

Efficient Nash Equilibrium Computation for Cybersecurity Games

Michael Lanier David Farmer Yevgeniy Vorobeychik
Washington University in St. Louis

Abstract

Computing Nash equilibria of simulation-based cybersecurity games with policy-space response oracles (PSRO) is bottlenecked by payoff estimation: every payoff-matrix entry costs Monte-Carlo rollouts of a slow simulator, while policies and restricted-game solves are cheap. We introduce Regret-Weighted Payoff Sampling (RWPS), a budgeted estimator that simulates only the cells an equilibrium is sensitive to and fills the rest with a surrogate trained on every entry simulated earlier in the run. The sup-norm error bound cannot evaluate such an estimator, because it is set by the cells left deliberately inaccurate. We prove an instance-dependent bound that weights error by the opponent’s equilibrium mixture, a certificate computable from simulation data alone, and a coverage result showing that once the deviation-relevant set is simulated, surrogate error cannot affect either player’s regret. On three 21×21 general-sum games, two synthetic and an asymmetric Colonel Blotto, the refined bounds are four to six times tighter on the estimator’s own output, and the coverage result predicts in advance which games are cheap: 18% of the matrix for small-support games against 82% for Blotto. In growing-pool PSRO, RWPS reaches lower exploitability than minimum-regret-first search, information-gain search, and progressive sampling at a matched budget, and on the CyGym and ANSG cyber simulators it is lowest at the smallest budgets.

1 Introduction

Autonomous cyber defence is increasingly studied through simulation-based security games: a defender and an attacker act over a network, neither observes full state, and the value of any strategy pair is defined only by executing a simulator. This infrastructure is now standard. The Cyber Operations Research Gym [30, 31] underpins the CAGE challenge series; Yawning Titan [1], CyberBattleSim [18], and FARLAND [20] provide alternatives at varying abstraction; and CyGym [15] casts cyber defence explicitly as a two-player general-sum stochastic game solved by policy-space response oracles (PSRO) [14], with MetaDOAR [16] scaling the best-response step.

In this setting PSRO’s three inner operations have costs separated by orders of magnitude. Generating a candidate best response is a bounded reinforcement-learning run and can be restarted at will. Solving the restricted game, once a payoff matrix exists, is a small numerical problem that fictitious play [6] dispatches in milliseconds. Estimating a *single payoff entry*, by contrast, requires Monte-Carlo rollouts of the full simulator under both policies, and a rebuild touches $\Theta(n_D n_A)$ entries for pools of size n_D, n_A , growing quadratically over a run. The community has already identified simulation throughput as the binding constraint: CybORG++ [9] is motivated substantially by execution speed, and its MiniCAGE re-implementation reports up to a thousandfold faster parallel execution precisely so that large-scale training and evaluation become feasible. Making the simulator faster is one response, but the usual route to speed is reduced fidelity: abstracting host, process, and packet-level detail, as in CybORG’s split between its simulator and its virtual-machine emulation [30] and the explicit fidelity–performance trade-offs of network environment design [20]. Lower fidelity is

its own problem, since the abstracted detail is often exactly the realism the analysis was commissioned to capture. The complementary response, taken here, is to leave the simulator alone and call it less.

The natural design is to spend a fixed rollout budget only on entries whose value can change the equilibrium, and to fill the rest with a learned surrogate; we call the resulting method *Regret-Weighted Payoff Sampling* (RWPS). It has two parts. A surrogate trained on every entry simulated earlier in the run fills the matrix, so each purchase keeps paying across PSRO iterations. A regret-weighted score decides which entries to buy, and matters most when the budget is smallest. Is the speedup worth its cost? Answering that requires knowing what error the budget induces in the computed equilibrium, and what guarantee survives it. The available answer comes from the analysis of estimated games [33]: an equilibrium of the estimated game is a 2τ -equilibrium of the true game, where τ is the worst payoff error over *all* cells. Applied to a budgeted surrogate this bound is so loose as to be uninformative. The estimator deliberately concentrates its error on cells it judged unable to affect the equilibrium, and τ is set by exactly those cells. Under the prior analysis, then, the question of how much equilibrium quality the speedup costs is effectively open; answering it requires a bound that distinguishes errors the equilibrium can feel from errors it cannot. What our results ultimately show is that a payoff the equilibrium depends on but which an estimator can predict well is not worth simulating, and neither is an uncertain payoff no equilibrium strategy plays. Our contributions are as follows:

1. **Instance-dependent bounds for estimated games.** We refine the sup-norm bound of Tuyls et al. [33] so that only payoff errors weighted by the opponent’s equilibrium mixture enter a player’s regret, and a signed form in which only underestimated deviations and overestimated support can hurt (Lemma 1). We show that once the deviation-relevant set of $B^* = n_D s_A + n_A s_D - s_D s_A$ cells is simulated, a surrogate of arbitrary quality may fill the rest without affecting either player’s regret (Corollary 2), and give an optimistic certificate computable from simulation data alone (Corollary 1). On three 21×21 general-sum games the refined bounds are four to six times tighter than the sup-norm bound on the estimator’s own output.
2. **Regret-Weighted Payoff Sampling.** We give a budgeted payoff estimator for PSRO that fills unsimulated cells with a surrogate trained on every entry simulated earlier in the run, and buys cells by a score built from smoothed bootstrap equilibrium marginals, with forced exploration that Theorem 2 shows is necessary. In growing-pool PSRO it reaches lower exploitability than minimum-regret-first search, information-gain search [12], and progressive sampling at a matched budget. The fill carries 62% of the margin and the score 38%; with the fill held fixed, the score beats Sokota et al. [28] on Colonel Blotto ($p = 0.038$, 13/16 seeds).
3. **The Agentic Network Security Game.** We introduce ANSG, which extends CyGym [15] from binary node access to networks of LLM agents compromised by goal subversion through indirect prompt injection, with general-sum payoffs and partial observability on both sides. In budget sweeps on CyGym and ANSG the estimator reaches the lowest exploitability at the smallest budgets; on ANSG, removing the acquisition score leaves exploitability four to five times higher at budgets 1–3.

2 Related Work

Several simulators target cyber defence. CybORG [30, 31] drives the CAGE challenges; Yawning Titan [1], CyberBattleSim [18], and FARLAND [20] model networks at varying fidelity; CyGym [15] adds the game-theoretic layer this paper targets. All share the property that evaluating a strategy pair means running the simulator, and the cost of doing so is a recognised obstacle: CybORG++ [9]

reports up to a thousandfold parallel speedup from a lightweight re-implementation, motivated by the infeasibility of large-scale evaluation on the original. We are orthogonal to simulator engineering: we reduce the number of calls the equilibrium computation needs, and the two approaches compose.

The double-oracle method [17] and PSRO [14] grow strategy pools by best response to the current meta-game equilibrium; extensions target oracle quality and diversity [4], and structural studies of pools [8] inform what equilibrium supports look like in practice, a quantity our Corollary 2 makes decision-relevant. MetaDOAR [16] scales the best-response step for cyber networks. Little of this line targets the evaluation step, where cost concentrates in simulation-based games.

Our theory refines a known result. Tuyls et al. [32, 33] prove that a Nash equilibrium of an estimated meta-game is a 2τ -approximate equilibrium of the true game under sup-norm error τ . We take this as our point of departure. Lemma 1 replaces the sup-norm with an opponent-support-weighted error, which makes the bound informative for an estimator that deliberately leaves irrelevant cells inaccurate. Rowland et al. [25] allocate samples for evaluation under incomplete information, primarily for α -rank [22] response graphs, using bandit rules without a generalising surrogate.

Query-efficient EGTA attacks the same cost by pruning rather than interpolating. Progressive-sampling algorithms allocate samples adaptively and drop profiles once they can be shown irrelevant to the learning goal [2, 3, 7, 19]. Their objective is a uniform ε -approximation over surviving profiles, and the saving comes from cells that never need sampling to that accuracy. Ours is the complementary regime: every cell is present in the matrix we solve, filled by a surrogate when unsimulated, and the certificate is evaluated at the equilibrium actually returned rather than uniformly over profiles. Neither guarantee subsumes the other.

Wellman et al. [39] still describe MRFS-style heuristic search as the practical route where exhaustive evaluation is infeasible, so we treat it as a live baseline. We compare against both algorithms of Jordan et al. [12] and reach lower exploitability than both in a growing pool (Section 6.1). Neither carries information between iterations beyond its cache, whereas our surrogate trains on every cell bought at every previous iteration, so our advantage accumulates over a run. On a fixed pool, where no such history exists, MRFS is stronger at low budgets.

Closest to us on the model-learning side, Wang et al. [37] pair double oracle with a learned game model and regularisation for sample efficiency, though for mean-field games and without an explicit payoff matrix, and Wang and Wellman [36] study the complementary axis of which strategies to add rather than which profiles to evaluate. Earlier work in this vein learns payoff or deviation-payoff functions over strategy spaces [28, 35] within EGTA [34, 38].

Sokota et al. [28] pair a learned deviation-payoff model with sampling around candidate equilibria, the closest instinct to ours; they build no payoff table and give no bound. Picheny et al. [23] apply Bayesian optimisation to Nash computation in continuous black-box games with a Gaussian-process surrogate and a fixed game. Adaptive payoff queries for α -rank [24, 25] target a different solution concept. Active learning [26, 27] and deep ensembles [13] supply our acquisition and uncertainty machinery. Appendix A.3.1 compares each in detail.

3 Method

3.1 Preliminaries and problem statement

A simulation-based security game is a two-player general-sum game between a defender and an attacker. Each player holds a set of pure strategies; in our setting a pure strategy is a complete policy for the underlying stochastic game, for instance a trained neural controller for the defender’s network actions or the attacker’s intrusion actions. The game is *simulation-based* because its payoffs

are not given analytically: the expected utility of a strategy pair (d, a) is

$$U^D(d, a) = \mathbb{E}[u(d, a)], \quad U^A(d, a) \text{ symmetrically,}$$

where the expectation is over the simulator’s stochasticity and each sample of it costs a full rollout. The game is general-sum, $U^D + U^A \not\equiv c$, reflecting asymmetric valuations of the same outcome; nothing below assumes zero-sum structure.

Restricted to finite pools $S_D = \{d_1, \dots, d_{n_D}\}$ and $S_A = \{a_1, \dots, a_{n_A}\}$, the game is a bimatrix game with payoff matrices $D_{ij} = U^D(d_i, a_j)$ and $A_{ji} = U^A(d_i, a_j)$, scaled without loss of generality to $[0, 1]$. A solution is a mixed-strategy Nash equilibrium: distributions $(p, q) \in \Delta_{n_D} \times \Delta_{n_A}$ from which neither player can profitably deviate,

$$p^\top Dq \geq e_i^\top Dq \quad \forall i, \quad q^\top Ap \geq e_j^\top Ap \quad \forall j.$$

Quality is measured by exploitability, the largest deviation gain either player forgoes,

$$\varepsilon(p, q) = \max \left\{ \max_i (Dq)_i - p^\top Dq, \max_j (Ap)_j - q^\top Ap \right\}, \quad (1)$$

so $\varepsilon = 0$ is an exact equilibrium and, with scaled payoffs, ε reads as a fraction of the payoff range. A profile whose exploitability is at most ε is called an ε -Nash equilibrium.

Policy-space response oracles solve the full game by alternating three operations: solve the current restricted game for (p, q) ; train a best response to the opponent mixture with a reinforcement-learning oracle and add it to the pool; and extend the payoff matrices with the new strategy’s entries. In the simulation-based setting the first operation is nearly free, the second is a bounded training run, and the third costs N_{MC} rollouts per cell over $\Theta(n_D n_A)$ cells. The problem this paper addresses is the third step under a budget: given pools (S_D, S_A) , a cache of previously simulated entries, and a budget of $B \ll n_D n_A$ new cell evaluations, produce estimates $(\hat{D}, \hat{A})$ whose equilibrium is close, in the sense of Eq. (1), to an equilibrium of the true (D, A) .

The design question is which B cells to simulate, and what to write into the rest.

3.2 The budgeted estimator

Algorithm 1 gives the full procedure; its components are explained below in the order they appear, with lines 5–12 constituting the acquisition step and line 14 the forced exploration that Section 4 shows to be load-bearing. We call this *Regret-Weighted Payoff Sampling* (RWPS): the budget is allocated by each cell’s estimated influence on equilibrium regret, measured through the smoothed bootstrap marginals rather than through per-cell payoff uncertainty. The method has two separable parts, the history-trained fill and the acquisition score. Section 6.3 measures each; the fill carries most of the margin, and the score matters most at the smallest budgets.

Line 1 gives each policy a simulation-free representation. Fixing a probe bank $\Xi = \{\xi_1, \dots, \xi_P\}$ of states per role, collected once from a reference trajectory, a policy with actor μ_π receives

$$\varphi(\pi) = [\text{mean}_p \mu_\pi(\xi_p) \parallel \text{std}_p \mu_\pi(\xi_p)] \in \mathbb{R}^d, \quad (2)$$

at the cost of P forward passes. Non-parametric strategies receive fixed tags. The fingerprint describes behaviour rather than identity. How much payoff information it carries is a property of the game, which Section 5 varies deliberately.

Lines 3–4 fit an ensemble of M small networks mapping fingerprint pairs to payoff pairs, on bootstrap resamples of every entry ever simulated. Entries paid for at one PSRO iteration are

Algorithm 1 Regret-Weighted Payoff Sampling (RWPS)

Require: pools S_D, S_A ; cache $\mathcal{C}$ of simulated entries; budget B ; batch size c ; exploration probability ε_x ; smoothing ν ; bootstrap size B_{boot} ; ensemble size M

- 1: $\varphi(s) \leftarrow$ fingerprint of every $s \in S_D \cup S_A$ $\triangleright$ Eq. (2): forward passes only, no rollouts
- 2: $B \leftarrow B - |\{(i, j) \in \mathcal{C}\}|$ if B is a coverage target $\triangleright$ spend on the *uncovered remainder*, not the whole set
- 3: **for** $r = 1, \dots, \lceil B/c \rceil$ **do**
- 4: fit ensemble $\{f_\psi^{(m)}\}_{m=1}^M$ on $\{(\varphi(d_i), \varphi(a_j)) \mapsto \mathcal{C}[i, j]\}$
- 5: $(\hat{u}_{ij}, \hat{\sigma}_{ij}) \leftarrow$ ensemble mean, spread for all $(i, j) \notin \mathcal{C}$
- 6: **for** $b = 1, \dots, B_{\text{boot}}$ **do** $\triangleright$ uncertainty of the *solution*, not the cells
- 7: subsample $k_D \times k_A$ strategies; draw $\tilde{D}_{ij} \sim \mathcal{N}(\hat{u}_{ij}, \hat{\sigma}_{ij}^2)$ on unsimulated cells, pin cached cells
- 8: $(p^{(b)}, q^{(b)}, v^{(b)}) \leftarrow$ fictitious play on the realization
- 9: **end for**
- 10: $\bar{p}_i \leftarrow \text{mean}_b p_i^{(b)}$; $\bar{q}_j \leftarrow \text{mean}_b q_j^{(b)}$ $\triangleright$ marginals, not the averaged product
- 11: $\tilde{p} \leftarrow (1 - \nu) \bar{p} + \nu/|S_D|$; $\tilde{q} \leftarrow (1 - \nu) \bar{q} + \nu/|S_A|$ $\triangleright$ smooth toward uniform
- 12: $\alpha(i, j) \leftarrow \tilde{p}_i \tilde{q}_j \hat{\sigma}_{ij}$ $\triangleright$ zero on cached cells
- 13: **for** $t = 1, \dots, c$ **do** $\triangleright c$ cells per refit; the surrogate is refit between batches
- 14: with probability ε_x : simulate a cell drawn uniformly at random, unsimulated first, then revisits averaged by running mean $\triangleright$ Thm. 2
- 15: otherwise: simulate the unsimulated cell of largest α
- 16: add to $\mathcal{C}$
- 17: **end for**
- 18: **end for**
- 19: $\hat{D}_{ij}, \hat{A}_{ji} \leftarrow \mathcal{C}[i, j]$ if cached, else ensemble mean
- 20: **return** $(\hat{D}, \hat{A})$

training data at the next, which amortises simulation across a run. Queries return the mean $\hat{u}_{ij}$ and disagreement $\hat{\sigma}_{ij}$, with a rank-one row-and-column fallback until enough labels exist. Cached entries always override surrogate values.

Lines 5–12 measure uncertainty at the level of the solution rather than the cell. Simulating wherever $\hat{\sigma}$ is large targets the wrong object: the equilibrium is a functional of the whole matrix, and a maximally uncertain cell can be irrelevant because no equilibrium strategy plays it. The cost asymmetry makes the right object affordable. Re-solving is nearly free, so we re-solve B_{boot} bootstrap realizations of the game, each with a resampled pool and a matrix realization drawn from the surrogate’s predictive distribution. By the envelope theorem $\partial v / \partial D_{ij} = p_i q_j$ at fixed support, so the equilibrium value is first-order sensitive to a cell in proportion to the product of the two players’ probabilities of playing it.

Reading that identity off a single re-solve would score only the cells the current support estimate already plays. Two departures spread the score past it. First the product is formed from the bootstrap *marginals*, $\bar{p}_i \bar{q}_j$, rather than from the averaged product $\text{mean}_b p_i^{(b)} q_j^{(b)}$; the two agree only when the marginals are independent across replicates, and they are not, because a replicate that plays d_i tends to be one that plays a particular a_j against it. The averaged product therefore concentrates on the diagonal block of replicate-consistent pairs, while the marginal product also scores cells that pair a strategy some replicate plays with an opponent strategy a *different* replicate plays. Those cross pairs are exactly the ones a support estimate errs about. Second each marginal is

mixed toward uniform by ν , which places a floor under every cell in a played row or column and bounds the score’s concentration when the bootstrap is confident and wrong. Setting $\nu = 0$ and returning the product inside the average recovers value-sensitivity weighting, the rule used in earlier versions of this work, which Section 6.3 retains as the $\nu = 0$ ablation.

Forced exploration is the only mechanism that can cover cells the surrogate cannot know it is wrong about. The sensitivity rule is the correct objective under a calibrated surrogate, but no surrogate is calibrated on cells it has not sampled: perfect targeting would require the very payoffs the budget is meant to discover. Theorem 2 makes this precise. Two policies can share a fingerprint while differing arbitrarily in payoff, so no acquisition rule routed through the embedding carries the asymptotic guarantee, and uniform draws are the only mechanism that does not consult it.

The cache persists across PSRO iterations: cached values override the surrogate, every fit trains on all cells bought earlier, and a coverage-target budget pays only for the part of the deviation-relevant set not already held. Sokota et al.’s neighbourhood sampler moves mass toward the simplex edges, whereas ν -smoothing moves it toward uniform; Appendix A.3.2 discusses the contrast, the per-draw exploration form, and consistency, and Appendix A.6 gives the full procedure and hyperparameters.

4 Theory: an Instance-Dependent Bound

We now ask what an equilibrium of the estimated game $(\hat{D}, \hat{A})$ is worth in the true game (D, A) . The known answer is the sup-norm bound of Tuyls et al. [32, 33]. Writing $\Delta_D = \hat{D} - D$, $\Delta_A = \hat{A} - A$ and $\tau = \max\{\|\Delta_D\|_\infty, \|\Delta_A\|_\infty\}$, any exact equilibrium of the estimated game is a 2τ -equilibrium of the true one. This is the wrong instrument for a budgeted estimator. Our estimator is designed to leave irrelevant cells inaccurate, so τ is set by precisely the cells that do not matter, and the resulting certificate is correspondingly loose (see Section 6). The refinements in this section let the equilibrium itself say which errors count. We successively strengthen the certificate from a two-sided worst case to a signed, optimism-compatible bound that can be computed from the data the estimator already holds. All proofs are deferred to Appendix A.1.

4.1 Support-weighted perturbation

Lemma 1 (Support-weighted perturbation). *Let $(\hat{p}, \hat{q})$ be an $\varepsilon_{\text{solve}}$ -approximate Nash equilibrium of $(\hat{D}, \hat{A})$, meaning neither player gains more than $\varepsilon_{\text{solve}}$ by deviating in the estimated game. Write $\Delta_D = \hat{D} - D$ and define the opponent-support-weighted errors $\tau_D(\hat{q}) = \max_i(|\Delta_D|_{\hat{q}})_i$ and $\tau_A(\hat{p}) = \max_j(|\Delta_A|_{\hat{p}})_j$, with $|\cdot|$ entrywise. Then the defender’s regret in the true game satisfies*

$$\begin{aligned}
\text{reg}_D - \varepsilon_{\text{solve}} &\leq \underbrace{\max_i(-\Delta_D \hat{q})_i + \hat{p}^\top \Delta_D \hat{q}}_{(a) \text{ signed}} \\
&\leq \underbrace{\tau_D(\hat{q}) + \hat{p}^\top |\Delta_D| \hat{q}}_{(b) \text{ two-term}} \\
&\leq \underbrace{2\tau_D(\hat{q})}_{(c) \text{ support-weighted}} \leq \underbrace{2\tau}_{(d) \text{ sup-norm}},
\end{aligned} \tag{3}$$

where $\tau = \max\{\|\Delta_D\|_\infty, \|\Delta_A\|_\infty\}$, and symmetrically for the attacker. Consequently $(\hat{p}, \hat{q})$ is an ε -Nash equilibrium of (D, A) for ε equal to $\varepsilon_{\text{solve}}$ plus the maximum over players of any of (a)–(d). Equality in the last step holds only when a worst-error cell lies in a played column, respectively row.

The chain contains four bounds. Form (d) is the sup-norm bound of Tuyls et al. [32, 33], recovered as the loosest link. Form (c) is its support-weighted refinement, and form (b) keeps both terms instead of doubling the larger. Form (a) drops the absolute values, so that only *underestimated deviations* and *overestimated support* contribute.

We use different parts of the chain at different points below, and the parts are not interchangeable. Theorem 1 and Corollary 2 are derived from (b), since the absolute values are what allow a row to be split over simulated and surrogate cells with each class bounded by a magnitude. Corollary 1 instead rests on the sign structure of (a), which is what makes an optimistic fill admissible. Note that the solver term $\varepsilon_{\text{solve}}$ is computable without ground truth, since it is just the regret of $(\hat{p}, \hat{q})$ in the matrices the estimator already holds. We report it alongside the bounds in Section 6. It matters most for form (a), which has no slack to absorb it.

4.2 Finite-budget guarantee

Under the estimator, entries are of two kinds: simulated cells carry statistical error and surrogate cells carry representational error. The support-weighted bound splits accordingly. Assume rollouts return truth plus independent σ -sub-Gaussian noise, each simulated cell holds an average of at least m visits, and surrogate cells have absolute error at most β (whatever combination of regression error and embedding non-decodability produces it; we make no attempt to certify β a priori and measure it in Section 6). One subtlety requires care: which cells are simulated, and how often, is chosen by the acquisition rule from earlier samples, so the design is *adaptive* rather than fixed. A fixed-design union bound would be optimistic under adaptive selection. We therefore state the statistical term with a time-uniform concentration bound [11], valid simultaneously for all cells and all stopping times, at the cost of a log log factor in the visit count; the guarantee below then holds regardless of how the acquisition rule selects cells.

Theorem 1 (Finite-budget guarantee). *With probability at least $1 - \delta$, simultaneously over all cells and all sample-dependent stopping rules, any $\varepsilon_{\text{solve}}$ -approximate equilibrium $(\hat{p}, \hat{q})$ of $(\hat{D}, \hat{A})$ satisfies*

$$\varepsilon \leq \varepsilon_{\text{solve}} + 2 \max_{\pi \in \{\hat{p}, \hat{q}\}} \left[\zeta(m, \delta) + \beta \cdot w_{\text{sur}}(\pi) \right],$$

where $\zeta(m, \delta) = \sigma \sqrt{\frac{2(\log(4n_D n_A / \delta) + \log \log_2(2m))}{m}}$ is a time-uniform confidence radius valid under adaptive sampling, and $w_{\text{sur}}(\hat{q}) = \max_i \sum_{j: (i,j) \text{ surrogate}} \hat{q}_j$ is the largest opponent-support mass any deviation row places on surrogate cells, and symmetrically for $w_{\text{sur}}(\hat{p})$.

The multiplier w_{sur} is the key quantity here. Representational error is discounted by how much equilibrium mass actually touches surrogate cells, rather than entering at full strength as it does in the sup-norm bound. Two of our later results turn on this. Computing w_{sur} requires only a cache query against the returned mixture and consumes no rollouts, so its vanishing gives a stopping rule that we can check rather than assume (Proposition 1). Once it vanishes the bound retains only its statistical term, which we can invert to obtain the budget that a target exploitability requires (Corollary 3).

4.3 An optimistic certificate

The unsigned forms of Eq. (3) charge errors in both directions, whereas the signed form (a) does not, and an estimator can exploit that sign structure. Only two events produce regret: a deviation row whose true value the estimate *understates*, and a support whose value the estimate *overstates*.

Errors in the harmless directions are free. This is what makes optimism admissible. If we fill the unsimulated cells of a player’s own matrix at the maximum payoff, in the tradition of R-max [5], then no deviation is ever understated, the first term collapses to statistical noise on simulated cells, and only the support term remains. That term is controlled once the $s_D s_A$ support cells are simulated.

Optimism enters our analysis in two places, with opposite consequences. Used as a post-hoc fill that only *evaluates* the bound it is nearly free, and this gives Corollary 1. Used as a fill inside the *solve* it is not, and we treat that case immediately after.

Corollary 1 (Computable optimistic certificate). *Let $(\hat{p}, \hat{q})$ be an $\varepsilon_{\text{solve}}$ -approximate equilibrium of any estimated matrices $(\hat{D}, \hat{A})$, and let $(D^{\text{opt}}, A^{\text{opt}})$ equal the measured values on simulated cells and the maximum payoff elsewhere. If rollout noise is σ -sub-Gaussian and each simulated cell averages m visits, then with probability at least $1 - \delta$,*

$$\varepsilon \leq \max \left\{ \max_i (D^{\text{opt}} \hat{q})_i - \hat{p}^\top \hat{D} \hat{q}, \right. \\ \left. \max_j (A^{\text{opt}} \hat{p})_j - \hat{q}^\top \hat{A} \hat{p} \right\} + 2\zeta_{m,\delta} + \varepsilon_{\text{solve}},$$

where $\zeta_{m,\delta} = \zeta(m, \delta)$ is the same time-uniform radius as in Theorem 1, valid under adaptive sampling, and every quantity on the right is computable from simulation data alone.

The two uses have opposite cost. In Corollary 1 the optimistic matrices $(D^{\text{opt}}, A^{\text{opt}})$ appear only in the deviation term, which is evaluated once on the equilibrium the estimator has already returned. The fill never influences which strategies are played, and it costs one row-maximum. Used inside the *solve*, the same fill pulls every best response toward unpaid cells and certifies only after nearly the whole matrix is bought (Appendix A.3.3).

Corollary 2 (Deviation-relevant coverage). *Let $s_D = |\text{supp}(\hat{p})|$, $s_A = |\text{supp}(\hat{q})|$. If every cell of $\{1:n_D\} \times \text{supp}(\hat{q})$ and of $\text{supp}(\hat{p}) \times \{1:n_A\}$ is simulated, then $w_{\text{sur}}(\hat{p}) = w_{\text{sur}}(\hat{q}) = 0$ and the bound of Theorem 1 reduces to its statistical and solver terms, independent of surrogate quality. The set has*

$$B^* = n_D s_A + n_A s_D - s_D s_A$$

cells, against $n_D n_A$ for a full rebuild.

The principle that a profile’s regret depends only on its unilateral deviations is standard in EGTA [38, 39], and underlies MRFS [12]; setting $s_D = s_A = 1$ recovers its $2n - 1$ confirmation set. What we add is the closed-form bimatrix count and the coupling to Theorem 1. In prior treatments the complement of the deviation set is unevaluated; here it is *filled* by a surrogate the solver reads, and the corollary shows that the fill cannot reach either player’s regret. This is what makes interpolation admissible in place of pruning. Because the supports it needs are estimated by bootstrap re-solves rather than rollouts, B^* also serves as a screen: a small B^* says the estimator is worth enabling, a B^* close to $n_D n_A$ says a rebuild is the better option. Appendix A.3.3 prices a target certificate from B^* (Corollary 3) and gives the extended discussion.

Exploration is necessary for the asymptotic guarantee. A fingerprint is a finite summary, so two policies can share one while differing arbitrarily in payoff, and any deterministic acquisition rule reading only fingerprints and the cache can be led to never simulate the cell that matters (Theorem 2, Appendix A.3.3). The uniform draws of Algorithm 1 are the only mechanism that does not consult the embedding.

5 Experimental Setup

Every claim is measured against ground truth, which requires games whose true payoff matrix and exact equilibrium are computable. On a cyber simulator neither exists for a growing pool: the true payoff of a pair is the very Monte-Carlo quantity the method avoids computing. The synthetic games below are instruments for measuring the estimator, not threat models. For the two cyber simulators we instead fix a small policy pool, simulate its full matrix once as ground truth, and sweep the budget against it (Section 6.2).

5.1 Games

All are 21×21 (441 cells), payoffs scaled to $[0, 1]$, rollout noise $\sigma = 0.10$ of the range, $N_{MC} = 4$ episodes per cell evaluation, so a full rebuild costs 1764 episodes. They differ in what the fingerprint encodes and in equilibrium support size, the two quantities the theory says matter.

Latent-quality, informative embedding. Strategies carry latent qualities $q_s \sim \mathcal{N}(0, 1)$ with bilinear general-sum payoffs $D_{ij} = q_i^D q_j^A + 0.3q_i^D$, $A_{ij} = -q_i^D q_j^A + 0.3q_j^A$; the fingerprint $[q, \tanh q, q^2, \sin q]$ makes payoff smooth in φ . Equilibrium supports are $(2, 2)$, so $B^* = 80$ cells, 18%.

Latent-quality, arbitrary embedding. Identical payoffs with fingerprints drawn independently of q ; an ablation of the fingerprint, reported in Appendix A.7.

Colonel Blotto, asymmetric values. $S = 5$ soldiers over $F = 3$ fields, fields worth different amounts to the two players drawn from $[0.5, 2]$, hence general-sum ($\text{std}(D+A) = 0.21$ across cells). The fingerprint is the normalised allocation; payoff is combinatorial in it. Supports are $(12, 12)$, so $B^* = 360$ cells, 82%: the theory predicts this game is expensive before any run.

CyGym and ANSG. CyGym [15] is a network security game in which an attacker compromises nodes by gaining binary access and a defender responds over a partially observed network. The Agentic Network Security Game (ANSg) extends it to networks whose nodes are LLM agents. Compromise becomes *goal subversion*: an attacker injects instructions into content an agent reads, by indirect prompt injection or malicious tool triggering, so the agent pursues attacker goals while its legitimate task may still complete. Subverted agents can propagate payloads downstream. The defender audits, reprompts, resets, or isolates nodes on noisy signals; the attacker cannot observe resets and must probe to learn which nodes remain subverted. Valuations are independent per player, so the game is general-sum. ANSG keeps CyGym’s per-node state and PSRO machinery. Full details are in Appendix A.4.

5.2 Measurements

Exploitability ε , Eq. (4), of the returned mixture in the true game; lower is better, 0 is exact, and scaling makes it a fraction of the payoff range:

$$\varepsilon = \max \left\{ \max_i (Dq)_i - p^\top Dq, \max_j (Ap)_j - q^\top Ap \right\}. \quad (4)$$

Bound tightness: realised ε against 2τ , $2\tau_{\text{supp}}$, and the sharp form of Lemma 1, all computed from the method’s own output. Error decomposition and coverage studies are in Appendices A.10 and A.8.

5.3 Baselines and protocol

Uniform sampling spends the same budget uniformly at random with a mean fill; *full rebuild* simulates everything, PSRO’s default and ours. We additionally compare against minimum-regret-first search

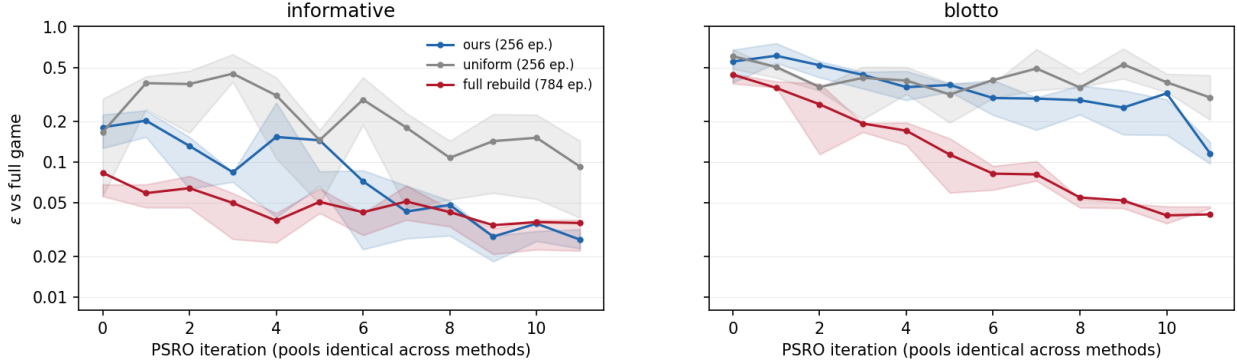


Figure 1: Exploitability against the full game over twelve PSRO iterations at a 5% per-build payoff budget. All methods draw the same initial pool and share a best-response oracle, but each responds to its own mixture, so the pools coincide at the first iteration and diverge thereafter. Means over sixteen seeds, interquartile bands.

and information-gain search [12] and the progressive-sampling family (Section 6.1). The latter is included to show that a uniform ε -approximation guarantee cannot be purchased at these budgets: its schedules require more samples per cell than the budget affords, so it returns the history-only estimate. The growing-pool comparison uses a 5% per-build budget over twelve iterations with pools growing from three to fourteen per side, five seeds; the fixed-pool diagnostic builds of Section 6.4 sweep $\{5, 10, 20, 40, 70\}\%$ of the matrix over sixteen seeds, measuring ε against the full game by zero-lifting the restricted mixture, with best-response episodes counted separately. All runs are deterministic given recorded seeds; τ -statistics are maxima over 441 cells and hence heavy-tailed, so we report medians with per-seed points shown. Seed counts are small and we present them as such. Code, data, and figures accompany the paper.

6 Results

All aggregates are mean $\pm$ one standard deviation unless a caption states otherwise; per-seed values appear as faint points in the figures. The growing-pool comparison uses sixteen seeds and the coverage evaluation of Table 6 sixteen; the fixed-pool builds used as a diagnostic bed in Section 6.4 use four. Each caption states its own count. The implementation, the verification suite, and the raw per-seed CSVs are released with the paper, and every number below regenerates from `run_all.py`.

6.1 Growing-pool PSRO

This is the setting the estimator is built for: pools grow, the cache persists, and every payoff entry costs a rollout. Table 1 reports twelve PSRO iterations at a 5% per-build budget over sixteen seeds, with exploitability measured against the full 21×21 game by zero-lifting the restricted mixture. All budgeted methods consume an identical 256 payoff episodes; the full rebuild is shown at its own 784 as a reference rather than a competitor.

The separation accumulates rather than appearing at any single build. Figure 1 plots the same runs iteration by iteration. All methods begin together, separate over the first four iterations, and hold their ordering for the rest of the run. This is what reuse of the cache predicts. On Blotto the estimator tracks the full rebuild for several iterations before the budget binds, while uniform sampling and the progressive-sampling family flatten early and stay flat.

Table 1: Growing-pool PSRO, 5% per-build budget, mean $\pm$ SD over sixteen seeds. ε_0 is the first iteration, where every method holds the same randomly drawn pool. Budget is cumulative payoff-rollout episodes over the run; every budgeted method is held to the same 256. The full rebuild has no budget—it simulates every uncached cell, costing 784 by iteration twelve—and is shown as a quality reference rather than a competitor. The three progressive-sampling members coincide because none prunes a single index at this budget.

Game	Payoff build	ε_0	ε_{12}	budget
Informative	RWPS (ours)	0.262	0.024 ± 0.017	256
	MRFS	0.180	0.091 ± 0.099	256
	IGS	0.168	0.113 ± 0.111	256
	GS / PSP / PS-REG-M	0.223	0.223 ± 0.174	256
	<i>full rebuild</i>	0.111	0.036 ± 0.023	784
Blotto	RWPS (ours)	0.525	0.082 ± 0.042	256
	MRFS	0.515	0.222 ± 0.114	256
	IGS	0.554	0.276 ± 0.133	256
	GS / PSP / PS-REG-M	0.627	0.627 ± 0.180	256
	<i>full rebuild</i>	0.398	0.045 ± 0.015	784

By iteration twelve the estimator reaches 0.024 against MRFS’s 0.091 and uniform’s 0.113, and on Blotto 0.082 against 0.222 and 0.293. Both margins are significant under a paired test over the sixteen seeds: against MRFS, $p = 0.022$ on the latent games and $p < 0.001$ on Blotto; against uniform, $p = 0.003$ and $p < 0.001$.

The estimator starts behind at iteration 0 (0.262 against MRFS’s 0.180), as expected with an empty cache and nothing to amortise; the gap then widens monotonically.

Against the full rebuild the result divides by game. On the latent games the estimator reaches 0.024 against the rebuild’s 0.036 at a third of the episodes, winning 13 of 16 seeds, but the paired difference is not significant ($p = 0.057$). We claim only that 5% of the cells cost no measurable quality, not that they buy more. On Blotto and on the arbitrary-embedding game the rebuild is ahead: 0.045 against 0.082 ($p = 0.004$), and 0.036 against 0.073 ($p = 0.036$). There the budget buys a real reduction in cost at a real cost in quality.

The result is end-to-end: each method’s oracle responds to its own mixture, so pools diverge after iteration 0, and the number credits the estimator for the strategies it leads PSRO to find. IGS and the progressive-sampling family trail for a common reason: both need more samples per profile than the budget affords. At a 14×14 pool a 5% budget is ten cell evaluations, against the 294 samples per cell PSP’s schedule requires, so GS, PSP and PS-REG-M prune nothing and coincide exactly. Appendix A.3.4 gives the details, including IGS converging toward MRFS as its per-profile allowance grows.

6.2 Budget sweeps on CyGym and ANSG

On both simulators RWPS reaches the lowest exploitability at the smallest budgets, and the other arms close the gap as the budget grows. On CyGym (Figure 2) both RWPS variants sit near the floor from budget 1, while uniform sampling and MRFS descend to it only around budget 8. IGS stays well above the floor across the sweep, and GS/PSP/PS-REG-M buys nothing at any budget shown. On ANSG (Figure 3) RWPS (adaptive) is lowest through budget 6; by budgets 11–13 the arms overlap within their bands.

The two simulators separate the acquisition score from the fill differently. On CyGym the

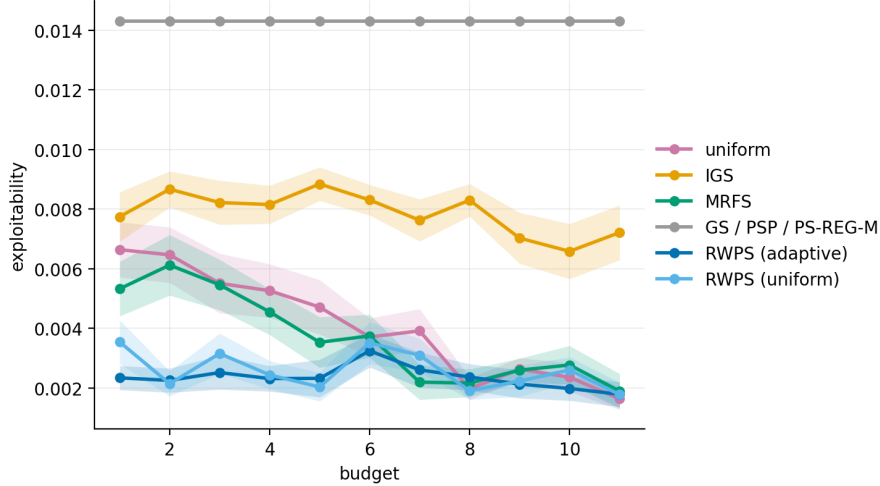


Figure 2: CyGym budget sweep. 6×6 pool with a 5×5 history pre-cached for every arm, leaving 11 cells to buy; 16 seeds. Exploitability is computed against a reference matrix estimated at 16 rollouts per cell. Bands are ± 1 standard error. GS/PSP/PS-REG-M is constant: its sampling schedule requires more samples per cell than the budget affords, so it returns the history-only estimate.

uniform-acquisition ablation tracks the adaptive rule closely, so the fill carries most of the margin. On ANSG the ablation is four to five times above the floor at budgets 1–3 and reaches it only at budget 4, so the score is what makes the first few purchases count. On all three simulators the ablation’s largest gap to the adaptive rule is at budget 1. Appendix A.5 reports the same sweep on Cyberwheel, where the pool is flat and every buying arm performs essentially the same.

6.3 Acquisition Ablation

Table 1 varies the payoff build as a whole, so its margin mixes two mechanisms: choosing which cells to simulate, and predicting the ones we skip. This section separates them. We fix the surrogate, the budget, the batch schedule, and the exploration rate, and vary only the acquisition score. Any difference between rows is therefore due to acquisition alone. All rules run on the same sixteen seeds, paired so that each begins from the same pool and the same oracle noise stream.

Uniform here samples cells at random but keeps the surrogate fill; Table 1’s uniform also replaces the fill with a flat value, so comparing the two isolates the fill.

The fill accounts for 62% of the total margin and the acquisition score for 38%. On the latent games the flat-fill baseline reaches 0.113, the surrogate-fill baseline 0.058, and the full method 0.024. On Blotto the same three are 0.293, 0.162, and 0.082. The split holds on both games to the nearest percentage point, although the two differ by a factor of three in difficulty and in whether the surrogate can fit the payoff surface at all. We did not tune for this and have no explanation for why it is so stable. Neither component is decorative.

The smoothed rule is first on both games. On Blotto it beats Sokota et al. [28] by 0.038, under both a paired t -test ($p = 0.038$) and a signed-rank test ($p = 0.016$), winning 13 of 16 seeds. On the latent games it leads by 0.007, which is not separable ($p = 0.224$). We report that as a tie.

Uncertainty sampling separates the two ingredients of the score. Scoring by surrogate disagreement alone beats uniform selection on both games, so $\hat{\sigma}$ carries real information about where the fill is wrong, but it loses to the full rule ($p = 0.021$ on Blotto, $p = 0.062$ on the latent games). Disagreement is therefore informative and not sufficient: a cell can be maximally uncertain and irrelevant because



Figure 3: ANSG budget sweep. 7×7 pool with a 6×6 history pre-cached for every arm, leaving 13 cells to buy; 16 seeds; exploitability computed exactly against the ground-truth matrix, so no measurement is clamped. Bands are ± 1 standard error. Two arms are omitted for legibility. GS/PSP/PS-REG-M is constant at 1.97×10^5 payoff units: its sampling schedule requires far more samples per cell than the budget affords, so it buys nothing and returns the history-only estimate. RWPS (uniform), the ablation with the acquisition score removed, averages 4.28×10^3 —it is 2.11×10^4 , 1.88×10^4 and 1.57×10^4 at budgets 1–3, and reaches the floor only at budget 4, where RWPS (adaptive) is already at it from budget 1.

no equilibrium strategy plays it, which is the argument of Section 3.2 measured rather than asserted. Appendix A.9.3 isolates the smoothing against the unsmoothed rule.

Two cautions belong with this table. We chose $\nu = 0.25$ on these games, an advantage the ported baseline does not have. The Blotto margin is larger than the spread of the ν sweep in Appendix A.9, but the latent-game ordering is within it. Uniform selection also has by far the widest spread of any rule, 0.064 against 0.017 for the smoothed rule on the latent games, so paired t -tests against it are underpowered relative to signed-rank tests. We report both rather than the more favourable one.

6.4 Why the estimator can afford to be wrong

Error off the equilibrium support is invisible to regret, and this holds on the estimator’s own output. Over all 240 budget-sweep runs every bound holds on every run (Figure 4). Our bound, the signed form (a) of Eq. (3) plus the measured solver term, averages 0.25 of the sup-norm bound, with a median overestimation factor of 2.3 against 11.4; forms (c) and (b) average 0.47 and 0.33, as the chain predicts. The estimator concentrates error on unplayed cells, the sup-norm is set by exactly those cells, and the support weighting removes them. The solver term is needed: without it the signed bound fails on three Blotto runs, each by less than that run’s solver regret (Appendix A.3.4). These are diagnostics, not certificates. Each form of Eq. (3) contains $\hat{D} - D$, so all 240 builds use a fixed-pool bed where the true matrix is available. Appendix A.10 gives per-game factors and the deployable certificate.

Table 2: Acquisition comparison. Identical surrogate, budget, batch schedule and forced-exploration rate; the rules differ only in the score used to rank uncached cells. Mean $\pm$ SD over sixteen paired seeds, ε_{12} . Uncertainty sampling scores by surrogate disagreement alone, the standard active-learning criterion [26, 29]. Uniform selection removes the score entirely while keeping the surrogate fill, so it is the control for the score rather than for the method. Appendix A.9.3 reports three further variants of our own rule.

Acquisition score	Informative	Blotto
ν -smoothed product (ours)	0.024 ± 0.017	0.082 ± 0.042
Sokota et al. [28]	0.032 ± 0.017	0.120 ± 0.045
uncertainty sampling ($\hat{\sigma}$ only)	0.040 ± 0.026	0.116 ± 0.050
uniform selection	0.058 ± 0.064	0.162 ± 0.077

7 Conclusion

Payoff estimation, not equilibrium solving, is what limits PSRO on cyber simulators, and this paper shows it no longer has to be paid in full. On a 5% per-build simulation budget, RWPS reaches lower exploitability than every budgeted alternative we tested. It does so with a guarantee. Our bounds are four to six times tighter than the sup-norm bound they replace; once we simulate every strategy against the opponent’s equilibrium strategies, the surrogate’s guesses for the remaining cells cannot change how exploitable the equilibrium is; and the bound can be computed from the simulations we ran, without knowing the true payoff matrix. On CyGym and on our LLM-agent game ANSG, RWPS is the lowest-exploitability method at the smallest budgets. Budgeted, certified equilibrium computation is now practical for simulation-based security games.

References

- [1] Alex Andrew, Sam Spillard, Joshua Collyer, and Neil Dhir. Developing optimal causal cyber-defence agents via cyber security simulation. *arXiv preprint arXiv:2207.12355*, 2022.
- [2] Enrique Areyan Viqueira, Amy Greenwald, Cyrus Cousins, and Eli Upfal. Learning simulation-based games from data. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1778–1780, 2019.
- [3] Enrique Areyan Viqueira, Cyrus Cousins, and Amy Greenwald. Improved algorithms for learning equilibria in simulation-based games. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 79–87, 2020.
- [4] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech M. Czarnecki, Julien Pérolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning (ICML)*, 2019.
- [5] Ronen I. Brafman and Moshe Tennenholtz. R-max: A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.
- [6] George W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 1951.

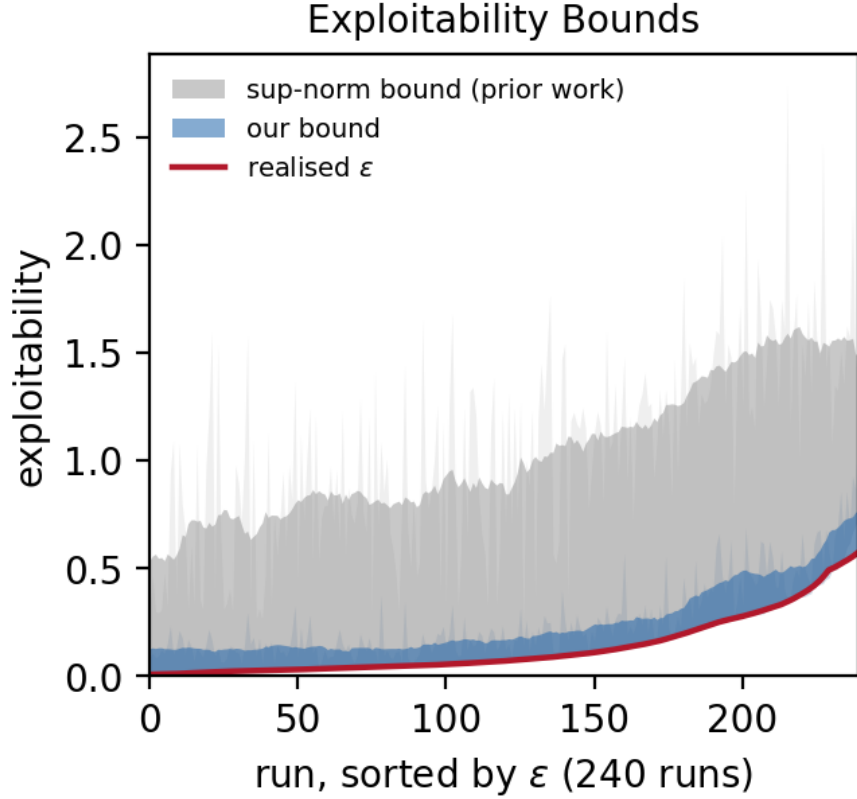


Figure 4: Realised exploitability and its bounds on the estimator’s own output, over all 240 fixed-pool budget-sweep runs on the three 21×21 games (the informative and arbitrary-embedding latent-quality games and Colonel Blotto; budgets $\{5, 10, 20, 40, 70\}\%$ of the matrix; 16 seeds each), sorted by ε . Shaded regions are each bound’s margin over ε , faint at full resolution and solid under a 21-run rolling mean; neither closes, and no bound is violated on any individual run.

- [7] Cyrus Cousins, Bhaskar Mishra, Enrique Areyan Viqueira, and Amy Greenwald. Computational and data requirements for learning generic properties of simulation-based games. *arXiv preprint arXiv:2208.06400*, 2022.
- [8] Wojciech M. Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [9] Harry Emerson, Liz Zhou, David Bowman, et al. CybORG++: An enhanced gym for the development of autonomous cyber agents. *arXiv preprint arXiv:2410.16324*, 2024.
- [10] Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. In *Journal of Machine Learning Research*, volume 13, pages 1809–1837, 2012.
- [11] Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021.
- [12] Patrick R. Jordan, Yevgeniy Vorobeychik, and Michael P. Wellman. Searching for approximate

- equilibria in empirical games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1063–1070, 2008.
- [13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
 - [14] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
 - [15] Michael Lanier and Yevgeniy Vorobeychik. CyGym: A simulation-based game-theoretic analysis framework for cybersecurity. In *Proceedings of GameSec*, LNCS 16223. Springer, 2026.
 - [16] Michael Lanier and Yevgeniy Vorobeychik. A scalable approach to solving simulation-based network security games. In *Proceedings of FLAIRS*, 2026.
 - [17] H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning (ICML)*, 2003.
 - [18] Microsoft Defender Research Team. CyberBattleSim. <https://github.com/microsoft/cyberbattlesim>, 2021.
 - [19] Bhaskar Mishra, Cyrus Cousins, and Amy Greenwald. Regret pruning for learning equilibria in simulation-based games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. arXiv:2211.16670.
 - [20] Andres Molina-Markham, Cory Minitier, Becky Powell, and Ahmad Ridley. Network environment design for autonomous cyberdefense. *arXiv preprint arXiv:2103.07583*, 2021.
 - [21] Sean Oesch, Amul Chaulagain, Brian Weber, Matthew Dixson, Amir Sadovnik, Benjamin Roberson, Cory Watson, and Phillipe Austria. Towards a high fidelity training environment for autonomous cyber defense agents. In *Proceedings of the 17th Cyber Security Experimentation and Test Workshop, CSET '24*, pages 91–99. Association for Computing Machinery, 2024. doi: 10.1145/3675741.3675752.
 - [22] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M. Czarnecki, Marc Lanctot, Julien Pérolat, and Rémi Munos. α -rank: Multi-agent evaluation by evolution. *Scientific Reports*, 9, 2019.
 - [23] Victor Picheny, Mickaël Binois, and Abderrahmane Habbal. A Bayesian optimization approach to find Nash equilibria. *arXiv preprint arXiv:1611.02440*, 2016.
 - [24] Tabish Rashid, Cheng Zhang, and Kamil Ciosek. Estimating α -rank by maximizing information gain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5673–5681, 2021. doi: 10.1609/aaai.v35i6.16712.
 - [25] Mark Rowland, Shayegan Omidshafiei, Karl Tuyls, Julien Pérolat, Michal Valko, Georgios Piliouras, and Rémi Munos. Multiagent evaluation under incomplete information. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

- [26] Burr Settles. Active learning literature survey. *University of Wisconsin-Madison Department of Computer Sciences Technical Report 1648*, 2009.
- [27] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016.
- [28] Samuel Sokota, Caleb Ho, and Bryce Wiedenbeck. Learning deviation payoffs in simulation-based games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.
- [29] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.
- [30] Maxwell Standen, Martin Lucas, David Bowman, Toby J. Richer, Junae Kim, and Damian Marriott. CybORG: A gym for the development of autonomous cyber agents. In *IJCAI-21 1st International Workshop on Adaptive Cyber Defense*, 2021.
- [31] TTCP CAGE Working Group. Cyber operations research gym. <https://github.com/cage-challenge/CybORG>, 2022.
- [32] Karl Tuyls, Julien Pérolat, Marc Lanctot, Joel Z. Leibo, and Thore Graepel. A generalised method for empirical game theoretic analysis. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018.
- [33] Karl Tuyls, Julien Pérolat, Marc Lanctot, Edward Hughes, Richard Everett, Joel Z. Leibo, Csaba Szepesvári, and Thore Graepel. Bounds and dynamics for empirical game theoretic analysis. *Autonomous Agents and Multi-Agent Systems*, 34(7), 2020.
- [34] Yevgeniy Vorobeychik and Michael P. Wellman. Stochastic search methods for Nash equilibrium approximation in simulation-based games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1055–1062, 2008.
- [35] Yevgeniy Vorobeychik, Michael P. Wellman, and Satinder Singh. Learning payoff functions in infinite games. *Machine Learning*, 67(1–2):145–168, 2007.
- [36] Yongzhao Wang and Michael P. Wellman. Evaluating strategy exploration in empirical game-theoretic analysis. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1346–1354, 2022.
- [37] Yongzhao Wang, Qiurui Ma, and Michael P. Wellman. Empirical game-theoretic analysis for mean field games. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1025–1033, 2023.
- [38] Michael P. Wellman. Methods for empirical game-theoretic analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2006.
- [39] Michael P. Wellman, Karl Tuyls, and Amy Greenwald. Empirical game-theoretic analysis: A survey. *Journal of Artificial Intelligence Research*, 2025. arXiv:2403.04018.
- [40] Bryce Wiedenbeck and Michael P. Wellman. Scaling simulation-based game analysis through deviation-preserving reduction. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 931–938, 2012.

- [41] Bryce Wiedenbeck, Ben-Alexander Cassell, and Michael P. Wellman. Bootstrap statistics for empirical games. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 597–604, 2014.

A Appendix

A.1 Proofs

This appendix collects the proofs of the results in Section 4, in order of appearance. Statements are not repeated; each proof is headed by a reference to the result it establishes.

Proof of Lemma 1. For any pure deviation i of the defender,

$$e_i^\top D\hat{q} = e_i^\top \hat{D}\hat{q} - e_i^\top \Delta_D \hat{q} \leq \hat{p}^\top \hat{D}\hat{q} + (|\Delta_D|\hat{q})_i = \hat{p}^\top D\hat{q} + \hat{p}^\top \Delta_D \hat{q} + (|\Delta_D|\hat{q})_i,$$

using optimality of $\hat{p}$ in the estimated game and the triangle inequality. Taking the maximum over i and bounding $\hat{p}^\top \Delta_D \hat{q} \leq \hat{p}^\top |\Delta_D| \hat{q} \leq \tau_D(\hat{q})$ gives the defender's bound; the attacker's is symmetric and no zero-sum structure is used. For the comparison, each row of $|\Delta_D|\hat{q}$ is a $\hat{q}$ -weighted average of entries bounded by τ , so $\tau_D(\hat{q}) \leq \tau$; columns of zero mass contribute nothing.

For (ii), repeat the chain without the triangle inequality: $e_i^\top D\hat{q} = e_i^\top \hat{D}\hat{q} - e_i^\top \Delta_D \hat{q} \leq \hat{p}^\top \hat{D}\hat{q} - e_i^\top \Delta_D \hat{q} = \hat{p}^\top D\hat{q} + \hat{p}^\top \Delta_D \hat{q} - e_i^\top \Delta_D \hat{q}$; take the maximum over i . The comparison follows since dropping absolute values can only decrease each term. $\square$

Proof of Theorem 1. Split $(|\Delta_D|\hat{q})_i$ over simulated and surrogate cells of row i . The simulated part is at most the concentration radius times $\sum_j \hat{q}_j \leq 1$, by mean concentration of m -visit averages and a union bound over all $2n_D n_A$ cells; the surrogate part is at most β times the $\hat{q}$ -mass row i places on surrogate cells. Take the maximum over rows, apply Lemma 1, and repeat for the attacker. $\square$

Proof of Corollary 1. With probability $1 - \delta$ all simulated entries lie within $\zeta_{m,\delta}$ of truth, so $D \leq D^{\text{opt}} + \zeta_{m,\delta}$ entrywise: on simulated cells by concentration and on unsimulated cells because payoffs are bounded by the fill. Hence $\max_i (D\hat{q})_i \leq \max_i (D^{\text{opt}}\hat{q})_i + \zeta_{m,\delta}$, while $\hat{p}^\top D\hat{q} \geq \hat{p}^\top \hat{D}\hat{q} - \zeta_{m,\delta}$ whenever the support cells are simulated, the deficit otherwise being visible in the computable first term. The attacker is symmetric. $\square$

Proof of Proposition 1. Coverage is monotone: a cell once simulated is never returned to the surrogate, so the set of cached cells is non-decreasing in t .

Let $\mathcal{R}_t$ denote the set consisting of every row simulated in all n_A columns together with every column simulated in all n_D rows, after round t . Round t simulates $\{1:n_D\} \times \text{supp}(\hat{q}_t)$, which completes every column of $\text{supp}(\hat{q}_t)$, and $\text{supp}(\hat{p}_t) \times \{1:n_A\}$, which completes every row of $\text{supp}(\hat{p}_t)$; hence $\text{supp}(\hat{p}_t) \cup \text{supp}(\hat{q}_t) \subseteq \mathcal{R}_t$.

Suppose the procedure does not halt after round $t + 1$, so $w_{\text{sur}}(\hat{q}_{t+1}) > 0$ or $w_{\text{sur}}(\hat{p}_{t+1}) > 0$. In the first case there are i and j with $\hat{q}_{t+1,j} > 0$ and (i, j) unsimulated, so column j is incomplete and therefore $j \notin \text{supp}(\hat{q}_t)$; round $t + 1$ simulates $\{1:n_D\} \times \text{supp}(\hat{q}_{t+1}) \ni (\cdot, j)$ and so adds j to $\mathcal{R}$. The second case is symmetric in rows. Every non-halting round therefore enlarges $\mathcal{R}$ by at least one element, and $|\mathcal{R}| \leq n_D + n_A$, which gives (i).

For (ii), halting means $w_{\text{sur}}(\hat{p}) = w_{\text{sur}}(\hat{q}) = 0$ at the mixture returned by the final solve, which is exactly the hypothesis of Corollary 2 evaluated at that mixture; the conclusion of Theorem 1 then loses its β term. For (iii), monotonicity bounds the total simulated set by the whole matrix. $\square$

Proof of the per-iteration accounting above. Write $\hat{p}, \hat{q}$ for the unchanged supports and $d_{\text{new}}, a_{\text{new}}$ for the added strategies. The set $\{1:n_D\} \times \text{supp}(\hat{q})$ acquires one new row, contributing the s_A cells (d_{new}, j) for $j \in \text{supp}(\hat{q})$; the set $\text{supp}(\hat{p}) \times \{1:n_A\}$ acquires one new column, contributing the s_D cells (i, a_{new}) for $i \in \text{supp}(\hat{p})$. The cell $(d_{\text{new}}, a_{\text{new}})$ lies in neither set, since by hypothesis neither new strategy is in a support. All other cells of the coverage set were simulated at the previous

iteration and are cached. A cached full rebuild instead simulates every newly exposed cell, of which there are $(k+1)^2 - k^2 = 2k+1$. $\square$

Proof of Corollary 3. Under coverage $w_{\text{sur}}(\hat{p}) = w_{\text{sur}}(\hat{q}) = 0$, so the bound of Corollary 1 reduces to $\varepsilon_{\text{solve}} + 2\zeta(m, \delta)$ and is at most ε_{tgt} exactly when $2\zeta(m, \delta) \leq \varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}}$. Since $\zeta(m, \delta)$ is strictly decreasing in m , the set of admissible m is an up-set and m^* is its minimum. The cost follows because coverage simulates B^* cells and each must carry m^* visits. For the rate, drop the $\log \log_2(2m)$ term, which is $o(\log m)$: solving $\sigma \sqrt{2 \log(4n_D n_A / \delta) / m} = (\varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}}) / 2$ gives $m = 8\sigma^2 \log(4n_D n_A / \delta) / (\varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}})^2$, and reinstating the $\log \log$ term inflates this by a factor $1 + o(1)$. $\square$

Proof of Theorem 3. For (i), coverage is monotone: a simulated cell is never returned to the surrogate. Let $\mathcal{R}$ be the set of rows simulated in every column together with the columns simulated in every row. By the argument of Proposition 1, whenever the iteration fails to halt it enlarges $\mathcal{R}$ by at least one element. Since $|\mathcal{R}| \leq n_D + n_A$ and $\mathcal{R}$ never shrinks, at most $n_D + n_A$ such extensions can occur across the whole sequence, however many times the mixture changes; after the last, every subsequent solve halts immediately and $w_{\text{sur}} = 0$ holds permanently.

For (ii), on the stated event Theorem 1 gives $\varepsilon \leq \varepsilon_{\text{solve}} + 2 \max_{\pi} [\zeta(m, \delta) + \beta w_{\text{sur}}(\pi)]$, and $w_{\text{sur}} = 0$ removes the second term. The radius $\zeta(m, \delta) = \sigma \sqrt{2(\log(4n_D n_A / \delta) + \log \log_2(2m)) / m}$ is decreasing in m with $\zeta \rightarrow 0$: the numerator grows doubly logarithmically while the denominator grows linearly, so $(\log(4n_D n_A / \delta) + \log \log_2(2m)) / m \rightarrow 0$ by L'Hôpital's rule on the ∞ / ∞ form. The $\log \log$ term therefore costs only a bounded factor at any practical m , and ζ retains the $m^{-1/2}$ decay of a fixed-design radius; this is why the budget of Corollary 3 inherits the fixed-design rate up to that factor. Crucially the bound is time-uniform: it holds simultaneously for all m and all sample-dependent stopping rules on a single event of probability $1 - \delta$, so the limit is taken on that event without a further union bound.

For (iii), $\varepsilon_{\text{solve}}$ is the exploitability of $(\hat{p}, \hat{q})$ in the estimated game and vanishes when the restricted game is solved exactly, giving $\varepsilon \rightarrow 0$. Since ε is exploitability in the true restricted game, the limit profile is an exact equilibrium of it. The cells outside the coverage set enter neither $\max_i (Dq)_i$ nor $\max_j (Ap)_j$ by Corollary 2, so their values are unconstrained. $\square$

Proof of Lemma 2. By Eq. (2), $\varphi(\pi)$ is a function of $\{\mu_{\pi}(\xi_p)\}_{p \leq P}$ alone, so any two policies agreeing on Ξ receive identical fingerprints whatever they do elsewhere. Payoffs, by contrast, depend on behaviour at all reachable states. Choosing π' to agree with π on Ξ and to realise the desired behaviour off it, which the hypothesis permits, gives the claim. $\square$

Proof of Theorem 2. Suppose for contradiction that some such $\mathcal{A}$, at budget B , simulates a trap cell on every twin-trap instance.

Consider the *all-duplicate* instance I_0 , in which a designated attacker strategy $a_{\star}$ is a genuine copy of a_{twin} : same fingerprint, same payoffs, likewise weak. Run $\mathcal{A}$ on I_0 and let $\mathcal{S}_B$ be the set of cells it simulates, so $|\mathcal{S}_B| \leq B < n_D n_A$. Since $\mathcal{S}_B$ omits at least one cell, and by enlarging the pool if necessary, choose an attacker column $j_{\star}$ that $\mathcal{S}_B$ does not meet.

By Lemma 2 there is a policy a_{trap} with $\varphi(a_{\text{trap}}) = \varphi(a_{\text{twin}})$ whose payoffs make it a dominant deviation. Let I_1 be the instance obtained from I_0 by installing a_{trap} in column $j_{\star}$, leaving everything else untouched. Then I_1 is a twin-trap instance, and I_0 and I_1 differ only in the payoffs of cells in column $j_{\star}$.

We claim $\mathcal{A}$ executes the same run on I_0 and I_1 . Argue by induction on rounds. The fingerprints agree, since $\varphi(a_{\text{trap}}) = \varphi(a_{\text{twin}}) = \varphi(a_{\star})$ and φ is a function of the actor alone, not of payoffs. Suppose the runs agree through round r , so the caches $\mathcal{C}_r$ agree as multisets of simulated cells. The

entries recorded there agree as values as well: every cell in $\mathcal{C}_r \subseteq \mathcal{S}_B$ lies outside column $j_\star$, where I_0 and I_1 are identical by construction. Hence $\mathcal{A}$ receives identical inputs at round $r + 1$ and, being deterministic, selects identical cells. The induction closes.

Therefore $\mathcal{A}$ simulates exactly $\mathcal{S}_B$ on I_1 , and $\mathcal{S}_B$ does not meet column $j_\star$, so $\mathcal{A}$ simulates no trap cell on I_1 . But I_1 is a twin-trap instance, on which the hypothesis asserts that $\mathcal{A}$ does simulate a trap cell. This is the contradiction, and it establishes the first claim.

For the exploitability claim, on I_1 the surrogate assigns column $j_\star$ its prediction for a_{twin} , since the two share a fingerprint and no label from $j_\star$ ever enters the cache. That prediction is uniformly weak, so the returned $\hat{q}$ places no mass on the trap. In the true game a_{trap} is a dominant deviation, so the attacker’s regret $\max_j (A\hat{p})_j - \hat{q}^\top A\hat{p}$ is at least $\eta := (A\hat{p})_{j_\star} - \hat{q}^\top A\hat{p}$, the gap between the trap’s true payoff against $\hat{p}$ and the value the estimated game reports. This is fixed by the instance and does not shrink with B .

Uniform draws escape the argument because the construction needs a deterministic $\mathcal{S}_B$ outside of which to place the trap. A rule visiting each unsimulated cell with probability bounded below admits no such placement, and reaches column $j_\star$ almost surely. $\square$

A.2 Fingerprint collisions

Lemma 2 (Fingerprint collisions exist). *Fix a probe bank $\Xi = \{\xi_1, \dots, \xi_P\}$ and let Π be a policy class in which, for any $\pi \in \Pi$ and any prescribed behaviour off Ξ , some $\pi' \in \Pi$ agrees with π on Ξ and realises that behaviour elsewhere. Then for every $\pi \in \Pi$ there is $\pi' \in \Pi$ with $\varphi(\pi') = \varphi(\pi)$ whose payoffs against a fixed opponent are arbitrary.*

The hypothesis on Π is mild. It asks only that agreement on P probe states does not determine behaviour everywhere, which fails only if Ξ happens to be a sufficient statistic for the policy class. For neural controllers on a state space larger than the probe bank, it holds.

A.3 Extended discussion

This appendix holds material condensed out of the main text.

A.3.1 Related work

We compare against three members and reach lower exploitability than all of them at a matched budget (Section 6.1). None prunes a single utility index at the budgets we study. PSP’s schedule begins at 294 samples per cell against the ten cell evaluations a 5% budget affords. A uniform PAC guarantee costs the sample complexity it requires, and in our regime it is not purchasable.

Sokota et al. [28] deserve particular mention, since they pair a learned model with an acquisition rule defined at the level of the solution, as we do. They train a network mapping mixed profiles to deviation payoffs and refine it by sampling in the neighbourhood of each candidate equilibrium, which is the same instinct that motivates our bootstrap sensitivity. Three things separate the settings. Their model takes a *mixture* as input and returns deviation payoffs directly, so no payoff table is built, whereas we estimate individual cells over policy embeddings and hand the completed matrix to a solver. They exploit role symmetry to scale in the number of players, while our games are two-player and asymmetric with pools that grow by best response. And their refinement is validated empirically rather than through a bound, which is the gap Theorem 1 and Corollary 2 are meant to close: we ask not only where to sample but what the resulting estimate can be shown to be worth.

Choosing which expensive evaluation to run is well trod [26, 27], and deep ensembles [13] supply our per-cell uncertainty. Picheny et al. [23] apply Bayesian optimisation to Nash equilibrium

computation in black-box games, the closest precedent for treating the equilibrium rather than a payoff as the object of interest; they target continuous strategy spaces with a Gaussian-process surrogate and a fixed game, where we target a finite pool that grows across PSRO iterations and reuse every entry ever simulated as training data. The departure common to both from standard BO is the target functional: we care about the equilibrium of the whole matrix, not pointwise accuracy, so uncertainty is measured by re-solving bootstrap realisations of the game, which the cost asymmetry makes nearly free. Their two criteria are probability of equilibrium and stepwise uncertainty reduction. The first has a discrete analogue we report as a variant in Appendix A.9.3, scoring a cell by the bootstrap frequency with which it enters a support rather than by a Gaussian-process posterior. The second does not port: it targets the probability that a cell *is* the equilibrium, which is defined for the pure equilibria of their discretised space but not for the mixed equilibria at supports (2, 2) and (9, 9) that our games have. Generalising stepwise uncertainty reduction to mixed equilibria is open, and we do not attempt it here.

A third line allocates payoff queries adaptively but targets α -rank rather than Nash: Rowland et al. [25] maintain confidence intervals over payoffs until the response graph is determined, and Rashid et al. [24] choose the next query by maximising information gain between an epistemic belief over α -ranks and the observed payoff. Exploitability against a Nash mixture is not what either optimises, so we do not run them as baselines; the information-gain criterion is in any case the one we do run, through Jordan et al. [12].

A.3.2 Method

The fill affects how quickly the coverage rule halts, not the validity of what it returns. Corollary 2 removes the fill from the bound, so this is worth stating precisely. The solver reads the complete matrix, so the fill decides which strategies are played, and therefore which cells the coverage rule of Proposition 1 asks for.

The construction is a response to the observation of Sokota et al. [28] that a learned payoff model should be accurate *around* a candidate equilibrium rather than at it. Their sampler perturbs the candidate and projects back onto the simplex, which moves mass toward the simplex edges and so concentrates it further; smoothing toward the uniform mixture moves in the opposite direction. Section 6.3 compares the two directly under an identical fill.

When the surrogate’s uncertainty is calibrated, a rule of this form inherits the behaviour of uncertainty-driven acquisition in Bayesian optimization, where any rule monotone in posterior uncertainty is consistent [10, 29]: a cell whose resolution would move the true support has nonzero sensitivity, is eventually queried, and drops out of w_{sur} (Theorem 1). The ν -floor strengthens this, since every cell in a played row or column keeps a score bounded below of the surrogate’s own uncertainty there.

Each selection independently ignores the score with probability ε_x and draws uniformly instead, with revisits averaged into the cache. The per-draw form matters at small batch sizes. A per-batch quota must be an integer, so at $c = 1$ it can only be 0 or 1, which either forces every draw to be uniform or front-loads all exploration into the opening batches and leaves the tail greedy. Theorem 2 requires each unsimulated cell to retain a probability bounded below of being drawn at *every* step, which the Bernoulli form supplies for any c .

RWPS interleaves surrogate fitting with simulation. Each of R rounds refits the ensemble on every cell ever simulated, then spends that round’s share of the budget on the cells its acquisition score ranks highest.

The cache persists across PSRO iterations and does three things. A fit at iteration k trains on everything bought earlier, which is what the amortisation result of Section 6 turns on. Cached values

always override surrogate predictions, so a cell simulated once is exact from then on. And when the budget is a coverage target rather than an exogenous constant, cached cells count toward it, so each iteration pays only for the part of the deviation-relevant set it does not already hold. The last of these makes the marginal cost constant in the pool size rather than growing with it. Until enough labels accumulate to fit the ensemble, predictions fall back to a rank-one row-and-column model, which is why the first acquisition round on a fresh pool is close to uninformed.

A.3.3 Theory

Using the same fill *inside* the solve, as R-max would, changes the object solved. Every unsimulated cell reads as maximal, so a best response points at it, it enters the support, is simulated, is revealed to be ordinary, and leaves – whereupon the next unsimulated cell is the maximal-looking one. Optimism inside the solve is a systematic pull toward whatever has not been paid for, with n_{Dn_A} cells to chase, and it certifies only after nearly the whole matrix has been bought (Appendix A.10). Optimism certifies a solution; it does not save simulation.

The principle behind this corollary is standard, and we do not claim it. That verifying a profile’s regret requires only that profile and its unilateral deviations is long established in EGTA [38]. Wellman et al. [39] state it directly, together with the observation that the count stays modest for mixed profiles when supports are small, and the same idea underlies minimum-regret-first search [12] and deviation-preserving reduction [40]. What we add is the closed-form count for the bimatrix case and, more consequentially, the coupling to Theorem 1.

That coupling is worth spelling out. In prior treatments the complement of the deviation set is *unevaluated*: it is pruned, deferred, or simply unknown, and the guarantee is stated over what has been sampled. In our setting it is instead *filled*, by a surrogate that the solver then reads, and the corollary establishes that the accuracy of that fill cannot reach either player’s regret. Note that w_{sur} is zero whatever β happens to be. This is what makes interpolation admissible in place of pruning, and it is what turns the set into a budget for a *budgeted estimator* rather than a stopping criterion for a sampler.

Read as a budget, the corollary converts equilibrium support size into a quantity we can screen before any rollout is spent. In particular, B^* is linear in the pool sizes and in the supports, so support size rather than matrix size sets the price. Section 6 evaluates the screen.

Corollary 3 (Budget for a target certificate). *Fix δ and a target $\varepsilon_{\text{tgt}} > \varepsilon_{\text{solve}}$. If the deviation-relevant set is covered, so that $w_{\text{sur}} = 0$, then the certificate of Corollary 1 is at most ε_{tgt} as soon as every simulated cell carries*

$$m \geq m^* = \min\{m : 2\zeta(m, \delta) \leq \varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}}\},$$

and the total rollout cost of achieving it is $B^ \cdot m^*$, with*

$$m^* = O\left(\frac{\sigma^2 \log(n_{Dn_A}/\delta)}{(\varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}})^2}\right)$$

up to the log log factor carried by the time-uniform radius.

We buy the two factors separately, and neither substitutes for the other. Here B^* fixes *which* cells are simulated and removes β , while m^* fixes *how tight* the bound is and shrinks ζ . Both are computable in advance, B^* from bootstrap supports and m^* by inverting ζ . A deployment can therefore price a guarantee before committing to it, instead of discovering after the fact what its budget bought. Section 6 reports that price. Because every quantity on the right of Corollary 1 is

computable from simulation data, the pair also applies where the true equilibrium is not available, which is the setting a deployment faces: a budget can be quoted against a target exploitability rather than the other way round. Coverage is cheap and precision is not. At $\sigma = 0.1$ on a 30×30 pool, 2ζ is roughly half the payoff range at $m = 4$ and reaching 0.05 needs $m^* \approx 512$, while pool size enters only logarithmically.

The relation to MRFS is exact rather than thematic, and we record it because it fixes the lineage. MRFS confirms a *pure* profile by evaluating its entire unilateral deviation set, which in a two-player $n \times n$ game is the row and the column through that cell, or $2n - 1$ evaluations. Setting $s_D = s_A = 1$ in Corollary 2 gives $n + n - 1 = 2n - 1$. In other words, the deviation-relevant set is MRFS’s confirmation criterion carried from a pure profile to a mixed one, with s_D and s_A replacing the singleton supports.

Theorem 1 carries two error sources, statistical and representational, and only the first is controllable by spending. The term β depends on whether fingerprints happen to be payoff-informative for the game at hand, it cannot be certified in advance, and by Theorem 2 it can be arbitrarily bad. Driving w_{sur} to zero deletes that term outright and leaves a purely statistical certificate, which is the guarantee a full rebuild would give, obtained without a full rebuild and without any assumption about surrogate quality.

The set that achieves this is exactly what Lemma 1 identifies as visible to regret. We need full columns on the opponent’s support, since those are the only columns that enter, and full rows on one’s own support, since deviations range over all rows. Nothing outside that set can affect either player’s regret, however wrong it is. Note that coverage does not simulate the matrix. The complementary $n_D n_A - B^*$ cells remain surrogate-filled and are read by the solver, and Corollary 2 establishes only that they carry zero weight in either player’s regret, not that they are absent.

Estimating B^* is cheap because it costs solves rather than rollouts. The support sizes it needs are not known before solving, but the bootstrap of Section 3.2 estimates them by re-solving many surrogate-filled realisations of the game. This consumes no simulation budget, since solving is negligible next to a rollout, and it follows the bootstrap-over-empirical-games methodology of Wiedenbeck et al. [41].

We can therefore use B^* as a *screen*. Before committing any rollouts, a practitioner can read off whether coverage is a small fraction of the matrix, in which case the estimator is worth enabling, or close to all of it, in which case a rebuild is the better option. That is the decision the method actually faces. Read as a prediction the estimate can of course be wrong, and Section A.8.2 shows that it does not have to be read that way.

Exploration is necessary

The asymptotic guarantee rests on the forced-exploration draws alone. They visit every cell infinitely often almost surely, running means converge, and Lemma 1 with $\tau_{\text{supp}} \rightarrow 0$ closes the argument; the two-phase Borel–Cantelli proof is standard and we omit it. This cannot be obtained any other way. A fingerprint is a finite summary, so distinct policies can share one while differing arbitrarily in payoff (Appendix A.2), and any rule reading only fingerprints and labels is defeated by that fact.

Theorem 2 (Necessity of exploration). *Call an attacker strategy a_{twin} a twin if it is densely simulated and uniformly weak, and a_{trap} a trap if $\varphi(a_{\text{trap}}) = \varphi(a_{\text{twin}})$ while its true payoffs make it a dominant deviation. Let $\mathcal{A}$ be any deterministic acquisition rule whose selections are a function of the fingerprints $\{\varphi(s)\}_{s \in S_D \cup S_A}$ and the cache $\mathcal{C}$ alone. Then for every budget $B < n_D n_A$ there is a twin–trap instance on which $\mathcal{A}$ simulates no trap cell, and on which the returned mixture has exploitability at least $\eta > 0$ with η determined by the instance and independent of B .*

A.3.4 Results

The result is end-to-end and does not isolate the matrix from the strategies. Pools are identical only at iteration 0. Each method’s best response is trained against its own mixture, so a better estimate steers the oracle to different strategies and the pools diverge. We measure exploitability against the full game by zero-lifting, which keeps pools of different composition comparable, but the number still credits the estimator for the strategies it leads PSRO to find. Appendix A.7 repeats the comparison with identical payoffs and uninformative fingerprints, which isolates what the embedding contributes.

Information-gain search finishes at 0.082 and 0.343, behind both MRFS and the estimator. IGS is the comparator Jordan et al. [12] recommend for noisy payoffs, where MRFS is their revealed-payoff algorithm, and they report IGS outperforming the ECVI benchmark. We implement it from their Section 6.2. The profile maximising the Kullback–Leibler information gain aggregated over its deviation set is sampled next, with p_s the probability that s has minimum regret under their point approximation, and an MRFS prefix supplies initial samples as in their IGS-MRFS-3.

The budget, not the algorithm, is what puts IGS behind. IGS allocates samples across profiles it has already visited; the original experiments run it at roughly twenty-eight samples per profile on a thirty-five-profile game. A budget that cannot visit every profile once denies it the quantity it reasons about. This is visible on a fixed pool: at 21×21 IGS improves from 0.177 to 0.041 as the allowance grows from 0.2 to 1.0 evaluations per profile, converging toward MRFS. We make the same observation about PSP and its relatives below.

The progressive-sampling family does not reach its own operating regime here. None of the three prunes a single utility index at a 5% per-build budget—at a 14×14 pool that budget is ten cell evaluations, against the 294 samples *per cell* PSP’s schedule requires before its bound certifies anything—so all three reduce to sampling a handful of profiles exhaustively and flat-filling the remainder, and coincide exactly. This is the regime argument of Section 6 reproduced in the deployment setting rather than a defect of those algorithms.

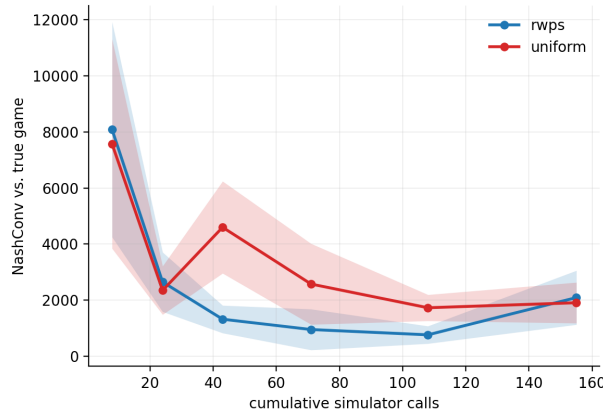


Figure 5: NashConv over cumulative simulator calls (lower is better) in CyGym. At low and high budgets, uniform sampling suffices, but across all budget levels, meets or exceeds performance.

The solver term earns its place. Without it the signed bound is violated on three runs, all on Blotto and all by less than the solver’s own regret on that run; fictitious play converges slowly on games with large supports, and raising its iteration count from 400 to 20,000 does not remove the residual (measured ϵ_{solve} averages 0.008 on Blotto against 0.002 on the informative game). Since ϵ_{solve} is computable from the estimated matrices alone, adding it costs nothing in applicability and restores validity on every run at a cost of 0.006 in mean tightness.

The table is needed because the rule of Sokota et al. [28] is a score over cells, not a payoff-build method, so it has no row in Table 1. Running it inside an otherwise identical build is the only fair comparison available to us.

The estimator starts behind. At iteration 0 it is at 0.262 against MRFS’s 0.180 and uniform’s 0.208. We expect this. The cache is empty at the first build, so there is nothing to amortise and the surrogate is a rank-one fit to almost no labels. What distinguishes the estimator is what it carries between builds, not what it does on one. The gap widens monotonically in the iteration index, which a per-build account does not predict.

Error off the equilibrium support is invisible to regret, and Figure 4 (Appendix A.3.4) measures that on the estimator’s own output. It plots all 240 budget-sweep runs sorted by realised exploitability, with the gap between ε and each bound shaded. No band closes, so every bound holds on every run. Our bound is the signed form (a) of Eq. (3) plus the measured solver term. It averages 0.25 of the sup-norm bound, with a median overestimation factor of 2.3 against 11.4, and the intermediate forms fall where the chain predicts: the support-weighted form (c) averages 0.47 and the two-term form (b) 0.33. The estimator concentrates error on unplayed cells, the sup-norm is set by exactly those cells, and the support weighting removes them. The asymmetry the design assumes is therefore real and not an artifact of a loose analysis.

Latent-quality, arbitrary embedding. Identical payoffs, fingerprints drawn independently of q : identity without payoff information. This isolates whether the surrogate’s benefit is generalisation across embedding space or amortisation of row and column effects. It is an ablation of the fingerprint rather than a third domain, and is reported as such in Appendix A.7.

Error decomposition: rebuild with one error source replaced by truth and re-solve, isolating statistical from representational contributions. **Coverage corollary:** simulate exactly the deviation-relevant set of Corollary 2, surrogate elsewhere, and compare ε to a full rebuild.

The uniform row here is not the uniform row of Table 1. Here uniform samples cells at random but still fills the rest with the surrogate. In Table 1 it also replaces the surrogate with a flat value. Comparing the two isolates the fill.

A.4 The Agentic Network Security Game

This appendix gives the full specification of the ANSG introduced in Section 5.

CyGym [15] models compromise as binary node access: the attacker exploits a node and thereby controls it. In a network of LLM agents the relevant failure is different. An agent processes workloads by invoking tools, and an adversary who plants instructions in content the agent reads can cause it to execute attacker-chosen goals *in addition to* the ones its principal asked for. The legitimate task still completes, so the subversion need not look like a failure at all. ANSG replaces access-compromise with this *goal subversion* model, keeping CyGym’s binary per-node state and its PSRO solution machinery while changing what compromise means and what each player can observe.

ANSG is a two-player general-sum partially observable stochastic game

$$\mathcal{G} = \langle \mathcal{S}, O^D, O^A, \mathcal{A}^D, \mathcal{A}^A, \mathcal{T}, U^D, U^A, \gamma \rangle$$

over a directed communication graph $G = (\mathcal{V}, \mathcal{E})$ whose nodes host agents. Each node carries a behavioural mode $b_v \in \{\text{benign}, \text{malicious}\}$ and a context age h_v . Both players choose policies simultaneously without observing the other’s, so Nash equilibrium is the solution concept and PSRO the approximation method, as in CyGym.

Attacker and defender valuations are independent rather than opposed: the attacker gains the value of the subverted goals it executes, while the defender loses the value of its workloads being corrupted, and these need not sum to a constant. Per-goal value factors differ by player. This is the

property that makes ANSG a natural test for a budgeted estimator whose theory assumes nothing about zero-sum structure.

The defender may *audit* a node, receiving a noisy signal whose detectability rises with the value of the subverted goal; *reprompt* it, a soft intervention with no CyGym analogue; *reset* it, which restores benign behaviour but erases context history at a cost; or *isolate* it, severing communication edges and halting propagation. The attacker may *inject* a payload into a workload, choosing a target, a goal rank, and a channel, or *probe* to estimate which previously compromised nodes remain malicious.

Injection carries a channel choice χ . Indirect prompt injection appends an attacker goal alongside the legitimate workload, which therefore completes normally; the only signal is a rank-dependent trace. Malicious tool triggering instead redirects an existing call, denying the legitimate output for a more damaging hit at higher detection probability. The rank choice sets value against exposure, and the channel choice compounds it.

Both players are partially informed, which is the substantive departure from CyGym. The defender sees only noisy audit signals. The attacker cannot observe resets, and because a reset benign node is behaviourally indistinguishable from one never compromised, must probe to estimate the current state. Compromise also propagates: a malicious agent can embed payloads in its outputs, infecting downstream agents along $\mathcal{E}$.

ANSG is the harder of the two deployment games for a budgeted build, for reasons the theory anticipates. Payoffs require full rollouts of a partially observable game, so simulation dominates iteration cost and the premise of Section 6 holds strongly. Equilibrium supports are not known in advance, so B^* must be estimated from bootstrap replicates rather than read off a solved game. And no ground-truth equilibrium is computable, so the certificate of Corollary 1 is the only available quality measure rather than a diagnostic checked against a known answer.

A.5 Budget sweep on Cyberwheel

When the pool is flat, every arm that buys cells returns essentially the same equilibrium, and budgeted acquisition has nothing to separate. Figure 6 repeats the budget sweep of Section 6.2 on Cyberwheel [21]. Uniform sampling, IGS, MRFS, and both RWPS variants all sit within about 0.015 of zero exploitability at every budget, with overlapping bands from budget 2 onward. The one visible separation is at budget 1, where RWPS (uniform) is highest of the buying arms. GS/PSP/PS-REG-M is constant near 0.16 because, as on CyGym and ANSG, its sampling schedule cannot afford a single cell and it returns the history-only estimate. The pool is flat in a measurable sense: on this 6×6 pool the payoff signal-to-noise ratio is below 1 for both roles, so no cell’s simulated value is distinguishable from rollout noise.

This is the outcome Corollary 2 predicts for a matrix on which no cell is much more deviation-relevant than another: if the surrogate is already accurate everywhere, which cells are bought barely changes the equilibrium. We report it as the regime in which RWPS offers no advantage over simple baselines, and in which it also costs nothing.

A.6 Implementation and computational procedure

This appendix gives the full RWPS procedure as configured for the experiments of Section 5; Section 3.2 summarises it.

Given a budget B over R rounds, each round performs four steps.

1. **Fit.** The ensemble of M networks is refit on the entire cache—every cell ever simulated, in this build and in all previous ones—each network on its own bootstrap resample of the labels. Inputs

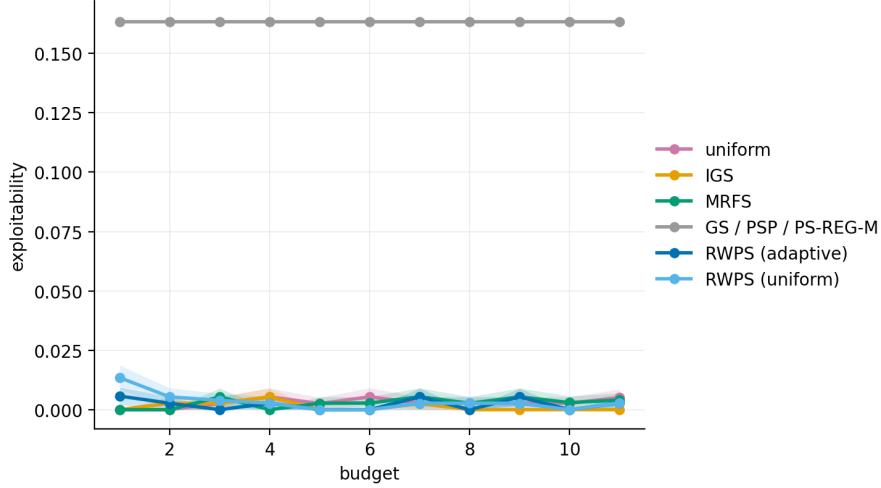


Figure 6: Cyberwheel budget sweep. 6×6 pool with a 5×5 history pre-cached for every arm, leaving 11 cells to buy; 16 seeds. Exploitability is computed against the stored payoff matrix of the pool. Bands are ± 1 standard error. All buying arms coincide within their bands; GS/PSP/PS-REG-M buys nothing and returns the history-only estimate.

and targets are standardised per fit.

2. **Predict.** The fitted ensemble predicts the full grid, returning a mean $\hat{u}_{ij}$ and a disagreement $\hat{\sigma}_{ij}$ on every unsimulated cell.
3. **Bootstrap.** B_{boot} realisations of the game are drawn and re-solved by fictitious play to give W and ς . This step consumes no simulation budget; it costs only solves, which are cheap by the cost asymmetry the method exploits.
4. **Simulate.** The round’s share of the budget is spent: c cells, each drawn uniformly at random with probability ε_x and otherwise taken by descending acquisition score. New labels enter the cache and become training data for the next round’s fit.

Real payoffs are therefore evaluated R times per build, in batches of roughly B/R cells, and the surrogate is refit $R + 1$ times: once per round, plus a final fit before the matrices are committed.

The cache lives on the simulator rather than on the builder, so it is persistent across PSRO iterations: a fit at iteration k trains on everything bought at iterations $1, \dots, k - 1$ as well. Cached values always override surrogate predictions, so a cell simulated once is exact from then on. Model state is also carried across builds: the ensemble is warm-started from the previous `build()` call rather than reinitialised, which leaves final exploitability unchanged (Welch $p = 0.81$ informative, $p = 0.62$ Blotto over eight seeds) and reduces payoff-build wall clock by 17–20%. The scope matters and is deliberately narrow: warm-starting the R internal rounds of a single build was measured to hurt, because the cache roughly doubles between rounds at a 40% budget while a +1 pool growth enlarges it by only about a fifth, so a short warm schedule tracks the second and not the first. Rounds $2..R$ and the final commit fit are therefore forced cold, and only a build’s opening fit continues from the previous build. Under a coverage budget the cache is also read for budget accounting: the target is B^* evaluated at the current supports, and the amount actually spent is that target less the deviation-relevant cells already in $\mathcal{C}$, so the per-iteration spend is the uncovered remainder rather than B^* itself. Repeat visits arise only from the exploration draws, and only once every unsimulated

Table 3: Growing-pool PSRO on games with identical payoffs and different fingerprints; final exploitability, mean over five seeds, all budgeted methods at a budget of 256. Only the estimator reads fingerprints, so only its row moves.

Payoff build	informative	uninformative	Δ
RWPS (ours)	0.045	0.060	+0.015
MRFS	0.053	0.053	0
IGS	0.082	0.082	0
GS / PSP / PS-REG-M	0.269	0.269	0
<i>full, unbudgeted</i>	0.035	0.035	0

cell has been visited; they are averaged into a running mean rather than overwriting, so a cell visited k times carries the mean of its k estimates and its effective N_{MC} is k times the nominal value.

Until the cache holds enough labels to fit the ensemble (eight, in our configuration), predictions fall back to a rank-one row-and-column model: row and column means of the observed entries, with the residual standard deviation serving as $\hat{\sigma}$. The first acquisition round of the first build on a fresh pool is therefore close to uninformed, which at the smallest budgets is the dominant source of seed-to-seed variance.

Unless stated otherwise we use $M = 3$ ensemble members, each a two-hidden-layer MLP of width 48 trained for 60 epochs with Adam at learning rate 5×10^{-3} ; $B_{boot} = 32$ bootstrap replicates subsampling 80% of each pool; $R = 3$ rounds; exploration probability $\varepsilon_x = 0.15$ applied as an independent Bernoulli per selection, batch size $c = \lceil B/3 \rceil$ unless stated (both ablated in Section A.9.2); and $\kappa = 0.25$. Fictitious play runs 400 iterations for the final solve and 200 within the bootstrap. Each simulation call averages $N_{MC} = 4$ rollouts, and rollout noise is Gaussian with $\sigma = 0.10$ on payoffs scaled to $[0, 1]$. Games are 21×21 , giving 441 cells; all reported budgets are fractions of that. Probe banks hold $P = 32$ states per role.

A.7 What the embedding contributes

The latent-quality game admits a controlled ablation of the fingerprint. Holding the payoff matrices fixed and redrawing the fingerprints independently of latent quality yields a game identical in every respect except that the embedding carries no payoff information. Table 3 repeats the growing-pool comparison of Section 6.1 on both versions.

Every method that does not consult the embedding is unchanged to the reported precision, which is a check on the construction as much as a result. The estimator is the only row that moves, so 0.015 is a direct measurement of what payoff-informative fingerprints are worth in this game.

The decomposition that follows is the substantive point, and it is not the one a reader would guess from the method description. The estimator’s advantage over uniform sampling splits into a part that survives an uninformative embedding, $0.092 \rightarrow 0.060$, and a part attributable to generalisation across embedding space, $0.060 \rightarrow 0.045$. The first is the larger. With fingerprints that are pure noise the surrogate still amortises row and column effects from cells already simulated—it continues to learn that a given strategy tends to score well or badly against the pool, which requires no embedding at all—and that alone recovers most of the gap. This tempers the natural reading of payoff regression over policy embeddings, and it is why the rank-one row-and-column fallback of Appendix A.6 is not merely a cold-start device: it is a substantial fraction of what the surrogate does throughout.

It also explains where the estimator stops winning. On the uninformative game MRFS reaches 0.053 against the estimator’s 0.060, inside one standard deviation. MRFS obtains the same row-and-column structure directly, by expanding a profile into its unilateral deviations, and needs no

embedding to do it. When the embedding contributes nothing there is nothing left for the estimator to win with, and the two methods are measuring the same thing by different means. We report this as the intended outcome of the ablation rather than as a defeat: the uninformative game exists to isolate the embedding term, and a near-tie with a model-free method is the correct result when that term is zero by construction.

A.8 Coverage as a fixed point

Corollary 2 identifies the deviation-relevant set B^* and Corollary 3 prices a target certificate against it. This appendix turns the pair into an iteration and gives the asymptotic guarantee that follows. The finite-budget result the paper relies on is Theorem 1, which does not depend on anything here.

A.8.1 The budget a target certificate requires

Corollary 1 reports what a completed build certifies. Run backwards, it answers the question a practitioner actually asks, namely how much simulation buys a stated guarantee. The answer is available before any of that simulation is spent.

A.8.2 Coverage as a fixed point

We do not treat B^* as a prediction that has to be got right. Instead we treat it as the first iterate of a procedure that comes with a free stopping test. The test is $w_{\text{sur}} = 0$, which holds exactly when the deviation-relevant set of the returned $(\hat{p}, \hat{q})$ has been covered, and which is a cache-membership query costing no rollouts. This dissolves the apparent circularity in reading B^* as an a-priori budget. The supports it needs are a property of the solve that it funds, but the procedure does not need them to be correct, only to be verified at the end.

Proposition 1 (Coverage iteration). *From any cache, iterate: solve $(\hat{D}_t, \hat{A}_t)$ for $(\hat{p}_t, \hat{q}_t)$; simulate every uncached cell of $\{1:n_D\} \times \text{supp}(\hat{q}_t) \cup \text{supp}(\hat{p}_t) \times \{1:n_A\}$; refit and repeat, halting when $w_{\text{sur}}(\hat{p}_t) = w_{\text{sur}}(\hat{q}_t) = 0$. Then*

- (i) *the procedure halts after at most $n_D + n_A$ rounds;*
- (ii) *at halting the hypothesis of Corollary 2 holds for the equilibrium actually returned, so $\varepsilon \leq \varepsilon_{\text{solve}} + 2\zeta(m, \delta)$ with no dependence on β ;*
- (iii) *total simulation never exceeds $n_D n_A$ cells, so the procedure is never more expensive than the full rebuild it replaces.*

Halting is by exhaustion of rows and columns, so the sequence of mixtures need not converge, and an oscillating equilibrium costs additional rounds and nothing else. The procedure is sound and terminating but not minimal, since an intermediate mixture can pull in rows for a support that later vacates. Note that the two error directions are not symmetric. Overshooting the support buys unnecessary cells while keeping the guarantee, whereas undershooting forfeits it, so we seed the first iterate with the *union* of the bootstrap supports rather than with a threshold on their average.

Theorem 3 (Convergence under coverage). *Consider a sequence of builds in which the iteration of Proposition 1 is run to halting whenever the returned mixture changes, and in which every cell of the coverage set continues to be sampled. Then, on the event of probability at least $1 - \delta$ on which Theorem 1 holds,*

- (i) *at most $n_D + n_A$ coverage extensions occur over the entire sequence, after which $w_{\text{sur}}(\hat{p}) = w_{\text{sur}}(\hat{q}) = 0$ holds permanently;*

- (ii) thereafter $\varepsilon \leq \varepsilon_{\text{solve}} + 2\zeta(m, \delta)$, and $\varepsilon \rightarrow \varepsilon_{\text{solve}}$ as $m \rightarrow \infty$;
- (iii) hence, with the restricted game solved exactly, the returned mixture converges to an exact equilibrium of that game, while the $n_D n_A - B^*$ surrogate-filled cells are never corrected and may remain arbitrarily wrong.

A single confidence event supports every m here, which is what the time-uniform radius buys us: a fixed-design bound holds at one prescribed sample size, so letting $m \rightarrow \infty$ under it would require a union over sample sizes. Neither half of the result is an assumption. We reach coverage in finitely many rounds using a stopping test that is a cache query, and the vanishing of ζ follows from continued sampling of the cells that coverage identifies. Coverage decides which cells convergence is possible on and visits decide how fast, so spending alone does not substitute for coverage. The contrast with Section A.3.3 is one of cost: that argument needs forced exploration to visit every cell infinitely often, $\Theta(n_D n_A)$ cells, whereas this one needs only the $\Theta(n_D s_A + n_A s_D)$ cells of the coverage set.

Coverage guarantees the bound, not the solve. Driving w_{sur} to zero removes β from Theorem 1 by construction, so the bound holds whatever the surrogate predicts. The certificate no longer depends on surrogate quality. The surrogate has not ceased to matter. Which equilibrium the solver arrives at is still a function of the fill, because the solver reads the complete matrix and the unsimulated cells are exactly the cells that determined they were unplayed. The surrogate therefore controls rounds to termination rather than the validity of what the iteration returns. The $\beta \cdot w_{\text{sur}}$ term cannot express this role, since coverage sets it to zero. At B^* coverage a surrogate fill and a flat fill carry the same guarantee, and Section 6 separates them empirically.

Coverage separates the budget into two parts that are priced independently. Coverage decides which cells are bought and removes $\beta \cdot w_{\text{sur}}$. Revisits decide how tight the bound is and shrink $2\zeta(m, \delta)$. Only the first is a question about the matrix, and it is the cheap one, costing $s_D + s_A$ cells per iteration under stable supports. The second is a visit count, and at small m it dominates, so a covered build certifies very little until m grows. A deployment can invert this: fix the exploitability bound required, take m to be the smallest visit count with $2\zeta(m, \delta)$ below it, and the rollout budget follows as the coverage set times m . Section 6 prices this.

The iteration trades a spending cap for a guarantee. A budget fixed as a fraction of the matrix bounds what a build can cost. The iteration instead spends what the game demands, which is $s_D + s_A$ cells per iteration under stable supports and, in the worst case, the full matrix. Under a hard rollout ceiling it may be truncated before it halts. Nothing is lost when that happens. Stopping with $w_{\text{sur}} > 0$ returns us to Theorem 1 with the representational term retained at the measured w_{sur} , which is the bound the estimator carries in any case. Halting is what removes β ; truncation degrades to the general bound rather than forfeiting it.

Fingerprint collisions do not defeat coverage. Theorem 2 constrains acquisition rules that read fingerprints, and its proof plants the trap in a column the rule never touches. Coverage admits no such column: $w_{\text{sur}}(\hat{p}) = 0$ requires $\text{supp}(\hat{p}) \times \{1:n_A\}$, so every attacker strategy is simulated against the defender's support whatever its fingerprint reports, and the construction has nowhere to place the trap. If a trap's true payoffs make it a profitable deviation against $\hat{p}$, covering reveals them, the re-solve moves mass onto it, w_{sur} becomes positive, and the next round completes its column. If it is profitable only against strategies outside $\text{supp}(\hat{p})$, then it does not enter $\max_j (A\hat{p})_j$ and is not a deviation by which the returned mixture can be exploited. Forced exploration remains load-bearing for the acquisition rule of Section 3.2, whose score is a function of the embedding, but it is not needed for the coverage guarantee.

Coverage costs $s_D + s_A$ cells per PSRO iteration whenever a new strategy pair leaves the supports unchanged. With a persistent cache the honest baseline is not a $\Theta(n_D n_A)$ rebuild. It is the $2k+1$

cells that a pool of size k per side newly exposes, an incremental-caching accounting standard in EGTA [38, 40]. The result is immediate: those are the only cells of the deviation-relevant set the cache does not already hold, which is to say the only cells at which $w_{\text{sur}} \neq 0$. The condition is support stability rather than support size, and it is the regime PSRO enters near termination. The saving accrues late in a run rather than eroding.

Each result above attaches to a different part of the estimator, and none of them derives the acquisition rule. The surrogate fill steers *which* equilibrium is solved for. Coverage makes the *bound* on that equilibrium independent of the fill (Proposition 1). Corollary 1 *certifies* the result post hoc in settings where no ground truth exists, and Corollary 3 inverts that certificate into a budget.

One caveat is worth stating. Our acquisition rule targets value sensitivity, through W on support $\times$ support, rather than the deviation-relevant set, so it under-covers relative to Corollary 2, with the exploration draws supplying the remainder. Closing that gap with a rule that targets deviation coverage directly is an immediate direction for future work.

A.9 Ablations and bound diagnostics

A.9.1 Error decomposition

Counterfactual decomposition (Figure 7) replaces one error source at a time with ground truth and re-solves. Error is representational at low budget and statistical at high, with the crossover below 20% on the informative game, between 20% and 40% on the arbitrary game, and near 40% on Blotto. This yields a tuning rule: below the crossover spend on coverage, above it on revisits.

The crossover need not be measured. Coverage and revisits buy different terms of Theorem 1 and are priced separately: coverage decides *which* cells and drives $\beta \cdot w_{\text{sur}}$ to zero, while revisits decide *how tight* and shrink $2\zeta(m, \delta)$. The rule in closed form is therefore to cover until $w_{\text{sur}} = 0$, which Proposition 1 bounds at $n_D + n_A$ rounds and costs $s_D + s_A$ cells per PSRO iteration under stable supports, and then to spend every remaining rollout revisiting covered cells until $2\zeta(m, \delta)$ meets the exploitability target. The two phases do not compete for the same cells: revisits fall only on cells coverage has already bought.

A.9.2 Batch size and exploration probability

The acquisition loop has two knobs. The batch size c is the number of cells bought between refits: $c = B$ commits the whole budget on a single ranking, $c = 1$ re-ranks after every cell. The exploration probability ε_x is the chance that a given selection ignores the score. Table 4 sweeps both at a 20% budget over four seeds.

Batch size is the parameter that matters, and it matters where the theory says it should. On Blotto, $c = 1$ reaches 0.073 against 0.163 at $c = 64$, a factor of 2.2; on the informative game the effect is within seed noise. The asymmetry follows from what a batch commits to: every cell in a batch is chosen from one ranking, so a large c spends that share of the budget on the surrogate’s state at the start of the batch. On a game the surrogate fits well that state is already good and the loss is small; on a combinatorial surface it is not, and re-ranking after each cell recovers most of the gap.

The exploration probability is close to flat, and $\varepsilon_x = 0$ is best or tied in several cells. This is the expected reading rather than a contradiction. Theorem 2 establishes that exploration is *necessary*, in the sense that no rule reading only fingerprints and the cache carries the asymptotic guarantee without it; it does not claim that exploration improves finite-budget exploitability on a benign instance. These three games are benign in exactly that sense—no fingerprint collision is constructed—so exploration buys the guarantee and not the number, and its cost is the small

Table 4: Exploitability against batch size c and exploration probability ε_x , 20% budget, mean over four seeds. Batch size is the operative knob; ε_x is close to flat.

c	ε_x			
	0.00	0.05	0.15	0.30
<i>Latent, informative</i>				
1	0.018	0.029	0.019	0.028
4	0.030	0.054	0.056	0.027
16	0.014	0.031	0.037	0.042
64	0.041	0.028	0.033	0.074
<i>Blotto</i>				
1	0.073	0.042	0.078	0.074
4	0.086	0.074	0.107	0.106
16	0.164	0.164	0.194	0.160
64	0.163	0.139	0.214	0.167

budget share it diverts. The instance where it earns its keep is the twin-trap construction, reported separately in Section 6. We therefore keep $\varepsilon_x = 0.15$ as the default: it is insurance whose premium is measured here and whose payout is measured there.

A.9.3 Variants of the acquisition score

Table 2 compares our rule against prior work and against no score at all. This subsection varies our own rule instead, under the same protocol: identical surrogate, budget, batch schedule and forced-exploration rate, sixteen paired seeds, ε_{12} . None of these is a baseline drawn from the literature; each removes or replaces one ingredient of the score in Algorithm 1.

Table 5: Variants of our acquisition score. The first row repeats Table 2 for reference. Mean $\pm$ SD over sixteen paired seeds.

Acquisition score	Informative	Blotto
ν -smoothed product (ours)	0.024 $\pm$ 0.017	0.082 $\pm$ 0.042
bootstrap inclusion	0.035 $\pm$ 0.022	0.100 $\pm$ 0.069
training exposure	0.039 $\pm$ 0.019	0.105 $\pm$ 0.044
value-sensitivity ($\nu = 0$)	0.048 $\pm$ 0.038	0.096 $\pm$ 0.058

Value-sensitivity ($\nu = 0$) isolates the smoothing, and the ordering reverses across games. It is the rule used in earlier versions of this work: no smoothing, and the product returned inside the bootstrap average. Smoothing is worth 0.024 on the latent games ($p = 0.007$, 12/16 seeds) and 0.014 on Blotto, where it is not significant ($p = 0.348$, 9/16). It therefore does its work where the support estimate is sharp and the unsmoothed score concentrates on a handful of cells, not where the support is already broad—on Blotto, at supports (12, 12), the unsmoothed score already spreads over most of the matrix and there is little left for ν to do.

Bootstrap inclusion replaces equilibrium mass with inclusion frequency, the fraction of bootstrap replicates in which a strategy appears in the support at all, and sums the two roles rather than multiplying them. This is the discrete analogue of the probability-of-equilibrium criterion of Picheny et al. [23], with the bootstrap standing in for their Gaussian-process posterior. The sum is our departure from it: a product is supported only on $\text{supp}(\hat{p}) \times \text{supp}(\hat{q})$, whereas a sum scores every

row against any plausible column, which is the shape of the deviation-relevant set of Corollary 2. It is the same neighbourhood idea as ν , learned from the bootstrap rather than imposed by a fixed constant, and it finishes second here—close enough that the choice between them is not settled by these games.

Training exposure is the PSRO-specific rule. Every strategy in the pool arrived as a best response to a known opponent mixture, so the cells pairing it with the opponents it was optimised against are cells whose values that optimisation has already partly determined; the cells it was never trained against are both less predictable and, by Corollary 2, deviation-relevant as soon as that strategy enters a support. The rule down-weights a cell by the training exposure $e(i, j)$, the weight the mixture that produced a_j placed on defender i , and symmetrically. It is the only rule considered anywhere in this paper that uses information no payoff-build baseline can see, since only the outer loop knows what each strategy was trained against, and it finishes fourth. We report it because the negative result is informative: a best response supplies one scalar constraint on a $\hat{p}$ -weighted average of a column, not per-cell information, and weighting cells by it asks the constraint to carry more than it does. The constraint is better used inside the surrogate fit, which we have not attempted here.

A.10 Bound and certificate diagnostics

These three studies use fixed pools as a diagnostic bed rather than the growing-pool setting of Section 6. The first shows that the deviation-relevant set predicts cost before any budget is spent. The second breaks Figure 4 down by game. The third gives what the computable certificate costs in optimism. None is needed to read Table 1, Table 2, or Figure 4.

A.10.1 Predicting difficulty before spending

The results above are retrospective: they report what a budget bought. The practical question is the prospective one, whether a game’s cost can be known before committing to it. Corollary 2 answers it with $B^* = n_{DS_A} + n_{AS_D} - s_{DS_A}$, computable from bootstrap support estimates alone, and Table 6 evaluates it directly: simulate exactly the deviation-relevant set, surrogate elsewhere.

Two claims are tested. First, B^* *orders the games correctly*. The two latent games have supports of size 2 on each side and cost 80 cells, 18% of the matrix; Blotto’s supports come out at (12, 12) under this solver, giving $B^* = 360$ cells, 82%. That ordering is available before any budget is spent, and it matches the ordering the growing-pool results of Section 6.1 reveal. Equilibrium support size, not matrix size, is the operative quantity. Second, simulating exactly that set *delivers full-rebuild quality*: on the informative game coverage reaches 0.017 ± 0.003 against 0.033 ± 0.014 for a full rebuild, at 18% of the simulation, and on the arbitrary game it is statistically indistinguishable (0.038 ± 0.020). Coverage is therefore not merely equal to a full rebuild on the informative game, it is slightly ahead of it, and the reason is worth stating because it looks at first like an error. A full rebuild measures every one of the 441 cells independently at $N_{MC} = 4$ and keeps the statistical error of all of them. Coverage measures 80 cells and predicts the rest from a model fitted to everything observed so far. When that model class contains the truth, its predictions pool information across the matrix and carry *lower* variance than an individual four-rollout average, so replacing measurements by predictions can reduce error rather than add it. The latent games are the extreme case, since their payoff matrices are exactly rank one by construction (Section 5) and the surrogate’s row-and-column fallback is correctly specified for them. Applying a rank-one projection to a full rebuild’s own noisy measurements reproduces the effect on its own, halving realised exploitability, which confirms that the gain comes from denoising rather than from anything the acquisition rule does.

Table 6: Deviation-relevant coverage (Corollary 2), mean $\pm$ SD over sixteen seeds, on 21×21 games (441 cells).

Game	supp.	B^* (%)	ε
Latent, informative	(2, 2)	80 (18)	0.017 ± 0.003
Latent, arbitrary	(2, 2)	80 (18)	0.038 ± 0.020
Blotto	(12, 12)	360 (82)	0.062 ± 0.034
<i>full rebuild, latent</i>		441 (100)	0.033 ± 0.014
<i>full rebuild, Blotto</i>		441 (100)	0.043 ± 0.021

The effect tracks how well the model class contains the truth, and it disappears when that fails. On the arbitrary game the payoffs are the same but the fingerprints carry no information, so only row and column effects can be pooled and coverage is indistinguishable from a rebuild. On Blotto it is worse. The payoff surface there has effective rank above four, the surrogate cannot fit it, and at 82% coverage there is little left to save in any case.

A.10.2 Bound tightness by game

Figure 8 reports the same 240 runs as per-game overestimation factors. The sup-norm bound (d) overestimates realised ε by mean factors of 22.7, 20.0, and 11.7 on the three games; the support-weighted form (c) reduces these to 11.1, 7.2, and 4.3; the signed form (a) with the measured solver term to 5.8, 3.4, and 2.0. Mean ratios carry heavy right tails, since a run with near-zero realised ε inflates the ratio however tight the bound is in absolute terms, so the seed spreads are large. The ordering (d) $>$ (c) $>$ (a) is uniform across games and budgets.

A.10.3 The computable certificate, and the cost of optimism

Corollary 1 is evaluated in both of the roles Section 4 distinguishes. Used post hoc at B^* coverage, the computable certificate held on all 48 runs, with values of 0.935 ± 0.059 and 0.938 ± 0.075 on the latent games and 0.516 ± 0.032 on Blotto against realised ε of 0.017–0.062 (Table 6). At $m = 4$ visits its slack is dominated by the time-uniform concentration radius $2\zeta_{4,0.05} = 0.481$ and by mismatch between the estimated and true supports, and it contracts as $1/\sqrt{m}$ with revisits. Proposition 1 removes the second of those two sources by construction: iterating coverage to $w_{\text{sur}} = 0$ leaves the statistical radius as the only slack.

That remaining slack is the operative constraint, and it is a visit count rather than a coverage question. At $m = 4$ the bound a fully covered build can certify is $\varepsilon_{\text{solve}} + 0.481$, roughly half the payoff range, which is why the certificates above sit near 0.94 against realised ε of 0.017–0.062: the estimator was buying the right cells and not enough visits. Inverting $2\zeta(m, \delta)$ prices the alternative. A certificate of 0.25 needs $m = 16$ and, at the 80 cells of B^* coverage, 1,280 rollouts—below the 1,764 a full rebuild spends at $m = 4$. A certificate of 0.10 needs $m = 101$ and 8,080 rollouts, and one of 0.05 needs $m = 408$ and 32,640, some 18 $\times$ a full rebuild. Coverage is cheap and the certificate is not; a deployment should choose the bound it needs and let the visit count follow, rather than choosing a cell budget and discovering what it certifies. Used inside the solve, R-max style, optimism certified only after simulating 90–97% of the matrix (395–430 of 441 cells), because every unsimulated cell looks maximal and the support chases fresh cells. The division of labour is therefore a three-way one: the surrogate fill decides which equilibrium is solved for, coverage of the deviation-relevant set makes the bound on that equilibrium independent of the fill, and optimism only certifies it. The B^* runs above isolate the middle role—every one of them carries the same β -free guarantee—while the

exploitability they reach still separates surrogate fill from the flat fill of the uniform baseline, which is the first role.

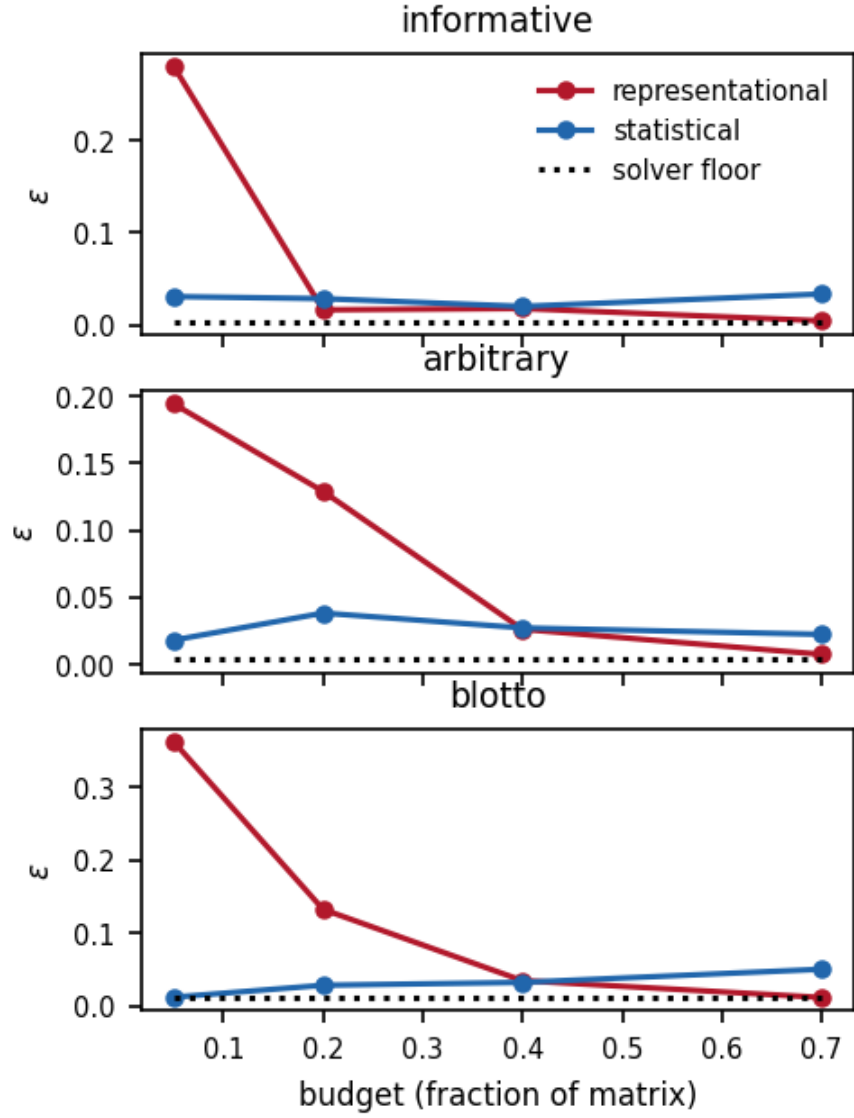


Figure 7: Counterfactual decomposition of ϵ into statistical and representational parts (seed means; dotted line is the solver floor).

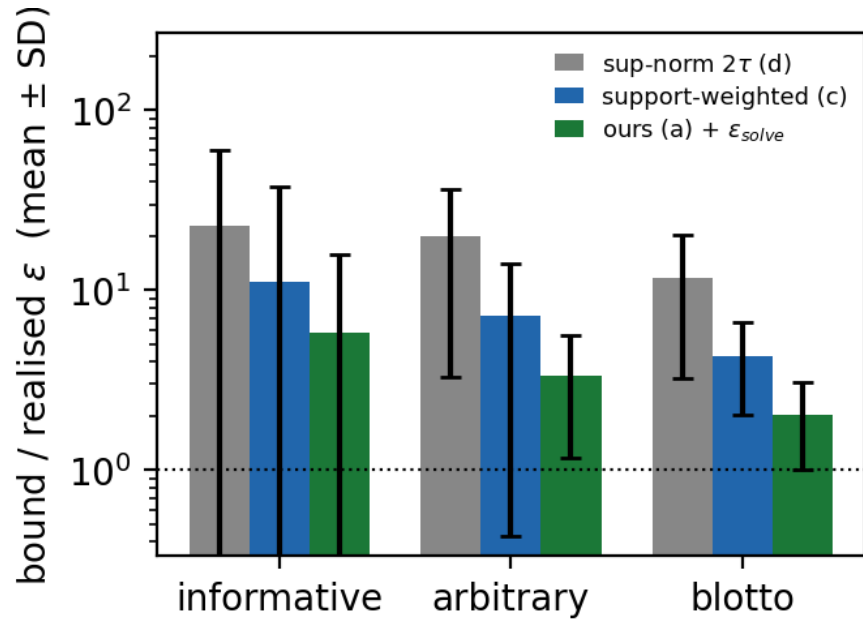


Figure 8: Overestimation factor, bound over realised ε , per game (mean $\pm$ SD over the 240 budget-sweep runs, log scale). Labels refer to the forms of Eq. (3); our bound is 4–6 $\times$ tighter than the sup-norm bound.