Strategic Choices in Optimization

Cheng-Wei Chou, Ping-Chiang Chou, Jean-Joseph Christophe, Adrien Couetoux, Pierre De Freminville, Nicolas Galichet, Chang-Shing Lee, Jialin Liu, David Saint-Pierre, Michèle Sebag, et al.

Strategic Choices in Optimization

CHENG-WEI CHOU[1], PING-CHIANG CHOU, JEAN-JOSEPH CHRISTOPHE[4,5], ADRIEN COUËTOUX[4,5], PIERRE DE FREMINVILLE[5], NICOLAS GALICHET[4], CHANG-SHING LEE[2], JIALIN LIU[4], DAVID L. SAINT-PIERRE[3], MICHÈLE SEBAG[4], OLIVIER TEYTAUD[4], MEI-HUI WANG[2], LI-WEN WU[2] AND SHI-JIM YEN[1]

[1]*Department of Computer Science and Information Engineering*
National Dong Hwa University
Hualien, 974 Taiwan
[2]*Department of Computer Science and Information Engineering*
National University of Tainan
Tainan, 700 Taiwan
[3]*Montefiore Institute*
Université de Liège, Belgium
[4]*Tao, Inria-CnrsS-Lri, Univ. Paris-Sud*
[5]*Artelys, 12 Bd 4 Septembre, Paris*

Many decision problems have two levels: one for strategic decisions, and another for tactical management. This paper focuses on the strategic level, more specifically the sequential exploration of the possible options and the final selection (recommendation) of the best option. Several sequential exploration and recommendation criteria are considered and empirically compared on real world problems (board games, card games and energy management problems) in the uniform (1-player) and adversarial (2-player) settings. W.r.t. the sequential exploration part, the classical upper confidence bound algorithm, the exponential exploration-exploitation algorithm, the successive reject algorithm (designed specifically for simple regret), and the Bernstein races, are considered. W.r.t. the recommendation part, the selection is based on the empirically best arm, most played arm, lower confidence bounds, based on the reward distribution or variants thereof designed for risk control. This paper presents a systematic study, comparing the coupling of the sequential exploration and recommendation variants on the considered problems in terms of their simple regret. A secondary contribution is that, to the best of our knowledge, this is the first win ever of a computer-kill-all Go player against professional human players [16].

Keywords: multi-armed bandit problems, metagaming, strategic choices, investment optimization, upper confidence bounds, simple regret, games

1. INTRODUCTION

Real world problems, ranging from electricity production and logistics to games, often involve two types of decisions: (i) tactical decisions, that is the (usually sequential) decision making problem involved in the everyday management of the system; (ii) strategic decisions, *i.e.* the long-term decisions such as investments, equipment (storage dimensioning) or contracts (buying options).

In principle, the general real-problem can be modelled as a classical optimization one, where the action space contains both tactical and strategic options, and the tactical

and strategic decisions are jointly optimized. In many cases however, it is much easier to distinguish between both types of decisions, considering the nested problem made of a strategic level, where each transition of the strategic level implies solving yet another tactical sequential decision making problem.

A simplified case is when the strategic decisions are taken once and for all, making the strategic problem a non sequential (thus simpler) optimization problem. The objective function considered at the strategic level however still requires one to solve the sequential tactical decision problem. The strategic objective function defines a computationally expensive optimization problem, possibly requiring fast approximations of the tactical level resolution and commonly involving stochastic or adversarial uncertainties.

In the last decade or so, a wide body of theoretical and experimental works have been tackling non-sequential and often unstructured decision making problems, stemmed from the so-called bandit approaches [3, 24]. Strategic decision problems, while they can be modelled as bandit problems, however feature a very restricted time budget. Furthermore, in the 2-player (adversarial) case, they have a huge sparsity in the sense that the optimal solution (Nash equilibrium) contains few actions.

The present paper proposes a systematic analysis of the bandit literature algorithms and its relevance w.r.t. strategic decision making problems, considering three representative benchmark problems. The performance criterion is the simple regret based on the general reward distribution possibly involving a risk criterion.

The rest of this section presents the formalization of the problem, a motivating example, and the background for the remainder of the paper. Section 2 describes the considered exploration and recommendation algorithms. Section 3 is devoted to the comparative analysis of their experimental performances analysis in the 1-player and 2-player cases, considering the kill-all Go problem, energy management and Urban Rivals. The paper concludes in Section 4.

1.1 Formalization

There is no clear-cut characterization of strategic versus tactical decision making problems, although it is generally agreed that since strategic choices are made at a higher level, their tradeoff between exploration and evaluation is computationally more expensive. After the game literature, the strategic (respectively, tactical) problem will be referred to as meta-gaming (resp., ingaming). Within the bandit framework (see below), the strategic problem is characterized by a small budget, specifically the number of iterations T is small compared to the number of options. In the 1-player case (K arms), the number T of iterations is such that $T < 100K$; the simple regret (as opposed to the cumulative regret [3]) is considered as performance criterion. In the 2-player case ($K \times K'$ options, an option is a pair of arms), the number of iterations is less than the number of options ($T \leq K \times K'$); the performance criterion is the average performance of the recommended distribution, naturally extending the simple regret to the 2-player case.

Let us consider a set of strategic choices, a.k.a. arms or options in the bandit setting denoted with no loss of generality $\{1, \ldots, K\}$. The time budget, a.k.a. Time horizon, T is finite. Given a performance criterion L, the exploration phase gathers T realizations $L(\theta_1)$, $L(\theta_2)$, …, $L(\theta_T)$, which sustain the selection of some θ^* in the recommendation phase. The metagame in the 1-player case is detailed in Fig. 1.

Note that a simulator (emulating the resolution of the tactical decision making problem) is needed to evaluate $L(\theta_i)$. Such a simulator relies on the assumption that one can simulate the tactical choices after the strategic choices have been made. The simulator assumedly enables to evaluate the tactical behaviors of both players. Indeed this assumption requires that we can extrapolate the opponent strategy, or efficiently approximate it (such that playing optimally against the opponent strategy or its approximation does not make significant differences). This assumption, though a standard one in the game literature, does not hold for *e.g.* poker (where opponent modeling is a key strategical ability) or Go (where human players outperform computer-Go algorithms).

The performance criterion is the simple regret, directly measuring the expected loss increase due to the strategic choices (to the extent that $L(\cdot, \cdot)$ is a good sampling of the possible outcomes).

- For $i = 1 \dots T$, the algorithm chooses $\theta_i \in \{1, \dots, K\}$ and gets a reward r_i distributed as $L(\theta_i)$;
- The algorithm chooses $\hat{\theta}$ based on $L(\theta_1), \dots L(\theta_T)$.
- The loss, termed simple regret, is $r_T = \max_\theta \mathbb{E}L(\theta) - \mathbb{E}L(\hat{\theta})$.

Fig. 1. Metagaming with one players.

- For $i = 1 \dots T$, the algorithm chooses $\theta_i \in \{1, \dots, K\}$ and $\theta'_i \in \{1, \dots, K'\}$.
- The algorithm gets a reward r_i distributed as $L(\theta_i, \theta'_i)$.
- The algorithm chooses $\hat{\theta}$ based on $L(\theta_i, \theta'_i)$.
- The loss, termed simple regret, is $r_T = \max_\theta \min_{\theta'} \mathbb{E}L(\theta, \theta') - \min_{\theta'} \mathbb{E}L(\hat{\theta}, \theta')$ (where here maxima and minima are for random variables θ, θ'; in the 2-player case we look for Nash equilibria and we expect optimal strategies to be non-deterministic).

Fig. 2. Metagaming with two players.

1.2 Motivating Example: High-Level Parameters of a Sequential Decision Making Problem

In Section 1.2.1, we introduce the 1-player Markov Decision Processes (MDP) setting, representative of the 2-player case as well. The following sections present three solvers: stochastic dynamic programming (SDP); direct policy search (DPS), and a hybrid approach which is the first contribution of our work.

1.2.1 Markov decision processes

Consider a system to be controlled. The system has an internal state x_t at time t (*e.g.* current temperature, current stock levels). The transition model of the system is governed by the state equation (Eq. 1) and the controller sets its control variables ut using some policy $u_t = u(x_t)$. The exogenous input i_t can be modeled as a Markov chain (possibly through increasing the state space dimensionality). The system is described along the following equations:

$$(c_t, x_{t+1}) = f(x_t, i_t, u_t) \quad \text{system equation} \quad (1)$$

$$C = \sum_{t \in T} c_t \quad \text{total cost} \quad (2)$$

$$u_t = u(x_t) \quad u \text{ is the sought strategy} \quad (3)$$

$$i_t = MarkovChain(i_{t\text{-}1}) \quad \text{a stochastic function} \quad (4)$$

The goal of such a Markov Decision Process (MDP) is to optimize the function u, such that the random variable C is "as small as possible". As C is a random variable in the general case, different minimization criteria have been considered, possibly taking into account risk.

1.2.2 Stochastic dynamic programming

[8] has shown that, in the MDP case and for an optimization on expectation, an optimal u for solving Eqs. (1)-(4) is defined as follows:

$$u(o) = \arg\min_u c_t(u) + V(x_{t+1}, t+1), \quad (5)$$

where o is the observation, *i.e.* (x_t, t) in the MDP case, and $V(x, t) = \inf_u c_t + c_{t+1} + \ldots + c_{time\ horizon\ T}$, with $x_t = x$ and for u an optimal-on-average policy for time steps $t' \geq t$.

The very expensive computation of $V(x, t)$ by backwards computation is the purpose of Stochastic Dynamic Programming (see *e.g.* [10] for a comprehensive introduction).

1.2.3 Direct policy search

While Stochastic Dynamic Programming (SDP) is quite efficient in the case of a linear or quadratic model, and accommodates large-dimensional decision u, it becomes very slow for high-dimensional state x (dimension > 250). This limitation (also addressed by Approximate Dynamic Programming) is the motivation for Direct Policy Search.

Let us define:

$$u(o) = \prod_\theta(o), \quad (6)$$

for some function $\prod_\theta$ parameterized by $\theta \in \Theta$ (*e.g.*, the weight vector of a neural network). Then θ is optimized by minimization of the *Simul* function below:

Function *Simul*(θ):

- Construct the decision function $o \mapsto u(o)$ as in Eq. (6);
- Compute *Simul*(θ) = cost C for u (using a single simulation).

Using human expertise for defining the $\prod_\theta$ function can make this approach quite efficient and anytime [9]. Note however that *Simul* is a stochastic function, as it depends on the random processes involved in the simulation, making the objective function $\theta \mapsto$ *Simul*(θ) a noisy one. While noisy optimization is a difficult problem (see *e.g.* [20]), a good structured function might make the problem "sufficiently" smooth and convex for the convergence rate of an estimate $\hat{\theta}$ as a function of the computational budget c to be of the form $\mathbb{E}[Simul(\theta) = O(1/c)]$.

1.2.4 Direct value function search

A variant of Direct Policy Search, based on the nice properties of the SDP representation, is proposed in this paper. Formally, while Eq. (5) is kept, the value function $V(x_t)$ is optimized in the spirit of the DPS, as follows:

$$V(x, t) = \prod_{\theta}(x, t), \tag{7}$$

where the optimal θ is optimized according to the *Simul* function below:

Function *Simul*(θ):

- Construct the decision function $o \mapsto u(o)$ as in Eq. (5) using $V(x, t)$ as in Eq. (7);
- Compute *Simul*(θ) = cost C for u (using a single simulation).

An extension is as follows:

$$u(x,t) = \inf_{u_t} \mathbb{E}c_t + \ldots + c_{t+k} + \prod_{\theta}(x_t, x_{t+k+1}, t+1) \tag{8}$$

where

- $\prod_{\theta}(x, x', t)$ is a non-linear function of t and x, enabling the representation of complex controls;
- $\prod_{\theta}(x, x', t)$ is a convex piecewise linear function of x', making Eq. (8) define a linear programming problem if all $c_{t'}$ are linear (quadratic extensions are possible).

This setting benefits from both the scalability of linear or quadratic programming, and the genericity of direct policy search. In particular, one can show that all trajectories can be modeled within this setting, despite the restrictions on the convex piecewise linearity or quadraticity of $\prod_{\theta}(x, x', t)$ w.r.t. x'.

Formally, the proposed approach satisfies the following properties:

1. If the cost c_t is linear as a function of decisions, under linear constraints; if $s \mapsto \prod_{\theta}(x, s)$ is convex piecewise linear; then decision making (Eq. 8) can be achieved in polynomial time.
2. If the cost c_t is Lipschitzian, then any trajectory prescription p: $s \mapsto s'$ can be forced by some function $\prod$ such that $\forall x, t$, the function $x \mapsto \prod_{\theta}(x, x', t)$ is convex piecewise linear with at most $d + 1$ linear cuts, where d is the dimension of the state space.

Proof: Just set $\prod_{\theta}(x, s) = 2l\|x - p(x)\|_1$ (with l the Lipschitz constant for c_t) for showing the result with $2d$ cuts. Optimizing the placement of linear cuts leads to $d + 1$ cuts. ❑

The main issue remains the (noisy) optimization of $\theta \mapsto \mathrm{E}\ \mathit{Simul}(\theta)$.

1.3 Formal Background

In the following, $\#E$ denotes the cardinal of the set E. $N_t(i)$ is the number of times

the parameter i has been tested at iteration t ($N_t(i) = \#\{j \leq t;\ \theta_j = i\}$). $\hat{L}_t(i)$ denotes the average reward for parameter i at iteration t ($\hat{L}_t(i) = \frac{1}{N_t(i)} \sum_{j \leq t;\theta_j = i} r_j$), which is well defined for $N_t(i) > 0$. Upper and lower confidence bounds on $\hat{L}_t(i)$ are given as: $UB_t(i) = \hat{L}_t(i) + \sqrt{\log(t) / N_t(i)}$; $LB_t(i) = \hat{L}_t(i) - \sqrt{\log(t) / N_t(i)}$ (often with a multiplicative constant factor plugged in front of the $\sqrt{\cdot}$).

These confidence bounds are statistically asymptotically consistent estimates of the lower and upper confidence bounds in the 1-player case for a confidence converging to 1. For the EXP3 algorithm (see below), a weighted average is considered with $\hat{W}_t(i) = \frac{1}{t} \sum_{j \leq t;\theta j = i} r_j / p_j(i)$, where $p_j(i)$ is the probability that i is chosen at iteration j given observations available at that time and $\forall i$, $\hat{W}_0(i) = 0$). In the 2-player case, the quantities related to the second player are indicated with a prime ($\hat{W}_t'(j)$, $\hat{L}_t'(j)$).

Following [13], the metagaming/strategic policy building involves:

i) an exploration module, aimed at choosing θ_i (and θ_i' in the 2-player case);

ii) a recommendation module, aimed at choosing $\hat{\theta}$.

The scope of this paper is restricted to the case where the detailed structure of the problem is unknown, and only available through high-level primitives such as the L function. It resumes an earlier work [11] that considered a different context and different bandit algorithms. The best performing algorithms in [11] are considered here together with new ones: the LCB recommendation, Bernstein races, Successive Rejects and Adapt-UCB-E.

2. BANDIT ALGORITHMS

A bandit algorithm involves an exploration and a recommendation strategy. Let us summarize the state of the art in exploration (Section 2.1) and recommendation (Section 2.2); theoretical bounds are briefly discussed in Section 2.3. We then discuss parallelization (Section 2.4) and risk analysis (Section 2.5).

2.1 Algorithms for Exploration

We present below several known algorithms for choosing θ_i, θ_i'. The **UCB (Upper Confidence Bound)** formula is well known since [3, 24]. It is optimal in the 1-player case up to some constants, for the criterion of cumulative regret. The formula is as follows, for some parameter α: $\theta_t = \mathrm{mod}(t, K) + 1$ if $t \leq K$; $\theta_t = \arg\max_i \hat{L}_{t-1}(i) + \alpha \sqrt{\log(t) / N_{t-1}(i)}$ otherwise. The **EXP3 (Exponential weights for Exploration and Exploitation)** algorithm is known in the 2-player case [5]. It converges to the Nash equilibrium of the strategic game. In our variant, $\theta_{t+1} = i$ with probability

$$\frac{\beta}{K\sqrt{t}} + (1 - \beta / \sqrt{t}) \frac{\exp(\sqrt{t}\hat{W}_{t-1}(i))}{\sum_{j \in \{1,\ldots,K\}} \exp(\sqrt{t}\hat{W}_{t-1}(j))}.$$

[13] has discussed the efficiency of the very simple **uniform exploration strategy** in the 1-player case, *i.e.* $\theta_t = \arg\min_{i \in \{1,\ldots,K\}} N_{t-1}(i)$; in particular, it reaches the provably-

optimal expected simple regret $O(\exp(-cT))$ for c depending on the problem. [13] also shows that it reaches the optimal regret, within logarithmic terms, for the non-asymptotic distribution independent framework, with $O(\sqrt{K\log(K)/T})$. [26] has revisited recently the progressive discarding of statistically weak moves, *i.e.* **Bernstein races**; in this paper, we choose, as a next arm for exploration, the arm with smallest number of simulations among arms which are not statistically rejected:

$$\theta_{t+1} = \underset{\substack{i\in\{1,\dots,K\}\\ UB_t(i)\geq \max_k LB_t(k)}}{\arg\min} N_t(i).$$

In many works, Bernstein bounds are used with a large set of arms, and coefficients in LB or UB formula above take into account the number of arms; we will here use the simple LB and UB above as our number of arms is moderate. **Successive Reject (SR)** is a simple algorithm, quite efficient in the simple regret setting; it explores uniformly non-discarded arms and at some given time steps discards the weakest arm; see [1, 16] for more details. **Adaptive-UCB-E** is a variant of UCB, with an adaptive choice of coefficients; see Fig. 3.

- Define $Z = \frac{1}{2} + \sum_{i=2}^{K} 1/i$ and $n_k = \lceil (1/Z)\frac{T-K}{K+1-k} \rceil$.
- Define $t_0 = 0$ and $t_1 = Kn_1$ and $t_k = n_1 + \cdots + n_{k-1} + (K - k + 1)n_k$.
- Define $B_{i,t}(a) = \hat{L}_{t-1}(i) + \sqrt{a/N_{t-1}(i)}$.
- For each epoch $k = 1, \dots, K-1$:
  - Let $H = K$ if $k = 0$, and $H = \max_{K-k+1\leq i\leq K} i\hat{\Delta}^{-2}(< i >, k)$ otherwise, where $\hat{\Delta}_{i,k} = (max_{1\leq j\leq K}\hat{L}_{t-1}(j) - \hat{L}_{t-1}(i)$, and $< i >$ is an ordering such that $\Delta_{<1>,k} \leq \cdots \leq \Delta_{<K>,k}$.
  - For $t = t_k + 1, \dots, t_{k+1}$ choose (exploration) arm i maximizing $B_{i,t}(cn/H)$.
- Recommend i maximizing $L_t(i)$.

Fig. 3. The adaptive-UCB-E algorithm from [1].

Importantly, we will also test progressive widening (see Section 3.1.2) and versions adapted for the risky case (see Section 2.5).

2.2 Algorithms for Final Recommendation

Choosing the final arm, used for the real case, and not just for exploration, might be very different from choosing exploratory arms. Typical formulas are: **Empirically best arm (EBA):** picks up the arm with best average reward. Makes sense if all arms have been tested at least once. Then the formula is $\hat{\theta} = \arg\max_i \hat{L}_T(i)$. **Most played arm**

(MPA): the arm which was simulated most often is chosen. This methodology has the drawback that it can not make sense if uniformity is applied in the exploratory steps, but as known in the UCT literature (Upper Confidence Tree [23]) it is more stable than EBA when some arms are tested a very small number of times (*e.g.* just once with a very good score – with EBA this arm can be chosen). With MPA, $\hat{\theta} = \arg\max_i N_T(i)$. **Upper Confidence Bound (UCB):** $\hat{\theta} = \arg\max_i UB_T(i)$. This makes sense only if $T \geq K$. UCB was used as a recommendation policy in old variants of UCT but it is now widely understood that it does not make sense to have "optimism in front of uncertainty" (*i.e.* the positive coefficient for $\sqrt{t / N_t(i)}$ in the UB formula) for the recommendation step. As Upper Confidence Bound, with their optimistic nature on the reward (they are increased for loosely known arms, through the upper bound), are designed for exploration more than for final recommendation, the **LCB (Lower Confidence Bound)** makes sense as well: $\hat{\theta} = \arg\max_i LB_T(i)$. EXP3 is usually associated with the *empirical* recommendation technique (sometimes referred to as **"empirical distribution of play"**), which draws an arm with probability proportional to the frequency at which it was drawn during the exploration phase; then $P(\hat{\theta} = i) = N_T(i)/T$. For the 2-player case, a variant of EXP3 benefiting from sparsity through truncation (**TEXP3,** Truncated EXP3) has been proposed [21]. It is defined in Fig. 6. For SR (successive reject), there are epochs, and one arm is discarded at each epoch; therefore, at the end there is only one arm, so there is no problem for recommendation.

2.3 Bounds

The interested reader is referred to [1, 14] for a detailed analysis of simple regret bandit problems. For large T values, the best simple regret bounds (in $O(K \log K/T)$ with a universal constant) are obtained by a uniform exploration and EBA as a recommendation. For a UCB exploration, weaker bounds (for large T) are obtained: $O(K / \sqrt{\log T})$ with EBA recommendation and $O(\sqrt{\frac{K \log(T)}{T}})$ with MPA recommendation. In practice however, UCB most usually outperforms uniform exploration by far, as UCB shortcomings are only observed for huge (impractical) values of T.

2.4 Parallel Settings

Let us consider the parallel setting where p arms (possibly redundant) are simultaneously pulled.

We can easily derive the following results from the state of the art:

- **Uniform exploration algorithms for simple regret:** [14] has shown distribution-independent bounds in $O(\sqrt{K \log(K) / n})$ for the simple regret with n rounds in a K-armed bandit with rewards in [0, 1]. This is obtained with uniform exploration; therefore the algorithm is fully parallel and with n rounds with p processors we get $O(\sqrt{K \log(K) /(pn)})$. A limitation of this positive result is that in spite of their theoretical optimality for huge values of n, algorithms with uniform allocation are not that convenient in the real world (see Section 2.3). It should however be mentioned that uniform allocation has obvious simplicity, robustness, parallelization advantages which, besides their mathematical foundation above, are recommended by practitioners in real-world cases [11].

- Upper Confidence Bound for Best Arm Identification: [2] provides a different algorithm, UCB-beta, which ensures that with probability at least $1 - \beta$, at least of the optimal arms is found after a number of iterations linear (within logarithmic factors) in

$$\log(K/\beta) \sum_{\theta \in \{1,\ldots,K\}; \delta_\theta > 0} \left(\frac{VarL(\theta)}{\Delta_\theta} + \frac{1}{\Delta_\theta} \right),$$

where $\Delta_k = \sup_\theta L(\theta) - L(\theta^*)$. When the optimal arm is found, then the simple regret becomes zero, so this is the number of iterations necessary for reaching regret zero.

If we use p processors, we can divide the variance by p, therefore the first term is divided by p, which suggests that we have a linear speed-up as long as the main term is the variance-based term. For $p = O(\inf_k \frac{Var\ L(\theta)}{\Delta_\theta})$ we get a linear speed-up.

2.5 Risk Analysis

Various frameworks for risk analysis are relevant in the metagaming context. For a given risk level α, let us define the conditional Value at Risk of a continuous random variable X, noted $CVaR\ X$, as the average over the quantile α of the X realizations. Then:

- One might want to find an arm $\hat{\theta}$ which is less risky, minimizing on average the $CVaR$ regret defined as $r_T = \max_\theta CVaR\ L(\theta) - CVaR\ L(\hat{\theta})$. In other words, one wants a good outcome result on average over runs. The performance criterion is then set to:

$$\mathbb{E} \max_\theta CVaR\ L(\theta) - CVaR\ L(\hat{\theta}). \tag{9}$$

- One might want to find with high probability an arm which is good on average [22]. An appropriate criterion then regards the quantile α of the worst runs from the point of view of the average outcome reward: $-CVaR\ -(\max_\theta \mathbb{E}L - \mathbb{E}L(\hat{\theta}))$.
- Finally, one might want to maximize the quantile of the outcome:

$$-CVaR(-L(\hat{\theta}^*)) - CVaR(-L(\hat{\theta}))$$

where $CVaR$ operates both on the randomness of the estimator $\hat{\theta}$ and on the random outcome L and where $\hat{\theta}^*$ is an optimal policy (including both an exploration policy and a recommendation policy) for minimizing $-CVaR - L(\hat{\theta}^*)$.

The focus in the following will be on Eq. (9), as our goal is to find on average a non-risky policy. To the best of our knowledge, this criterion has not yet been considered in the bandit literature. We refer to [2, 27] for the risk analysis for cumulative regret.

3. EXPERIMENTAL RESULTS

All presented algorithms are investigated in the 1-player (playing against a stochastic process) and 2-player (playing against an opponent, *i.e.* worst case robustness also referred to as adversarial) case.

3.1 One-Player Case: Playing Against Random

Results on killall-Go were presented in [16]. We here provide results on Energy Management.

3.1.1 Energy management

In this section, we apply the methodology proposed in Section 1.2.4, *i.e.* decision policy $u(o) = \arg\min_u \mathbb{E}(c_t(u) + ... + c_{t+k}) + \prod_\theta(x_{t+k}, x_{t+k+1}, t + k + 1)$.

We here use

$$\prod_\theta(x_t, x_{t+1}, t + 1) = \theta x_{t+1}, \tag{10}$$

which is a very simple model with constant marginal costs; the optimal value of θ was empirically chosen.

We work with $K = 20$ options, corresponding to different investment choices. Each option is associated with $m = 117$ realizations. We here present the results using MPA, EBA, LCB as recommendation rules. We use UCB and Upper Confidence Bound tuned (UCBt) [4] as exploration rules, detailed in Section 2. UCBt is defined as follow: For some parameter α: $\theta_t = \mathrm{mod}(t, K) + 1$ if $t \leq K$;

$$\theta_t = \arg\max_i \hat{L}_{t\text{-}1}(i) + \alpha\sqrt{\log(t) / N_{t-1}(i) \min\{1/4, V_i(N_{t-1}(i))\}},$$

$$V_i(s) = (\tfrac{1}{s}\sum_{\tau=1}^{s} X_{i,\tau}^2) - \bar{X}_{i,s}^2 + \sqrt{\frac{2\log(t)}{s}}$$

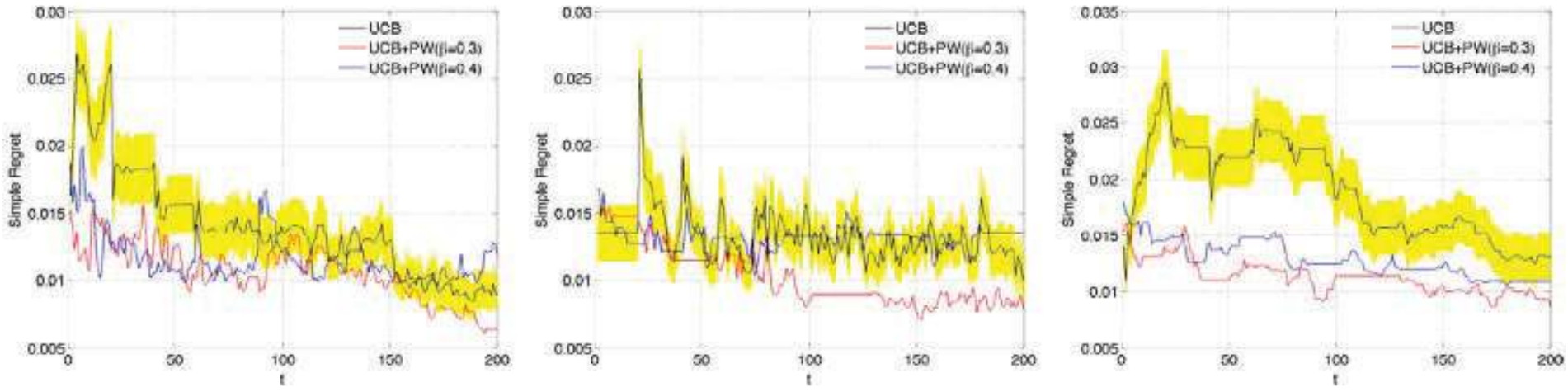
otherwise. We here consider budget $T \leq 100K = 2000$.

3.1.2 Huge number of options or small budget

When $T < K$ or T of the same order as T, it is not reasonable to work with UCB and LCB as if T was large. There should be a compromise between the number of studied arms and the precision of the analysis on different arms.

This is termed progressive widening (PW). The principle of progressive widening is quite simple; instead of pulling the arm with best score at iteration t, we pull the arm with best score among arms with index at most $C\lfloor t^\beta \rfloor$. We here use the same data and same rules as in the Section 3.1.1.

We here consider moderate budget $T \leq 10K = 200$. Fig. 4 shows a comparison between the simple regret without PW and with PW when $C = 2$, $\beta = 0.3$ and $\beta = 0.4$. When β small, for example, $\beta = 0.3$, algorithm using PWoutperforms the simple regret; on the other hand, with $\beta = 0.4$, results are good. In particular, when using UCB as exploration rule and LCB as recommendation rule. In comparison, Successive Reject (SR) has a simple regret of -0.0153 ± 0.0022 (results are averaged on 100 trials). UCB with Progressive widening outperforms Successive Reject for moderate budgets ($5K \leq T \leq 100K$). For smaller budgets, Successive Reject outperforms UCB with Progressive Widening.



(a) Simple Regret UCB-EBA. (b) Simple Regret UCB-MPA. (c) Simple Regret UCB-LCB.

Fig. 4. Results for simple regret with Progressive Widening (PW) using UCB ($\alpha = \sqrt{2}$) as exploration rules. EBA, MPA and LCB (from left to right) are used as recommendation rules. Here, $T = 10K = 200$. Note black line is result without PW, the range of standard deviation of the mean is marked in yellow. Red line is result with PW and $\beta = 0.3$, blue line is result with PW and $\beta = 0.4$. Abscissa is iteration number, ordinate is simple regret. All plots are averages over 100 trials. When T is small, PW outperforms other algorithms in spite of some instabilities when T corresponds to a newly added arm.

3.1.3 Energy management: comparing different master plans

Consider a problem in which the tactical level is hard. Then, the strategic level has to decide, for each simulation at the tactical level, the computational power devoted to this simulation. Formally, $L(\theta)$ is replaced by $L(\theta, c)$ where c is a time budget for the simulation. c large implies more precision (less bias) on the estimation; however, it also reduces the budget T for a given runtime, because the runtime will be nearly $T \times c$. Classical simple regret bounds (see *e.g.* [14] for a survey) show that the optimal expected precision is $\leq C \times K \log K/T$ (reached *e.g.* by uniform exploration and EBA), where C is an absolute (*i.e.* distribution-independent) constant; therefore, if we want a precision ϵ, we know that we need T such $CK \log K/T \leq \epsilon$. Deciding the runtime c necessary for a bias upper bounded by ϵ is more difficult as existing bounds are far from being practical on non-trivial sequential decision making problems.

In our work, we consider the following problem: the agent needs to decide how to distribute a limited number $K + 1$ of sites to energy production facilities. A site can be either a thermal power plant (TPP) or a hydro-electric plant (HEP), with the constraint that we must have at least one of each facility. This means that there are exactly K different possible configurations (the order does not matter): 1 TPP, 2 TPP, …, K TPPs. The resulting problem is a non linear stochastic sequential decision making problem (inflows and energy demand are random).

Here, each possible configuration of the problem is an arm. The real value of such an arm is the expected cost of such a configuration under optimal management of the production facilities.

In our experiment, we used a version of continuous MCTS (Monte-Carlo Tree Search [18]), as described in [17], to evaluate each arm. The longer MCTS runs, the less noisy and the closer it gets to the optimal value of a given configuration. There is an obvious dilemma, between spending more time on each arm evaluation, and playing more arms. This is not the focus of this paper, but we believe this should be addressed in a future work.

Table 1. (a) "real" value of each arm; computed by giving MCTS a large budget (100s); (b) expected simple regret of different bandit algorithms, as a function of the budget allowed to arm plays and to *T*.

(a)

k	Cost
1	3.54 ± 0.025
2	3.20 ± 0.015
3	2.94 ± 0.039
4	2.36 ± 0.11
5	1.94 ± 0.11
6	$\mathbf{1.04} \pm 0.14$
7	$4.00 \cdot 10^2 \pm 0.059$

(b)

MCTS budget (s) T	3.2 16	3.2 64	3.2 256	12.8 16	12.8 64
UCB+EBA	1.78 ± 0.11	2.34 ± 0.056	.	0.62 ± 0.12	1.32 ± 0.12
UCB+MPA	1.86 ± 0.12	2.36 ± 0.046	.	0.50 ± 0.091	1.50 ± 0.11
UCB+LCB	1.79 ± 0.11	2.34 ± 0.055	.	0.50 ± 0.091	1.50 ± 0.11
UCB+median+MPA	1.68 ± 0.11	2.27 ± 0.075	2.40 ± 0.059	0.39 ± 0.095	1.41 ± 0.13
UE+EBA	1.03 ± 0.13	2.18 ± 0.078	.	$\mathbf{0.21 \pm 0.067}$	0.93 ± 0.14
UE+LCB	1.67 ± 0.16	2.19 ± 0.077	.	0.26 ± 0.093	0.93 ± 0.14
SR	1.47 ± 0.12	2.28 ± 0.068	2.40 ± 0.036	0.32 ± 0.076	1.15 ± 0.15
SR+median	1.11 ± 0.13	1.04 ± 0.15	$\mathbf{0.70 \pm 0.095}$	0.28 ± 0.080	$\mathbf{0.075 \pm 0.071}$

We considered a problem with 8 locations, and at least one HEP and one TPP (K = 7). The time horizon is of 5 time steps. An approximation of the real value of each arm is shown in Table 1.

This problem is particularly difficult for two reasons. First, all the K configurations of the problem are not equally easy to solve for MCTS. Hence, for a small tactical budget c, it is very likely that $\theta^*_c = \arg\max_\theta \mathbb{E}L(\theta, c)$ will be different from $\theta^* = \arg\max_\theta \mathbb{E}L(\theta)$. This means that if c is not big enough, the bandit algorithm is going to converge towards a suboptimal arm. Secondly, the distribution followed by each arm is very far from satisfying common assumptions of bounded rewards and variances. Our problem does not have bounded rewards. And, because for small tactical budgets, MCTS will occasionally make mistakes, the distribution of some arms is heavy tailed. This is particularly harmful to the stability of the mean estimator, that is crucial in most bandit algorithms studied here.

We compared five different bandit strategies: UCB, Successive Reject, and uniform exploration, UCB and SR using the median of the rewards instead of the mean reward. When it made sense, we used up to three different algorithms for final recommendation: LCB, EBA, and MPA. Our results are shown in Table 1.

Discussion From our results, it seems like the most determinant factor in reducing the simple regret is the budget given to MCTS. At a MCTS budget of 3.2 or 12.8 seconds, an increase in T actually decreases the performances of most algorithms. The only bandit algorithm that works in this setting is SR using the median. Looking closely at our data, it appears that the main reason for the poor performance of all the other algorithms is due to the heavy tail distribution of the arms. In such a setting, the mean estimator does not work. One needs to use a truncated version of the mean.

Most likely, increasing the MCTS budget to a very high value until the arms are no longer heavy tailed would make the mean based algorithms more efficient. But this could prove impractical. We think that an important future work would be to find an adaptive method to progressively increase the bandit budget and the MCTS budget, in order to avoid wasteful increases in bandit budget, as seen in our results (see Table 1).

3.1.4 Energy management with risk

We here work on the risky case, as discussed in Section 2.5, *i.e.* we aim at minimizing

$$\mathbb{E}(\max_{\theta} CVaRL(\theta) - CVaRL(\hat{\theta})). \quad (11)$$

We work on the same data as in Section 3.1.1. We modify recommendation rules and exploration rules as follows for this case:

- EBA, LCB become EBA_{risk} and LCB_{risk} by replacement of $\hat{L}_T$ by the empirical estimate $\widehat{CVaRL_T}$ of $CVaR\ L_T$.
- UCB becomes UCB_{risk} by defining θ_i as follows

$$\theta_i = \arg\max_i \hat{C}VaR\ L_t(i) + \sqrt{\alpha \log(t-1) / N_{t-1}(i)} \text{ otherwise.}$$

Experiments have been conducted to assess the ability of the proposed modified strategies to minimize the risk criterion defined by Eq. (11). Their results obtained with a selection of parameters are given by Fig. 5.

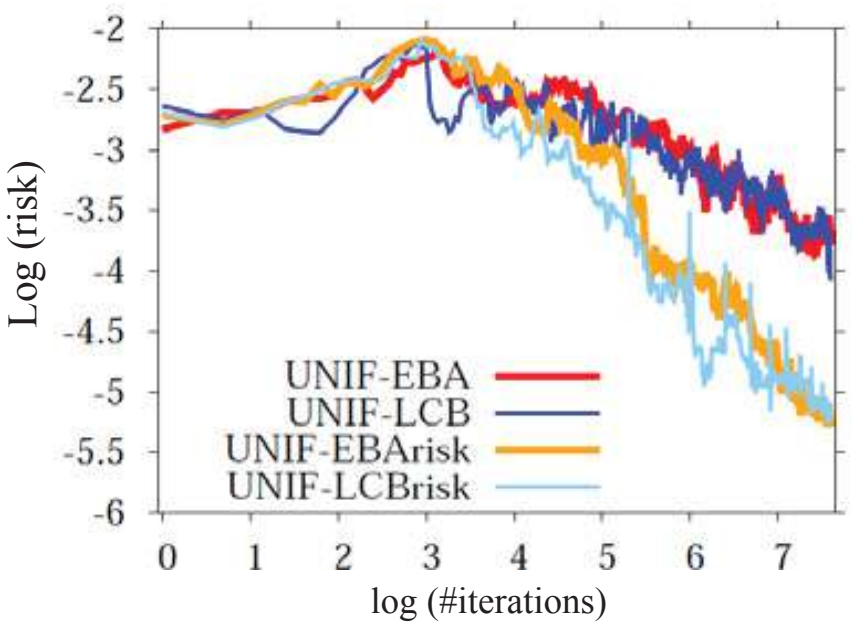
In the first experience (Fig. 5 (a)), the exploration strategy is fixed to uniform. The comparison between the EBA/LCB and their risk EBA_{risk}/LCB_{risk} counterparts demonstrates that recommendation rules can be specifically derived for risk minimization. Moreover, these modified rules outperform the previously presented approaches by the only change of the recommendation strategy. In the same manner, by fixing the recommendation procedure to MPA, Fig. 5 (b) shows that one can minimize risk by only switching exploration rule from UCB to UCB_{risk}. Risk can thus be minimized through an alternative choice of exploration and/or recommendation strategy. Furthermore, Fig. 5 (c) also indicates that uniform exploration with risk-sensitive recommendations (EBA_{risk} or LCB_{risk}) leads to comparable performances than UCB_{risk} with MPA recommendation. Nevertheless, these results emphasize as well the importance of the combination of rules. For instance, Fig. 5 (b) points out that UCB_{risk}-EBA configuration results are close to results for UCB-MPA and are clearly outperformed by the UCB-EBA_{risk} approach. Results show that it is worth using EBA_{risk} and LCB_{risk} rather than EBA and LCB, even with moderate values of T. Also, experiences suggest, in a risk minimization perspective, that uniform exploration should always be associated with EBA_{risk} or LCB_{risk} and UCB_{risk} with EBA_{risk}, LCB_{risk} or MPA recommendation. In the last case, Fig. 5 (d) shows that the combination of UCB_{risk} with EBArisk or LCB_{risk} doesn't lead to an increase of performances in comparison to UCBrisk-MPA.

3.2 Two-Player Case: Sparse Adversarial Bandits

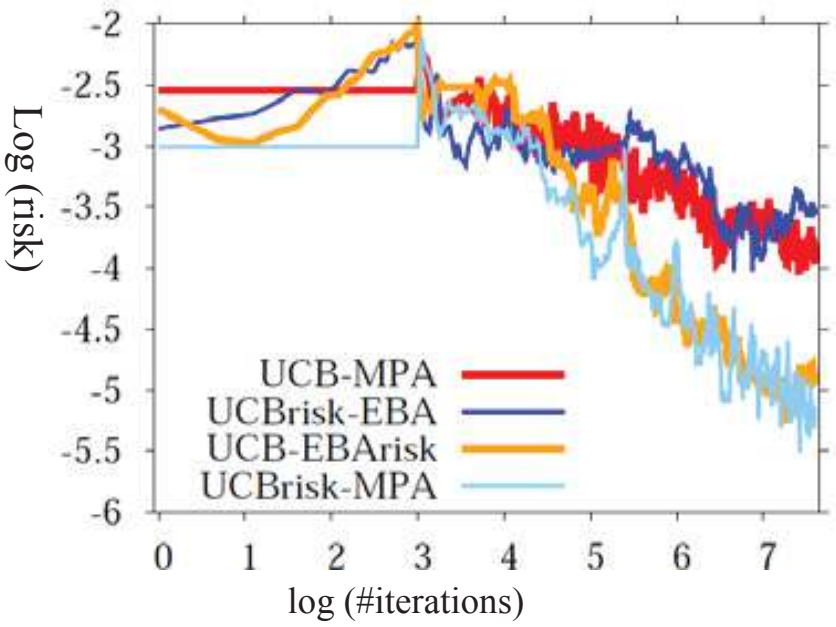
This section is devoted to experiments around strategic optimization in 2-player cases. Section 3.2.1 discusses the impact of sparsity. Section 3.2.2 presents experimental results on Urban Rivals.

3.2.1 Sparsity in 2-player games

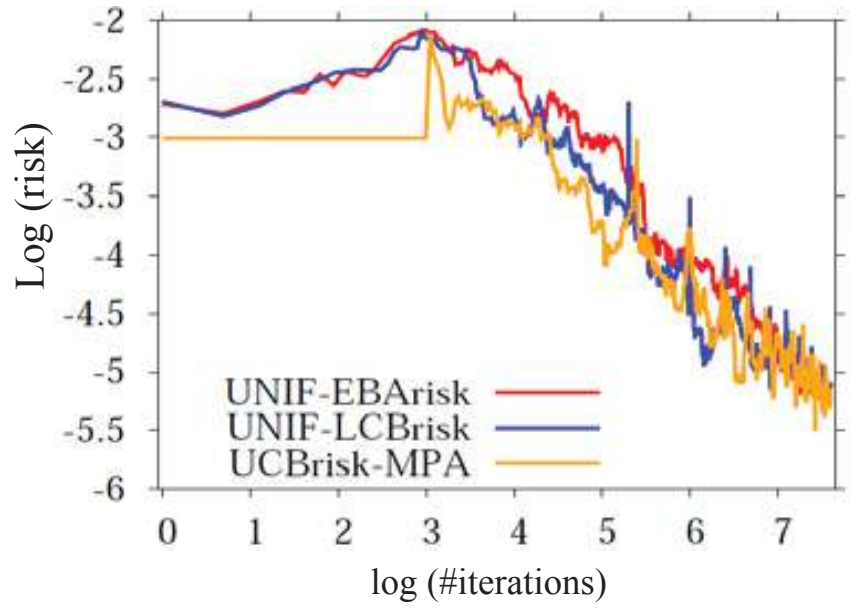
Recently [21] proposed a variant of EXP3 called TEXP3. TEXP3 takes its root into the fact that decision making algorithms in games rarely have enough time to reach the nice asymptotic behavior guaranteed by EXP3. Also, EXP3 fails to exploit that in most games, the number of good moves is rather low compared to the number of possible



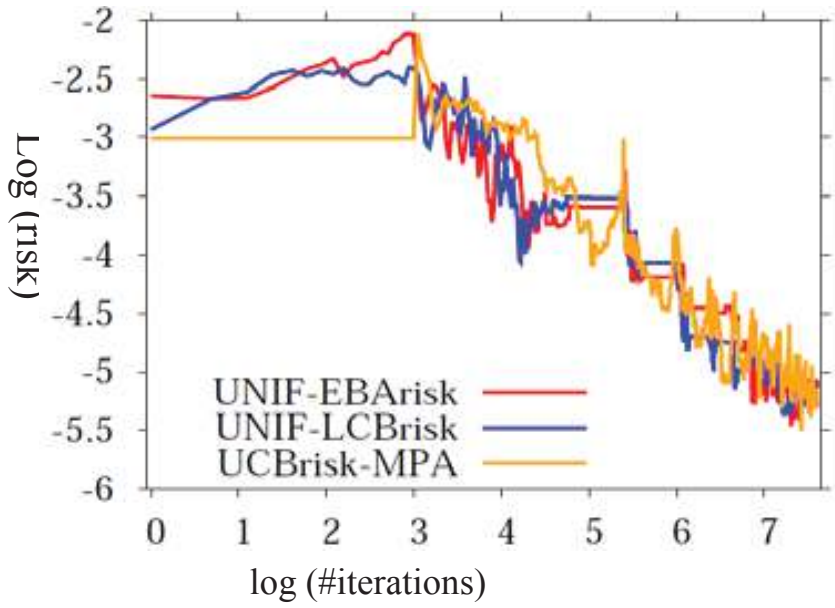
(a) Comparison of LCB/EBA and LCB_{risk}/ EBA_{risk} with an uniform exploration.



(b) Comparison of UCB/UCB_{risk}-MPA with UCB-EBA_{risk} and UCB_{risk}-EBA.



(c) Comparison of UNIF-EBA_{risk}/LCB_{risk} with UCB_{risk}-MPA.



(d) Comparison of LCB_{risk}, EBA_{risk} and MPA with UCB_{risk} exploration.

Fig. 5. Results for risk log (risk) is plotted against log (#iterations).

moves K. TEXP3 is an attempt to exploit these two characteristics.

$T = 100 \times K = 2000$. The risk level is fixed to $\alpha_{risk} = 0.1$. For UCB, LCB , UCB_{risk} and LCB_{risk}, $\alpha = \sqrt{2}$. Plots are averaged over 100 trials.

It uses the outcome of EXP3 and truncates the arms that are unlikely to be part of the solution. Algorithm 6 describes the implementation. The constant c is chosen as $1/T$ $\max_i(Tx_i)^{\alpha}$ for some $\alpha \in]0, 1[$ (and d accordingly), as in [21], while T is the number of iterations executed. We set $\alpha = 0.7$ in our experiments, following [7, 21].

The natural framework of EXP3 is a 2-player game.

3.2.2 Urban rivals

In this section we apply EXP3 and TEXP3 to Urban Rivals, a stochastic card games available for free on Internet and that fits the framework. The game is as follows: (1) player 1 choose a combination $\theta_1 \in \{1, \ldots, K_1\}$ of cards; (2) simultaneously, player 2 choose a combination $\theta' \in \{1, \ldots, K'\}$ of cards; (3) then the game is resolved (ingaming). We consider a setting in which two players choose 4 cards from a finite set of 10 cards. There exists 10^4 combinations, yet by removing redundant arms, we remain with 715

different possible combinations (both $K_1 = K_2 = 715$), allowing the same card to be used more than once. The first objective is to test whether EXP3 (and TEXP3) is stronger than a random player for different numbers of iterations T. We are specifically interested in situation where T is small (compared to $K_1 \times K_2$) as it is typically the case in games. Table 7 (left) present the score (in %) of EXP3 versus a random player. EXP3 significantly beats the random player when $T > 25000$. It can thus execute a strategic choice that outperforms a random player when they have similar tactical capabilities. As T grows, the strategic choice becomes better. Next we look into a way to make an even better choice with a smaller T. Recently TEXP3 has been proven to outperform a random player with less information than EXP3 (experimentally in [21], theoretically in [7]). Table 7 (right) presents the performance of TEXP3 against a random player under the same settings as EXP3 above. These results are in line with previous studies; however, the improvement is much better – probably because we have here a highly sparse problem. Even with the lowest setting (T = 10000), TEXP3 managed a strong performance against a random player. Again, with little information ($T << K_1 \times K_2$), TEXP3 can make strategic choices that influence the outcome of the game positively; furthermore, it clearly outperforms EXP3.

Let x and y be the approximate Nash equilibria as proposed by EXP3 for the row and column players respectively. Truncate as follows

$x_i' = x_i$ if $x_i > c$, $x_i' = 0$ otherwise;
$y_i' = y_i$ if $y_i > c$, $y_i' = 0$ otherwise.

Renormalized: $x'' = x'/\sum_i x_i'$; $y'' = y'/\sum_i y_i'$.
Output x'', y''.

Fig. 6. TEXP3 (truncated EXP3), offline truncation post-EXP3.

T	Score $\pm 1\sigma$
10 000	0.5042 ± 0.001
25 000	0.5278 ± 0.001
50 000	0.5421 ± 0.002
100 000	0.5749 ± 0.004

T	Score $\pm 1\sigma$
10 000	0.7206 ± 0.005
25 000	0.7238 ± 0.003
50 000	0.7477 ± 0.002
100 000	0.7871 ± 0.006

Fig. 7. EXP3 vs Random (left) and TEXP3 vs Random (right).

4. CONCLUSIONS AND PERSPECTIVES

The investigation of the strategic decision making problem **in the 1-player case** demonstrates that the most important component is the exploration one, and that **UCB generally is the best exploration algorithm** (except for very small budget, see below). This result, consistent with earlier results on artificial problems [13], is surprising as the performance criterion is the simple regret (the natural one for strategic choices): Despite their asymptotic optimality (within logarithmic factors) for both a fixed distribution and a distribution-free setting w.r.t simple regret, uniform exploration algorithms are outperformed by UCB by far. It is worth noting that the UCB variant referred to as Adapt-UCB-E, designed for parameter free simple regret, behaves well. Lastly and importantly, **Successive Reject outperforms UCB variants for the small time budget**; SR, designed for simple regret, shows a very stable behavior (never very bad). However, for very small time budgets, Progressive widening is necessary; see below.

Regarding the **recommendation** algorithms, the results confirm the usual practice in Monte-Carlo Tree Search: the **lower confidence bound** performed very well; the most played arm and the empirically best arm also performed well. It must be emphasized that many practitioners in the Computer-Go literature (based on heavily tuned bandit algorithms) use combinations of EBA and MPA and LCB as recommendation arms for optimal performance. As could have been expected, and was also reported in [11], EBA becomes weaker as the number of arms increases. Bernstein races performed moderately well; further work will be devoted to check whether hyper-parameter tuning can improve their results. Adapt-UCB-E performs well, though it remains comparable to SR or other UCB variants. Last but not least, **Progressive Widening is shown to be highly efficient for moderate *K* values.**

Risk. Several risk criteria have been defined (Section 3.1.4), relatively to the algorithm stochasticity and relatively to the outcome distribution; these criteria are not equivalent. The performance was shown to significantly improve when using risk-sensitive criteria in both exploration and recommendation, even with a moderate time budget.

Progressive widening and very small time budget. Progressive widening, first proposed by [19, 28] to handle a large number of options, was extended to stochastic tree search with continuous domains in [17]: at iteration t, one selects the arm with best score with index at most $C\lfloor t^\beta \rfloor$. Significant evidence confirms the relevance of the PW heuristics (with an exponent β circa 0.3). Notably, PW is the only relevant algorithm for $T < K$, contrasting with other algorithms, including successive reject.

In the 2-player case, EXP3 performed very well, confirming earlier results [6, 12, 21]; this was expected as EXP3 is dedicated to the adversarial setting. The efficiency of the truncation algorithm TEXP3 on sparse problems [21] was also confirmed (as mentioned, metagaming often yields sparse problems). Unexpectedly, the TEXP3 results were much better than in the original paper [21], which is explained from the higher sparsity of the benchmark problems. Results include experiments on a real game, Urban Rivals, where the strategic choice consists in choosing the cards.

Further work. No prior knowledge about the structure of the arm space (*e.g.* domain knowledge, similarity) was involved in this paper. In the 1-player case, the literature on how to use prior knowledge in a bandit setting (*e.g.* progressive widening, progressive unpruning) is advancing at rapid pace. How to use a structure on the arm space (*e.g.* similarity or distance between arms) [15, 17, 19, 25] is among our perspectives for further research.

In the 2-player case, both the structure of the search space, and the prior knowledge on the adversarial strategic choices, have been to a large extent ignored in the literature. How to handle such prior knowledge is the main goal of our further work.

REFERENCES

1. J.-Y. Audibert and S. Bubeck, “Best arm identification in multi-armed bandits,” in

Proceedings of Annual Conference on Learning Theory, 2010, pp. 41-53.
2. J.-Y. Audibert, R. Munos, and C. Szepesvari, "Use of variance estimation in the multi-armed bandit problem," in *Proceedings of Workshop on On-line Trading of Exploration and Exploitation*, 2006, pp. 1-8.
3. P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite time analysis of the multiarmed bandit problem," *Machine Learning*, Vol. 47, 2002, pp. 235-256.
4. P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, Vol. 47, 2002, pp. 235-256.
5. 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*, 1995, pp. 322-331.
6. D. Auger, "Multiple tree for artially observable Monte-Carlo tree search," in C. D. Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcázar, J. J. M. Guervós, F. Neri, M. Preuss, H. Richter, J. Togelius, and G. N. Yannakakis, ed., *Evo Applications* (1), LNCS, Vol. 6624, 2011, pp. 53-62.
7. D. Auger, S. Ruette, and O. Teytaud, "Sparse bandit algorithms," http://www.math.u-psud.fr/~ruette/articles/sparsebandit.pdf.
8. R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.
9. Y. Bengio, "Using a financial training criterion rather than a prediction criterion," *International Journal on Neural Systems*, Vol. 8, 1997, pp. 433-443.
10. D. Bertsekas and J. Tsitsiklis, *Neuro-dynamic Programming*, Athena Scientific, 1996.
11. A. Bourki, M. Coulm, P. Rolet, O. Teytaud, and P. Vayssière, "Parameter tuning by simple regret algorithms and multiple simultaneous hypothesis testing," in *Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics*, Vol. 1, 2010, pp. 169-173.
12. B. Bouzy and M. Métivier, "Multi-agent learning experiments on repeated matrix games," in *Proceedings of International Conference on Machine Learning*, 2010, pp. 119-126.
13. S. Bubeck, R. Munos, and G. Stoltz, "Pure exploration in multi-armed bandits problems," in *Proceedings of the 20th international conference on Algorithmic Learning Theory*, 2009, pp. 23-37.
14. S. Bubeck, R. Munos, and G. Stoltz, "Pure exploration in finitely-armed and continuous-armed bandits," *Theoretical Computer Science*, Vol. 412, 2011, pp. 1832-1852.
15. G. Chaslot, M. Winands, J. Uiterwijk, H. van den Herik, and B. Bouzy, "Progressive strategies for Monte-Carlo tree search," in *Proceedings of the 10th Joint Conference on Information Sciences*, 2007, pp. 655-661.
16. C.-W. Chou, P.-C. Chou, C.-S. Lee, D. L. Saint-Pierre, O. Teytaud, M.-H. Wang, L.-W. Wu, and S.-J. Yen, "Strategic choices: Small budgets and simple regret," in *Proceedings of Conference on Technologies and Applications of Artificial Intelligence*, 2012, pp. 182-187.
17. A. Couetoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, 2011, pp. 433-445.
18. R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search,"

in *Proceedings of the 5th International Conference on Computers and Games*, 2006, pp. 72-83.
19. R. Coulom, "Computing elo ratings of move patterns in the game of go," in *Proceedings of Computer Games Workshop*, 2007, pp. 198-208.
20. V. Fabian, "Stochastic approximation of minima with improved asymptotic speed," *Annals of Mathematical Statistics*, Vol. 38, 1967, pp. 191-200.
21. S. Flory and O. Teytaud, "Upper confidence trees with short term partial information," in *Proceedings of EvoGames*, accepted, 2011, pp. 152-162.
22. V. Gabillon, M. Ghavamzadeh, and A. Lazaric, "Best arm identification: A unified approach to fixed budget and fixed confidence," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, 2012, pp. 3221-3229.
23. L. Kocsis and C. Szepesvari, "Bandit based Monte-Carlo planning," in *Proceedings of the 15th European Conference on Machine Learning*, 2006, pp. 282-293.
24. T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, Vol. 6, 1985, pp. 4-22.
25. C.-S. Lee, M.-H. Wang, G. Chaslot, J.-B. Hoock, A. Rimmel, O. Teytaud, S.-R. Tsai, S.-C. Hsu, and T.-P. Hong, "The computational intelligence of MoGo revealed in Taiwan's computer Go tournaments," *IEEE Transactions on Computational Intelligence and AI in Games*, 2009, pp. 73-89.
26. V. Mnih, C. Szepesv´ari, and J.-Y. Audibert, "Empirical Bernstein stopping," in *Proceedings of the 25th ACM International Conference on Machine Learning*, 2008, pp. 672-679.
27. A. Sani, A. Lazaric, and R. Munos, "Risk-aversion in multi-armed bandits," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, 2012, pp. 3284-3292.
28. Y. Wang, J.-Y. Audibert, and R. Munos, "Algorithms for infinitely many-armed bandits," *Advances in Neural Information Processing Systems*, Vol. 21, 2008, pp. 1729-1736.