Strategic Choices: Small Budgets and Simple Regret

Cheng-Wei Chou, Ping-Chiang Chou, Chang-Shing Lee, David L. Saint-Pierre, Olivier Teytaud, Mei-Hui Wang, Li-Wen Wu, Shi-Jim Yen

Strategic Choices: Small Budgets and Simple Regret

Cheng-Wei Chou [1], Ping-Chiang Chou, Chang-Shing Lee [2],
David Lupien Saint-Pierre [3], Olivier Teytaud [4], Mei-Hui Wang [2],
Li-Wen Wu [2] and Shi-Jim Yen [2]

1Dept. of Computer Science and Information Engineering NDHU, Hualian, Taiwan

2Dept. of Computer Science and Information Engineering National University of Tainan, Taiwan

3Montefiore Institute Université de Liège, Belgium

4TAO (Inria), LRI, UMR 8623(CNRS - Univ. Paris-Sud) bat 490 Univ. Paris-Sud 91405 Orsay, France

March 18, 2013

Abstract

In many decision problems, there are two levels of choice: The first one is strategic and the second is tactical. We formalize the difference between both and discuss the relevance of the bandit literature for strategic decisions and test the quality of different bandit algorithms in real world examples such as board games and card games. For exploration-exploitation algorithm, we evaluate the Upper Confidence Bounds and Exponential Weights, as well as algorithms designed for simple regret, such as Successive Reject. For the exploitation, we also evaluate Bernstein Races and Uniform Sampling. As for the recommandation part, we test Empirically Best Arm, Most Played, Lower Confidence Bounds and Empirical Distribution. In the one-player case, we recommend Upper Confidence Bound as an exploration algorithm (and in particular its variants adaptUCBE for parameter-free simple regret) and Lower Confidence Bound or Most Played Arm as recommendation algorithms. In the two-player case, we point out the commodity and efficiency of the EXP3 algorithm, and the very clear improvement provided by the truncation algorithm TEXP3. Incidentally our algorithm won some games against professional players in kill-all Go (to the best of our knowledge, for the first time in computer games).

Keywords: Go; Bandit problem; Recommendation policy; Selection policy; Metagaming.

1 Introduction

Many important optimization problems can be separated in two parts: strategic decisions and tactical behavior. Table 1 provides several examples in industry and games. In the recent years, a wide body of theoretical and experimental work, namely the bandit literature, has been developed around one-step and often unstructured decision making. However, strategic decisions are specific bandit problems; they usually have a very restricted time budget and, in the two-player case, a huge sparsity (in the sense that the optimal solution, namely a Nash equilibrium, is usually sparse). This paper is devoted to the analysis of the relevance of this literature for strategic decisions.

In this section we will formalize the problem (1.1) and present the notations (1.2). In Section 2 we will present classical algorithms. In Section 3 we will test them experimentally; we will conclude in Section 4.

1.1 Formalization of the problem

There's no clear formal definition of what is a strategic choice, compared to a tactical choice. However, the idea is that a strategic choice is at a higher level; we will formalize this as follows: in a strategic bandit problem, the number of iterations T is not huge compared to the number of options (K in the one player case, or $K \times K'$ in the two player case). In the one-player case, we will therefore focus on $T \leq 100K$, and in

Real world examples		
Electricity Production	Choosing the maintenance dates	Choosing (real-time) how to satisfy the demand (switching on/off the plants)
Logistics	Warehouse/factory positioning	Trucks/boats/trains scheduling
Military operations	Choosing the date & the planning	Military tactics
Games		
Handicap Go	Placing the handicap stones	Standard Go gameplay
Batoo	Opening stones	Batoo variant of Go gameplay
Chess	Opening choice	Chess gameplay
New card games (Pokemon, Urban Rivals)	Choosing the deck	Choosing cards/attacks

Table 1: Problems with strategy/tactics decomposition. Sometimes, it would be possible to define more levels (e.g. deciding investments for electricity production). Batoo is a recent yet popular game with strong strategic decision: the choice of initial stones.

the two-player case $T \leq K \times K'$. Also, we use simple regret, and not cumulative regret, for the one-player case; and average performance of the recommended distribution for the two player case, which is somehow a natural extension of simple regret for the two player case.

Let us consider a set of strategic choices, also termed arms or options in the bandit literature, denoted without loss of generality by $\{1, \ldots, K\}$. We want to choose $\theta \in \{1, \ldots, K\}$ for some performance criterion. We have a finite time budget T (also termed horizon), which means that we can have access to T realizations $L(\theta_1), L(\theta_2), \ldots, L(\theta_T)$ and we then choose some $\hat{\theta}$. This is the metagame in the one-player case; it is detailed in Fig. 1. There are several remarks on this framework:

- The algorithm chooses $\theta_1 \in \{1, \ldots, K\}$.
- The algorithm gets a reward r_1 distributed as $L(\theta_1)$.
- The algorithm chooses $\theta_2 \in \{1, \ldots, K\}$.
- The algorithm gets a reward r_2 distributed as $L(\theta_2)$.
- ...
- The algorithm chooses $\theta_T \in \{1, \ldots, K\}$.
- The algorithm gets a reward r_T distributed as $L(\theta_T)$.
- The algorithm chooses $\hat{\theta}$.
- The loss, termed simple regret, is $r_T = \max_\theta \mathbb{E}L(\theta) - \mathbb{E}L(\hat{\theta})$.

Figure 1: Metagaming with one player.

- For evaluating $L(\theta_i)$, we need a simulator, including the tactical decisions. This possibility is based on the assumption that we can simulate the tactical choices once the strategic choices have been made.
- Without loss of generality, the simple regret is always positive, and the goal is to have a simple regret as small as possible.

In the two player case, the framework is detailed in Fig. 2. As in the one-player case, the loss is always positive (without loss of generality), and the goal is to have a loss as small as possible. There are several remarks on this framework:

- As in the one-player case, we assume that we can simulate the tactical behaviors (included the tactical behavior of the opponent). Basically, this is based on the assumption that the opponent has a strategy that we can nearly simulate, or the assumption that the difference between the strategy we choose for the opponent and the opponent's real strategy is not a problem (playing optimally against the first is nearly equivalent to playing optimally against the latter). This is a classical assumption in many game algorithms; however, this might be irrelevant for e.g. Poker, where opponent modelization is a crucial component of a strategy for earning money; it might also be irrelevant in games in which humans are by far stronger than computers, as e.g. the game of Go.

- The algorithm chooses $\theta_1 \in \{1, \dots, K\}$ and $\theta'_1 \in \{1, \dots, K'\}$.
- The algorithm gets a reward r_1 distributed as $L(\theta_1, \theta'_1)$.
- The algorithm chooses $\theta_2 \in \{1, \dots, K\}$ and $\theta'_2 \in \{1, \dots, K'\}$.
- The algorithm gets a reward r_2 distributed as $L(\theta_2, \theta'_2)$.
- ...
- The algorithm chooses $\theta_T \in \{1, \dots, K\}$ and $\theta'_T \in \{1, \dots, K'\}$.
- The algorithm gets a reward r_T distributed as $L(\theta_T, \theta'_T)$.
- The algorithm chooses a random variable $\hat{\theta}$.
- 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).

Figure 2: Metagaming with two players.

- We use a simple regret algorithm; this is somehow natural (under assumptions above) as the simple regret is directly the expected increase of loss due to the strategic choice (at least, if we trust assumptions above which ensure that $L(\cdot, \cdot)$ is a good sampling of possible outcomes).

In the game literature, the non-strategic part is usually termed "ingaming" for pointing out the difference with the metagaming.

1.2 Terminology, notations, formula

Useful notations: $\#E$ is the cardinal of the set E. $N_t(i)$ is the number of times the parameter i has been tested at iteration t, i.e. $N_t(i) = \#\{j \leq t; \theta_j = i\}$. $\hat{L}_t(i)$ is the average reward for parameter i at iteration t, i.e. $\hat{L}_t(i) = \frac{1}{N_t(i)} \sum_{j \leq t; \theta_j = i} r_j$ (well defined if $N_t(i) > 0$). Confidence bounds will be useful as well: $UB_t(i) = \hat{L}_t(i) + \sqrt{\log(t)/N_t(i)}$; $LB_t(i) = \hat{L}_t(i) - \sqrt{\log(t)/N_t(i)}$.
Various constants are sometimes plugged into these formula (e.g. a multiplicative factor in front of the $\sqrt{.}$). These confidence bounds are statistically asymptotically consistent estimates of the lower and upper confidence bounds in the one-player case for a confidence converging to 1. In some cases, we need a weighted average as follows (with $\forall i, \hat{W}_0(i) = 0$): $\hat{W}_t(i) = \frac{1}{t} \sum_{j \leq t; \theta_j = i} r_j / p_j(i)$ where $p_j(i)$ is the probability that i is chosen at iteration j given observations available at that time. This will in particular be useful for EXP3. When there are two players, similar notations with a ' are used for the second player: $\hat{W}'_t(j)$, $\hat{L}'_t(j)$,...

As we can see in Figs 1 and 2, specifying a metagaming algorithm implies specifying several components:

- The tactical simulators (necessary for computing L), which, given the sequence of strategic and tactical decisions, provide the loss; this is part of the problem specification.
- The simulator of our tactical strategy; this is also necessary for computing L. We will not work on this part, which is precisely not the meta-gaming part.
- For the two-player case, the simulator of the opponent's strategy as well. This could be considered as a part of metagaming because the uncertainty on the opponent's strategy should, in a perfect world, be taken into account in the strategic module. However, we simplify the problem by assuming that such a simulator is given and fixed and satisfactory.
- The two components which are the core of metagaming/strategic choices (following the terminology of [1]):
  - exploration module, aimed at choosing θ_i and θ'_i (the latter in the two-player case);
  - recommendation module, aimed at choosing $\hat{\theta}$.

The underlying assumption in this paper is that we do not seek to work on the detailed structure or the problem, and we just want to have access to it through high-level primitives like the L function. [2] has done a similar comparison, with a different family of bandits and a different context; we here use their best performing bandits, and add some new ones (the LCB recommendation, Bernstein races which were cited but not tested, Successive Rejects and Adapt-UCB-E).

- Define $Z = \frac{1}{2} + \sum_{i=2}^{K} 1/i$ and $A = \{1, \ldots, K\}$ and $n_0 = 0$ and $n_k = \lceil (1/Z) \frac{T-K}{K+1-k} \rceil$ for $k \geq 1$.
- For each epoch $k = 1, \ldots, K-1$:
  - For each $i \in A$, choose (exploration) arm i during $n_k - n_{k-1}$ steps.
  - Then, remove from A the arm with worse average reward.
- Recommend the unique remaining element of A.

Figure 3: The Successive Reject algorithm from [7] for K arms and T iterations.

2 Algorithms

We summarize below the state of the art, for exploration and for recommendation.

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, 4]. It is optimal in the one player case up to some constants, for the criterion of cumulative regret. The formula is as follows, for some parameter α: $\theta_t = mod(t, K) + 1$ 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 two-player case[5]. It converges to the Nash equilibrium of the strategic game. In our variant, $\theta_{t+1} = i$ with probability
$$\frac{\beta}{K\sqrt{t}} + (1 - \beta/\sqrt{t}) \frac{\exp(\sqrt{t}\hat{W}_{t-1}(i))}{\sum_{j \in \{1,\ldots,K\}} \exp(\sqrt{t}\hat{W}_{t-1}(j))}.$$
- [1] has discussed the efficiency of the very simple **uniform exploration strategy** in the one-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. [1] also shows that it reaches the optimal regret, within logarithmic terms, for the non-asymptotic distribution independent framework, with $O(\sqrt{K \log(K)/T})$.
- [6] has revisited recently the progressive discarding of statistically weak moves, i.e. **Bernstein races**; in this paper, we choose the arm with smallest number of simulations among arms which are not statistically rejected:
$$\theta_{t+1} = \arg\min_{\substack{i \in \{1,\ldots,K\} \\ UB_t(i) \geq \max_k LB_t(k)}} N_t(i).$$
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; see Fig. 3.
- Adaptive-UCB-E is a variant of UCB, with an adaptive choice of coefficients; see Fig. 4

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:

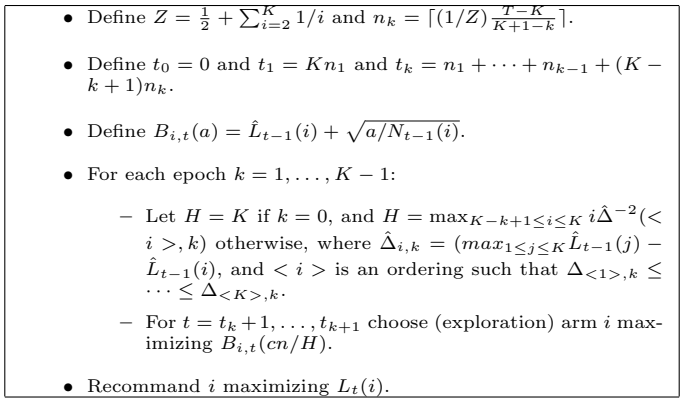

Figure 4: The Adaptive-UCB-E algorithm from [7].

- **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[8]) 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 $P(\hat{\theta} = i) = \frac{N_T(i)}{T}$.
- For the two-player case, a variant of EXP3 benefit from sparsity through truncation (**TEXP3**, Truncated EXP3) has been proposed [9]. It is defined in Fig. 7.
- 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.

3 Experimental results

We experiment algorithms above in the one-player case (with kill-all go, in which the strategic choice is the initial placement of stones for the black player) and in the two-player case in sections below.

3.1 One-player case: killall Go

We refer to classical sources for the rules of Go; KillAll Go is the special case in which black is given an advantage (some initial stones), but has a more difficult goal: he must kill all opponent stones on the board. So, one player only has initial stones to set up; the game is then played as a standard Go game. We refer to two different killall-Go frameworks: 7x7, 2 initial stones for black (Section 3.1.1); 13x13, 8 or 9 initial stones for black (Section 3.1.2). The human opponent is Ping-Chiang Chou (5p professional player).

3.1.1 7x7 killall Go

Here, the black player must choose the positioning of two initial stones. Human experts selected 4 possibilities: (1) a black stone in C5, and next black stone chosen by the tactical system as a first move; (2) a black stone in C4, and next black stone chosen by the tactical system as a first move; (3) a black stone in D4 (center), and next black stone chosen by the tactical system as a first move; (4) two black stones in C4 and E4. We tested intensively each of these strategic choices by our tactical system (ColdMilk program, by Dong Hwa university), in order to get a reliable estimate of the winning rate in each case (Table 2). Then, we simulated (using these estimates as ground truth) what would happen if we used various strategic tools for choosing the initial placement, for various limited budgets ($T = 16,\ 64,\ 256$). Results, for kill-all Go as explained above and also for various artificial settings with the same number of arms, are presented in Tables 5 and 5. Arms are randomly rotated for getting rid of trivial bias.

Table 5 contains both real-world and artificial results; artificial results are as follows: (1) random uniform probabilities of winning for each arm; (2) all arms have probability 0 of winning except one arm which has probability 0.1 of winning (3) all arms have probability 1 of winning except one arm which has probability 0.9 (4) all arms have probability 0.5 except one which has probability 0.4 (5) all arms have probability 0.5 except one which has probability 0.6. Please note that in the artificial cases, the index of the special arm (the arm with different reward) is randomly drawn and is indeed not necessarily the first.

Placement of black stones	Score for black
C5+choice by program	27.9% ±2.3%
C4+choice by program	33.4% ±2.4%
D4+choice by program	36.2% ±3.0%
C4+E4	44.8% ±2.7%

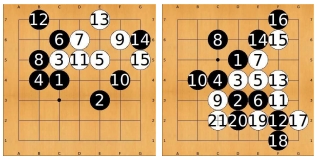

Table 2: Left: Efficiency of each strategic choice for black in killall Go. These numbers will be used as ground truth for experiments below (Table 5). Center and Right: Games played by our program MoGoTW as White (center) and as Black (right) in 7x7 killall Go. The center game is a win for the program whereas the right game is a loss.

Two 7x7 killall-go games were then played against Ping-Chiang Chou (5P), with one win of the computer as White and one loss of the computer as Black (i.e. White won both). Results are presented in Fig. 2 (right). The pro player did not make the same strategic choice as our program (he chose C4 E3 instead of our choice C4 E4) but agreed, after discussion, that C4 E4 is better.

3.1.2 13x13 killall Go

We reproduced the experiments with 13x13 initial placement of stones. Fig. 5 presents the five different handicap placements considered in the experiment. As for 7x7, heavy computations allowed us to find an approximate ground truth, and then experiments are run on this ground truth. Experimental results for various bandit approaches on this 13x13 killall Go metagaming are given in Table 3. We also test on artificial problems.

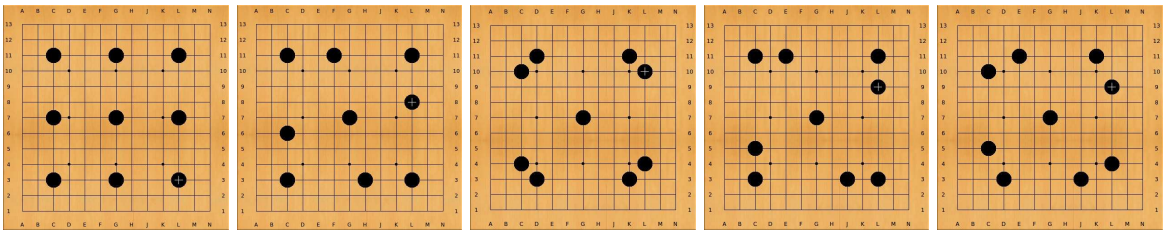

Figure 5: Five different handicap placements proposed by Go experts for 13x13 killall Go with 9 initial stones.

We then show in Fig. 6 the games played against Ping-Chiang Chou by our program as White with 8 and 9 initial stones respectively; we see on these games the strategic choice made by Ping-Chiang Chou (5P),

Exploration / Recommendation Algorithm	Average simple regret (16,64,256 time steps)
Strategic choice in 13x13 killall Go (5 possible choices)	
uniform sampling, EBA	**0.0201** 0.0204 0.0139
UCB+MPA	0.0215 **0.0192** 0.0147
UCB+UCB	0.0336 0.0274 0.0213
UCB+LCB	0.0224 0.0202 **0.0137**
Bernstein+LCB	0.0206 0.0206 0.0146
UCB+EBA	0.0221 0.0206 **0.0137**
EXP3+EDP	0.0369 0.0359 0.0357
SR	0.0239 0.0225 0.0119
adaptUCBE, EBA	0.0235 0.0199 0.0138

Table 3: Experimental results of average simple regret when comparing five different stone placements for 9 stones in 13x13 as shown in Fig. 5. All experiments are reproduced 1000 times.

which is the same as the strategic choice by our program, i.e. the first choice in Fig. 5.

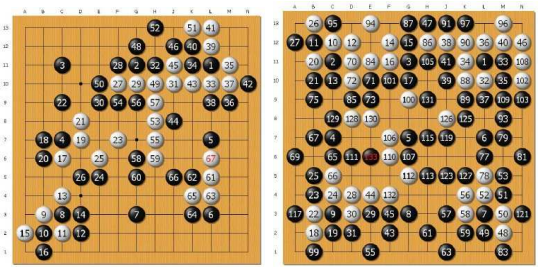

Figure 6: These two games are the killall-go games played by our program as White against Ping-Chiang Chou (5P). The program won with 8 initial black stones and lost with 9 initial stones. As Black, the program lost both H8 and H9.

3.2 Two-player case: Sparse Adversarial Bandits for Urban Rivals

Recently [9] 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 guarantied 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 moves K. TEXP3 is an attempt to exploit these two characteristics. It uses the outcome of EXP3 and truncates the arms that are unlikely to be part of the solution. Alg. 7 describes the implementation. The constant c is chosen as $\frac{1}{T}\max_i(Tx_i)^\alpha$ for some $\alpha \in]0,1[$ (and d accordingly), as in [9], while T is the number of iterations executed. We set $\alpha = 0.7$ in our experiments, following [9, 10]. The natural framework of EXP3 is a two-player game. 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 follow: (1) player 1 choose a combination $\theta_1 \in \{1,\dots,K_1\}$; (2) simultaneously, player 2 choose a combination $\theta' \in \{1,\dots,K'\}$; (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

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

$$\begin{aligned} x'_i &= x_i \text{ if } x_i > c, x'_i = 0 \text{ otherwise;} \\ y'_i &= y_i \text{ if } y_i > d, y'_i = 0 \text{ otherwise.} \end{aligned}$$

Renormalize: $x'' = x'/\sum_i x'_i$; $y'' = y'/\sum_i y'_i$.
Output x'', y''.

Figure 7: TEXP3 (truncated EXP3), offline truncation post-EXP3.

remain with 715 different possible combinations (both $K_1 = K_2 = 715$) if we allow 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 4 (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

Table 4: EXP3 vs Random (left) and TEXP3 vs Random (right).

T	Score $\pm 1\sigma$	T	Score $\pm 1\sigma$
10 000	0.5042 ± 0.001	10 000	0.7206 ± 0.005
25 000	0.5278 ± 0.001	25 000	0.7238 ± 0.003
50 000	0.5421 ± 0.002	50 000	0.7477 ± 0.002
100 000	0.5749 ± 0.004	100 000	0.7871 ± 0.006

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 [9], theoretically in [10]). Table 4 (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.

4 Conclusions for small time budgets: adversarial and stochastic frameworks, cumulative regret or simple regret, and the importance of sparsity

We compared various algorithms for strategic choices including widely played games (Killall Go, a classical exercise of Go schools, and Urban Rivals); we defined strategic choices in terms of moderate exploration budget for a simple regret criterion. We distinguished the one-player case and the two-player case; this distinction, in bandit terms, is a distinction between stochastic and adversarial bandits.

As clearly shown by the good performance of UCB/LCB variants, SR, and EXP3 on their original frameworks (one-player and two-player cases respectively), and by the poor performance of EXP3 in the one-player case, this distinction is relevant. Consistently with theory, bandits designed for the stochastic case (typically UCB) performed well in the one-player case and bandits designed for the adversarial case (typically EXP3) performed well in the two-player case. The distinction between simple regret and cumulative regret is less striking; yet, successive rejects, which was designed for simple regret algorithms, performed very well in particular for very small budgets.

We also show the relevance of a careful recommendation algorithm; UCB is a good exploration algorithm, but it should be accompanied by a good recommendation strategy like LCB or MPA as soon as the number of options is not negligible compared to the number of time steps; otherwise weak poorly explored arms can be recommended. This is however less critical than in Monte-Carlo Tree Search, where bandits are applied many times per run (once per move in a control problem or in a game).

The results in the two-player case also suggest that sparsity should be used whenever possible in the adversarial case; the superiority of TEXP3 over EXP3 in this context is the most clearest contrast in this work. Whereas simple regret and cumulative regret make little difference, even in the context of small time budget, sparse or not sparse makes a big difference, as much as distinguishing one-player case and two-player case. A main further work is the analysis of cases where structure on options or a priori ranking of options

is available. We conclude below with more details for the one-player and two-player case respectively.

4.1 One-player case

There are two crucial components under test: exploration algorithm, and recommendation algorithm. The most important component in strategic choices is the exploration formula. In many of our tests (with the notable exception of very small budget, very relevant here for our setting), **the best algorithm for *exploration* is UCB**, which is designed for the one-player case with cumulative regret; the surprising thing is that we here work on the simple regret, which is the natural notion of regret for the framework of strategic choices. Nonetheless, the variant of UCB termed Adapt-UCB-E, designed for parameter free simple regret, performs correctly. Consistently with artificial tests in [1], UCB is non-asymptotically much better than uniform exploration variants (which are nonetheless proved asymptotically optimal within logarithmic factors both for a fixed distribution and in a distribution free setting, in the "simple regret" setting). The asymptotic behavior is far from being a good approximation here. Importantly for our framework, **Successive Reject, designed for simple regret, is very stable (never very bad) and outperforms UCB variants for the smallest budgets.**

Consistently with some folklore results in Monte-Carlo Tree Search, the recommendation should not be made in a UCB manner; in fact, **the lower confidence bound performed very well; we also got good results with the *most played arm* or the *empirically best arm*, *as recommendation rules***. We point out that many practitioners in the Computer-Go literature (which is based on heavily tuned bandit algorithms) use combinations of EBA and MPA and LCB as recommendation arms for optimal performance. Consistently with intuition, EBA becomes weaker with larger numbers of arms. This is consistent with experiments in [2]. Bernstein races performed moderately well; there was no effort for tuning them and maybe they might be improved by some tuning. Adapt-UCB-E performed well as a variant of UCB dedicated to simple regret, but not better than SR or other UCB variants.

Results include games won against a professional player, in 7x7 Killall Go and in 13x13 Killall Go; in this case, the strategic decision is the initial choice.

4.2 Two-player case

In the two-player case, EXP3 (dedicated to this adversarial setting) naturally performed well.We made experiments confirming the good behavior of the algorithm, following [9, 11, 12]. As metagaming is a good candidate for providing sparse problems, we tested the efficiency of the truncation algorithm TEXP3 [9], with indeed much better results here than in the original paper (this is certainly due to the fact that, in our metagaming context, we have a much more sparse benchmark than [9]).

Results include experiments on a real game, namely Urban Rivals; the strategic choice consists in choosing the cards, which is directly a strategic choice setting.

References

[1] S. Bubeck, R. Munos, and G. Stoltz, "Pure exploration in multi-armed bandits problems," in *ALT*, 2009, pp. 23–37.

[2] A. Bourki, M. Coulm, P. Rolet, O. Teytaud, and P. Vayssière, "Parameter Tuning by Simple Regret Algorithms and Multiple Simultaneous Hypothesis Testing," in *ICINCO2010*, funchal madeira, Portugal, 2010, p. 10. [Online]. Available: http://hal.inria.fr/inria-00467796/en/

[3] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, pp. 4–22, 1985.

[4] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2/3, pp. 235–256, 2002.

[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*. IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 322–331.

[6] V. Mnih, C. Szepesvári, and J.-Y. Audibert, "Empirical Bernstein stopping," in *ICML '08: Proceedings of the 25th international conference on Machine learning*. New York, NY, USA: ACM, 2008, pp. 672–679.

[7] J.-Y. Audibert and S. Bubeck, "Best Arm Identification in Multi-Armed Bandits," in *COLT 2010 - Proceedings*, Haifa, Israël, 2010, p. 13 p. [Online]. Available: http://hal-enpc.archives-ouvertes.fr/hal-00654404

[8] L. Kocsis and C. Szepesvari, "Bandit based Monte-Carlo planning," in *15th European Conference on Machine Learning (ECML)*, 2006, pp. 282–293.

[9] S. Flory and O. Teytaud, "Upper confidence trees with short term partial information," in *Procedings of EvoGames 2011*. Springer, 2011, p. accepted.

[10] D. Auger, S. Ruette, and O. Teytaud, "Sparse bandit algorithms," *submitted*, 2012.

[11] B. Bouzy and M. Métivier, "Multi-agent learning experiments on repeated matrix games," in *ICML*, J. Fürnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 119–126.

[12] D. Auger, "Multiple tree for partially observable monte-carlo tree search," in *EvoApplications (1)*, ser. Lecture Notes in Computer Science, 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, Eds., vol. 6624. Springer, 2011, pp. 53–62.

Exploration / Recommendation Algorithm	Average simple regret (16,64,256 time steps)		
Strategic choice in 7x7 killall Go (with symetry-breaking; 4 possible choices)			
unif. sampling, EBA	0.092	0.0603	0.0244
UCB+MPA	0.079	0.0563	0.022
UCB+UCB	0.0673	0.0523	0.0304
UCB+LCB	0.0751	0.0466	0.0222
Bernstein+LCB	0.0633	0.0537	0.0226
UCB+EBA	0.0744	0.0474	**0.0185**
EXP3+EDP	0.0849	0.0809	0.0748
SR	**0.0627**	**0.0448**	0.021
adaptUCBE, EBA	0.0707	0.0483	0.0188
4 artificial options, with reward unif. in $[0,1]$			
unif. sampling, EBA	0.0652	0.0197	0.00394
UCB+MPA	0.0535	0.0198	0.00453
UCB+UCB	0.064	0.0386	0.00931
UCB+LCB	0.0495	0.0142	0.00626
Bernstein+LCB	0.0563	0.0191	0.00465
UCB+EBA	**0.0454**	0.0175	**0.00401**
EXP3+EDP	0.184	0.145	0.11
SR	0.0611	0.0205	0.00681
adaptUCBE, EBA	0.0505	**0.014**	0.00478
4 artificial options, $(0.1,0,0,0)$			
unif. sampling, EBA	0.0509	**0.0129**	0.0002
UCB+MPA	0.0519	0.0148	0.0001
UCB+UCB	0.0695	0.0277	0.0049
UCB+LCB	0.0503	0.0155	**0**
Bernstein+LCB	**0.0483**	0.0136	0.0004
UCB+EBA	0.0501	0.014	0.0001
EXP3+EDP	0.0706	0.062	0.0524
SR	0.0532	0.0409	0.012
adaptUCBE, EBA	0.0528	0.014	0.0001
Exploration / Recommendation Algorithm	Average simple regret (16,64,256 time steps)		
4 artificial options, $(0.9,1,1,1)$			
unif. sampling, EBA	0.0151	0.0045	0
UCB+MPA	0.0189	0.0061	0.0001
UCB+UCB	0.0179	0.0045	0.0246
UCB+LCB	0.0167	0.006	0.0001
Bernstein+LCB	0.0168	0.0048	0
UCB+EBA	0.0165	0.0048	0.0001
EXP3+EDP	0.0209	0.0211	0.0214
SR	0.0118	0.0033	0
adaptUCBE, EBA	0.0152	0.0057	0
4 artificial options, $(0.4,0.5,0.5,0.5)$			
unif. sampling, EBA	0.0176	0.0088	0.0019
UCB+MPA	**0.0128**	0.0095	0.0027
UCB+UCB	0.0157	0.0114	0.0065
UCB+LCB	0.0142	**0.0078**	**0.0012**
Bernstein+LCB	0.0167	0.0084	0.0028
UCB+EBA	0.016	0.0094	0.002
EXP3+EDP	0.0206	0.0189	0.0174
SR	0.0175	0.0105	0.0025
adaptUCBE, EBA	0.0153	0.0081	0.0018
4 artificial options, $(0.6,0.5,0.5,0.5)$			
unif. sampling, EBA	0.0637	0.0527	0.0277
UCB+MPA	0.0636	0.053	0.0246
UCB+UCB	0.0675	0.0561	0.0346
UCB+LCB	0.0621	**0.0494**	**0.0244**
Bernstein+LCB	0.0643	0.0498	0.0284
UCB+EBA	**0.061**	0.05	0.0257
EXP3+EDP	0.0715	0.0709	0.0665
SR	0.0631	0.0531	0.03
adaptUCBE, EBA	0.0642	0.0509	0.0247

Exploration / Recommendation Algorithm	Average simple regret (16,64,256 time steps)		
Strategic choice in 7x7 killall Go (without symetry-breaking; 11 possible choices)			
unif. sampling, EBA	0.121	0.0973	0.0488
UCB+MPA	0.127	0.0677	0.0235
UCB+UCB	**0.0835**	0.0826	0.0543
UCB+LCB	0.0976	0.0656	**0.0213**
Bernstein+LCB	0.116	0.076	0.0488
UCB+EBA	0.104	0.0657	0.0222
EXP3+EDP	0.1	$\widehat{0.1}$	0.094
SR	0.0987	**0.0557**	0.0232
adaptUCBE, EBA	0.103	0.067	0.023
11 artificial options, with reward unif. in $[0,1]$			
unif. sampling, EBA	0.172	0.0614	0.017
UCB+MPA	0.219	**0.0263**	0.00829
UCB+UCB	0.202	0.0837	0.0366
UCB+LCB	0.165	0.0286	0.00758
Bernstein+LCB	0.185	0.0513	0.0111
UCB+EBA	0.168	0.0273	**0.00708**
EXP3+EDP	0.289	0.238	0.223
SR	**0.123**	0.0336	0.0118
adaptUCBE, EBA	0.154	0.0267	0.0083
11 artificial options, $(0.1,0,\ldots,0)$			
unif. sampling, EBA	0.0787	0.0474	**0.0073**
UCB+MPA	0.0787	0.0509	0.0089
UCB+UCB	0.089	0.0773	0.038
UCB+LCB	0.0776	0.048	0.0074
Bernstein+LCB	**0.0764**	0.0493	0.009
UCB+EBA	0.0788	0.0498	0.0094
EXP3+EDP	0.0862	0.0814	0.0765
SR	0.0788	0.0619	0.0319
adaptUCBE, EBA	**0.0764**	**0.0465**	0.0079
Exploration / Recommendation algorithm	Average simple regret (16,64,256 time steps)		
11 artificial options, $(0.9,1,\ldots,1)$			
unif. sampling, EBA	0.0069	0.0045	0.0007
UCB+MPA	0.0072	0.005	0.0005
UCB+UCB	0.0082	0.0051	0.0005
UCB+LCB	0.0065	0.0041	0.0006
Bernstein+LCB	0.0074	0.0048	0.0003
UCB+EBA	0.0072	0.005	0.0009
EXP3+EDP	0.0076	0.0086	0.0063
SR	**0.0052**	**0.0011**	**0**
adaptUCBE, EBA	0.0072	0.0041	0.0003
11 artificial options, $(0.4,0.5,\ldots,0.5)$			
unif. sampling, EBA	0.0055	0.0042	0.0011
UCB+MPA	0.0071	0.0032	**0.0008**
UCB+UCB	0.0067	0.0037	0.0032
UCB+LCB	0.0055	**0.0017**	0.0004
Bernstein+LCB	**0.0045**	0.0039	0.0018
UCB+EBA	0.0075	0.003	0.0003
EXP3+EDP	0.0074	0.0071	0.0066
SR	0.0062	0.0023	0.001
adaptUCBE, EBA	0.0049	0.0025	0.0009
11 artificial options, $(0.6,0.5,\ldots,0.5)$			
unif. sampling, EBA	0.0892	0.0824	0.0686
UCB+MPA	0.0888	**0.0764**	0.0563
UCB+UCB	0.087	0.0843	0.0726
UCB+LCB	0.0875	0.0766	**0.0556**
Bernstein+LCB	0.0869	0.0812	0.0691
UCB+EBA	**0.0862**	0.0783	0.0567
EXP3+EDP	0.0887	0.0869	0.0895
SR	0.0868	0.0817	0.0622
adaptUCBE, EBA	0.0868	0.0776	0.0569

Table 5: Left: Average simple regret for various exploration/recommendation methodologies. Performance of various strategic systems for choosing initial placement for Black in 7x7 killall-Go. The first row is the real-world case, with 4 arms; then, we consider various cases with the same number of arms (see text). Each experiment is reproduced 1000 times and standard deviations are less than 0.04. Right: similar to Left, but we do not remove the symetries; this increases the number of possible choices to 11. This is obviously not what we should do from the point of view of the application; we just do this in order to generate a new test case. The same artificial cases as in the 4-options case are reproduced with 11 options. All experiments are reproduced 1000 times and standard deviations are less than 0.004.