Sharing Information in Adversarial Bandit

David L. Saint-Pierre, Olivier Teytaud

Sharing Information in Adversarial Bandit

David L. St-Pierre[1] and Olivier Teytaud[2]

[1]Montefiore Institute, Department of Electrical Engineering and Computer Science, Liège University, B-4000 Liège, BE
[2]TAO, Inria, Université Paris-Sud, Paris, UMR CNRS 8623, FR

Abstract. 2-Player games in general provide a popular platform for research in Artificial Intelligence (AI). One of the main challenges coming from this platform is approximating a Nash Equilibrium (NE) over zero-sum matrix games. While the problem of computing such a Nash Equilibrium is solvable in polynomial time using Linear Programming (LP), it rapidly becomes infeasible to solve as the size of the matrix grows; a situation commonly encountered in games. This paper focuses on improving the approximation of a NE for matrix games such that it outperforms the state-of-the-art algorithms given a finite (and rather small) number T of oracle requests to rewards. To reach this objective, we propose to share information between the different relevant pure strategies. We show both theoretically by improving the bound and empirically by experiments on artificial matrices and on a real-world game that information sharing leads to an improvement of the approximation of the NE.
Keywords: Bandit Problem, Monte-Carlo, Nash Equilibrium, Games.

1 Introduction

2-Player games in general provide a popular platform for research in Artificial Intelligence (AI). One of the main challenges coming from this platform is approximating the Nash Equilibrium (NE) over zero-sum matrix games. To name a few games where the computation (or approximation) of a NE is relevant, there is Rock-Paper-Scissor, Battleship, partially observable variants of Chess [10, 4], and all games with a simultaneous metagaming part (e.g. choosing a deck in a card game) or simultaneous moves [12]. Such a challenge is not only important for the AI community. To efficiently approximate a NE can also help solving several real life problems. One can think, for example, of financial applications [5] or psychology [7].

While the problem of computing a Nash Equilibrium is solvable in polynomial time using Linear Programming (LP), it rapidly becomes infeasible to solve as the size of the matrix grows; a situation commonly encountered in games. Thus, an algorithm that can approximate a NE faster than polynomial time is required. [6, 3, 8] show that it is possible to ϵ-approximate a NE for a zero-sum game by accessing only $O(K\frac{\log(K)}{\epsilon^2})$ elements in a $K \times K$ matrix. In other words, in far less than the total number of elements in the matrix.

The early studies assume that there is an exact access to reward values for a given element in a matrix. It is not always the case. In fact, the exact value of an element can be difficult to know, as for instance when solving difficult games. In such cases, the value

is only computable approximately. [1] considers a more general setting where each element of the matrix is only partially known from a finite number of measurements. They show that it is still possible to ϵ-approximate a NE provided that the average of the measurements converges quickly enough to the real value.

[12, 11] propose to improve the approximation of a NE for matrix games by exploiting the fact that often the solution is sparse. A sparse solution means that there are many pure (i.e. deterministic) strategies, but only a small subset of these strategies are part of the NE. They used artificial matrix games and a real game, namely *Urban Rivals*, to show a dramatic improvement over the current state-of-the art algorithms. The idea behind their respective algorithms is to prune uninteresting strategies, the former in an offline manner and the latter online.

This paper focuses on further improving the approximation of a NE for zero-sum matrix games such that it outperforms the state-of-the-art algorithms given a finite (and rather small) number T of oracle requests to rewards. To reach this objective, we propose to share information between the different relevant strategies. To do so, we introduce a problem dependent measure of *similarity* that can be adapted for different challenges. We show that information sharing leads to a significant improvement of the approximation of a NE.

The rest of the paper is divided as follow. Section 2 formalizes the problem and introduces notations. The algorithm is defined in Section 3. Section 4 evaluates our approach from a theoretical point of view. Section 5 evaluates empirically the proposed algorithm and Section 6 concludes.

2 Problem Statement

We now introduce the notion of Nash Equilibrum in Section 2.1, define a generic bandit algorithm in Section 2.2 and Section 2.3 states the problem that we address in this paper.

2.1 Nash Equilibrium

Consider a matrix M of size $K_1 \times K_2$ with rewards bounded in $[0, 1]$, player 1 chooses an action $i \in K_1$ and player 2 chooses an action $j \in K_2$[1]. In order to keep the notations short and because the extension is straightforward, we will assume that $K_1 = K_2 = K$. Then, player 1 gets reward $M_{i,j}$ and player 2 gets reward $1 - M_{i,j}$. The game therefore sums to 1. We consider games summing to 1 for commodity of notations, but 0-sum games are equivalent. A NE of the game is a pair (x^*, y^*) both in $[0, 1]^K$ such that if i and j are chosen according to the distribution x^* and y^* respectively (i.e $i = k$ with probability x^*_k and $j = k$ with probability y^*_k with $k \in K$), then neither player can expect a better average reward through a change in their strategy distribution.

As mentioned previously, [12, 11] observe that in games, the solution often involves only a small number of actions when compared to the cardinality of the set K. In other words, often $\{i; x^*_i > 0\}$ and $\{j; y^*_j > 0\}$ both have cardinality $<< K$. The sparsity assumption is not a necessity to ensure convergence, but it makes convergence faster.

[1] From here on in, we will do a small abuse of notation by stating $K_p = [[1, K_p]]$ $\forall$ player p and where $[[\cdot, \cdot]]$ represents a discrete set.

2.2 Generic Bandit Algorithm

The main idea behind a bandit algorithm (adversarial case) is that it iteratively converges towards a NE. Bandit algorithms have the characteristic of being 'anytime', which means they can stop after any number of iterations and still output a reasonably good approximation of the solution. For a given player $p \in P$ where $P = [[1, 2]]$ for a 2-player game, each possible action is represented as an arm $a_p \in K_p$ and the purpose is to determine a probability distribution θ_p over the set of actions, representing a mixed (randomized) strategy as a probability distribution over deterministic (pure) strategies.

During the iteration process, each player selects an arm from their own set of actions K_p, forming a pair of action (a_1, a_2), according to their current distribution θ_p and their selection policy $\pi_p(\cdot)$. A selection policy $\pi_p(\cdot) \in K_p$ is an algorithm that selects an action $a_p \in K_p$ based on the information at hand. Once the pair of action (a_1, a_2) is selected, a reward r_t is computed for the t^{th} iteration. Based upon the reward, both distributions θ_1 and θ_2 are updated. A detailed description of the selection policies and the distribution updates used in this paper are provided in Section 3.

Such a process is repeated until the allocated number of iterations T has been executed. Afterward, the action to be executed consists in choosing an arm $\hat{a}_p$ according to the information gathered so far. The pseudo code for a generic bandit algorithm up to the recommendation of $\hat{a}_p$ is provided in Algorithm 1.

Algorithm 1 Generic Bandit Algorithm. The problem is described through the "get reward" function and the action sets. The "return" method is formally called the recommendation policy. The selection policy is also commonly termed exploration policy.

Require: $T > 0$: Computational budget
Require: $P = [[1, 2]]$: Set of players
Require: K_p: Set of actions specific for each $p \in P$
Require: π_p: Selection policy
  Initialize θ_p: Distribution over the set of actions K_p
  for $t = 1$ to T **do**
    Select $a_p \in K_p$ based upon $\pi_p(\theta_p)$ (for $p \in [[1, 2]]$)
    Get reward $r_t = getReward(a_1, a_2)$: player 1 receives r_t and player 2 receives $1 - r_t$.
    Update θ_p using r_t (for $p \in [[1, 2]]$)
  end for
  Return $\hat{a}_p$

2.3 Problem Statement

In most of the bandit literature, it is assumed that there is no structure over the action set K_p. Consequently, there is essentially only one arm updated for any given iteration $t \in T$. In games however, the reasons for sharing information are threefolds. First, each game possesses a specific set of rules. As such, there is inherently an underlying structure that allows information sharing. Second, the sheer number of possible actions can be too large to be efficiently explored. Third, to get a precise reward r_t can be a difficult task. For instance, computing r_t from a pair of arms (a_1 and a_2) can be time consuming or/and involve highly stochastic processes. Under such constraints, sharing information along K_p seems a legitimate approach. Given $\psi = (\psi_1, \psi_2)$ that describes

some structure of the game, we propose an algorithm α_ψ that shares information along the set of actions K_p. To do so, we propose to include a measure of similarity $\psi_p(\cdot,\cdot)$ between actions of player p. Based upon the measure $\psi_p(\cdot,\cdot)$, the algorithm α_ψ shares the information with all other arms deemed similar. The sharing process is achieved by changing the distribution update of θ_p.

3 Selection Policies and Updating rules

As mentionned in Section 2.2, a selection policy $\pi(\cdot)$ is an algorithm that selects an action $a_p \in K_p$ based upon information gathered so far. There exist several selection policies in the context of bandit algorithms, [9] studied the most popular, comparing them in a Monte-Carlo Tree Search architecture. Here we develop a variant of a selection policy $\pi(\cdot)$ relevant in the adversarial case called $EXP3$ [3]. Throughout this section, the reference to a specific player p is avoided to keep the notation short.

Section 3.1 describes the $EXP3$ selection policy. Section 3.2 presents a recommendation policy, $TEXP3$ [12] dedicated to sparse Nash Equilibria. Finally, Section 3.3 introduces the notion of similarity and define our new updating rule.

3.1 EXP3

This selection policy is designed for adversarial problems. For each arm $a \in K$, we gather the following quantities:

- t_a, the number of simulations involving arm a, or its visit count.
- θ_a, the current probability to select this arm
- w_a, a weighted sum of rewards

The idea is to keep a cumulative weighted sum of reward per arm and use it to infer a distribution of probability over the different arm. An interesting fact is that it is not the probability θ_a that converges to the Nash, but the counter t_a. More formally, every time an arm a receives a reward r_t, the value w_a is updated as follows:

$$w_a \leftarrow w_a + \frac{r_t}{\theta_a} \tag{1}$$

for the player which maximizes its reward (r_t is replaced by $1 - r_t$ for the opponent).

At any given time, the probability θ_a to select an action a is defined as:

$$\theta_a \simeq (1-\gamma)\frac{\exp(\eta w_a)}{\sum\limits_{k\in K}\exp(\eta w_k)} + \frac{\gamma}{C}, \tag{2}$$

where $\eta > 0$ and $\gamma \in]0;1]$ and $C \in \mathbb{R}$ are three parameters to tune. $\simeq$ stands for "is proportional to".

3.2 TEXP3

This recommendation policy is an extension of $EXP3$. It is a process that is executed only once before choosing $\hat{a}$, the arm to be pulled. Basically, it uses the property that, over time, the probability to pull an arm a, given by the ratio $\frac{t_a}{T}$, that is not part of the

optimal solution will tend toward 0. Therefore, for all arms $a \in K$ deemed to be outside the optimal solution, it artificially truncates these arms. The decision whether an arm is part of the NE is based upon a threshold c. Following [12], the constant c is chosen as $\max_{a \in K} \frac{(T \times t_a)^\alpha}{T}$, where $\alpha \in]0, 1]$. If the ratio $\frac{t_a}{T}$ of an arm $a \in K$ is below such threshold, it is removed and the remaining arms have their probability rescaled accordingly.

3.3 Structured EXP3

As mentioned previously, one of the main reason for sharing information is to exploit a priori regularities that are otherwise time consuming to let an algorithm find by itself. The core idea is that simliar arms are likely to produce similar results. The sharing of information is mostly important in the early iterations because afterwards the algorithm gathers enough information to correctly evaluate each individual relevant arms.

EXP3 uses an exponential at its core combined with cumulative rewards. One must be careful about the sharing of information under such circumstance. The exponential makes the algorithm focus rapidly on a specific arm. The use of cumulative reward is also problematic. For example, sharing several times a low reward can, over time, mislead the algorithm into thinking an arm is better than one that received only once a high reward. To remedy this situation, we only share when the reward is interesting. To keep it simple, the decision whether to share or not is made by a threshold ζ that is domain specific.

Lets define $\varphi_a \subseteq K$ as a set of arms that are considered similar to a based upon the measure $\psi(a, k)$, i.e. $\varphi_a = \{k; \psi(a, k) > 0\}$. If $r_t > \zeta$, for all $k \in \varphi_a$ we update as follow:

$$w_k \leftarrow w_k + \frac{r_t}{\theta_k}. \tag{3}$$

The probability θ_k to select an action k is still defined as:

$$\theta_k \simeq (1 - \gamma) \frac{\exp(\eta w_k)}{\sum_{k' \in K} \exp(\eta w_{k'})} + \frac{\gamma}{C}, \tag{4}$$

where $\eta > 0$, $\gamma \in]0; 1]$ and $C \in \mathbb{R}$ are three parameters to tune. In the case where $r_t \leq \zeta$, the update is executed following (1) and (2).

4 Theoretical Evaluation

In this section we present a simple result showing that structured-EXP3 performs roughly S times faster when classes of similar arms have size S. The result is basically aimed at showing the rescaling of the update rule.

A classical EXP3 variant (from [1]) uses, as explained in Alg. 2, the update rule

$$\theta_a = \gamma/C + (1 - \gamma) \frac{\exp(\eta \omega_a)}{\sum_{k \in K} \exp(\eta \omega_k)},$$

$$\text{where} \quad C = K, \gamma = \min(0.8 \sqrt{\frac{\log(K)}{Kt}}, 1/K) \text{ and } \eta = \gamma.$$

Algorithm 2 The EXP3 algorithm as in [1] (left) and TEXP3, sEXP and sTEXP3 variants (right). Strategies are given for player 1. Player 2 use $1 - r_t$ instead of r_t.

for each iteration $t \in [[1, T]]$ **do**
  Selection policy π: choose arm a with probability θ_a (Eq. 4).
  Get reward r_t.
  Update ω_a:

$$w_a \leftarrow w_a + \frac{r_t}{\theta_a}.$$

end for
Recommendation: choose arm $\hat{a}$ with probability $n_T(a) = \frac{t_a}{T}$.

for each iteration $t \in [[1, T]]$ **do**
  Selection policy π: choose arm a with probability θ_a (Eq. 4).
  Get reward r_t.
  Update ω_a: $w_a \leftarrow w_a + \frac{r_t}{\theta_a}$.
  if sEXP3 or sTEXP3 and $r_t < \zeta$ **then**
    for each $b \in \varphi(a) \setminus a$ **do**

$$w_b \leftarrow w_b + \frac{r_t}{\theta_b}.$$

    end for
  end if
end for
if TEXP3/sTEXP3 **then**
  if $t_a/T \leq c$ **then**
    Set $t_a = 0$.
  end if
  Rescale t_a: $t_a \leftarrow t_a / \sum_{b \in K} t_b$.
end if
Recommendation: choose arm $\hat{a}$ with probability $n_T(a) = \frac{t_a}{T}$.

Note that the parameters γ and η depend on t, removed for shorter notation.

The pseudo-regret L after T iterations is defined as:

$$L_T = \max_{k \in K} \mathbb{E} \left(\sum_{t=1}^{T} r_t(k) - r_t \right)$$

where r_t is the reward obtained at iteration t and $r_t(k)$ is the reward which would have been obtained at iteration t by choosing arm k at iteration t. Essentially, the pseudo-regret is non-negative, and is zero if we always choose an arm that gets optimal reward. With this definition, EXP3 verifies the following[2, 1]:

Theorem 1: pseudo-regret L of EXP3.
Consider a problem with K arms and 1-sum rewards in $[0, 1]$. Then, EXP3 verifies

$$L_T \leq 2.7\sqrt{TKln(K)}.$$

It is known since [6] that it is not possible to do better than the bound above, within logarithmic factors, in the general case.

We propose a variant, termed Structured-EXP3 or $sEXP3$, for the case in which for each arm a, there is a set φ_a (of cardinality S) containing arms similar to a. Under mild assumptions upon φ_a, the resulting algorithms has some advantages over the baseline EXP3. The parameters γ and η for $sEXP3$ are defined by Eq. (5) ($C = K$ is preserved,

as in EXP3).

$$\gamma = \min(0.8\sqrt{\frac{S\log(K/S)}{Kt}}, S/K) \text{ and } \eta = \gamma/S. \quad (5)$$

In other words, γ is designed (as detailed in the theorem below) for mimicking the values corresponding to the problem with K/S arms instead of K arms and η is designed for avoiding a too aggressive pruning.

The following theorem is aimed at showing that parameters in Eq. (5) ensure that Structured-EXP3 emulates EXP3 on a bigger problem with a particular structure.

Theorem 2: Structured-EXP3 and pseudo-regret.

Consider a problem where there are K' classes of S similar arms i.e. $K = K' \times S$ (arms from different classes have no similarity); φ_a is the set of arms of the same class as arm a. Assume that all arms in a class have the same distribution of rewards, i.e. $a \in \varphi_b$ implies that a and b have the same distribution of rewards against any given strategy of the opponent. Set $\zeta = -\infty$. Then, Structured-EXP3 verifies

$$L_T \leq 2.7\sqrt{T(K/S)ln(K/S)},$$

where S is the cardinal of φ_a (whereas the EXP3 bound is $2.7\sqrt{TKlnK}$).

Proof: For this proof, we compare the Structured-EXP3 algorithm with K arms including classes of S similar arms (i.e. $\forall\, a \in [[1, K]], \varphi_a = S$) and an EXP3 algorithm working on an ad hoc problem with K/S arms. The ad hoc problem is built as follows.

Instead of arms $A = [[1, K]]$ (ordered by similarity, so that blocks of S successive arms are similar) for the Structured-EXP3 bandit, consider arms $A' = [[1, S+1, 2S+1, \ldots, K-S+1]]$ for the EXP3 bandit. Consider the same reward as in the Structured-EXP3 bandit problem.

Any mixed strategy on the EXP3 problem can be transformed without changing its performance into a mixed strategy on the Structured-EXP3 problem by arbitrarily distributing the probability of choosing arm $k \in A'$ onto arms $[[k, k+1, \ldots, k+S-1]] \subset A$.

Let us use $\theta'_{a'}$, the probability that EXP3 chooses $a' \in A'$; and $\omega'_{a'}$, the sum of rewards associated to $a' \in A'$ for EXP3 (notations with no "prime" are for Structured-EXP3). We now show by induction that

- $\omega_a = S \times \omega'_{a'}$ for $a \in A$ similar to $a' \in A'$ when EXP3 and Structured-EXP3 have the same history[2];
- $\theta_a = \frac{1}{S}\theta'_{a'}$ for $a \in A$ similar to $a' \in A'$.

The proof is based on the following steps, showing that when the induction properties hold at some time step then they also hold at the next time step. We assume that $\omega_a = S \times \omega'_{a'}$ (for all $a' \in A'$ similar to a) at some iteration, and we show that it implies $\theta_a = \frac{1}{S}\theta'_{a'}$ at the same iteration (also for all $a' \in A'$ similar to a) and that $\omega_a = S \times \omega'_{a'}$ at the next iteration (also for all $a' \in A'$ similar to a). More formally, we show that

$$\forall (a, a') \in A \times A', a \in \varphi_{a'}, \omega_a = S \times \omega'_{a'} \quad (6)$$

$$\text{implies } \forall (a, a') \in A \times A', a \in \varphi_{a'}, \theta_a = \frac{1}{S} \times \theta'_{a'} \quad (7)$$

[2] The set of arms are not the same in Structured-EXP3 and EXP3. By same history we mean up to the projection $a \to a' = \lfloor (a-1)/S \rfloor + 1$.

and at next iteration Eq. (6) still holds. The properties of Eq. (6) and Eq. (7) hold at the initial iteration (we have only zeros) and the induction from one step to the next is as follows:

- **Let us show that Eq.** (6) **implies Eq.** (7)**, i.e. if, for all** a**,** ω_a **for Structured-EXP3 is** S **times more than** $\omega'_{a'}$ **for** $a' \in A'$ **similar to** a**, then the probability for Structured-EXP3 to choose an arm** $a \in A$ **similar to** $a' \in A'$ **is exactly** S **times less than the probability for EXP3 to choose** a'**.** The probability that Structured-EXP3 chooses arm a at iteration t given an history $a_1, \ldots, a_{t-1}$ of chosen arms with rewards $r_1, \ldots, r_{t-1}$ until iteration $t-1$ is

$$\theta_a = (1-\gamma)\frac{\exp(\eta w_a)}{\sum\limits_{k \in K} \exp(\eta w_k)} + \frac{\gamma}{K}, \tag{8}$$

  which is exactly S times less than the probability that EXP3 chooses arm $\lfloor (a-1)/S \rfloor + 1$ given a history $\lfloor (a_1-1)/S \rfloor + 1$, $\lfloor (a_2-1)/S \rfloor + 1$,..., $\lfloor (a_{t-1}-1)/S \rfloor + 1$. Thus,

$$\theta_a = \frac{1}{S}\theta'_{a'}.$$

  This is the case because the S additional factor in ω_a is compensated by the S denominator in Eq. (5) so that terms in the exponential are the same as in the Structured-EXP3 case; but the numerator is S times bigger. This concludes the proof that Eq. (7) holds.
- We now show that **the probability that Structured-EXP3 chooses an arm in** $[[a', a'+1, \ldots, a'+S-1]]$ **similar to** $a' \in A$ **is the same as the probability that EXP3 chooses** $a' \in A'$ **(given the same history).** The update rule in Structured-EXP3 ensures that $\omega_a = \omega_b$ as soon as a and b are similar. So the probability of an arm of the same class as $a \in A$ to be chosen by Structured-EXP3 is exactly the probability of $a' \in A'$ (similar to A) being chosen by EXP3:

$$\sum_{a \text{ similar to } a'} \theta_a = \theta'_{a'}. \tag{9}$$

- **Let us now show that Eq.** (6) **and Eq.** (7) **implies Eq.** (6) **at the next iteration, i.e. the weighted sum of rewards** ω_a **for** $a \in A$ **is** S **times more than the weighted sum of rewards** $\omega'_{a'}$ **for** a **similar to** a' **(given the same histories)**. This is because (i) probabilities that Structured-EXP3 chooses an arm a among those similar to an arm a' is the same as the probability that EXP3 chooses a' (given the same history), as explained by Eq. (9), and (ii) probabilities used in the update rule are divided by S in the case of Structured-EXP3 (updates have K at the denominator). This concludes the induction, from an iteration to the next.□

5 Experiments

This section describes a set of experiments that evaluates the quality of our approach. The first testbed is automatically generated sparse matrices and the results are presented

in section 5.1. The second testbed, presented in section 5.2, is the game Urban Rivals (UR), an internet card game. Throughout this section, we used 3 baselines: $EXP3$, $TEXP3$ and $Random$. The parameters γ, η and C were tuned independantly for each testbed (and each algorithm) to ensure they are performing as good as they can. For the automatically generated sparse matrices, the set of parameters that gave the best results are $\eta = \frac{1}{\sqrt{t}}$, $\gamma = \frac{1}{\sqrt{t}}$, $C = 0.65$ and $\alpha = 0.8$. In the game Urban Rivals, the best values found for the parameters are $\eta = \frac{1}{\sqrt{t}}$, $\gamma = \frac{1}{\sqrt{t}}$, $C = 0.7$ and $\alpha = 0.75$.

As a reminder, we add the prefix s when we exploit the notion of distance. The distance $\psi(\cdot, \cdot)$ is specific to the testbed and is thus defined in each section.

5.1 Artificial experiments

First we test on automatically generated matrices that have a sparse solution and contain an exploitable measure of distance between the arms. We use matrix M defined by $M_{i,j} = \frac{1}{2} + \frac{1}{5}(1 + cos(i \times 2 \times \pi/100))\chi_{i \bmod \omega} - \frac{1}{5}(1 + cos(j \times 2 \times pi/100))\chi_{j \bmod \omega}$, where $\omega \in \mathbb{N}$ is set to 5 and M is of size 50×50. The distance $\psi(\cdot, \cdot)$ is based on the position of the arm in the matrix. It is defined such that the selected arm a and $k \in K$ have a similarity

$$\psi(a, k) = \begin{cases} 1 & \text{if } (k' - a') \bmod \omega = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

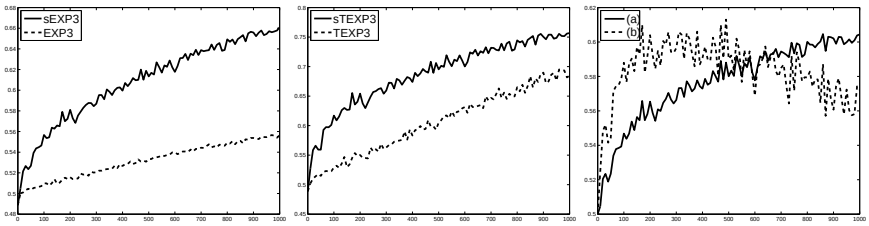
where k' and a' are the position of respectively k and a in the matrix M. The set φ_a includes all k where $\psi(a, k) = 1$. At any $t \in T$ the reward is given by the binomial distribution $r_t \sim \mathrm{B}(20, M(i, j))$, where 20 is the number of Bernoulli trials with parameter $M(i, j)$. The threshold ζ is fixed at 0.8 and for any given T, the experiment is repeated 100 times.

Figure 1 analyses the score ($\%$) in relation to the maximal number of iterations T of our approach playing against the baselines. Figure 1(a) presents the results of $sEXP3$ and $EXP3$ playing against the baseline $Random$. Figure 1(b) shows $sTEXP3$ and $TEXP3$ also playing against the baseline $Random$. Figure 1(c) depicts the results of $sEXP3$ playing against $EXP3$ and $sTEXP3$ playing against $TEXP3$.

Figure 1(a) shows that $sEXP3$ significantly outperforms $EXP3$. It requires as little as $T = 30$ iterations to reach a significant improvement over $EXP3$. As the maximal number of iterations T grows, $sEXP3$ still clearly outperforms its counterpart $EXP3$.

Figure 1(b) shows again a clear improvement of exploiting the structure (as in $sTEXP3$) versus not (as in $TEXP3$). It requires $T = 30$ iterations to reach a significant improvement. The score in Figure 1(b) are clearly higher than in Figure 1(a), which is in line with previous findings. Moreover, a Nash player would score 87.64% versus the $Random$ baseline. The best score 75.68% is achieved by $sTEXP3$ which is fairly close to the Nash, using only $1\,000$ requests to the matrix.

The results in Figure 1(c) are in line with Figure 1(a) and 1(b). The line representing $sEXP3$ versus $EXP3$ (labeled (a) in reference to Figure 1(a)) shows that even after $T = 1\,000$ iterations, $EXP3$ does not start to close the gap with $sEXP3$. The line representing $sTEXP3$ versus $TEXP3$ shows that it takes around $T = 500$ iterations for $TEXP3$ to start filling the gap with the algorithm that shares information $sTEXP3$. Yet even after $T = 1\,000$ it is still far from performing as well.



(a) Without Truncation (b) With Truncation (c) Exploit Structure vs Not

Fig. 1. Performance $(\%)$ in relation to the number of iterations T of our approach compared to different baselines. Each of the 99 different positive abscissa is an independent run, so the null hypothesis of an average ordinate $\leq 50\%$ is less than 10^{-29}. We see that (a) sEXP3 converges faster than EXP3 (in terms of success rate against random), (b) sTEXP3 converges faster than TEXP3 (in terms of success rate against random), (c.a) sEXP3 outperforms EXP3 (direct games of sEXP3 vs EXP3) and (c.b) sTEXP3 outperforms TEXP3 (direct games of sTEXP3 vs TEXP3).

Overall for this testbed, the sharing of information greatly increases the performance of the state-of-the-art algorithms. The good behavior of sparsity techniques such as TEXP3 is also confirmed.

5.2 Urban Rivals

Urban Rivals (UR) is a widely played internet card game, with partial information. As pointed out in [12], UR can be consistently solved by a Monte-Carlo Tree Search algorithm (MCTS) thanks to the fact that the hidden information is frequently revealed. A call for getting a reward leads to 20 games played by a Monte-Carlo Tree Search with $1\ 000$ simulations before an action is chosen. Reading coefficients in the payoff matrices at the root is quite expensive, and we have to solve the game approximately.

We consider a setting in which two players choose 4 cards from a finite set of 10 cards. We use two different representations. In the first one, each arm $a \in K$ is a combination of 4 cards and $K = 10^4$. In the second representation, we remove redundant arms. There remain $K = 715$ different possible combinations if we allow the same card to be used more than once in the same combination.

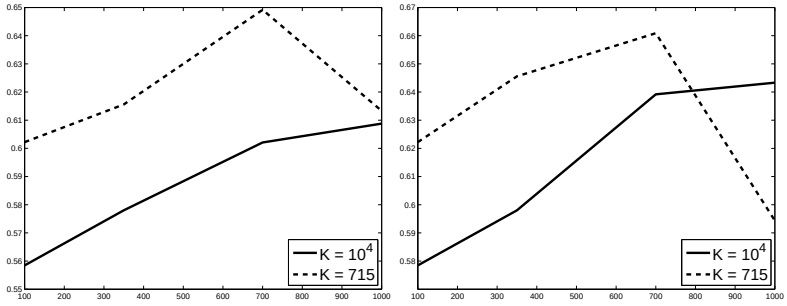
There are two baseline methods tested upon UR, namely $EXP3$ and $TEXP3$.

The distance $\psi(\cdot, \cdot)$ is based on the number of similar cards. It is defined such that the selected arm a and $k \in K$ have a distance $\psi(a, k) = 1$if k and a share more than 2 cards and 0 otherwise. The set φ_a includes all k where $\psi(a, k) = 1$. At any $t \leq T$ the reward r_t is given by 20 games played with the given combinations. The threshold ζ is fixed at 0.8 and for any given T.

For a given number of iterations T, each algorithm is executed 10 times and the output is saved. To compute the values in Figure 2, we play a round-robin (thus comparing 10×10 different outputs) where each comparison between two outputs consist in repeating 100 times the process of selecting an arm and executing 20 games.

Figure 2 presents the score $(\%)$ in relation to the maximal number of iterations T of our approach playing against their respective baselines. Figure 2(a) presents the results of $sEXP3$ playing against $EXP3$. Figure 2(b) shows $sTEXP3$ playing against

$TEXP3$. In both cases, we present the results for 2 different representations ($K = 10^4$, and $K = 715$).



(a) sEXP3 vs EXP3 (b) sTEXP3 vs TEXP3

Fig. 2. Performance ($\%$) in relation to the number of iterations T of our approach compared to different baselines. Standard deviations are smaller than 1%. We see that (a) sEXP3 outperforms EXP3 in both versions (game with 10K arms and game with 715 arms) (b) sTEXP3 outperforms TEXP3 in both versions (game with 10K arms and game with 715 arms).

Figure 2(a) shows that $sEXP3$ significantly outperforms $EXP3$ independently of the representation since the values are far behond 50%. Even at the lowest number of iterations ($T = 100$), there is a significant improvement over $EXP3$ with both representations ($K = 10^4$ and $K = 715$). As the maximal number of iterations T grows, $sEXP3$ still clearly outperforms its counterpart $EXP3$. Moreover, Figure 2(a) shows that the representation impacts greatly on the quality of the results. The discrepancy between the two lines is probably closely related to the ratio $\frac{T}{K}$. For instance, when $T = 1\ 000$ and $K = 10^4$ the score is equal to 60.88%. If we compare such a result to $T = 100$ and $K = 715$, a ratio $\frac{T}{K}$ relatively close, the score (60.22%) is rather similar.

Figure 2(b) shows that $sTEXP3$ significantly outperforms $TEXP3$ independently of the representation since the values are also far beyond 50%. The conclusion drawn from Figure 2(b) are quite similar to the ones from Figure 2(a). However, the sudden drop at $T = 1\ 000$ and $K = 715$ indicates that $TEXP3$ also start to converge toward the Nash Equilibrium, thus bringing the score relatively closer to the 50% mark.

For the game UR, it seems that sharing information does also greatly improve the performance of the state-of-the-art algorithms.

6 Conclusion

In this paper, we present an improvement over state-of-the-art algorithms to compute an ϵ-approximation of a Nash Equilibrium for zero-sum matrix games. The improvement consist in exploiting the similarities between arms of a bandit problem through a notion of distance and share information among them.

From a theoretical point of view, we compute a bound for our algorithm that is better than the state-of-the-art by a factor roughly based on the number of similar arms.

Moreover, empirical results on the game of Urban Rival and automatically generated matrices show a significant better performance of the algorithms that share information compared to the ones that do not. This is when results are compared on the basis of EXP3 parameters that are optimized on the application.

As future work, the next step is to create a parameter free version of our algorithm, for instance by automatically fixing the parameter ζ. Also, so far we solely focus on problem where the total number of iterations T is too small for converging to the NE. As the maximal number of iterations T gets bigger, there would be no reason for sharing information anymore. A degradation function can be embedded into the updating rule to ensure convergence. We do not know for the moment whether we should stop sharing information depending on rewards (using ζ; if the game is symmetric, this implicitly eventually stops sharing), depending on iterations (using a limit on t/T) or more sophisticated criteria.

References

1. J. Audibert and S. Bubeck. Minimax policies for adversarial and stochastic bandits. In *22nd annual conference on learning theory (COLT)*, Montreal, Jun 2009.
2. P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.
3. P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, Los Alamitos, CA, 1995.
4. P. Ciancarini and G. P. Favini. Monte carlo tree search in kriegspiel. *Artif. Intell.*, 174(11):670–684, 2010.
5. S. R. Grenadier. Option exercise games: An application to the equilibrium investment strategies of firms. *Review of financial studies*, 15(3):691–721, 2002.
6. M. D. Grigoriadis and L. G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58, 1995.
7. T. Hedden and J. Zhang. What do you think i think you think?: Strategic reasoning in matrix games. *Cognition*, 85(1):1–36, 2002.
8. R. J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 36–41. ACM, 2003.
9. P. Perrick, D. St-Pierre, F. Maes, and D. Ernst. Comparison of different selection strategies in Monte-Carlo tree search for the game of Tron. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2012)*, Granada, Spain, 2012.
10. S. Russell and J. Wolfe. Efficient Belief-State AND-OR Search, with Application to Kriegspiel. In *IJCAI*, pages 278–285, 2005.
11. D. St-Pierre, Q. Louveaux, and O. Teytaud. Online sparse bandit for card game. In *Proceedings of Advanced in Computer Games 2011 (ACG 2011)*, pages 295–305, 2011.
12. O. Teytaud and S. Flory. Upper confidence trees with short term partial information. *Applications of Evolutionary Computation; EvoGames*, pages 153–162, 2011.