proposed a new probabilistic notion of security policy and level and derive bounds on the sample sizes that guarantees the discovery of a security

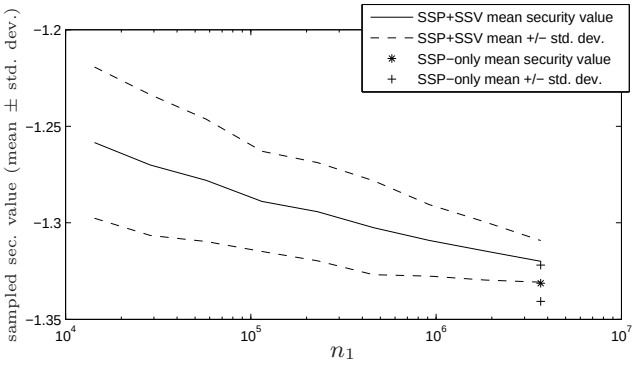

Fig. 2. Plot similar to that in Figure 1, with the exception that we took $\bar{n}_2 = 1000$. Each mean and standard deviation was estimated using 300 Monte Carlo samples.

policy with high probability. The bounds provided consider both the case where the two players sample policies using the same and different distributions. The applicability of the results is illustrated with a combinatorial hide-and-seek game.

This work suggests a number of future directions of research. One promising direction is to explore incremental opmization techniques to reduce the bound on the size of the submatrices and/or the number of entries of the submatrices that are needed to compute the sampled security policies. Another direction for future research regards the choice of the distributions used to sample policies to minimize the sample-size bounds and maximize the probability of finding adequate policies. In the context of the example in Section 6, we are currently exploring closed-loop versions of the hide-and-seek game that involve the searcher taking measurements regarding the location of the treasure as she moves from point to point (cf. [8]).

References

[1] T. Alamo, R. Tempo, and E. F. Camacho. Randomized strategies for probabilistic solutions of uncertain feasibility and optimization problems. *IEEE Trans. on Automat. Contr.*, 54(11):2545–2559, November 2009.

[2] T. Alamo, R. Tempo, and A. Luque. On the sample complexity of randomized approaches to the analysis and design under uncertainty. In *Proc. of the 2010 Amer. Contr. Conf.*, pages 4671–4676, Baltimore, MD, USA, June 2010.

[3] T. Basar and G. J. Olsder. *Dynamic Non-Cooperative Game Theory*. SIAM, Philadelphia, PA, USA, 1999.

[4] R. Bellman. Dynamic programming treatment of the traveling salesman problem. *J. Assoc. Comput. Mach.*, 9:61–63, 1962.

[5] S. D. Bopardikar, A. Borri, J. P. Hespanha, M. Prandini, and M. D. Di Benedetto. Randomized sampling for large zero-sum games. In *Proc. of the 49th Conf. on Decision and Contr.*, pages 7675–7680, Atlanta, GA, USA, December 2010.

[6] S. D. Bopardikar, A. Borri, J. P. Hespanha, M. Prandini, and M. D. Di Benedetto. Randomized sampling for large zero-sum games. Technical report, University of California at Santa Barbara, November 2012.

[7] S. D. Bopardikar and J. P. Hespanha. Randomized solutions to partial information dynamic zero-sum games. In *Proc. of the 2011 Amer. Contr. Conf.*, June 2011.

[8] A. Borri, S. D. Bopardikar, J. P. Hespanha, and M. D. Di Benedetto. Hide-and-seek with directional sensing. In *Proc. of the 18th World Congress of Int. Federation of Automat. Contr.*, Milan, Italy, August 2011.

[9] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.

[10] G. Calafiore. On the expected probability of constraint violation in sampled convex programs. *Journal of Optimization Theory and Applications*, 143(2):405–412, 2009.

[11] G. Calafiore. Random convex programs. *SIAM Journal of Optimization*, 20(6):3427–3463, 2010.

[12] G. C. Calafiore and M. C. Campi. The scenario approach to robust control design. *IEEE Trans. on Automat. Contr.*, 51(5):742–753, May 2006.

[13] M. C. Campi and G. C. Calafiore. Notes on the scenario design approach. *IEEE Trans. on Automat. Contr.*, 54(2):382–385, February 2009.

[14] M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of robust convex programs. *SIAM J. Contr. Optimization*, 19(3):1211–1230, 2008.

[15] M. C. Campi, S. Garatti, and M. Prandini. The scenario approach for systems and control design. *Annual Reviews in Control*, 33(2):149–157, December 2009.

[16] D. P. de Farias and B. V. Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, August 2004.

[17] E. Erdoğan and G. Iyengar. Ambiguous chance constrained problems and robust optimization. *Mathematical Programming*, 107(1–2):37–61, December 2006.

[18] A. Frank. *Heuristic programming in Artificial Intelligence 2: The second computer olympiad*, chapter Brute force search in games of imperfect information, pages 204–209. Ellis Horwood, 1989.

[19] I. Frank and D. Basin. Optimal play against best defence: Complexity and heuristics. In H.J. Herik and H. Iida, editors, *Computers and Games*, volume 1558 of *Lecture Notes in Computer Science*, pages 50–73. Springer Berlin Heidelberg, 1999.

[20] Y. Fujisaki and Y. Kozawa. Probabilistic robust controller design: Probable near-minimax value and randomized algorithm. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1938–1943, Maui, Hawaii, USA, December 2003.

[21] M. Ginsberg. Partition search. In *Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 228–233, 1996.

[22] J. P. Hespanha and M. Prandini. Nash equilibria in partial-information games on Markov chains. In *Proc. of the 40th Conf. on Decision and Contr.*, pages 2102–2107, December 2001.

[23] Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: A tool for automatic verification of probabilistic systems. In Holger Hermanns and Jens

Palsberg, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer Berlin / Heidelberg, 2006.

[24] P. Khargonekar and A. Tikku. Randomized algorithms for robust control analysis and synthesis have polynomial complexity. In *Proc. of the 35th Conf. on Decision and Contr.*, volume 3, pages 3470–3475, Kobe, Japan, December 1996.

[25] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, , and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. Wellesley, MA, 2001.

[26] R. J. Lipton and N. E. Young. Simple strategies for large zero-sum games with applications to complexity theory. In *Twenty-sixth annual ACM Symposium on Theory of Computing*, pages 734–740, 1994.

[27] K.-W. Lye and J. Wing. Game strategies in network security. *International Journal of Information Security*, 4:71–86, 2005.

[28] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[29] A. Parker, D. Nau, and V. S. Subrahmanian. Game-tree search with combinatorially large belief states. In *International Joint Conference on Artificial Intelligence*, pages 254–259, 2005.

[30] R. Tempo, E. W. Bai, and F. Dabbene. Probabilistic robustness analysis: Explicit bounds for the minimum number of samples. *Syst. & Contr. Lett.*, 30(5):237–242, 1997.

[31] R. Tempo, G. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer-Verlag, London, 2004.

[32] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[33] M. Vidyasagar. Statistical learning theory and randomized algorithms for control. *IEEE Contr. Syst. Mag.*, 18(6):69–85, December 1998.

[34] M. Vidyasagar and V. D. Blondel. Probabilistic solutions to some NP–hard matrix problems. *Automatica*, 37(9):1397–1405, September 2001.

[35] J. Von Neumann. Zur theorie der gesellschaftsspiele. *Math. Annalen.*, 100:295–320, 1928.