Reducing Exploration of Dying Arms in Mortal Bandits

Stefano Tracà
Massachusetts Institute of Technology
stet@alum.mit.edu

Cynthia Rudin
Duke University
cynthia@cs.duke.edu

Weiyu Yan
Duke University
weiyu.yan@duke.edu

Abstract

Mortal bandits have proven to be extremely useful for providing news article recommendations, running automated online advertising campaigns, and for other applications where the set of available options changes over time. Previous work on this problem showed how to regulate exploration of new arms when they have recently appeared, but they do not adapt when the arms are about to disappear. Since in most applications we can determine either exactly or approximately when arms will disappear, we can leverage this information to improve performance: we should not be exploring arms that are about to disappear. We provide adaptations of algorithms, regret bounds, and experiments for this study, showing a clear benefit from regulating greed (exploration/exploitation) for arms that will soon disappear. We illustrate numerical performance on the Yahoo! Front Page Today Module User Click Log Dataset.

Keywords: Multi-armed bandit, exploration-exploitation trade-off, retail management, recommender systems, regret bounds.

1 INTRODUCTION

In many applications of multi-armed bandits, the bandits are *mortal*, meaning that they do not exist for the full period over which the algorithm is running. In advertising, ads and coupons can come and go; in news article recommendation, the news is perpetually changing; in website optimization, the content changes to keep viewers interested. Chakrabarti et al. [2009] introduced and formalized the notion of mortal bandits, and there has been a body of work following this. This work has proved to be valuable in the setting of advertising (see Agarwal et al. [2009] and Féraud and Urvoy [2012]) and in other areas such as communications underlaying cellular networks (see Maghsudi and Stańczak [2015]). Bnaya et al. [2013] propose an adaptation to the mortal settings of the popular UCB algorithm introduced by Auer et al. [2002]. While these algorithms are designed to adapt exploration based on when arms *appear*, they do not adapt when arms *disappear* (for example, in the work of Bnaya et al. [2013], new arms are immediately played, even for arms that may soon die, which could be a poor strategy). In strategic implementations of mortal bandits, *we should not be exploring arms that are soon going to disappear*.

In the applications discussed above (advertising, news article recommendation, website optimization) and others, we often know in advance when arms will appear or disappear. For coupons and discount sales, we launch them for known periods of time (e.g., a one day sale), whereas for news articles, we could choose to place them in a pool of possible featured articles for mobile devices for one day or one week. If the lifespans of the arms are not known, they can often be estimated. For instance, we can observe the distribution of the lifespans of the arms to determine when an arm is old relative to other arms. Alternatively, external features can be used to estimate the remaining lifespan of an arm.

This work provides algorithms for the mortal bandit setting that reduce exploration for dying arms. In Section 2 we introduce two algorithms: the AG-L algorithm (adaptive greedy with life regulation) and the UCB-L algorithm (UCB mortal with life regulation). We present finite time regret bounds (proofs are in the Supplement[1]) and intuition on the meaning of the bounds. In Section 3 we discuss numerical performance on the publicly available Yahoo! Front Page Today Module User Click Log

[1]The Supplement is available in the GitHub repository: https://github.com/5tefan0/Supplement-to-Reducing-Exploration-of-Dying-Arms-in-Mortal-Bandits

Dataset. The experiments show a clear benefit in final rewards when the algorithms reduce exploration of arms that are about to expire. This confirms the intuition that it is useless to gain information about arms if they are going to disappear soon anyway.

2 ALGORITHMS FOR REGULATING EXPLORATION OVER ARM LIFE

Formally, the mortal stochastic multi-armed bandit problem is a game played in n rounds. At each round t the algorithm chooses an action I_t among a finite set M_t of possible choices called *arms* (for example, they could be ads shown on a website, recommended videos and articles, or prices). When arm $j \in M_t$ is played, a random reward $X_j(t)$ is drawn from an unknown distribution. The distribution of $X_j(t)$ does not change with time (the index t is used to indicate in which turn the reward was drawn) and it is bounded in $[a, b]$ (and we denote with r the range $r = b - a$), while the set M_t can change: arms may become unavailable (they "die") or new arms may arrive (they "are born"). At each turn, the player suffers a possible regret from not having played the best arm: the mean regret for having played arm j at turn t is given by $\Delta_{j,i_t^*} = \mu_{i_t^*} - \mu_j$, where $\mu_{i_t^*}$ is the mean reward of the best arm available at turn t (indicated by i_t^*) and μ_j is the mean reward obtained when playing arm j. Let us call $I(j)$ the set of turns during which the algorithm chose arm j. At the end of each turn the algorithm updates the estimate of the mean reward of arm j:

$$\hat{X}_j = \frac{1}{T_j(t-1)} \sum_{s \in I(j)}^{T_j(t-1)} X_j(s), \tag{1}$$

where $T_j(t-1)$ is the number of times arm j has been played before round t starts.

Let us define M_t as the set of all available arms at turn t (M_1 is the starting set of arms). $M_I = \{1, 2, \ldots, m_I\} \subset M_1, M_I \neq \emptyset$ is the set of arms that are initialized over the first m_I iterations (i.e., the algorithm plays one time all of them following the order of their index). The quantity that a policy tries to minimize is the cumulative regret R_n that is given by

$$R_n = \sum_{j \in M_I} \Delta_{j,i_j^*} + \sum_{t=m_I+1}^{n} \sum_{j \in M_t} \Delta_{j,i_t^*} \mathbb{1}_{\{t \in I(j)\}}, \tag{2}$$

where $\mathbb{1}_{\{t \in I(j)\}}$ is an indicator function equal to 1 if arm j is played at time t (otherwise its value is 0). The first summation in (2) is the regret that the algorithm suffers during the initialization phase when each arm in M_I is pulled once yielding a regret of Δ_{j,i_j^*} (the arms in M_I are played in order of their index and i_j^* denotes the best arm available at that turn). For the rest of the game ($t \in \{m_I + 1, \cdots, n\}$), the algorithm incurs Δ_{j,i_t^*} regret at time t only when arm j is available ($j \in M_t$) and it is pulled ($t \in I(j)$). Let us call $M = \bigcup_{t=1}^{n} M_t$ the set of all arms that appear during the game and $L_j = \{s_j, s_j + 1, \cdots, l_j\}$ the set of turns that arm j is available. Then, we can also write (2) as

$$R_n = \sum_{j \in M_I} \Delta_{j,i_j^*} + \sum_{j \in M} \sum_{\substack{t \in L_j \\ t > m_I}} \Delta_{j,i_t^*} \mathbb{1}_{\{t \in I(j)\}}. \tag{3}$$

Depending on the algorithm used, one formulation may be more convenient than the other when computing a bound on the expected cumulative regret $\mathbb{E}[R_n]$. A complete list of the symbols used throughout the paper can be found in Supplement E.

2.1 THE ADAPTIVE GREEDY WITH LIFE REGULATION (AG-L) ALGORITHM

In Algorithm 1 we extend the adaptive greedy algorithm (which we abbreviate with AG) presented in Chakrabarti et al. [2009]. We call this new algorithm the *adaptive greedy with life regulation algorithm*, which we abbreviate with AG-L. AG-L handles rewards bounded in $[a, b]$, and regulates exploration based on the remaining life of the arms (that is, the algorithm avoids exploring arms that are going to disappear soon). During the initialization phase, the algorithm plays each arm in the initialization pool M_I once. After that, to determine whether to explore arms, AG-L draws from a Bernoulli random variable with parameter

$$p = 1 - \frac{\max_{j \in M_t} \hat{X}_j - a}{b - a},$$

which intuitively means that if the algorithm has a good available arm (i.e., an arm that has a high mean estimate) the probability of exploration is very low and the algorithm will exploit by playing the best available arm so far (ignoring also arms that were excluded in the initialization phase or have been born and never played). If the value of the Bernoulli random variable is 1, then AG-L proceeds by playing an arm at random among those arms whose remaining life is long enough: we call this set $M_t(\mathcal{L})$. One way to set $M_t(\mathcal{L})$ is to pick all arms in M_t such that their remaining lifespan is in the top 30% of the distribution of all remaining lifespans (we chose 30% because we tuned this parameter by trying different values on a small subset of data). $M_t(\mathcal{L})$ can also contain arms that have never been played before or that were excluded in the initialization phase. As mentioned earlier, if the value of the Bernoulli random variable is 0, then AG-L exploits the arm that has the highest average reward.

Algorithm 1: AG-L algorithm

Input : number of rounds n, initialization set of arms M_I, set M_t of available arms at time, rewards range $[a, b]$

Initialization: play all arms in M_I once, and initialize $\hat{X}_j$ for each $j = 1, \cdots, m_I$

for $t = m_I + 1$ **to** n **do**

Draw a Bernoulli B r.v. with parameter

$$p = 1 - \frac{\max_{j \in M_t} \hat{X}_j - a}{b - a};$$

if $B = 1$ **then**

Play an arm at random from $M_t(\mathcal{L})$;

else

Play an arm j with highest $\hat{X}_j$;

end

end

Get reward $X_j(t)$;

Update $\hat{X}_j$;

end

Note that this algorithm is not relevant to sleeping bandits (see Kleinberg et al. [2010] and Kanade et al. [2009]) because those arms do not die, they simply sleep. For sleeping bandits, we would want to explore them until they fall asleep because the estimate of the arm's mean reward would still be useful when the arm wakes up again.

In order to derive a finite time regret bound we introduce $\mathcal{H}_{t-1}$ as the set of all possible histories (after deterministic initialization) of the game up to turn $t-1$:

$$\mathcal{H}_{t-1} = \left\{ h = \begin{bmatrix} b_{m_I+1} & b_{m_I+2} & \dots & b_{t-1} \\ i_{m_I+1} & i_{m_I+2} & \dots & i_{t-1} \end{bmatrix} \text{ such that } b_s \in \{0,1\},\ i_s \in M_s,\quad \forall s \in \{m_I+1, \dots, t-1\} \right\}.$$

Each element h of $\mathcal{H}_{t-1}$ is a possible history of pulls before turn t and tells exactly what arm was pulled and if it was an exploration turn or an exploitation turn. If $b_s = 1$ we say that the algorithm explored at time s, if $b_s = 0$ we say that the algorithm exploited at time s, while i_s is the index of the arm that was played at time s. Let us define the linear transformation $g(p) = b + (a-b)p$ (used to standardize rewards to the interval $[0, 1]$) and use a result from Vaughan and Venables [1972] for the PDF (or PMF) $f_{M(h,s)}(g(p))$ of the maximum of the estimated mean rewards at time s given that each arm has been pulled according to history h up to time $s-1$:

$$f_{M(h,k)}(x) = \frac{1}{(m_t - 1)!} \operatorname{perm} \left(\begin{bmatrix} F_1(x) & \dots & F_{m_k}(x) \\ \vdots & \ddots & \vdots \\ F_1(x) & \dots & F_{m_k}(x) \\ f_1(x) & \dots & f_{m_k}(x) \end{bmatrix} \right)$$

where the matrix has a total of m_k rows (and columns), $f_1(x), \cdots, f_{m_k}(x)$ and $F_1(x), \cdots, F_{m_k}(x)$ are the PDFs (or PMFs) of the distributions of the average rewards (which we can compute knowing the distribution from which rewards are drawn). For each h, we indicate how many times arm j has been pulled up to time k with

$$t_j(h,k) = \mathbb{1}_{\{j \in M_I\}} + \sum_{s'=m_I+1}^{k} \mathbb{1}_{\{i_{s'} \in I(j)\}}.$$

Similarly to when we defined the regret, let us call $\Delta(i, i_s) = \mu_i - \mu_{i_s}$. Then, consider the following quantities (see Supplement A for how to compute them given the mean rewards):

- $u_s(h, i_s)$ **is an upper bound on the probability that arm i_s is considered to be the best arm at time s given the history of pulls** (according to h) up to time $s-1$:

$$u_s(h, i_s) = \prod_{i:\mu_i > \mu_{i_s}} \left(\exp\left\{ -\frac{t_{i_s}(h,s)\Delta(i,i_s)^2}{2r} \right\} + \exp\left\{ -\frac{t_i(h,s)\Delta(i,i_s)^2}{2r} \right\} \right),$$

  where range of rewards r is defined as $r = b - a$.

- $U_k(h, i_k)$ **is an upper bound on the probability that arm i_k would be pulled at time k given the history of pulls** (according to h) up to time $k-1$: When $k < t$, then $U_k(h, i_k) =$

$$\int_0^1 \left(\tfrac{p}{m_k} \mathbb{1}_{\{b_k=1\}} + (1-p) u_k(h, i_k) \mathbb{1}_{\{b_k=0\}} \right) \times f_{M(h,k)}(g(p))\, \mathrm{d}p, \tag{4}$$

  and when $k = t$, then $U_t(h, i_t) =$

$$\int_0^1 \left(\frac{p}{m_t} + (1-p) u_t(h, i_t) \right) f_{M(h,t)}(g(p))\, \mathrm{d}p. \tag{5}$$

- $U_t(h, j)$ **is an upper bound on the probability that arm j would be pulled at time t given the history of pulls** (according to h) up to time $t-1$:

$$U_t(h, j) = \int_0^1 \left(\frac{p}{m_t} + (1-p) u_t(h, j) \right) f_{M(h,t)}(g(p))\, \mathrm{d}p. \tag{6}$$

In standard regret bounds, the bound is usually in terms of the mean rewards μ_j and Δ_j for each arm j, which are not known in the application. Our bounds analogously depend on the μ_j's and $\Delta(i, i_s)$'s (where i_s is the arm played at time s, and i is another arm with higher mean reward). While standard bounds usually

have a simple dependence on μ_j's, our bounds have a more complicated dependence on the μ_j's. On the other hand, they depend on the same quantities as the standard bounds; once we have the μ_j terms, the bound can be computed using the same information that is available in the standard bounds. For instance $u_s(h, i_s)$, $U_s(h, i_s)$, and $U_t(h, j)$ do not require any additional information other than the μ_j's.

Theorem 2.1 presents a finite time upper bound on the regret for the AG-L algorithm (Supplement A has the proof).

Theorem 2.1. *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \sum_{j \in M_I} \Delta_{j,i_j^*} + \sum_{t=m_I+1}^{n} \sum_{j \in M_t(\mathcal{L})} \Delta_{j,i_t^*} \times \quad (7)$$

$$\sum_{h \in \mathcal{H}_{t-1}} \left(U_t(h, j) \prod_{s=m_I+1}^{t-1} U_s(h, i_s) \right). \quad (8)$$

The standard case, when there is no exploration regulation based on remaining arms life, can be recovered by setting $M_t(\mathcal{L}) = M_t$. (This is the case where we are not excluding arms that are about to disappear). In that standard case, Theorem 2.1 is a novel finite time regret bound for the standard AG algorithm introduced by Chakrabarti et al. [2009].

The first summation in (7) represents the total mean regret suffered during the initialization phase. Intuitively, it is the summation of the mean regrets Δ_{j,i_t^*} for having pulled an arm j that is in the initialization set $M_I = \{1, 2, \ldots, m_I\}$. The second triple summation in (7) and (8) represents the total mean regret suffered after the initialization phase. Intuitively, it is the summation of all the mean regrets Δ_{j,i_t^*} for having pulled an arm j weighted by the bound on the probability of pulling arm j (the term that appears in (8)). The bound on the probability of pulling arm j is computed by considering all possible histories of pulls up to turn $t-1$ (hence the summation over $\mathcal{H}_{t-1}$). For each history h in the sum, the bound of choosing arm j at time t is given by multiplying the bound $U_t(h, j)$ on the probability of pulling arm j at turn t given h with the bound on the probability of that particular history h (given by the product of $U_s(h, i_s)$ up to turn $t-1$).

To intuitively see why this regret bound is better than the one that arises from the standard AG policy, we look at the quantities in Equation (4). The integrand has two main terms that are mutually exclusive (i.e., one appears during exploration turns and the other during exploitation turns):

- $1/m_s$ (recall that m_s is the number of arms available at turn s): this is a constant appearing during exploration phases (when $b_s = 1$).
- $u_s(h, i_s)$: this is a product of negative exponentials that decreases quickly, becoming smaller than $1/m_s$ after enough pulls on arm i_s. It appears during exploitation turns (when $b_s = 0$).

The two terms are mutually exclusive, and the AG algorithm that explores more often will have the term $1/m_s$ appear more often in the integrand of Equation (4). A larger integrand will yield a larger regret bound.

Conversely, the AG-L algorithm considers only the set $M_t(\mathcal{L})$ of arms with long life, and the term $1/m_s$ will appear less often than the smaller quantity $u_s(h, i_s)$, yielding a smaller regret bound.

Algorithms with smaller regret bounds generally lead to smaller regrets in practice. We will show how this is realized in the experiments later.

We can see the bound's intuition by restating Theorem 2.1 with dependence on the $1/m_s$ and $u_s(h, i_s)$ terms notated explicitly:

Theorem 2.2. *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \mathcal{O}(1) + \sum_{t=m_I+1}^{n} \sum_{j \in M_t(\mathcal{L})} \Delta_{j,i_t^*} \times$$

$$\sum_{h \in \mathcal{H}_{t-1}} F\left(\frac{1}{m_1}, \cdots, \frac{1}{m_t}, u_1(h, i_1), \cdots, u_t(h, i_t)\right),$$

where

$$F\left(\frac{1}{m_1}, \cdots, \frac{1}{m_t}, u_1(h, i_1), \cdots, u_t(h, i_t)\right)$$
$$= U_t(h, j) \prod_{s=m_I+1}^{t-1} U_s(h, i_s)$$

is an increasing function of all its arguments.

Intuitively, $u_1(h, i_1), \cdots, u_t(h, i_t)$ are smaller than $\frac{1}{m_1}, \cdots, \frac{1}{m_t}$ since they decrease at a fast rate (they are products of negative exponentials). By regulating exploration on arms that live longer, the bound of Algorithm 1 presents the smaller terms more times than the larger ones, yielding an overall better expected regret. Reducing exploration on dying arms tends not to impact the other reward terms unless the dying arms have signif-

icantly better rewards than the long-lived arms, which generally is not the case in real applications.

A thought experiment with good and bad arms. Let us conduct a thought experiment to provide intuition for why it is beneficial to limit exploration only among arms with short remaining life. Consider two different standard games, where arms are always available: the first with 100 arms, and the second with 10 arms. The quality of the arms come from the same distribution, for example we know that 30% of the arms have high expected rewards, and 70% have instead low expected rewards. The probability of picking a bad arm at random is the same in both games. However, one of these games is much more difficult than the other one in practice: in the 10-arm game, we can allocate more pulls to each arm, and thus it is much easier to determine when an arm is bad based on its mean reward estimate. For the 10-arm game, the algorithm will explore less (the term $1/m_s$ will appear less often) than in the 100-arm game, and thus the 10-arm game will have better bounds on the probability of playing suboptimal arms (the terms $u_s(h, i_s)$ decrease more quickly). Thus, it is easier to play the standard game with fewer arms.

When there is a mixture of long-lived and short-lived arms, AG-L may (in essence) reduce the full game to a smaller, easier one that considers only the long-lived arms.

In real applications, at each time, we expect there to be a mixture of arms with short remaining life and long remaining life. Intuitively, AG-L would reduce the game to an easier game by playing (among approximately good arms) mainly the long-lived arms.

Algorithm 2: UCB-L algorithm

Input : number of rounds n, initialization set of arms M_I, set M_t of available arms at time, rewards range $[a, b]$

Initialization: play all arms in M_I once, and initialize $\hat{X}_j$ for each $j = 1, \cdots, m_I$

for $t = m_I + 1$ **to** n **do**

  Play arm with highest $\hat{X}_j + \psi(j,t)\sqrt{\frac{2\log(t-s_j)}{T_j(t-1)}}$;

  Get reward $X_j(t)$;

  Update $\hat{X}_j$;

end

2.2 THE MORTAL UCB WITH LIFE REGULATION ALGORITHM (UCB-L)

Algorithm 2 extends the UCB algorithm of Auer et al. [2002] to handle life regulation. In the standard UCB algorithm, the arm with the highest upper confidence bound above the estimated mean is played. In this new version, the upper confidence bound has been modified so that it can be used in the mortal setting. It gradually shrinks the estimated UCB as the life of the arm comes to an end. Exploration is thus encouraged only on arms that have a long lifespan. In this way, arms that are close to expiring are played only if their estimated mean is high. Let s_j and l_j be the first and last turn at which arm j is available, and let $\psi(j,t)$ be a function proportional to the remaining life of arm j, which decreases over time. An example for $\psi(j,t)$ is $c\log(l_j - t + 1)$, where c is a positive constant (note that $\psi(j,t)$ approaches zero as the game gets closer to the expiration of arm j). New arms are initialized by using the average performance of past arms (i.e., if in the past, many bad arms appeared, new arms are considered more likely to be bad), and their upper confidence bound is built as if they have been played once. We abbreviate this algorithm by UCB-L.

Theorem 2.3 presents a finite time regret bound for the UCB-L algorithm (proof in Supplement B).

Theorem 2.3. *Let $\bigcup_{z=1}^{E_j} L_j^z$ be a partition of L_j into epochs with different best available arm, s_j^z and l_j^z be the first and last step of epoch L_j^z, and for each epoch let $u_{j,z}$ be defined as*

$$u_{j,z} = \max_{t \in \{s_j^z, \cdots, l_j^z\}} \left\lceil \frac{8\psi(j,t)\log(t - s_j)}{\Delta_{j,z}^2} \right\rceil,$$

where

$$\Delta_{j,z} = \Delta_{j,i_t^*} \text{ for } t \in L_j^z.$$

Then, the bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by

$$\begin{aligned}
&\mathbb{E}[R_n] \\
&\leq \sum_{j \in M_I} \Delta_{j,i_j^*} \\
&+ \sum_{j \in M} \sum_{z=1}^{E_j} \Delta_{j,z} \min\Big((l_j^z - s_j^z)\ ,\ u_{j,z} \\
&\quad + \sum_{\substack{t \in L_j^z \\ t > m_I}} (t - s_{i_t^*})(t - s_j - u_{j,z} + 1) \\
&\quad \times \left[(t - s_j)^{-\frac{4}{r^2}\psi(j,t)} + (t - s_{i_t^*})^{-\frac{4}{r^2}\psi(i_t^*,t)} \right] \Big).
\end{aligned}$$

The first summation $\sum_{j \in M_I} \Delta_{j,i_j^*}$ is the regret suffered during the initialization phase (the arms in M_I are played in order of their index and i_j^* denotes the best arm available at that turn). Intuitively, $u_{j,z}$ is the number of pulls required to be able to distinguish arm j from the best arm in epoch z. In the second double summation, the mean

Table 1: extracted dataframe from the original text record

timestamp	id	clicked	number of arms
1317513291	id-560620	0	26
1317513291	id-565648	0	26
1317513291	id-563115	0	26
1317513292	id-552077	0	26
1317513292	id-564335	0	26

regret for pulling arm j is multiplied by the minimum between the epoch length and the upper bound on the probability that the arms *appears* to be the best available one. This upper bound is a combination of the probability that we are either underestimating the best arm in epoch z or we are overestimating arm j (see Supplement B for more details). If the game is such that no new arms are born during the game and all arms expire after turn n, then this regret bound reduces to the standard UCB bound (see Auer et al. [2002]).

3 EXPERIMENTS ON Yahoo! NEWS ARTICLE RECOMMENDATION

We tested the performance the new AG-L and UCB-L algorithms versus the standard AG and UCB algorithms using the dataset from the Yahoo! Webscope program. The dataset consists of a stream of recommendation events that display articles randomly to users. At each time, the dataset contains information on the action taken (which is the article shown to the human viewing articles on Yahoo!), the outcome of that action (click or no click), the candidate arm pool at that time (the set of articles available) and the associated timestamp. We preprocessed the original text file into a structured data frame (see an extract of the data frame in Table 1).

In each game, the algorithms are tested on the same data. The recommender algorithms play for a fixed number of turns. We record the accumulated rewards of each algorithm. At each time, a reward can be calculated only when the article that was displayed to the human user matches the action of the algorithm. (We do not know the outcome of actions not recorded in the dataset.) This means rewards can only be calculated at a fraction of times that the algorithm is playing. Therefore, while still playing the same number of turns, some algorithms will have more evaluations than others. In particular, an algorithm can be unlucky, in that most of its actions are discarded by chance. However, when the dataset was constructed, articles were shown uniformly at random to the human user, and overall, the difference between the number of evaluations per algorithm is small. More details on the experiment can be found in Supplement D.

For ad serving or article serving in practice, the AG-L and UCB-L algorithms would be told when articles/advertisements/coupons are scheduled to appear and expire. Accordingly, we provided the algorithms with the beginning and end of life for each arm.

Separately, we consider the case where we do not have the life of each arm in advance. In that case, we add a step to the algorithms, which estimates the lifespan of new arms by the mean lifespan of expired arms.

In order to obtain a distribution for performance (rather than a single performance measurement), we ran the AG and AG-L algorithms many times to plot the distribution of rewards. The AG and AG-L algorithms are non-deterministic, since they choose arms randomly from the candidates with enough remaining life. On the other hand, UCB and UCB-L algorithms are deterministic because they always pick the arm with the best upper confidence bound. Running UCB and UCB-L many times on the same dataset will always give the same result. Therefore, to obtain a distribution for performance, we ran UCB and UCB-L for different sliding windows of time (i.e., we started the algorithms at many different points in time), which explains the multi-modal shape of UCB-L rewards distribution in Figure 2.

Figures 1 and 2 show the empirical distribution of rewards for the algorithms. Each algorithm played 100 games with 100000 turns per game. Each game consumed millions of data rows, because many actions could not be evaluated, as discussed above (they did not match the action shown to the Yahoo! user at that time).

The algorithms with life-regulation dramatically outperform the standard ones. Among AG-Ls, knowing the exact lifespan of each article (rather than using an estimated lifespan) improves performance. This result would have been obvious in retrospect: more information given to the algorithm allows it to make better decisions.

The AG-L strategy adopted here was part of a high-scoring entry of one of the Exploration-Exploitation competitions. The entry scored second place, with a score that was not statistically significantly different from the first place entry. In this competition, AG-L was one of two key strategies contributing to the high score. Both key strategies were based on incorporating time series information about article behavior, which added more strategic value than other types of information available during the competition.

4 CONCLUSIONS

In this work, we have shown that it is possible to leverage knowledge about the lifetimes of the arms to improve the

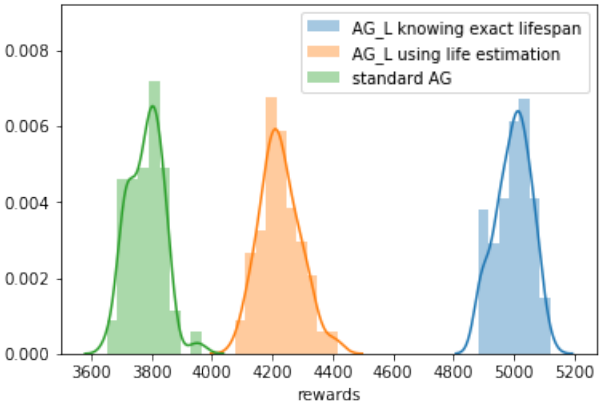

Figure 1: AG's playing the game 100 times. Randomness arises from the AG algorithms.

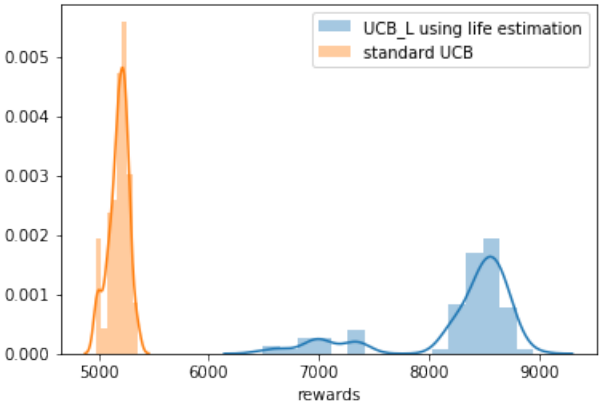

Figure 2: UCB's playing a set of 100 slightly different games. Randomness arises not from the algorithms but from random starting time.

quality of exploration and exploitation in mortal multi-armed bandits. Our algorithms focus on exploring the arms that will be available longer, leading to substantially increased rewards. In cases where we do not know the lifetimes of the arms but can estimate them, these techniques are still able to substantially increase rewards. We have presented novel finite time regret bounds and numerical experiments on the publicly available Yahoo! Webscope Program Dataset that show the benefit of reducing exploration on arms that are about to disappear soon.

References

Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. Explore/exploit schemes for web content optimization. In *Ninth IEEE International Conference on Data Mining (ICDM)*, pages 1–10, 2009.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Zahy Bnaya, Rami Puzis, Roni Stern, and Ariel Felner. Volatile multi-armed bandits for guaranteed targeted social crawling. In *AAAI (Late-Breaking Developments)*, 2013.

Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 273–280, 2009.

Raphaël Féraud and Tanguy Urvoy. A stochastic bandit algorithm for scratch games. In *JMLR: Workshop and Conference Proceedings, Asian Conference on Machine Learning*, volume 25, pages 129–143, 2012.

Varun Kanade, H Brendan McMahan, and Brent Bryan. Sleeping experts and bandits with stochastic action availability and adversarial rewards. 2009.

Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

Setareh Maghsudi and Slawomir Stańczak. On channel selection for energy-constrained rateless-coded d2d communications. In *Proc. 23rd European Signal Processing Conference (EUSIPCO)*, pages 1028–1032. IEEE, 2015.

R. J. Vaughan and W. N. Venables. Permanent expressions for order statistic densities. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2): 308–310, 1972.