Journal of Machine Learning Research 22 (2021) 1-99 Submitted 12/17; Revised 4/20; Published 1/21

Regulating Greed Over Time in Multi-Armed Bandits

Stefano Tracà STET@ALUM.MIT.EDU
Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Cynthia Rudin CYNTHIA@CS.DUKE.EDU
Department of Computer Science
Duke University
Durham, NC 27708, USA

Weiyu Yan WEIYU.YAN@ALUMNI.DUKE.EDU
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708, USA

Editor: Shie Mannor

Abstract

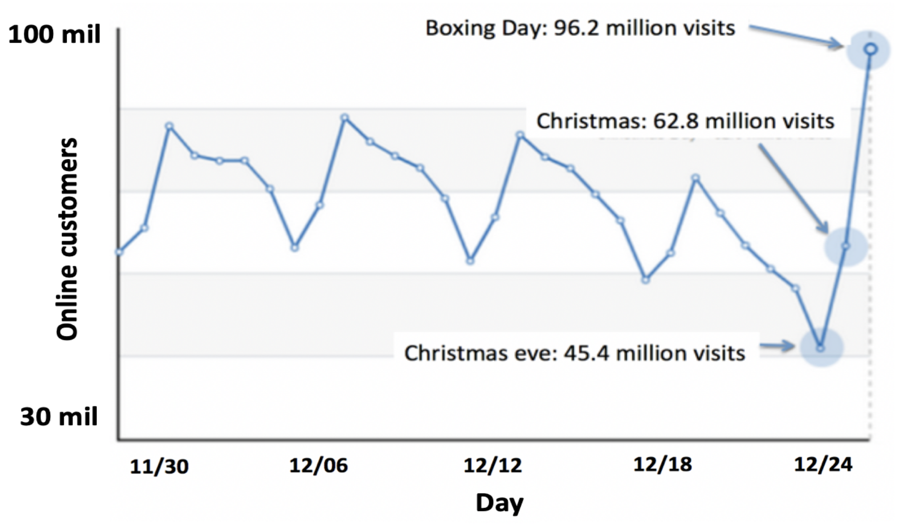
In retail, there are predictable yet dramatic time-dependent patterns in customer behavior, such as periodic changes in the number of visitors, or increases in customers just before major holidays. The current paradigm of multi-armed bandit analysis does not take these known patterns into account. This means that for applications in retail, where prices are fixed for periods of time, current bandit algorithms will not suffice. This work provides a remedy that takes the time-dependent patterns into account, and we show how this remedy is implemented for the UCB, ε-greedy, and UCB-L algorithms, and also through a new policy called the variable arm pool algorithm. In the corrected methods, exploitation (greed) is regulated over time, so that more exploitation occurs during higher reward periods, and more exploration occurs in periods of low reward. In order to understand why regret is reduced with the corrected methods, we present a set of bounds that provide insight into why we would want to exploit during periods of high reward, and discuss the impact on regret. Our proposed methods perform well in experiments, and were inspired by a high-scoring entry in the Exploration and Exploitation 3 contest using data from Yahoo! Front Page. That entry heavily used time-series methods to regulate greed over time, which was substantially more effective than other contextual bandit methods.

Keywords: Multi-armed bandit, exploration-exploitation trade-off, retail management, online applications, regret bounds, incorporating time-series into bandits.

1. Introduction

Consider a classic pricing problem faced by retailers, where the price of a new product on a given day is chosen to maximize the expected profit. The optimal price is learned asymptotically through a mix of exploring various pricing choices and exploiting those known to yield higher profits, potentially through the use of a multi-armed bandit (MAB). The retailer is not permitted to change the price for the day once it has been set. The demand (the number of customers) is approximately known in advance, since we assume the retailer knows the daily trend of the number of customers over time that visit the store. This information can be leveraged in order produce a better exploration/exploitation scheme. For instance, if we know that many customers will come to the store on the week before Christmas, we would not want to explore new prices on those days. We might even stop exploring all together during that week. Our setting violates the classic assumptions of random rewards with a static probability distribution that is typically considered in multi-armed bandits. Since the retailer cannot change the price more than once each day, daily rewards are correlated through the trends in customer behavior; a large number of customers on a given day means a possible larger

Figure 1: Trend of English users shopping online in the weeks prior to Christmas. Retailers can observe clear weekly patterns in sales. In particular, customers tend not to shop right before Christmas and on Christmas day, but they delay their purchases to Boxing Day. When using a multi-armed bandit approach, a retailer would not want to be in exploration phase when there is a peak in visits like the one observed on Boxing Day. Source: ispreview.co.uk.



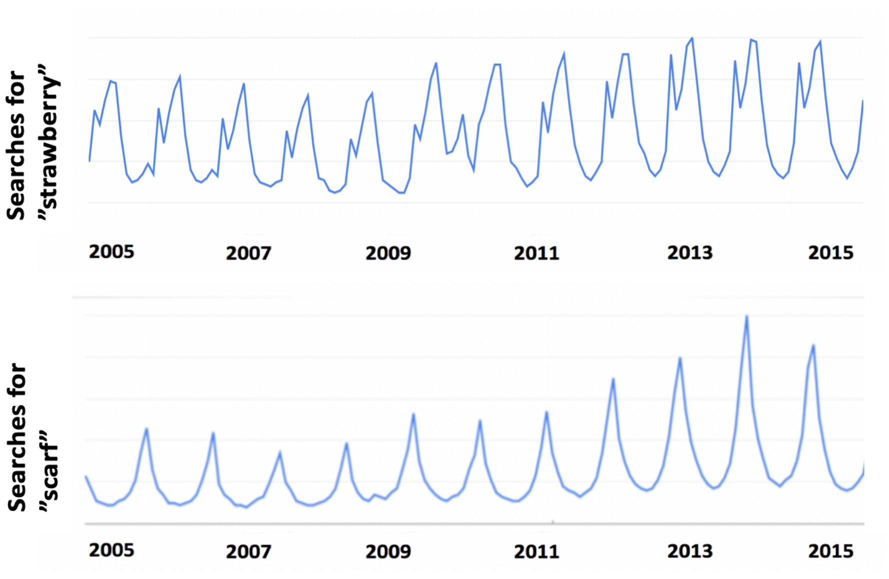
reward for that day (but also a larger possible regret for that day if the price is chosen sub-optimally). If one uses a standard MAB algorithm in the case where trends are dramatic, the result could be bad; an example is the case where the number of customers at the store will have a predictably large spike on a given day (e.g., for boxing day in England, shown in Figure 1), where the classic MAB algorithm could choose a suboptimal price on that particular day for the purpose of exploration. For retailers, there are almost always clear trends in customer arrivals, and they are often periodic or otherwise predictable. Some examples are in Figure 2. These dramatic trends might have a substantial impact on which policy we would use to price products.

The main contributions of this work are: i. A new framework that illustrates when it is beneficial to stop or limit exploration to favor exploitation; ii. Algorithms that show how to adapt existing policies to regulate greed over time, and a new algorithm (variable arm pool) that regulates greed over time; iii. Theoretical regret bounds for the algorithms; iv. Numerical comparisons, both in a simulated environment (Section 4 and Appendix G) and in a real-data environment (Section 6 and Appendix H). To help the reader navigate the algorithms and their theoretical results, we listed the detailed contributions in Table 1. We compare the performance of our algorithms to “smarter” versions of the classic ε-greedy algorithm (Algorithm 6) and UCB algorithm (Algorithm 7). Since cumulative rewards are impacted by trends, the standard algorithms incorrectly estimate the mean rewards of the arms. The “smarter” versions fix this issue, and thus are a reasonable baseline to compare with. However, the “smarter” algorithms do not regulate greed over time and their performance is worse than the algorithms that do this regulation. We also show experiments when the reward multiplier is not known but is estimated using time series analysis.

In our setting, the behavioral information about customers is distilled so that it takes the form of a *reward multiplier* $G(t)$, where we assume $G(t)$ is known or can be well-estimated before the decision is made at time t. $G(t)$ should be thought of as the number of customers in the store on day t. If $G(t)$ is not known but could be well-approximated, the regret bounds weaken accordingly.

The new algorithms, that take advantage of knowing $G(t)$, are not a simple extension of the ε-greedy algorithm and the UCB algorithm. They anticipate the number of customers and choose how much exploration to allow at that timestep. Some multiplier functions will work better than others for regulating greed over time,

Figure 2: Google searches for the word "scarf" and the word "strawberry" over a ten-year period. Retailers can observe clear yearly patterns in the quantity of searches. When using a multi-armed bandit approach, a retailer could use the periods with low searches to explore new strategies (such as price, or location of the items in a store, or coupons), and then exploit the information gained during high seasons. Source: Google Trends.



but the theorems provided will reveal when the cumulative regret will be high due to the form of $G(t)$, which (as we mentioned) is known in advance.

As a result of the reward multiplier function, theoretical regret bound analysis of the multi-armed bandit problem becomes more complicated, because now the distribution of rewards depends explicitly on time. We not only care *how many* times each suboptimal arm is played, but exactly *when* they are played. For instance, if suboptimal arms are played only when the reward multiplier is low, intuitively it should not hurt the overall regret. A strength of our theorems is that they tell us when and when not to use our algorithms, depending on the multiplier function; for some multiplier functions we can determine in advance when the algorithms are likely to produce large regret.

A Python implementation of the algorithms is available online.[1].

2. Related Work

Multi-armed bandit algorithms were introduced by Lai and Robbins (1985). There are several surveys and books that cover the many streams of work on multi-armed bandit problems (e.g., Bergemann and Valimaki, 2006; Cesa-Bianchi and Lugosi, 2006; Gittins et al., 2011; Bubeck et al., 2012).

The setup of this work differs from other works considering time-dependent multi-armed bandit problems – we do not assume the mean rewards of the arms exhibit random changes over time (they are static in our setting, as in the non-time-dependent problem), and we assume that the reward multiplier is known in advance. Other works consider different scenarios where reward distributions can change over time, but in a way that is not known in advance. For these settings, the algorithm needs to compensate for changes in the reward distribution after the change, rather than altering their strategy in advance of the change. Along these lines, Liu et al. (2013) consider a problem where each arm transitions in an unknown Markovian way to a different

1. The implementation of the algorithms used in the simulated environment is available at `https://github.com/5tefan0/Regulating-Greed-Over-Time`
The implementation of the algorithms used in the real data setting is available at `https://github.com/ShrekFelix/Regulating-Greed-Over-Time`.

Table 1: Location of algorithms and theoretical results. The proofs are in the Appendix.

Section	Algorithm	Related Theoretical Results	Proof
3.2	ε-z greedy algorithm	Theorem 1: Regret bound for Alg. 1 Theorem 2: Reformulation of Theorem 1 Corollary 3: Corollary of Theorem 1	A
3.3	Soft ε-greedy algorithm	Theorem 4: Regret bound for Alg. 2 Theorem 5: Reformulation of Theorem 4	B
3.4	UCB-z algorithm	Lemma 6: Minimum pulls with UCB Theorem 7: Lower bound for Lemma 6 Theorem 8: Regret bound for Alg. 3 Corollary 17: Extension of Lemma 6 Corollary 18: Lower bound for Corollary 17 Corollary 19: Corollary of Corollary 18	C
3.5	Soft UCB algorithm	Theorem 9: Regret bound for Alg. 4 Theorem 10: Reformulation of Theorem 9 Lemma 22: Minimum pulls with Soft UCB Lemma 23: Extension of Lemma 22	D
3.6	Variable pool algorithm	Theorem 12: Regret bound for Alg. 5	E
5	Soft UCB mortal algorithm	Theorem 13: Regret bound for Alg. 8	F

reward state when it is played, and evolves according to an unknown random process when it is not played. Garivier and Moulines (2008) presented an analysis of a discounted version of the UCB and a sliding window version of the UCB, where the distribution of rewards can have abrupt changes and stays stationary in between. Besbes et al. (2014) considers the case where the mean rewards for each arm can change, where the variation of that change is bounded. Slivkins and Upfal (2007) consider an extreme case where the rewards exhibit Brownian motion, leading to regret bounds that scale differently than typical bounds (linear in T rather than logarithmic). One of the works that is relevant to ours is that of Chakrabarti et al. (2009) that considers "mortal bandits," where arms disappear or appear. In the mortal setting, we compare the performance of Algorithm 8 with the UCB-L Algorithm introduced in Tracà et al. (2019), which regulates exploration based on the remaining lifespans of the available arms.

A particularly interesting setting is discussed by Komiyama et al. (2013), where there are lock-up periods when one is required to play the same arm several times in a row. There is a similarity of that problem to the one studied here. In our setting, we fix the price of the product for an entire day, each day is a timestep, and there is not a separate timestep for each customer. If we were instead to take a timestep for each customer, we would need to lock the arm over the course of the day in order to keep the same price throughout the entire day. In other words, in our scenario, the micro-lock-up periods occur at each step of the game, and their effective lengths are given by $G(t)$. (The difference between these situations is that playing the same arm $G(t)$ times is not equivalent to playing an arm once and then multiplying the reward by $G(t)$; if we multiply a single step's reward by $G(t)$, we do not learn the same amount as if we had taken $G(t)$ pulls and viewed the regret $G(t)$ separate times.) In the work of Komiyama et al. (2013), lock periods are present but there is no regulating greed based on the size of the lock periods.

The setting studied in Bouneffouf and Feraud (2016) also consider the case of rewards that follow a known trend, but it is fundamentally different because in our setting the multiplier function is exogenous (in the sense that it does not depend on the arms). In Bouneffouf and Feraud (2016), the trend of the rewards of an arm depends on how many times the arm has been played (for example, people may not like a song the first time they hear it, but they may like it more after a few times), so there are no particular periods where it is more important to regulate greed and stop exploration, in order to exploit more during high-reward periods.

The ideas in this paper were inspired by a high scoring entry in the Exploration and Exploitation 3 Phase 1 data mining competition, where the goal was to build a better recommendation system for Yahoo! Front

Page news articles. At each time, several articles were available to choose from, and these articles would appear only for short time periods and would never be available again. One of the main ideas in this entry was simple yet effective: if any article gets more than 9 clicks out of the last 100 times we show the article, and keep displaying it until the clickthrough rate goes down. This alone increased the clickthrough rate by almost a quarter of a percent. In the Yahoo! advertising problem, the high reward period was created by the availability of an article (an arm), which is different than the retail store case, but the same effect is present, where regulating the rate of exploitation (i.e., *greed*) over time is beneficial to overall rewards. Here also, it is useful to stop exploring during times when a function like $G(t)$ is high. For Yahoo! Front Page, articles have a short lifespan and some articles are much better than others, in which case, if we find a particularly good article, we should exploit by repeatedly showing that one, and not explore new articles. The framework here distills the problem, allowing us to isolate and study this effect of a time dependent function that we can use to regulate greed over time.

3. Algorithms for regulating greed over time

This section illustrates the problem, the proposed algorithms to regulate greed over time, and theoretical results on the bound on the expected regret of each policy.

3.1 Problem setup

Formally, the stochastic multi-armed bandit problem with regulated greed is a game played in n rounds. At each round t the player chooses an action among a finite set of m possible choices called *arms*. For retail, these arms could be prices set once per day, coupons offered for a fixed time period, or what products to put on sale for the day (these are all actions chosen at time t that will affect all $G(t)$ customers arriving within that round). Note that prices, coupons, and sales can not be chosen per customer, so the standard setting of the multi-armed bandit does not apply here. When arm j is played ($j \in \{1, \cdots, m\}$) an *unscaled* random reward $X_j(t)$ is drawn from an unknown distribution and the player receives the *scaled* reward $X_j(t)G(t)$ where $G(t)$ is the *multiplier function*. The distribution of $X_j(t)$ does not change with time (the index t is just used to indicate the turn in which the reward was drawn), while $G(t)$ is a known function of time assumed to be bounded (this is, for instance, the number of customers in the store on day t). At each turn, the player suffers also a possible regret from not having played the best arm. The unscaled mean regret for having played arm j is given by $\Delta_j = \mu_* - \mu_j$, where μ_* is the mean reward of the best arm (indicated by "*") and μ_j is the mean reward obtained when playing arm j; the scaled mean regret is given by $G(t)\Delta_j$. At the end of each turn the player can update their estimate of the mean reward μ_j and use it in the next turn t:

$$\widehat{X}_j = \frac{1}{T_j(t-1)} \sum_{s=1}^{t-1} X_j(s) \mathbb{1}_{\{I_s=j\}}, \tag{1}$$

where $T_j(t-1)$ is the number of times arm j has been played before round t starts, and $\mathbb{1}_{\{I_s=j\}}$ is an indicator function equal to 1 if arm j has been played at time s (otherwise its value is 0). This update will help the player in choosing a good arm in the next round. The total regret at the end of the game is given by

$$\text{Total regret} = \sum_{t=1}^{n} \sum_{j=1}^{m} \left(X_*(t) - X_j(t)\right) G(t) \mathbb{1}_{\{I_t=j\}}.$$

If the choice of the arm played at time t is fixed a priori by the sequence $\mathcal{I} = \{I_t\}_{t=1}^{n}$, the total mean regret of a game $\mathcal{I}$ is defined as

$$R_n = \mathbb{E}_{X_1,\dots,X_m}[\text{Total regret}] = \sum_{t=1}^{n} \sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{I_t=j\}},$$

where the expectation is taken over the distribution of the rewards of each arm. When the choice of the arm at time t is not fixed but given a policy π, the expected cumulative regret is defined as $\mathbb{E}_\pi[R_n]$, where now the expectation is taken over the distribution induced by the policy on the choice of the arms to play. When there is not confusion on the policy used, we simply indicate this quantity by $\mathbb{E}[R_n]$ and we simply call this quantity "expected regret" or "mean regret". The strategies presented in the following sections aim to minimize the expected cumulative regret $\mathbb{E}[R_n]$ by regulating *exploitation* (i.e., *greed*) of the best arm found so far, and *exploration*, based on the values of the multiplier function $G(t)$. In general, when the multiplier function is high, the player risks incurring a high regret if a bad arm is played. We show that it is beneficial to stop exploration in this situation and resume exploration when rewards and regrets are lower. A complete list of the symbols used throughout the paper can be found in Appendix J.

For any strategy (not just the ones in this paper), it is problematic if $G(t)$ is high early in the game. This would be analogous to opening a new store just before the sales peak at Christmas, and being expected to have already optimized the price at that time. In that case, no strategy could have explored enough to perform well. The strategies presented in this work are useful for cases where we can explore enough before $G(t)$ becomes large to determine what the optimal arm should be. When $G(t)$ is large, our strategies exploit. In the exposition that follows, we present the strategy as is, without imposing additional exploration to deal with high $G(t)$ at early times. This serves a useful purpose: it allows one to see directly in the bounds that forcing exploitation when $G(t)$ is high may not work early on. But in fact, *any* algorithm, whether exploring or exploiting, will have a poor regret guarantee when $G(t)$ is high early on. Even early on, it still may be better in practice to exploit the best arm so far, rather than to try a risky arm when the stakes are high.

3.2 Regulating greed with threshold using an ε-greedy algorithm

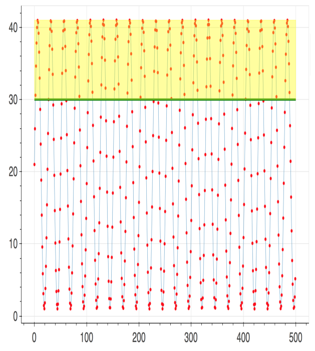
In Algorithm 1 we present the ε-z greedy algorithm, a variation of the ε-greedy algorithm of Auer et al. (2002), in which a threshold z has been introduced in order to regulate greed. This is the simplest method we know that would allow regulating greed over time. It has the disadvantage of adding one more parameter z, though we usually choose the heuristic of z being 75% of the maximum of $G(t)$. A good value for z can also be estimated by running the algorithm on past data and by finding the one that gives the lowest regret. At each turn t, when the rewards are "high" (i.e., the $G(t)$ multiplier is above the threshold z) the algorithm exploits the best arm found so far, that is, arm j with the highest mean estimate as defined in Equation (1). When the rewards are "low" (i.e., the $G(t)$ multiplier is under the threshold z), the algorithm will explore an arm at random (each arm having probability $1/m$ of being selected) with probability $\varepsilon_t = \min\{1, km/\tilde{t}\}$, where $\tilde{t}$ counts how many times the multiplier function has been under the threshold up to time t, and k is a constant greater than 10 and such that $k > 4/(\min_j \Delta_j^2)$. The examples in Figure 3 show how the threshold z (in green) determines which turns are in the high reward periods (region in yellow, when $G(t) \geq z$ we want to be greedy) or in the low reward periods ($G(t) < z$, where we balance exploration and exploitation).

The following theorem provides a finite-time bound on the mean regret of the ε-z greedy algorithm (proof in Appendix A) in the form of:

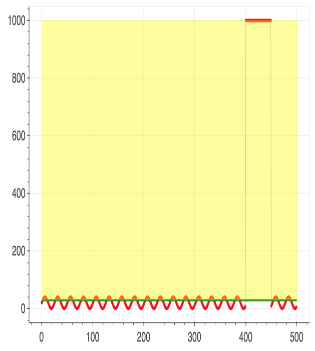
$$\begin{aligned} \mathbb{E}[R_n] &\leq \text{Regret during initialization phase} \\ &+ \text{Regret under the threshold} \\ &+ \text{Regret above the threshold.} \end{aligned}$$

Figure 3: Examples of how a multiplier function $G(t)$ (in red) can be divided into high reward periods and low reward periods by the threshold $z = 30$. Refer to Section 4 for the analytical expression of the "Wave Greed", "Christmas Greed", and "Step Greed" multiplier functions.

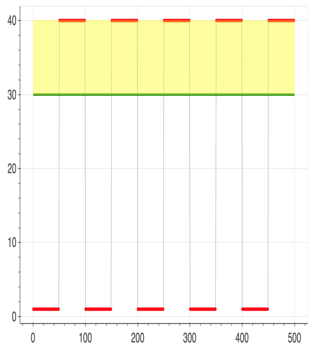
(a) The *Wave Greed*



(b) The *Christmas Greed*



(c) The *Step Greed*



```
Algorithm 1: $\varepsilon$-z greedy algorithm
  Input          : number of rounds $n$, number of arms $m$, threshold $z$, a constant $k > 10$ such that
                   $k > 4/(\min_j \Delta_j^2)$, sequences $\{\varepsilon_t\}_{t=1}^n = \min\{1, km/\tilde{t}\}$ and $\{G(t)\}_{t=1}^n$
  Initialization : play all arms once and initialize $\widehat{X}_j$ (defined in (1)) for each $j = 1, \dots, m$
  for $t = m + 1$ to $n$ do
      if $G(t) < z$ then
          with probability $\varepsilon_t$ play an arm uniformly at random (each arm with probability $1/m$),
            otherwise (with probability $1 - \varepsilon_t$) play arm $j$ such that $\widehat{X}_j > \widehat{X}_i \; \forall i$;
      else
          play arm $j$ such that $\widehat{X}_j > \widehat{X}_i \; \forall i$;
      end
      Get reward $G(t)X_j$;
      Update $\widehat{X}_j$;
  end
```

Theorem 1 (Regret bound of the ε-z greedy algorithm) *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\begin{aligned}\mathbb{E}[R_n] \quad &\leq \quad \sum_{j=1}^{m} G(j)\Delta_j &(2)\\ &+ \quad \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)<z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \left(\varepsilon_t \frac{1}{m} + (1-\varepsilon_t)\beta_j(\tilde{t})\right) &(3)\\ &+ \quad \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)\geq z\}} \sum_{j:\mu_j<\mu_*} \Delta_j\beta_j(\tilde{t}), &(4)\end{aligned}$$

where

$$\beta_j(\tilde{t}) = k\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k}{10}} \log\left(\frac{\tilde{t}}{mke}\right) + \frac{4}{\Delta_j^2}\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k\Delta_j^2}{4}}. \qquad (5)$$

The sum in (2) is the exact mean regret during the initialization phase of Algorithm 1. Addend (3) is a bound on the expected regret for turns that present low values of $G(t)$, where the quantity in the parenthesis is an upper bound on the probability of playing arm j: $\beta_j(\tilde{t})$ in (5) is an upper bound on the probability that arm j is considered to be the best arm at round t, and $1/m$ is the probability of choosing arm j when the choice is made at random. The reason for the choice of $k > 4/(\min_j \Delta_j^2)$ is to make $\beta_j(\tilde{t})$ a quantity that is $o(1/\tilde{t})$. By setting the parameter k accordingly, we can ensure the logarithmic bound on the expected cumulative regret over the number of rounds (because the ε_t are $\theta\left(1/t\right)$ and their sum over time is logarithmically bounded, and $\beta_j(\tilde{t})$ is $o\left(1/\tilde{t}\right)$). Finally, in (4) we have a bound on the expected regret for turns with high values of $G(t)$, and in this case we consider only the upper bound $\beta_j(\tilde{t})$ on the probability that arm j is considered to the best arm since we do not explore during high reward periods. The usual ε-greedy algorithm as introduced by Auer et al. (2002) is a special case when $G(t) = 1\ \forall t$ and $z > 1$. Notice that ε_t is a quantity $\theta\left(1/\tilde{t}\right)$, while $\beta_j(\tilde{t})$ is $o\left(1/\tilde{t}\right)$, so that an asymptotic logarithmic bound in n holds for $\mathbb{E}[R_n]$ if $\tilde{t}$ grows at the same rate as t (because of the logarithmic upper bound of the harmonic series[2]).

The following version of Theorem 1 shows the order of magnitude of the regret at the end of the game.

Theorem 2 (Regret bound of the ε-z greedy algorithm) *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \mathcal{O}(1) + \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)<z\}}\left(\theta\left(\frac{1}{t}\right) + o\left(\frac{1}{t}\right)\right) + \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)\geq z\}} o\left(\frac{1}{t}\right).$$

Intuitively, this bound is better than the standard ε-greedy bound because, when $G(t)$ is low, the regret of choosing a suboptimal arm is multiplied by a quantity that is of the order $\theta\left(1/t\right) + o\left(1/t\right)$, while when $G(t)$ is high it is multiplied by a $o\left(1/t\right)$ quantity. In contrast, in the standard ε-greedy algorithm, the regret of choosing a suboptimal arm is always multiplied by a quantity that is of the order $\theta\left(1/t\right) + o\left(1/t\right)$.

3.2.1 COMPARISON WITH A SMARTER VERSION OF THE STANDARD ε-GREEDY ALGORITHM

We want to compare this bound with the one of the standard version of the ε-greedy algorithm but, since it is not well suited for the setting in which the rewards are altered by the multiplier function, we discount the rewards obtained at each round (by dividing them by $G(t)$) so that it can also produce accurate estimates of

2. Logarithmic upper bound of the harmonic series: $\sum_{t=1}^{n} \frac{1}{t} \leq \log(n) + 1$

the mean reward for each arm. This "smarter" version of the ε-greedy algorithm is presented in Algorithm 6 (Section 4). The bound on the probability of playing a suboptimal arm j for the standard ε-greedy algorithm is given by $\beta_j(t)$ (i.e., $\beta_j(\tilde{t})$ when $\tilde{t} = t$) and we refer to it as $\beta_j^{\text{old}}(t)$. In general, $\beta_j^{\text{old}}(t)$ is lower than $\beta_j(\tilde{t})$ (since $\tilde{t} \leq t$). Intuitively, this reflects the fact that the new algorithm performs fewer exploration steps. Moreover, in the standard ε-greedy algorithm, the probability of choosing a suboptimal arm j at time t is given by

$$\mathbb{P}\left(I_t^{\text{old}} = j\right) = \varepsilon_t \frac{1}{m} + (1 - \varepsilon_t)\beta_j^{\text{old}}(t),$$

which is less than the probability of the new algorithm in case of low $G(t)$ (even if the rate of decay is the same),

$$\mathbb{P}\left(I_t^{\text{new}} = j\right) = \varepsilon_t \frac{1}{m} + (1 - \varepsilon_t)\beta_j(\tilde{t}),$$

but can easily be higher than the probability of the new algorithm in case of high rewards (which is given by only $\beta_j(\tilde{t})$). In fact, for suboptimal arm j, when $G(t) > z$, we have

$$\begin{aligned}\mathbb{P}\left(I_t^{\text{old}} = j\right) - \mathbb{P}\left(I_t^{\text{new}} = j\right) &= \varepsilon_t \frac{1}{m} + (1 - \varepsilon_t)\beta_j^{\text{old}}(t) - \beta_j(\tilde{t}) \\ &= \frac{1}{m}\min\left\{1, \frac{km}{t}\right\} - \beta_j^{\text{old}}(t)\min\left\{1, \frac{km}{t}\right\} + \beta_j^{\text{old}}(t) - \beta_j(\tilde{t}). \qquad (6)\end{aligned}$$

If $t > km$ we get

$$(6) = \frac{1}{m}\frac{km}{t} + \beta_j^{\text{old}}(t)\left(1 - \frac{km}{t}\right) - \beta_j(\tilde{t}), \qquad (7)$$

if $t \leq km$ we get

$$(6) = \frac{1}{m} - \cancel{\beta_j^{\text{old}}(t)} + \cancel{\beta_j^{\text{old}}(t)} - \beta_j(\tilde{t}), \qquad (8)$$

and for t large enough both expressions are positive since $\beta_j(\tilde{t})$ is $o\left(1/\tilde{t}\right)$. Having (7) and (8) positive means that when we are in a high-rewards period the probability of choosing a suboptimal arm decreases faster when using Algorithm 1. In that case, Algorithm 1 would have lower regret than the standard ε-greedy algorithm.

If past data are available, a good value for z can be chosen using cross validation techniques, i.e., by trying different thresholds with the available data and by choosing the one that yields the best performance, or using the heuristic of the 75% of the maximum value of $G(t)$ as mentioned earlier (we used this heuristic in the real-data experiment in Section 6).

3.2.2 REGRET BOUND FOR A SIMPLE SCENARIO

The following corollary illustrates the benefits of the bound in a simple scenario (shown in Figure 3c) when the multiplier function can only take two values and the regulating threshold divides the higher value from the lower one (this is the case of the "Step" multiplier function used in the numerical experiments in Section 4).

Corollary 3 *Suppose the greed function $G(t)$ takes only two values: g_{low} and g_{high}. It takes the value g_{low} for a fraction q of the turns played, and the value g_{high} for the remaining $n - qn$ turns. Then, the bound in Theorem 1 of the expected regret at turn n reduces to*

$$\mathbb{E}[R_n] \leq \mathcal{O}(1) + g_{low}\,(qn)\left(\theta\left(\frac{1}{t}\right) + o\left(\frac{1}{t}\right)\right) + g_{high}\,(n - qn)\,o\left(\frac{1}{t}\right).$$

The term $\theta(1/t)$ that hurts regret most is multiplied only by g_{low}, and not by g_{high}. When the rewards are high (and so is the possible regret), only terms of order $o(1/t)$ are present. If exploration were permitted during the high reward zone (as is the case when using the standard ε-greedy algorithm), large terms g_{high} would affect the regret, which is what Algorithm 1 is designed to avoid. In this case it does not matter where we put the threshold as long as it is above g_{low} and below g_{high}. In Appendix I.1 we generalize Theorem 1 to the case when the function $G(t)$ is not known but estimated by $H(t)$. (The method used in Appendix I.1 is easily applied to the following theorems too when $G(t)$ is not known).

3.3 Soft ε-greedy algorithm

We present in Algorithm 2 a "soft version" of the ε-greedy algorithm where greed is regulated gradually (in contrast with the hard threshold z of the previous section). This algorithm has the advantage of an adaptive threshold which the user does not need to choose. Again, in high reward zones, exploitation will be preferred, while in low reward zones the algorithm will explore the arms more. Let us define the following function

$$\psi(t) = \frac{\log\left(1 + \frac{1}{G(t)}\right)}{\log\left(1 + \frac{1}{\min_{s\in\{m+1,\cdots,n\}} G(s)}\right)}, \tag{9}$$

and let $\gamma = \min_{s\in\{m+1,\cdots,n\}} \psi(s)$. Notice that $0 < \psi(t) \leq 1\ \forall t$ and that its values are close to 0 when $G(t)$ is high, while they are close to 1 for low values of $G(t)$. The new probabilities of exploration during the game are given at each turn t by $\varepsilon_t = \min\{\psi(t), km/t\}$. In this way, we still maintain the decay of the probabilities of exploration, but we push them closer to zero when the multiplier function $G(t)$ is high to avoid high regrets. We generally assume that $\min_{s\in\{m+1,\cdots,n\}} G(s)$ is not smaller than 1. This particular functional form of $\psi(t)$ has been chosen because it is well-suited to be used for determining probabilities (of exploration), the standard ε-greedy algorithm of Auer et al. (2002) is recovered when $G(t) = 1$ for all t, and it makes the regret bound easier to prove.

Algorithm 2: Soft ε-greedy algorithm

```
Input          : number of rounds $n$, number of arms $m$, a constant $k > 10$, such that
                 $k > 4/(\min_j \Delta_j^2)$, sequences $\{\varepsilon_t\}_{t=1}^n = \min\{\psi(t), km/t\}$ and $\{G(t)\}_{t=1}^n$
Initialization : play all arms once and initialize $\widehat{X}_j$ (defined in (1)) for each $j = 1, \ldots, m$
for $t = m + 1$ to $n$ do
    With probability $\varepsilon_t$ play an arm uniformly at random (each arm with probability $\frac{1}{m}$),
    otherwise (with probability $1 - \varepsilon_t$) play arm $j$ such that $\widehat{X}_j > \widehat{X}_i\ \forall i$;
    Get reward $G(t)X_j$;
    Update $\widehat{X}_j$;
end
```

The following theorem (proved in Appendix B) shows that a logarithmic bound holds in this case too (because the ε_t are $\theta(1/t)$ and their sum over time is logarithmically bounded, while the $\beta_j^S(t)$ term is $o(1/t)$).

Theorem 4 (Regret-bound for Soft ε-greedy algorithm) *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \sum_{j=1}^{m} G(j)\Delta_j \tag{10}$$

$$+ \sum_{t=m+1}^{n} G(t) \sum_{j:\mu_j<\mu_*} \Delta_j \left(\varepsilon_t \frac{1}{m} + (1-\varepsilon_t)\beta_j^S(t)\right) \tag{11}$$

where

$$\beta_j^S(t) = k\left(\frac{\gamma t}{mke}\right)^{-\frac{k}{10}} \log\left(\frac{\gamma t}{mke}\right) + \frac{4}{\Delta_j^2}\left(\frac{\gamma t}{mke}\right)^{-\frac{k\Delta_j^2}{4}}. \tag{12}$$

The sum in (10) is the exact mean regret during the initialization of Algorithm 2. For the rounds after the initialization phase, the quantity in the parenthesis of (11) is the upper bound on the probability of playing arm j (where $\beta_j^S(t)$ is the bound on the probability that arm j is the best arm at round t, and $1/m$ is the probability of choosing arm j when the choice is made uniformly at random).

Theorem 4 can be stated in a simpler form that shows the order of magnitude of the bounding quantity:

Theorem 5 (Regret bound of the Soft ε greedy algorithm) *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \mathcal{O}(1) + \sum_{t=m+1}^{n} G(t)\left[\min\left(\psi(t), \theta\left(\frac{1}{t}\right)\right) + o\left(\frac{1}{t}\right)\right]. \tag{13}$$

Intuitively, the advantage of Algorithm 2 over the standard ε-greedy algorithm is that, when $G(t)$ is high, only the $o(1/t)$ term contributes significantly to the regret (because $\psi(t)$ pushes the term $\min(\psi(t), \theta(1/t))$ to zero).

3.3.1 COMPARISON WITH A SMARTER VERSION OF THE STANDARD ε-GREEDY ALGORITHM

As before, we want to compare this bound with the "smarter" version of the ε-greedy algorithm presented in Algorithm 6. In the usual ε-greedy algorithm, after the "critical time" $n' = km$, the probability $\mathbb{P}(\widehat{X}_{j,T_j(t-1)} \geq \widehat{X}_{i,T_i(t-1)})$ of arm j being considered as the current best arm can be bounded by a quantity $\beta_j^{\text{old}}(t)$ that is $o(1/t)$. Before time n', the decay of $\mathbb{P}(\widehat{X}_{j,T_j(t-1)} \geq \widehat{X}_{i,T_i(t-1)})$ is exponential: $\theta(e^{-t})$ (see Remark 15 in Appendix A). The probability of choosing a suboptimal arm j changes as follows:

- if $t < n'$, $\mathbb{P}(I_t = j) = \frac{1}{m}$;
- if $t \geq n'$, $\mathbb{P}(I_t = j) = \frac{k}{t} + \left(1 - \frac{km}{t}\right)\beta_j^{\text{old}}(t)$, which is $\theta\left(\frac{1}{t}\right)$ as t grows.

In the Soft ε-greedy algorithm, before time $w = \min\{s : f(s) < \gamma\, f(s) = km/s\}$, we have that $\beta_j^S(t)$, which is the bound on the probability $\mathbb{P}(\widehat{X}_{j,T_j(t-1)} \geq \widehat{X}_{i,T_i(t-1)})$ of arm j being the current best arm, is a quantity that decays exponentially with rate $\theta(e^{-\gamma t})$ (see Remark 16 in Appendix B). After w, $\beta_j^S(t)$ can be bounded by a quantity that is $o(1/(\gamma t))$ as t grows. The bound on the probability of choosing a suboptimal arm j changes as follows:

- if $t < n'$, $\mathbb{P}(I_t = j) = \frac{1}{m}\psi(t) + (1-\psi(t))\beta_j^S(t)$;
- if $n' \leq t \leq w$, $\mathbb{P}(I_t = j) = \frac{1}{m}\min\left\{\psi(t), \frac{km}{t}\right\} + \left(1 - \min\left\{\psi(t), \frac{km}{t}\right\}\right)\beta_j^S(t)$;

Table 2: Summary of the **decay rate of the bound on the probabilities of choosing a suboptimal arm** for the Soft ε-greedy algorithm and the standard ε-greedy algorithm (supposing it is taking in account the time-patterns, i.e., a "smarter" version). The decay depends on the turn number of the game (see Figure 4). The probability of choosing a suboptimal arm decays much faster for the Soft ε-greedy algorithm when $G(t)$ is high. For the other cases, the rate of decay is the same.

round t	$G(t)$	$\mathbb{P}(I_t = j)^{\text{old}}$	$\mathbb{P}(I_t = j)^{\text{soft}}$	$\mathbb{P}(I_t = j)^{\text{soft}} < \mathbb{P}(I_t = j)^{\text{old}}$?
$t < n'$	high	$\frac{1}{m}$	$\theta(e^{-\gamma t})$	yes, faster decay
	low	$\frac{1}{m}$	close to $\frac{1}{m}$	no, but not by much
$n' \leq t \leq w$	high	$\theta\left(\frac{1}{t}\right) + o\left(\frac{1}{t}\right)$	$\theta(e^{-\gamma t})$	yes, faster decay
	low	$\theta\left(\frac{1}{t}\right) + o\left(\frac{1}{t}\right)$	$\theta\left(\frac{1}{t}\right) + \theta(e^{-\gamma t})$	yes, but same rate of decay
$t > w$	high	$\theta\left(\frac{1}{t}\right) + o\left(\frac{1}{t}\right)$	$\theta\left(\frac{1}{t}\right) + o\left(\frac{1}{\gamma t}\right)$	no, but same rate of decay
	low	$\theta\left(\frac{1}{t}\right) + o\left(\frac{1}{t}\right)$	$\theta\left(\frac{1}{t}\right) + o\left(\frac{1}{\gamma t}\right)$	no, but same rate of decay

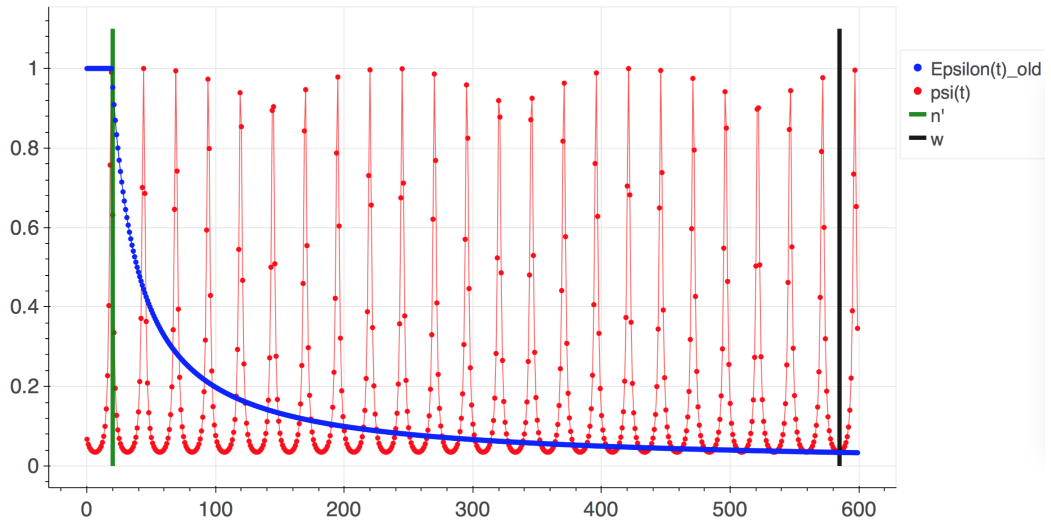
- if $t > w$, $\mathbb{P}(I_t = j) = \frac{k}{t} + \left(1 - \frac{k}{t}\right) \beta_j^S(t)$.

In order to interpret these quantities, let us see what happens for high or low values of the multiplier $G(t)$ as t grows in Table 2. For brevity, we abuse notation when using Landau's symbols, because in some cases t is not allowed to go to infinity; it is convenient to still use this notation to compare the decay rates of the probabilities of choosing a suboptimal arm, which also gives a qualitative explanation of what happens when using the algorithms. Let $\mathbb{P}(I_t = j)^{\text{soft}}$ be the probability of choosing a suboptimal arm for the Soft ε-algorithm. When comparing to the standard ε-greedy algorithm, the rate of decay of $\mathbb{P}(I_t = j)^{\text{soft}}$ is faster when $G(t)$ is high and $t \leq w$ (for the other cases, the rate of decay is the same as $\mathbb{P}(I_t = j)^{\text{old}}$, the probability of choosing a suboptimal arm for the standard ε-greedy algorithm). Notice that the parameter γ slows down the decay. This is a direct consequence of the slower exploration. An example of a typical behavior of $\psi(t)$ and $\varepsilon_t^{\text{old}}$ is shown in Figure 4, where $G(t) = 20 + 19\sin(t/2)$. The blue curve shows the probability of exploration under the usual ε-greedy algorithm that does not regulate greed, while the red curve shows how the function $\psi(t)$ oscillates depending on the value of the multiplier function. If $\psi(t) < km/t$, then $\psi(t)$ is the probability of exploration at time t which drops when $G(t)$ is high (which means higher rewards but also higher regrets), while it is bounded by km/t when $G(t)$ is low (which means lower rewards and regrets).

3.4 Regulating greed with threshold in the UCB algorithm

Before we provide results for our algorithm, we need to introduce some novel preliminary results on the standard UCB algorithm. Specifically, we need the minimum number of times that each arm will be pulled on or before turn t (Lemma 6). This quantity is required each time $G(t)$ exceeds the threshold. It is used for computing an upper bound on the probability that the algorithm thinks arm j is the best. (We cannot use the typical result of the expected number of times that each arm is pulled, as is used in standard UCB analysis.)

Figure 4: Comparison of probabilities of exploration over the number of rounds. Before n', $\varepsilon_t^{\text{old}}$ is 1 and always greater than $\psi(t)$. After w, $\varepsilon_t^{\text{old}}$ is always less than $\psi(t)$. When $G(t)$ is high, $\psi(t)$ pushes the probability of exploration towards zero.



3.4.1 PRELIMINARY RESULTS ON THE UCB ALGORITHM

The following lemma provides the minimum number of times each arm is pulled when using the standard UCB algorithm.

Lemma 6 *Suppose the rewards of the arms are bounded in $[a, b]$ and let us define $r = b - a$ the range of the possible rewards. When using the UCB policy for a game consisting of n turns, each arm will be pulled at least x_n times, where*

$$\begin{aligned} x_n &= \max\{y \in \mathbb{N} : (m-1)\phi(r,y) \le n\} + 1, \\ \phi(r,y) &= \min\left\{ t \ge \tau(r,y),\, t \in \mathbb{N} : t > \frac{2\,\log(t)}{r^2 + \frac{2\,\log(t)}{y} - 2r\sqrt{\frac{2\,\log(t)}{y}}} \right\}, \\ \tau(r,y) &= \min\left\{ t \in \mathbb{N} : \sqrt{\frac{2\log(t)}{y}} \ge r \right\}. \end{aligned}$$

Corollary 17 in Appendix C considers the case where random rewards of each arm take values in different ranges. Theorem 7 gives a lower bound on the previous quantity, showing that the minimum number of times the algorithm pulls an arm grows at least logarithmically.

Theorem 7 *Suppose the rewards of the arms are bounded in $[a, b]$ and let us define $r = b - a$ the range of the possible rewards. Let x_n be the minimum number of times each arm will be pulled when using the UCB policy for a game consisting of n turns ($n > m$). Then,*

$$x_n \in \Omega(\log(n)).$$

Corollary 18 in Appendix C considers the case where random rewards of each arm take values in different ranges.

3.4.2 UCB WITH THRESHOLD

Following what has been presented to improve the ε-greedy algorithm in the setting with multiplier function $G(t)$, we introduce in Algorithm 3 a modification of the UCB algorithm. We again set a threshold z and, if the multiplier of the rewards $G(t)$ is above this level, the new algorithm exploits the best arm. When $G(t)$ is under the threshold, the algorithm is going to play the arm with the highest upper confidence bound on the mean estimate. The threshold z can be chosen as suggested in Section 3.2.

Algorithm 3: UCB-z algorithm

```
Input          : number of rounds $n$, number of arms $m$, threshold $z$, sequence $\{G(t)\}_{t=1}^{n}$
Initialization : play all arms once and initialize $\widehat{X}_j$ (as defined in (1)) for each $j = 1, \cdots, m$
for $t = m+1$ to $n$ do
    if $G(t) < z$ then
        play arm $j$ with the highest $\widehat{X}_{j,T_j(t-1)} + \sqrt{\frac{2\log t}{T_j(t-1)}}$;
    else
        play arm $j$ such that $\widehat{X}_j > \widehat{X}_i \; \forall i$;
    end
    Get reward $G(t)X_j$;
    Update $\widehat{X}_j$;
end
```

Let $B = \{t : G(t-1) < z, G(t) \geq z\}$ be the set of rounds where the high-reward zone is entered. Let us call $y_1, y_2, \cdots, y_{|B|}$ the elements of B and order them in increasing order such that $y_1 < y_2 < \cdots < y_{|B|}$. Let us also define for every $k \in \{1, \ldots, |B|\}$ the set $Y_k = \{t : t \geq y_k, G(t) \geq z, t < y_{k+1}, y_{|B|+1} = n\}$ of times in the high-reward period entered at time y_k. Now, given a game of n total rounds, let us call $\tilde{n}$ the number of rounds played under the threshold z by the end of the game. The regret bound showed in Theorem 8 is the sum of the regret in the high reward zone and the regret in the low reward zone and is of the form

$$\begin{aligned} \mathbb{E}[R_n] \;&\leq\; \text{Regret during initialization phase} \\ &+\; \text{Regret under the threshold} \\ &+\; \text{Regret above the threshold.} \end{aligned}$$

(When $G(t) = 1$ for all t the usual regret bound for the UCB policy is recovered.) The regret bound also uses Lemma 6 which counts the number of times arm j has been pulled when using an UCB policy under the threshold. Usually, for applications of the UCB algorithm, the expected number of pulls is sufficient for theoretical results, but our framework is different, since we need to compute the probability of playing an arm at each turn t the threshold is exceeded.

Theorem 8 (Regret-bound for the UCB-z algorithm) *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\begin{aligned}\mathbb{E}[R_n] &\leq \sum_{j=1}^{m} G(j)\Delta_j && (14)\\ &+ z\left[8\sum_{j:\mu_j<\mu_*}\left(\frac{\log\tilde{n}}{\Delta_j}\right)+\left(1+\frac{\pi^2}{3}\right)\left(\sum_{j=1}^{m}\Delta_j\right)\right] && (15)\\ &+ \sum_{j=1}^{m}\Delta_j\sum_{k=1}^{|B|}\sum_{t\in Y_k} G(t)2\beta_j^U(t), && (16)\end{aligned}$$

where

$$\beta_j^U(t)=\frac{2}{\Delta_j^2}e^{-\frac{\Delta_j^2}{2}(x_t-1)}$$

and x_t is the minimum amount of pulls for each arm at time t (see Lemma 6.)

The sum in (14) represents the exact regret coming from the initialization phase, in (15) we have a logarithmic regret (in the number of turns played under the threshold) that comes from the classic UCB policy, where z is the upper bound on $G(t)$ for low reward turns and the square brackets is the sum of the regret of each arm multiplied by an upper bound on the expected number of times that arm is played. Finally, in (16) we have the regret that comes from turns in the high reward zone, where $2\beta_j^U(t)$ is an upper bound on the probability of playing arm j at turn t. Theorem 7 guarantees that β_j^U decreases fast enough since x_t grows at least logarithmically. For the proof of Theorem 8, Theorem 7, and Lemma 6 see Appendix C.

3.5 The Soft UCB algorithm

In Algorithm 4, present a “soft version" of the UCB algorithm where greed is regulated gradually (in contrast with the hard threshold of the previous section). Again, in high reward zones, exploitation will be preferred, while in low reward zones the algorithm will explore the arms.
Let us define the following function:

$$\xi(t)=\left(1+\frac{t}{G(t)}\right). \tag{17}$$

At each turn t of the game, the algorithm plays the arm with the highest upper confidence bound on the mean estimate, but, with the introduction of $\xi(t)$, the confidence interval around $\widehat{X}_{j,T_j(t-1)}$ is built in a way such that, when $G(t)$ is high, it collapses upon the mean reward estimate, forcing the player to choose the arm with the highest mean estimate (thus, leading to a pure exploitation policy). In contrast, when the multiplier $G(t)$ is low, the confidence interval around $\widehat{X}_{j,T_j(t-1)}$ stretches out, allowing the player to explore arms with more uncertainty.

One of the main difficulties of the formulation of these bounds is to define a correct functional form for the upper confidence bounds so that it is possible to obtain smoothness in the arm decision, reasonable Hoeffding's inequality bounds while working out the proof (see Appendix D), and a convergent series (the second summation in (19)). This particular choice for the functional form of $\xi(t)$ was chosen because of the following:

- $\xi(t)$ correctly stretches or shrinks the upper confidence bound based on the multiplier function $G(t)$;
- when $G(t)$ is a constant equal to 1, the standard UCB algorithm is almost recovered (“almost” because the upper confidence bound, when $G(t)=1$, would have a $\log(t)$ factor for standard UCB and a $\log(1+t)$ factor for soft UCB, where we have an extra 1 to avoid a negative argument of the logarithm);

- $\xi(t)$ is well-suited to be the argument of the logarithm that appears under the square root of the upper confidence bound (because it is positive);
- we can easily use Hoeffding's inequality at the beginning of the proof of the regret bound (see Appendix D).

Algorithm 4: Soft UCB algorithm

```
Input          : number of rounds $n$, number of arms $m$, sequence $\{G(t)\}_{t=1}^n$
Initialization : play all arms once and initialize $\widehat{X}_j$ (as defined in (1)) for each $j = 1, \cdots, m$
for $t = m + 1$ to $n$ do
    play arm $j$ with the highest $\widehat{X}_{j,T_j(t-1)} + \sqrt{\frac{2\log\xi(t)}{T_j(t-1)}}$;
    Get reward $G(t)X_j$;
    Update $\widehat{X}_j$;
end
```

Also in this case, it is possible to achieve a bound that grows logarithmically in n.

Theorem 9 (Regret-bound for soft-UCB algorithm) *Let $S = \{m+1, \ldots, n\}$. The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \sum_{j=1}^{m} G(j)\Delta_j \tag{18}$$

$$+ \max_{t\in S} G(t)\left[\sum_{j:\mu_j<\mu_*} \frac{8}{\Delta_j}\log\left(\max_{t\in S}\xi(t)\right) + \sum_{j=1}^{m}\Delta_j\left(1 + \sum_{t=m+1}^{n}\frac{2(t-1-m)^2}{\xi(t)^4}\right)\right]. \tag{19}$$

Theorem 9 can be stated in a simpler form that shows the order of magnitude of the bounding quantity.

Theorem 10 (Regret-bound for soft-UCB algorithm) *Let $S = \{m+1, \ldots, n\}$. The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \mathcal{O}(1) \tag{20}$$

$$+ \max_{t\in S} G(t)\left[\mathcal{O}\left(\log\left(\max_{t\in S}\xi(t)\right)\right) + \mathcal{O}(1)\right]. \tag{21}$$

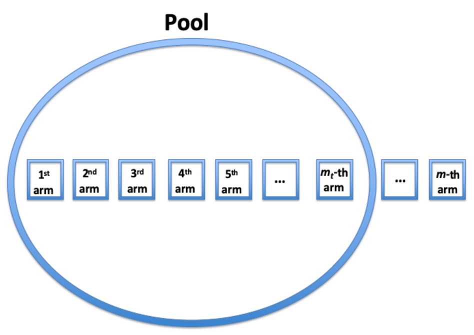
The first sum in (18) is the exact mean regret of the initialization phase of Algorithm 4. For the rounds after the initialization phase, the mean regret is bounded by the quantity in (19), which is almost identical to the bound of the usual UCB algorithm if we assume $G(t) = 1$ (i.e., rewards are not modified by the multiplier function). Similarly to the UCB algorithm with threshold, it is possible to compute the minimum number of times an arm will be pulled before or on turn t (see Lemma 22 and Lemma 23 in Appendix D).

3.6 Regulating greed with variable arm pool

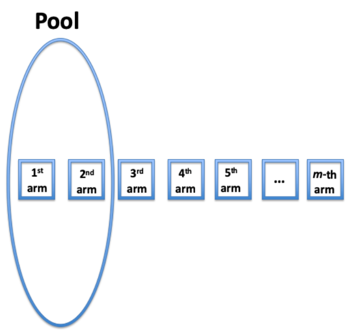
In Algorithm 5 we present a policy that regulates greed by varying the size m_t of the pool of arms from which we are allowed to choose (uniformly at random). When the greed function is high, the pool size of arms m_t shrinks (possibly to just one arm: the one with the highest mean reward so far), so that we choose randomly among the arms that performed best. When the greed function is low we choose randomly among a larger pool (which could possibly contain all the arms). Unlike ε-greedy algorithms that can go back to very bad arms

Figure 5

(a) The arm pool when $G(t)$ is low: most arms are included, and the algorithm is in exploration mode.



(b) The arm pool when $G(t)$ is high: only arms with the highest estimated mean reward are included, and the algorithm is in exploitation mode.



when exploring, this algorithm only explores among all arms when $G(t)$ is very small. The size of the pool is given by

$$m_t = \min\left(m, \max\left(1, \left\lfloor \frac{cm}{tG(t)} \right\rfloor\right)\right), \tag{22}$$

where $c > 1$. Figure 5a and 5b show an example of the pool behavior when $G(t)$ is low or when it is high.

Algorithm 5: variable arm pool algorithm

```
Input : number of rounds $n$, number of arms $m$, a constant $c > 1$, and $\{G(t)\}_{t=1}^{n}$;
for $t = m + 1$ to $n$ do
    Set pool size to $m_t = \min\left(m, \max\left(1, \left\lfloor \frac{cm}{tG(t)} \right\rfloor\right)\right)$;
    Play arm $j$ at random from the pool ;
    Get reward $G(t)X_j$;
    Update $\widehat{X}_j$;
end
```

Let us define

$$\lambda_t = \frac{1}{2m} \sum_{s=1}^{t} \mathbb{1}\{m_s = m\},$$

where $\sum_{s=1}^{t} \mathbb{1}\{m_s = m\}$ is the number of times that the pool contained all the arms up to turn t. For the following theorem we require that the multiplier function $G(t)$ is such that $\lambda_t > \gamma \log(t)$ for some $\gamma > 5$.

Remark 11 *If $G(t)$ does not satisfy the requirement that $\lambda_t \geq \gamma \log(t)$ it is easy to construct a new multiplier function $G'(t)$ by first finding the set $S = \{t : \lambda_t > \lceil \gamma \log(t-1) \rceil\}$ and then by defining*

$$G'(s) = \begin{cases} (c-1)/s & \text{for } s \in \{t, t+1, \cdots, t+2m\} \text{ if } s \in S \\ G(s) & \text{otherwise.} \end{cases}$$

If we do use $G'(t)$ instead of $G(t)$ in the algorithm, the bound that we present below holds for $G'(t)$ instead of $G(t)$.

The following theorem provides a finite-time bound on the regret after n rounds.

Theorem 12 (variable pool algorithm) *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \sum_{t=1}^{n} \sum_{j=1}^{m} \Delta_j G(t) \frac{2}{m_t} \beta_j^{VP}(t), \tag{23}$$

where

$$\beta_j^{VP}(t) = \gamma \log(t)(t)^{-\gamma/5} + \frac{2}{(\Delta_j^{*-m_t})^2} t^{-\frac{\gamma(\Delta_j^{*-m_t})^2}{2}}. \tag{24}$$

Here β_j^{VP} is the upper bound on the probability of choosing arm j. This algorithm tends to perform well in the experiments. Like previous algorithms, it also regulates greed based on the multiplier function (i.e., if $G(t)$ is high your pool may just contain the arm with the highest mean reward and that translates into exploitation of the best arm), but it goes back to play bad arms less often than ε-greedy or UCB policies.

4. Experimental results: simulated environment

We consider three types of multiplier function $G(t)$:

- The *Wave Greed* (Figure 6a): in a Wave-type greed function, rewards are multiplied following the trend of a periodic wave: $G(t) = 21 + 20\sin(0.25t)$ for $t \in \{1, \cdots, n\}$. We aim to exploit the best arm found so far during the peaks, while balancing exploration and exploitation during low-rewards periods. This behavior mimics weekly patterns.
- The *Christmas Greed* (Figure 6b): similarly to the Wave greed, rewards are multiplied following the trend of a wave, but a big peak (which we call "Christmas", in analogy to the phenomenon of the boom of customers during the Christmas holidays) appears towards the end of the game. Formally, $G(t) = 21 + 20\sin(0.25t)$ if $t \in [1, 0.8n) \cup (0.9n, n]$, and $G(t) = 1000$ if $t \in [0.8n, 0.9n]$.
- The *Step Greed* (Figure 6c): a Step-type greed function assumes only two values: one low and one high. In our case, we choose $G(t) = 40$ (high value) if $t \in [0.1n, 0.2n) \cup [0.3n, 0.4n) \cup [0.5n, 0.6n) \cup [0.7n, 0.8n) \cup [0.9n, n]$, otherwise $G(t) = 1$ (low value). We aim to exploit the best arm so far when the greed function assumes its high value, while balancing exploration and exploitation when it assumes its low value.

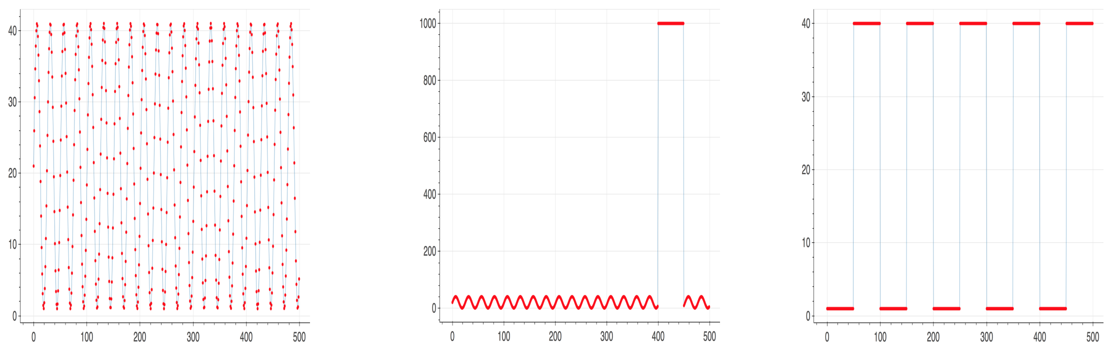
In this section we report the mean results (over 50 games) of the final cumulative rewards of games consisting of 1500 turns and 200 arms. A complete set of results with different number of arms (25, 50, 100, and 200) and different game-lengths (500, 1000, and 1500 turns) can be found in Appendix G.

We consider rewards coming from two different distributions:

- Bernoulli distribution: when rewards come from Bernoulli distributions, each arm is assigned a probability of success p_j drawn randomly from an uniform distribution on $[0, 1]$;

Figure 6: Shapes of the multiplier functions used in the experiments.

(a) The *Wave Greed* (b) The *Christmas Greed* (c) The *Step Greed*



- Truncated-Normal distribution: when rewards come form Truncated-Normal distributions, each arm is assigned a mean reward μ_j drawn randomly from an uniform distribution on $[0, 1]$, and a standard deviation $\sigma = 0.1$. Rewards for arm j are then drawn from a Normal $N_j \sim \mathcal{N}(\mu_j, 1)$, but are bounded in $[0, 1]$ so that the reward at time t is given by $X_j(t) = \max(0, \min(1, N_j))$.

In the algorithms, it is also possible to use Normal distributed rewards, but since the theorems about UCB algorithms require the rewards to be bounded, we preferred to keep this assumption also in the experiments.

The standard UCB and ε-greedy algorithms are not suitable for the setting in which the rewards are altered by the multiplier function. Thus, in their current form, it would not be fair to compare directly with them. The fact that rewards are multiplied by $G(t)$ would irremediably bias all the estimations of the mean rewards, leading standard UCB and ε-greedy to choose arms whose rewards were artificially inflated because they happened to be played in a high reward period. For example, suppose we show an ad on a website at lunch time: many people will see it because at that time the web-surfing is at its peak (i.e., the $G(t)$ multiplier is high). So even if the ad was bad, we may register more clicks than a good ad showed at 3:00AM (i.e., when the $G(t)$ multiplier is low). To obtain a fair comparison, we created "smarter" versions of the UCB and ε-greedy algorithms in which the rewards are discounted at each round by $G(t)$; then, the old version of the algorithms can be smarter in that they can produce accurate estimates of the mean reward for each arm. The smarter version of the usual UCB algorithm is presented in Algorithm 7 and the one for the ε-greedy algorithm is shown in Algorithm 6.

In Figure 7, 8, and 9 we show the average cumulative rewards at the end of the game. The red part of the bar indicates what portion of the rewards came from "pure exploitation." The definition of "pure exploitation" depends on the algorithm used:

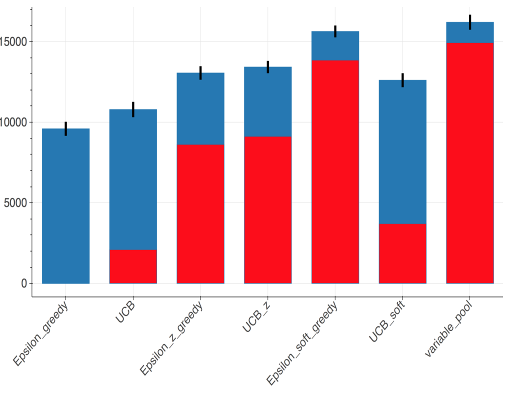
- pure exploitation in ε-greedy algorithms: when the algorithms decide to exploit. This is forced in algorithms with threshold when the greed function is above that threshold;
- pure exploitation in UCB algorithms: when the arm played has the highest estimated mean reward. This is forced in algorithms with threshold when the greed function is above that threshold;
- pure exploitation in the variable-pool algorithm: when the pool size is 1.

.

In Figure 10a and 10b we show the average increase in final rewards when regulating greed over time. The comparison is with respect to the (smarter) ε-greedy algorithm when the greed function is of the Wave type. In Figure 11a and 11b we show the average increase in final rewards when regulating greed over time. This time the comparison is with respect to the (smarter) UCB algorithm when the greed function is of the Wave type. The results show the power of regulating greed over time: by exploiting more when the greed function is high

Figure 7: Comparison of average final rewards in games with 200 arms, 1500 turns, and a Wave-type greed function. The first two bars refer to the smarter version of the ε-greedy and UCB algorithms, while the other five bars refer to our algorithms that regulate greed over time.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

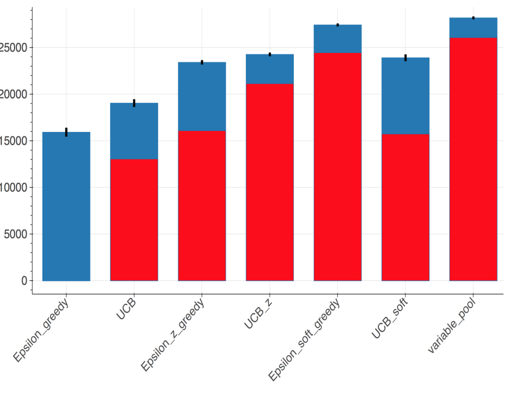
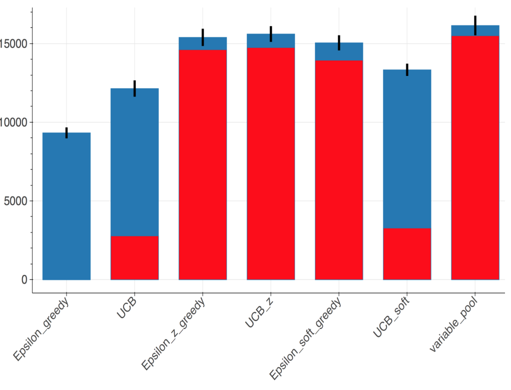


Figure 8: Comparison of average final rewards in games with 200 arms, 1500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

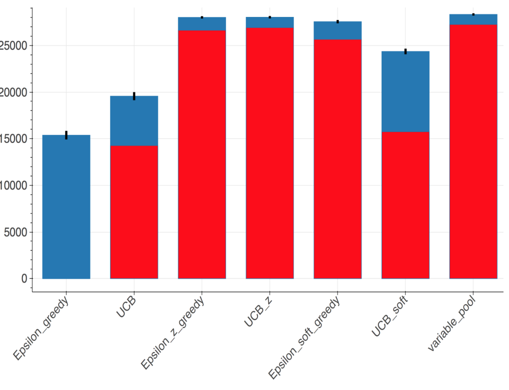


Figure 9: Comparison of average final rewards in games with 200 arms, 1500 turns, and a Christmas-type greed function.

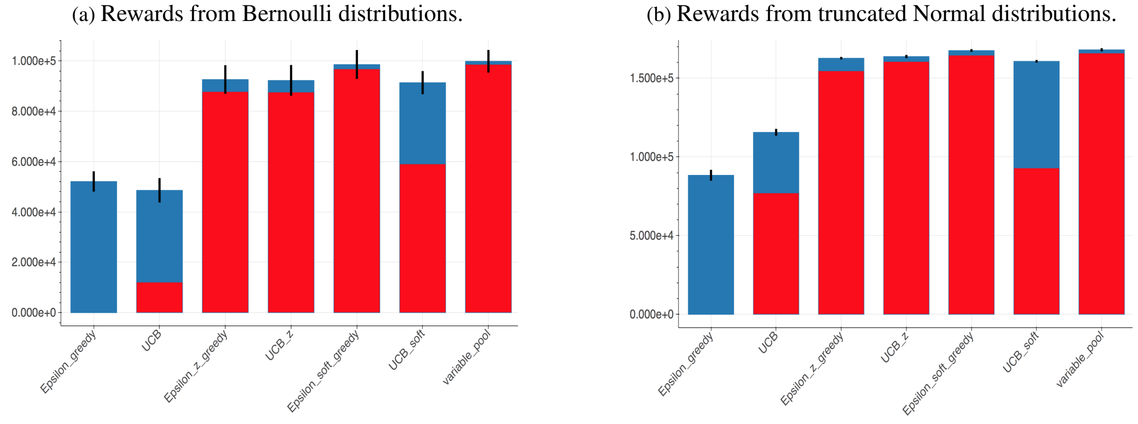
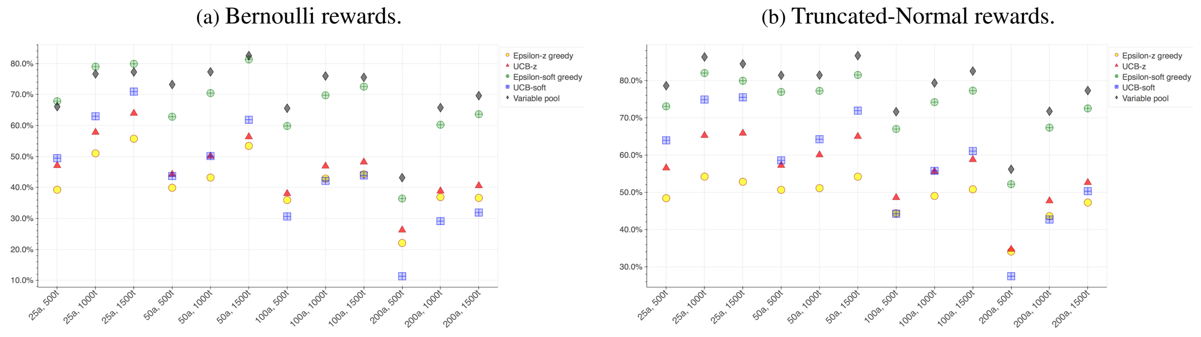


Figure 10: Cumulative Reward increase when regulating greed over time compared to the (smarter) ε-greedy algorithm (Wave-greed function).

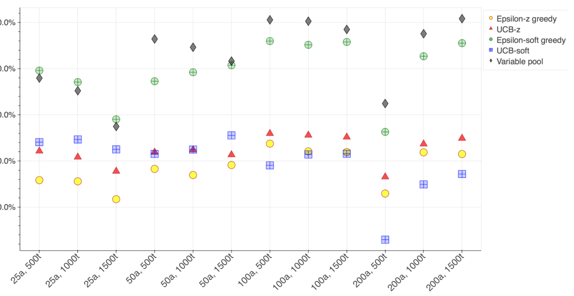


and exploring more when it is low, the final cumulative rewards are much higher than those of algorithms that do not regulate greed over time. Refer to the full set of experiments in Appendix G for the comparisons with the other types of greed functions.

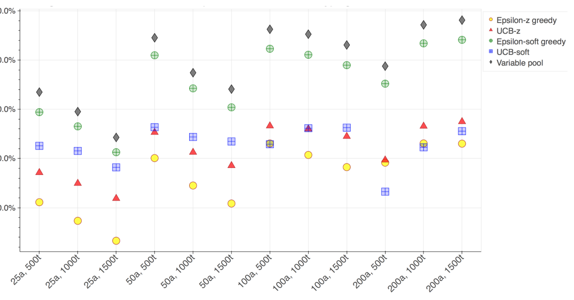
```
Algorithm 6: Smarter version of the standard $\varepsilon$-greedy algorithm
Input          : number of rounds $n$, number of arms $m$, a constant $c > 10$, a constant $d$ such that
                 $d < \min_j \Delta_j$ and $0 < d < 1$, sequences $\{\varepsilon_t\}_{t=1}^n = \min\{1, cm/d^2t\}$ and $\{G(t)\}_{t=1}^n$
Initialization : play all arms once and initialize $\widehat{X}_j$ (as defined in (1)) for each $j = 1, \cdots, m$
for $t = m + 1$ to $n$ do
    with probability $\varepsilon_t$ play an arm uniformly at random (each arm with probability $\frac{1}{m}$), otherwise
     (with probability $1 - \varepsilon_t$) play arm $j$ such that $\widehat{X}_j > \widehat{X}_i \; \forall i$;
    Get reward $G(t)X_j(t)$;
    Update $\widehat{X}_j$;
end
```

Figure 11: Cumulative Reward increase when regulating greed over time compared to the (smarter) UCB algorithm (Wave-greed function).

(a) Bernoulli rewards.



(b) Truncated-Normal rewards.



```
Algorithm 7: Smarter version of the standard UCB algorithm
  Input          : number of rounds $n$, number of arms $m$, sequence $\{G(t)\}_{t=1}^n$
  Initialization : play all arms once and initialize $\widehat{X}_j$ (as defined in (1)) for each $j = 1, \cdots, m$
  for $t = m + 1$ to $n$ do
      play arm $j$ with highest $\widehat{X}_{j,T_j(t-1)} + \sqrt{\frac{2\log(t)}{T_j(t-1)}}$;
      Get reward $G(t)X_j(t)$;
      Update $\widehat{X}_j$;
  end
```

5. Soft UCB mortal algorithm

Since the dataset from the Yahoo! Webscope program has arms that become available and unavailable at different times, we also introduce the mortal bandit setting and how the UCB algorithm can be adapted to this scenario.

In the mortal case, the algorithm chooses in turn t an arm among a set of available arms M_t. The set M_t can change: arms may become unavailable (they "die") and can never be played again, or new arms may arrive (they "are born"). Since the pool of available arms varies, the optimal arm to pull is not fixed anymore during the duration of the game. Let us denote the best arm in turn t by i_t^*. Let L_j be the set of turns during which arm j is available, and let s_j and l_j be the first and last turns in L_j. We assume that the lifespan of each arm is given (for example, an arm could be a coupon that has a known expiration date). Algorithm 8 is a modification of the UCB algorithm that has two main advantages. The first is to encourage exploration of arms that will be available over turns with high multiplicative reward $G(t)$. Each arm j has a score given by the sum of the values of $G(t)$ over its lifespan:

$$\text{score}_j = \sum_{t \in L_j} G(t).$$

At each turn t, the upper confidence bound for arm j will be scaled by the fraction of arms available that have score below that of j:

$$\psi_{\text{future}}(j,t) = \frac{c}{|M_t|} \sum_{i \in M_t} \mathbb{1}\{\text{score}_i \leq \text{score}_j\},$$

where $c \geq 1$ is a positive constant (in the experiments, $c = 2.0$). The function $\psi_{\text{future}}(j,t)$ compares the scores of the available arms and it is high if arm j will be alive during high values of $G(t)$.

The second main advantage of Algorithm 8 is that it encourages exploitation over turns with high multiplicative reward $G(t)$. Similarly to the Soft UCB algorithm, the function $\xi_{\text{present}}(t)$ will shrink the upper confidence bound in the presence of high values of $G(t)$:

$$\xi_{\text{present}}(t) = \left(1 + \frac{t}{G(t)}\right). \tag{25}$$

In the initialization phase, a subset of arms $M_I \subset M_1$ are played. M_I can be chosen by selecting arms with the highest values of $\psi_{\text{future}}(j, 1)$ (for simplicity, in the experiments we set $M_I = M_1$). Theorem 13 provides a finite-time regret bound for Algorithm 8.

Algorithm 8: Soft UCB mortal algorithm

```
Input          : number of rounds $n$, set $M_t$ of available arms at turn $t$, rewards range $[a, b]$, with
                 $r = b - a$
Initialization : play all arms in $M_I$ once, and initialize $\widehat{X}_j$ for each $j = 1, \cdots, |M_I|$
for $t = m_I + 1$ to $n$ do
    Play arm with highest $\widehat{X}_j + \psi_{\text{future}}(j, t)\sqrt{\frac{2\log \xi_{\text{present}}(t-s_j)}{T_j(t-1)}}$;
    Get reward $G(t)X_j(t)$;
    Update $\widehat{X}_j$;
end
```

Theorem 13 *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*

$$u_{j,z} = \max_{t \in \{s_j^z, \cdots, l_j^z\}} \left\lceil \frac{8\psi_{\text{future}}(j, t) \log \xi_{\text{present}}(t - s_j)}{\Delta_{j,z}^2} \right\rceil, \tag{26}$$

where

$$\Delta_{j,z} = \Delta_{j,i_t^*} \text{ for } t \in L_j^z. \tag{27}$$

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

$$\mathbb{E}[R_n] \leq \sum_{j \in M_I} G(j)\Delta_{j,i_t^*} \tag{28}$$

$$+ \sum_{j \in M} \sum_{z=1}^{E_j} \left(\max_{t \in E_j} G(t)\right) \Delta_{j,z} \min\left(l_j^z - s_j^z \ , \ \beta_j^M\right), \tag{29}$$

where

$$\beta_j^M = u_{j,z} + \sum_{\substack{t \in L_j^z \\ t > m_I}} (t - s_{i_t^*})(t - s_j - u_{j,z} + 1)\left[\xi_{\text{present}}(t - s_j)^{-\frac{4}{r^2}\psi_{\text{future}}(j,t)} + \xi_{\text{present}}(t - s_{i_t^*})^{-\frac{4}{r^2}\psi_{\text{future}}(i_t^*,t)}\right]. \tag{30}$$

In Theorem 13, we split each lifespan of each arm j into E_j epochs that have a different best arm. For each epoch L_j^z $(z = 1, \ldots, E_j)$, we denote with $\Delta_{j,z}$ the difference of the mean reward of the best available arm during epoch z and the mean reward of arm j. The quantity $u_{j,z}$ in (26) intuitively represents the number of times we need to have pulled a suboptimal arm j to be able to lower its probability of being chosen. The first summation in (28) is the regret suffered by the algorithm during the initialization phase, while the quantity in (29) bounds the regret for the rest of the game. The bound is computed by considering each epoch partition of each arm lifespan. If an epoch is too short (i.e., $\min(l_j^z - s_j^z \ , \ \beta_j^M) = l_j^z - s_j^z$), then the bound is achieved simply by multiplying the largest value of the multiplier function ($\max_{t \in E_j} G(t)$) by the length of

Table 3: An extract of the Yahoo! Webscope program dataset used in the simulation.

event index	timestamp	displayed article	clicked?	article pool	$G(t)$
1	1317513291	id-560620	0	id-560620,...	32
2		id-565648	0	id-565648,...	
3		id-563115	0	id-563115,...	
4	1317513292	id-552077	0	id-552077,...	13
5		id-564335	0	id-564335,...	

the epoch ($l_j^z - s_j^z$) and the mean regret of choosing arm j (which is $\Delta_{j,z}$). If an epoch is not too short (i.e., $\min(l_j^z - s_j^z\ ,\ \beta_j^M) = \beta_j^M$), then we can use the quantity defined in (30) instead of $l_j^z - s_j^z$. The proof of Theorem 13 is in Appendix F.

Algorithm 8 differs from the UCB-L algorithm (see Algorithm 9) introduced in Tracà et al. (2019) in that UCB-L regulates exploration based on the remaining life of the arms (i.e., if arms have a short lifespan or are close to their expiration then UCB-L favors exploration of arms that have longer lifespan) while the Soft UCB mortal algorithm encourages exploration of arms that are available when $G(t)$ is high (even if their lifespan might be short). If the arms never expire then the Soft UCB mortal algorithm reduces to the Soft UCB algorithm by setting $c = 1$ (because the score of all the arms is the same).

6. Experimental Results: real data environment

In this section, we show the performance of the proposed algorithms along with the widely used standard ε-greedy and UCB algorithms, using the event log data from the Yahoo! Webscope program. The dataset consists of a stream of recommendation events that display articles to users. Each entry of the dataset contains information on:

- the arm pulled (which is the article shown to the human viewing articles on Yahoo!);
- the outcome (whether the article was clicked or not);
- the pool of arms (articles) available at that time and the associated timestamp.

We preprocessed the original text file into a structured data frame (for a small extract of the data frame see Table 3). We created time bins of the duration of one second. We set $G(t)$ as the number of customers who appear during time bin t.

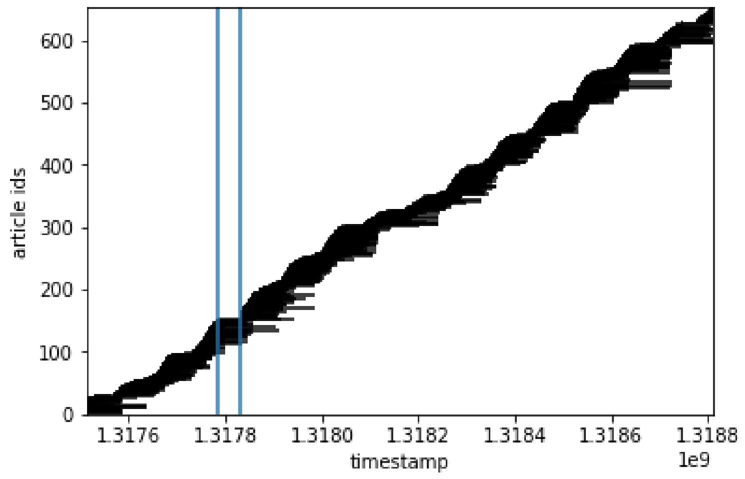
A unique property of this dataset is that the articles shown to the visitors were chosen uniformly at random from the pool of available articles, so that it is possible to use an unbiased offline evaluation methodology for our algorithms (see Li et al., 2011): at each time, the reward is calculated only when the article that was displayed to the human user matches the article chosen by the algorithm (otherwise, the record is discarded). More details on the offline evaluation can be found in Appendix H.

Note that articles in this dataset appear in the candidate pool at some time and often become unavailable shortly afterward. For this reason, the dataset creates a natural setting for mortal bandit algorithms. In Figure 12, the 652 little black horizontal bars represent the lifespans of the 652 articles that appear in the dataset. Since the ε-z greedy, Soft ε-greedy, UCB-z, and Soft UCB algorithms do not consider mortal bandits, we can run these algorithms only on a slice of the dataset where the candidate pool is invariant, so that we can create a non-mortal setting for the algorithms.

We also conducted experiments with variations of the UCB algorithm that are well-suited for a mortal setting in order to show the value of regulating greed over time:

- Soft UCB mortal (Section 5);
- A "smarter" version of UCB-L: Algorithm 9. The UCB-L algorithm regulates exploration based on the remaining lifespan of the arms, i.e., if an arm has short lifespan or will expire soon then the algorithm

Figure 12: The 652 little black horizontal bars represent the lifespans of the articles that appear in the dataset. Since the articles come and go, in order to recreate a non-mortal setting, we selected the longest slice with the most amount of arms always available (arms between the two vertical lines).



will prefer to play other arms that have longer life in order to gain information that can be used in the future (see Tracà et al., 2019, where UCB-L was one of the algorithms with the best performance).

We also included results of the standard UCB algorithm, since it is used in industry for mortal settings (even though it is not meant for a mortal setting) and to show the disadvantage of using an algorithm not tailored for the mortal setting.

We ran the algorithms 100 times to derive an empirical distribution of the rewards. For the ε-greedy algorithms and the variable pool algorithm, different articles are chosen during each run of the algorithms, because during exploration phases they are chosen at random. For those, we use one slice of the dataset (the time period between the two blue lines in Figure 12), which is the longest slice with the largest number of available arms. Also, since the offline evaluation consumes records very quickly (all records for which the algorithm did not choose the article recommended by the website must be discarded because no label is available), we duplicated this slice to increase the size of the dataset. The UCB algorithms are deterministic in their choices because they always pick the arm with the best upper confidence bound, i.e., running the UCB family many times on the same dataset will always give the same result. Therefore, we used a sliding window to cover different portions of the dataset to evaluate the performance of the UCB algorithms for each game. The randomness of their final rewards arises from the different portions of dataset used.

Figures 13-15 show the distribution of the rewards for each algorithm discussed above. We computed mean reward and conducted t-test to verify that the average reward of the algorithms that regulate exploration based on the values of the multiplier function $G(t)$ are significantly higher than the average rewards of the standard algorithms (a summary of the results is in Tables 4, 5, and 6). The algorithms that regulate greed outperform the standard ones.

6.1 Discussion on cases when the algorithms do not apply or reduce to standard cases

We have introduced several algorithms, each with their own regret bound. The regret bounds should guide us on the choice of algorithm, but our experiments indicate that the variable pool algorithm seems to perform consistently well in our experiments.

It is possible to construct pathological cases where bandit algorithms can perform badly in our new setting. Cases, for instance, where $G(t)$ is very high at the beginning and then decrease rapidly are examples where no bandit strategy would perform well, since the important decisions need to be made very early. In that case,

Table 4: Performance comparison of the ε-greedy algorithms in immortal circumstances. The mean reward in each turn is computed as the ratio of the total cumulative rewards and the number of turns.

Algorithm	Mean reward in each turn	p-value (vs. standard ε-greedy)
Standard ε-greedy	0.0406	-
ε-z greedy	0.0493	< 0.0001
Soft ε-greedy	0.0588	< 0.0001
Variable Pool	0.0688	< 0.0001

Table 5: Performance comparison of the UCB algorithms in immortal circumstances. The mean reward in each turn is computed as the ratio of the total cumulative rewards and the number of turns.

Algorithm	Mean reward in each turn	p-value (vs. standard UCB)
Standard UCB	0.0365	-
UCB-z	0.0441	< 0.0001
Soft UCB	0.0396	< 0.0001

Table 6: Performance comparison of variations of the UCB algorithm in mortal circumstances. The mean reward in each turn is computed as the ratio of the total cumulative rewards and the number of turns.

Algorithm	Mean reward in each turn	p-value (vs. standard UCB)
standard UCB	0.0523	-
UCB-L	0.0787	< 0.0001
UCB soft mortal	0.0840	< 0.0001

Figure 13: ε-greedy algorithms playing 100 games with 10000 turns per game. The best algorithm is the Soft ε-greedy algorithm, followed by variable pool and ε-z greedy. The standard ε-greedy algorithm has the lowest average reward.

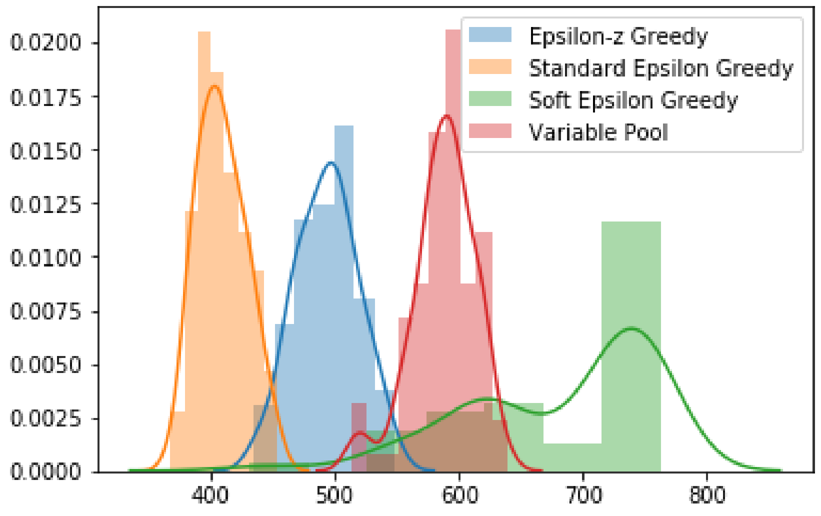


Figure 14: UCB algorithms playing a set of 100 slightly different games with 10000 turns per game under immortal circumstances (invariant arms pool). The best algorithm is the UCB-z algorithm, followed by Soft UCB. The standard UCB algorithm has the lowest average reward.

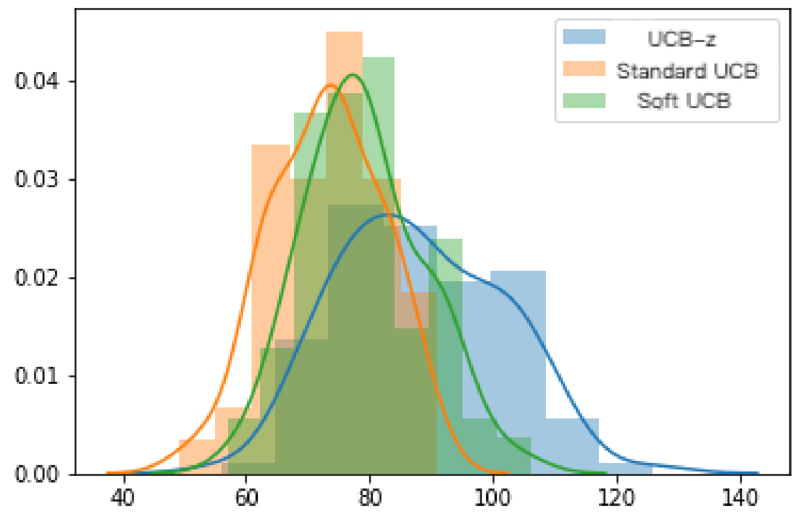
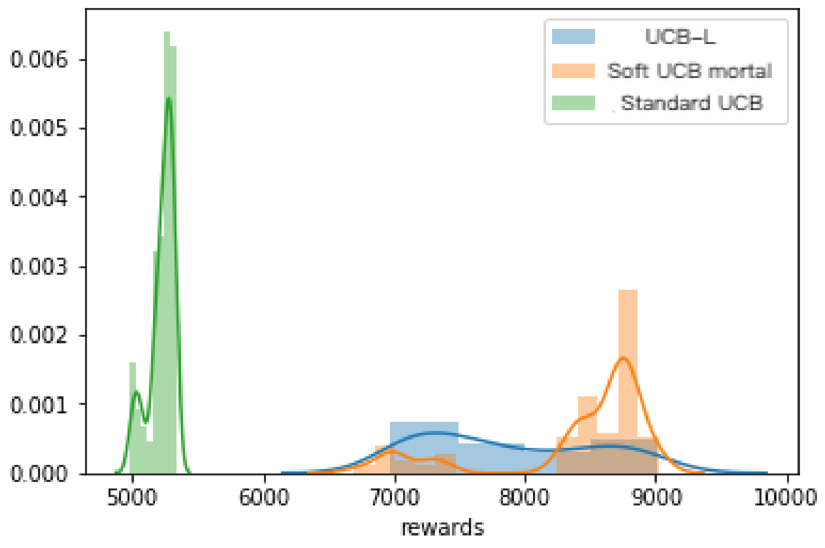


Figure 15: UCBs playing a set of 100 slightly different games with 10000 turns per game under mortal circumstances (variable arms pool). The best algorithm is the Soft UCB mortal algorithm, followed by UCB-L. The standard UCB algorithm has the lowest average reward.



```
Algorithm 9: Smarter version of the UCB-L algorithm
  Input          : number of rounds $n$, set $M_t$ of available arms at turn $t$, $c > 0$, rewards range $[a, b]$,
                   $\{G(t)\}_{t=1}^{n}$
  Initialization : play all arms in $M_I$ once, and initialize $\widehat{X}_j$ for each $j = 1, \cdots, |M_I|$
  for $t = m_I + 1$ to $n$ do
      Play arm with highest $\widehat{X}_j + \psi(j,t)\sqrt{\frac{2\log \xi(t-s_j)}{T_j(t-1)}}$,  where $\psi(j,t) = c\log(l_j - t + 1)$;
      Get reward $G(t)X_j(t)$;
      Update $\widehat{X}_j$;
  end
```

regulating greed may make the problem worse, but this is transparent from the regret bounds and thus known beforehand. Note that any bandit algorithm will also perform badly if rewards are very high at the beginning when little information is available.

Note also that if the number of turns is very high, regulating greed algorithms and standard MAB algorithms will all start to behave similarly because eventually they all figure out the best arms to play; experiments on bandits are only relevant in the non-asymptotic regime.

When $G(t) = 1$, all of the algorithms introduced here, except for the variable pool algorithm, reduce to standard MAB algorithms with same regret bound. For the variable pool algorithm, when $G(t) = 1$, the pool starts with all of the arms, and gradually eliminates arms from the pool as t increases.

6.2 Discussion on the Yahoo! contest

The motivation of this work comes from a high scoring entry in the Exploration and Exploitation 3 contest, where the goal was to build a better recommender system for Yahoo! Front Page news articles. The competition data is the same data used in our experiments above. These data and its competition setup had several challenging characteristics, including mortal arms (as studied here), broad trends over time in click-through-rate, the inability to access data from the correct distribution in order to cross-validate during the competition, and other complexities. This paper does not aim to handle all of these, but it does describe the key insight that led to increased performance for one of the leading teams, and that insight is precisely the regulation of greed over time (Rudin et al., 2012). Although there were features available for each timestep, this team was not able to successfully use the features to substantially boost performance, and the exploration/exploitation aspects turned out to be more important. Here are the main insights leading to large performance gains, all involving regulating greed over time:

- "Peak grabber": Stop exploration when a good arm appears. Specifically, when the article was clicked 9/100 times, keep showing it and stop exploration all together until the arm's click through rate drops below that of another arm. Since this strategy does not handle the massive global trends we observed in the data, it needed to be modified to the dynamic peak grabber strategy described next.

- "Dynamic peak grabber": Stop exploration when the click through rate of one arm is at least 15% above that of the global click through rate. The global rate was estimated by exponentially weighted moving averages.

- Stop exploring old articles: We can determine approximately how long the arm is likely to stay, and we reduce exploration gradually as the arm gets older.

- Do not fully explore new arms: When a new arm appears, do not use 1 as the upper confidence bound for the probability of click, which would force a UCB algorithm to explore it, use .88 instead. This allows the algorithm to continue exploiting the arms that are known to be good rather than exploring new ones.

The peak grabber strategies inspired the abstracted setting here, where one can think of a good article appearing during periods of high $G(t)$, where we would want to limit exploration; however, the other strategies are also relevant cases where the exploration/exploitation tradeoff is regulated over time. There were no "lock-up" periods in the contest dataset, though as discussed earlier, the $G(t)$ function is also relevant for modeling that setting. The large global trends we observed in the contest data click through rates are very relevant to the $G(t)$ model, since one might want to explore less when the click rate is high in order to get more clicks overall.

7. Conclusions

The dynamic trends we observe in most retail and marketing settings are dramatic. It is possible that understanding these dynamics and how to take advantage of them is central to the success of multi-armed bandit algorithms in a very large class of situations that occur in practice (such as retail). We showed in this work how to adapt algorithms and regret bound analysis to this setting, where we now need to consider not only the average number of times that an arm was pulled in the past, but precisely when the arm was pulled. The key element of our algorithms is that they regulate greed (exploitation) over time, where during high reward periods, less exploration is performed. The algorithms that can regulate greed outperform significantly standard ones, and the results are supported by simulations and a study on the Yahoo! event log dataset that was created specifically for evaluating the performance of multi-armed bandit algorithms. We chose to show how to adapt very well known algorithms to this setting (in theory and in practice), but there are many possible extensions to this work to many other important multi-armed bandit algorithms.

Acknowledgments

We thank team members Ed Su and Virot Ta Chiraphadhanakul for their work on the ICML Exploration and Exploitation 3 contest that inspired the work here. This project was partially funded by the Ford-MIT alliance.

Appendix A. Regret-bound for ε-z greedy algorithm with hard threshold

Proposition 14 is also used in Auer et al. (2002) for the proof of the regret bound for the ε-greedy algorithm.

Proposition 14 *Let us define the following events:*

$$\begin{aligned} A &= \left\{\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{*,T_*(t-1)}\right\}, \\ B &= \left\{\widehat{X}_{*,T_*(t-1)} < \mu_* - \frac{\Delta_j}{2}\right\}, \\ C &= \left\{\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j}{2}\right\}. \end{aligned}$$

Then,

$$A \subset (B \cup C)\,. \tag{31}$$

Intuitively, inclusion (31) means that we play arm j when we underestimate the mean reward of the best arm, or when we overestimate that of arm j. Assume for the sake of contradiction that there exists an element $\omega \in A$ that does not belong to $B \cup C$. Then, we have that $\omega \in (B \cup C)^{\mathcal{C}}$

$$\begin{aligned} \Rightarrow\ \omega &\in \left(\left\{\widehat{X}_{*,T_*(t-1)} < \mu_* - \frac{\Delta_j}{2}\right\} \cup \left\{\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j}{2}\right\}\right)^{\mathcal{C}} \\ \Rightarrow\ \omega &\in \left\{\widehat{X}_{*,T_*(t-1)} \geq \mu_* - \frac{\Delta_j}{2}\right\} \cap \left\{\widehat{X}_{j,T_j(t-1)} \leq \mu_j + \frac{\Delta_j}{2}\right\}. \end{aligned} \tag{32}$$

By definition we have $\mu_* - \frac{\Delta_j}{2} = \mu_* - \frac{\mu_* - \mu_j}{2} = \frac{\mu_* + \mu_j}{2} = \mu_j + \frac{\Delta_j}{2}$. From the inequalities given in (32) it follows that

$$\widehat{X}_{*,T_*(t-1)} \geq \mu_* - \frac{\Delta_j}{2} = \mu_j + \frac{\Delta_j}{2} \geq \widehat{X}_{j,T_j(t-1)},$$

but this contradicts our assumption that $\omega \in A = \left\{\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{*,T_*(t-1)}\right\}$.
Therefore, all elements of A belong to $B \cup C$.

Theorem 1 *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\begin{aligned} \mathbb{E}[R_n] \ &\leq\ \sum_{j=1}^{m} G(j)\Delta_j \\ &+\ \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)<z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \left(\varepsilon_t \frac{1}{m} + (1-\varepsilon_t)\beta_j(\tilde{t})\right) \\ &+\ \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)\geq z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \beta_j(\tilde{t}), \end{aligned}$$

where

$$\beta_j(\tilde{t}) = k\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k}{10}} \log\left(\frac{\tilde{t}}{mke}\right) + \frac{4}{\Delta_j^2}\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k\Delta_j^2}{4}}.$$

First step: Decomposition of $\mathbb{E}[R_n]$.

The total mean regret of a game $\mathcal{I} = \{I_t\}_{t=1}^{n}$ at round n is given by

$$R_n = \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{I_t=j\}}, \tag{33}$$

where $G(t)$ is the greed function evaluated at time t, $\mathbb{1}_{\{I_t=j\}}$ is an indicator function equal to 1 if arm j is played at time t (otherwise its value is 0) and $\Delta_j = \mu^* - \mu_j$ is the difference between the mean of the best arm reward distribution and the mean of the j's arm reward distribution. By considering the threshold z which determines which rule is applied to decide what arm to play, we can rewrite the regret as

$$\begin{aligned} R_n \quad &= \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{G(t)<z\}} \mathbb{1}_{\{I_t=j\}} + \\ &+ \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{G(t)\geq z\}} \mathbb{1}_{\{I_t=j\}}. \end{aligned}$$

By taking the expectation with respect to the policy, we have that

$$\begin{aligned} \mathbb{E}[R_n] \quad &= \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{G(t)<z\}} \mathbb{P}(\{I_t=j\}) + \\ &+ \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{G(t)\geq z\}} \mathbb{P}(\{I_t=j\}), \end{aligned}$$

which can be rewritten as

$$\begin{aligned} \mathbb{E}[R_n] \quad &= \quad \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{G(t)<z\}} \left[\varepsilon_t \frac{1}{m} + (1-\varepsilon_t) \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \ \ \forall i \right) \right] \\ &+ \quad \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{G(t)\geq z\}} \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \ \ \forall i \right). \end{aligned} \tag{34}$$

For the rounds of the algorithm where $G(t) < z$, we are in the standard setting, so for those times, we follow the standard proof of Auer et al. (2002). For the times that $G(t)$ is over the threshold, we need to create a separate bound.

Second step: Upper bound for $\mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \ \ \forall i \right)$.

Let us now bound the probability of playing the sub-optimal arm j at time t when the greed function is above the threshold z. From Proposition 14 we have that

$$\begin{aligned} \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \ \ \forall i \right) \quad &\leq \quad \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{*,T_*(t-1)} \right) \\ &\leq \quad \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j}{2} \right) + \mathbb{P}\left(\widehat{X}_{*,T_*(t-1)} < \mu_* - \frac{\Delta_j}{2} \right). \end{aligned} \tag{35}$$

Let us consider the first term of (35) (the computations for the second term are similar),

$$\begin{aligned} \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j}{2} \right) \quad &= \quad \sum_{s=1}^{t-1} \mathbb{P}\left(T_j(t-1) = s, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2} \right) \\ &= \quad \sum_{s=1}^{t-1} \mathbb{P}\left(T_j(t-1) = s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2} \right) \mathbb{P}\left(\widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2} \right) \\ &\leq \quad \sum_{s=1}^{t-1} \mathbb{P}\left(T_j(t-1) = s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2} \right) e^{-\frac{\Delta_j^2}{2}s}, \end{aligned} \tag{36}$$

where in the last inequality we used Hoeffding's[3] bound.

3. **Hoeffding's bound:** Let $X_1, \cdots, X_n$ be r.v. bounded in $[a_i, b_i]$ $\forall i$. Let $\widehat{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[\widehat{X}]$. Then, $\mathbb{P}\left(\widehat{X} - \mu \geq \varepsilon \right) \leq \exp\left\{ -\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2} \right\}$.

Third step: Upper bound for $\sum_{s=1}^{t-1} \mathbb{P}\left(T_j(t-1) = s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right) e^{-\frac{\Delta_j^2}{2}s}$.

Let us define $T_j^R(t-1)$ as the number of times arm j is played at random when exploring (note that $T_j^R(t-1) \le T_j(t-1)$ and that $T_j^R(t-1) = \sum_{s=1}^{t-1} B_s$ where B_s is a Bernoulli r.v. with parameter ε_s/m), and let us define

$$\lambda_t = \frac{1}{2m}\sum_{s=1}^{\tilde{t}} \varepsilon_s,$$

where $\tilde{t}$ is the number of rounds played under the threshold z up to time t. Then,

$$\begin{aligned}
(36) \quad &\le \sum_{s=1}^{\lfloor \lambda_t \rfloor} \mathbb{P}\left(T_j(t-1) = s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right) + \sum_{s=\lfloor \lambda_t \rfloor+1}^{t-1} e^{-\frac{\Delta_j^2}{2}s} \\
&\le \sum_{s=1}^{\lfloor \lambda_t \rfloor} \mathbb{P}\left(T_j(t-1) = s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right) + \frac{2}{\Delta_j^2} e^{-\frac{\Delta_j^2}{2}\lfloor \lambda_t \rfloor} \\
&\le \sum_{s=1}^{\lfloor \lambda_t \rfloor} \mathbb{P}\left(T_j^R(t-1) \le s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right) + \frac{2}{\Delta_j^2} e^{-\frac{\Delta_j^2}{2}\lfloor \lambda_t \rfloor} \\
&\le \lfloor \lambda_t \rfloor \mathbb{P}\left(T_j^R(t-1) \le \lfloor \lambda_t \rfloor\right) + \frac{2}{\Delta_j^2} e^{-\frac{\Delta_j^2}{2}\lfloor \lambda_t \rfloor} \qquad (37)
\end{aligned}$$

where for the first $\lfloor \lambda_t \rfloor$ terms of the sum we used 1 as upper bound on $e^{-\frac{\Delta_j^2}{2}s}$, and for the remaining terms we used the fact that $\sum_{s=x+1}^{\infty} e^{-ks} \le \frac{1}{k}e^{-kx}$, where in our case $k = \frac{\Delta_j^2}{2}$.

Fourth step: Upper bound for $\mathbb{P}\left(T_j^R(t-1) \le \lfloor \lambda_t \rfloor\right)$.

We have that

$$\mathbb{E}[T_j^R(t-1)] = \frac{1}{m}\sum_{s=1}^{\tilde{t}} \varepsilon_s, \quad Var(T_j^R(t-1)) = \sum_{s=1}^{\tilde{t}} \frac{\varepsilon_s}{m}\left(1 - \frac{\varepsilon_s}{m}\right) \le \frac{1}{m}\sum_{s=1}^{\tilde{t}} \varepsilon_s = \mathbb{E}[T_j^R(t-1)],$$

and, using the Bernstein inequality $\mathbb{P}(S_n \le \mathbb{E}[S_n] - a) \le \exp\{-\frac{a^2/2}{\sigma^2 + a/2}\}$ with $S_n = T_j^R(t-1)$ and $a = \frac{1}{2}\mathbb{E}[T_j^R(t-1)]$,

$$\begin{aligned}
\mathbb{P}(T_j^R(t-1) \le \lfloor \lambda_t \rfloor) &= \mathbb{P}\left(T_j^R(t-1) \le \mathbb{E}[T_j^R(t-1)] - \frac{1}{2}\mathbb{E}[T_j^R(t-1)]\right) \\
&\le \exp\left\{-\frac{\frac{1}{8}(\mathbb{E}[T_j^R(t-1)])^2}{\mathbb{E}[T_j^R(t-1)] + \frac{1}{4}\mathbb{E}[T_j^R(t-1)]}\right\} \\
&= \exp\left\{-\frac{4}{5}\frac{1}{8}\mathbb{E}[T_j^R(t-1)]\right\} = \exp\left\{-\frac{1}{5}\lfloor \lambda_t \rfloor\right\}. \qquad (38)
\end{aligned}$$

Fifth step: Lower bound for $\lfloor \lambda_t \rfloor$.

To get an upper bound for (38), we need a lower bound on $\lfloor \lambda_t \rfloor$. Let us define $n' = \lfloor km \rfloor$ and consider the case when $\tilde{t} \geq n$. The case when $\tilde{t} < n$ leads to an exponential decay of the bound (intuitively, at the beginning of the game the values of ε_s that compose λ_t are still high enough, see Remark 15). Then,

$$\begin{aligned}
\lambda_t &= \frac{1}{2m}\sum_{s=1}^{\tilde{t}} \varepsilon_s \\
&= \frac{1}{2m}\sum_{s=1}^{\tilde{t}} \min\left\{1, \frac{km}{s}\right\} \\
&= \frac{1}{2m}\sum_{s=1}^{n'} 1 + \frac{1}{2m}\sum_{s=n'+1}^{\tilde{t}} \frac{km}{s} \\
&= \frac{n'}{2m} + \frac{1}{2m}\left(\sum_{s=1}^{\tilde{t}} \frac{km}{s} - \sum_{s=1}^{n'} \frac{km}{s}\right) \\
&\geq \frac{n'}{2m} + \frac{k}{2}\big(\log(\tilde{t}+1) - (\log(n') + \log(e)\big) \\
&\geq \frac{k}{2}\log\left(\frac{n'}{m}\frac{1}{k}\right) + \frac{k}{2}\log\left(\frac{\tilde{t}}{n'e}\right) \\
&= \frac{k}{2}\log\left(\frac{\tilde{t}}{mke}\right). \qquad (39)
\end{aligned}$$

Remark 15 *Note that if $\tilde{t}$ (or t in the usual ε-greedy algorithm) was less than n', then we would have $\lambda_t = \tilde{t}/2m$, yielding an exponential decay of the bound on the probability of j being the best arm. To see this, $\tilde{t} < n'$ would imply that, using* (37) *and* (38)*,*

$$(37) \leq \frac{\tilde{t}}{2m}\exp\left\{-\frac{1}{5}\frac{\tilde{t}}{2m}\right\} + \frac{2}{\Delta_j^2}\exp\left\{-\frac{\Delta_j^2}{2}\frac{\tilde{t}}{2m}\right\}.$$

Intuitively, if $\tilde{t} < n'$ then λ_t is already big enough to guarantee an exponential decay. The interesting case is instead when $\tilde{t} \geq n'$, for which we need to provide a lower bound for λ_t to prove that it will still be large enough so that the bound on the probability of choosing a suboptimal arm j is of order $o(1/t)$.

Sixth step: Bringing together all bounding quantities.

Continuing the proof, we obtain a bound on the first term in (35) as follows. Using (39) combined with (38) in (37), we get that

$$\text{first addend of (35)} \leq \frac{k}{2}\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k}{10}}\log\left(\frac{\tilde{t}}{mke}\right) + \frac{2}{\Delta_j^2}\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k\Delta_j^2}{4}}. \qquad (40)$$

Since the computations for the second term in (35) are similar, a bound on $\mathbb{P}(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \;\; \forall i)$ is given by

$$\beta_j(\tilde{t}) = k\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k}{10}}\log\left(\frac{\tilde{t}}{mke}\right) + \frac{4}{\Delta_j^2}\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k\Delta_j^2}{4}}. \qquad (41)$$

We can use this to easily bound $\mathbb{P}(\widehat{X}_{j,T_j(t-1)} \geq \widehat{X}_{i,T_i(t-1)} \;\; \forall i)$ in (34) which yields the following bound on the mean regret at time n (recall that the first m turns are used in the initialization phase, each yielding a regret of $G(j)\Delta_j$):

$$\begin{aligned}
\mathbb{E}[R_n] \;\; &\leq \;\; \sum_{j=1}^{m} G(j)\Delta_j \\
&+ \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)<z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \left(\varepsilon_t \frac{1}{m} + (1-\varepsilon_t)\beta_j(\tilde{t})\right) \\
&+ \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)\geq z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \beta_j(\tilde{t}).
\end{aligned}$$

Appendix B. Regret bound for Soft ε-greedy algorithm

Theorem 4 *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\begin{aligned}\mathbb{E}[R_n] &\le \sum_{j=1}^{m} G(j)\Delta_j \\ &+ \sum_{t=m+1}^{n} G(t) \sum_{j:\mu_j<\mu_*} \Delta_j \left(\varepsilon_t \frac{1}{m} + (1-\varepsilon_t)\beta_j^S(t)\right)\end{aligned}$$

where

$$\beta_j^S(t) = k\left(\frac{\gamma t}{mke}\right)^{-\frac{k}{10}} \log\left(\frac{\gamma t}{mke}\right) + \frac{4}{\Delta_j^2}\left(\frac{\gamma t}{mke}\right)^{-\frac{k\Delta_j^2}{4}}.$$

First step: Derivation of $\mathbb{E}[R_n]$.

The total mean regret of a game $\mathcal{I} = \{I_t\}_{t=1}^n$ at round n is given by

$$R_n = \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{I_t=j\}}, \tag{42}$$

where $G(t)$ is the greed function evaluated at time t, $\mathbb{1}_{\{I_t=j\}}$ is an indicator function equal to 1 if arm j is played at time t, and $\Delta_j = \mu^* - \mu_j$ is the difference between the mean of the best arm reward distribution and the mean of the j's arm reward distribution. By taking the expectation over the policy, we have that

$$\mathbb{E}[R_n] = \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{P}(\{I_t = j\})$$

Second step: Upper bound for $\mathbb{P}(\{I_t = j\})$.

At each round t, arm j is played with probability

$$\frac{\varepsilon_t}{m} + (1-\varepsilon_t)\mathbb{P}\left(\widehat{X}_j > \widehat{X}_i \ \forall i\right),$$

where $\varepsilon_t = \min\left\{\psi(t), \frac{km}{t}\right\}$ and

$$\psi(t) = \frac{\log\left(1 + \frac{1}{G(t)}\right)}{\log\left(1 + \frac{1}{\min_{s\in\{m+1,\cdots,n\}} G(s)}\right)}.$$

Recall that $\gamma = \min_{1\le t\le n} \psi(t)$.
Let us bound the probability $\mathbb{P}(\{I_t = j\})$ of playing the sub-optimal arm j at time t. We have that

$$\begin{aligned}\mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \ \forall i\right) &\le \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{*,T_*(t-1)}\right) \\ &\le \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j}{2}\right) + \mathbb{P}\left(\widehat{X}_{*,T_*(t-1)} < \mu_* - \frac{\Delta_j}{2}\right).\end{aligned} \tag{43}$$

Third step: Upper bound for (43).

For the bound on the two addends in (43), we have identical steps to the proof for Theorem 1, and thus

$$\mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j}{2}\right) \leq \lfloor\lambda_t\rfloor \mathbb{P}\left(T_j^R(t-1) \leq \lfloor\lambda_t\rfloor\right) + \frac{2}{\Delta_j^2} e^{-\frac{\Delta_j^2}{2}\lfloor\lambda_t\rfloor} \tag{44}$$

$$\mathbb{P}\left(\widehat{X}_{*,T_*(t-1)} < \mu_* - \frac{\Delta_j}{2}\right) \leq \lfloor\lambda_t\rfloor \mathbb{P}\left(T_j^R(t-1) \leq \lfloor\lambda_t\rfloor\right) + \frac{2}{\Delta_j^2} e^{-\frac{\Delta_j^2}{2}\lfloor\lambda_t\rfloor} \tag{45}$$

and, similarly to the proof of the ε-greedy algorithm with hard threshold (Appendix A), we have

$$\mathbb{P}\left(T_j^R(t-1) \leq \lfloor\lambda_t\rfloor\right) \leq \exp\left\{-\frac{1}{5}\lfloor\lambda_t\rfloor\right\}. \tag{46}$$

Fourth step: Lower bound for $\lfloor\lambda_t\rfloor$.

Now we need a lower bound on $\lfloor\lambda_t\rfloor$. Let $t > w$ (see Remark 16 for the case $t \leq w$) where $w = \min\{s : \frac{cm}{d^2 s} < \gamma\}$. Then,

$$\begin{aligned}
\lambda_t &= \frac{1}{2m}\sum_{s=1}^{t}\varepsilon_s \\
&= \frac{1}{2m}\sum_{s=1}^{t}\min\left\{\psi(s), \frac{km}{s}\right\} \\
&\geq \frac{1}{2m}\sum_{s=1}^{w}\gamma + \frac{1}{2m}\left(\sum_{s=1}^{t}\frac{km}{s} - \sum_{s=1}^{w}\frac{km}{s}\right) \\
&\geq \frac{w\gamma}{2m} + \frac{k}{2}\left(\log(t+1) - (\log(w) + \log(e)\right) \\
&\geq \frac{k}{2}\log\left(\frac{w\gamma}{m}\frac{1}{k}\right) + \log\left(\frac{t}{we}\right) \\
&= \frac{k}{2}\log\left(\frac{\gamma t}{mke}\right).
\end{aligned} \tag{47}$$

Remark 16 *Simalarly to what noted in Remark 15, if $t \leq w$ then we would have $\lambda_t = \gamma t/2m$, yielding an exponential decay of the bound on the probability of j being the best arm. In fact, for $t \leq w$, using* (46) *combined with* (44) *and* (45)*, we have that*

$$(43) \leq \frac{\gamma t}{2m}\exp\left\{-\frac{1}{5}\frac{\gamma t}{2m}\right\} + \frac{2}{\Delta_j^2}\exp\left\{-\frac{\Delta_j^2}{2}\frac{\gamma t}{2m}\right\}.$$

Intuitively, if $t \leq w$ then λ_t is already big enough to guarantee an exponential decay. The interesting case is instead when $t > w$, for which we need to provide a lower bound for λ_t to prove that it will still be large enough so that the bound on the probability of choosing a suboptimal arm j is of order $o(1/t)$.

Fifth step: Bringing together all bounding quantities.

Using the upper bound for λ_t given in (46) and the lower bound in (47), from (43) the bound on $\mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \;\; \forall i\right)$ is given by

$$\beta_j^S(t) = k \log\left(\frac{\gamma t}{mke}\right)\left(\frac{\gamma t}{mke}\right)^{-\frac{k}{10}} + \frac{4}{\Delta_j^2}\left(\frac{\gamma t}{mke}\right)^{-\frac{k\Delta_j^2}{4}}.$$

Since the mean regret is given by

$$\mathbb{E}[R_n] = \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{P}\left(\{I_t = j\}\right), \tag{48}$$

the bound on the mean regret at time n is given by (recall that the first m turns are used in the initialization phase, each yielding a regret of $G(j)\Delta_j$):

$$\begin{aligned} \mathbb{E}[R_n] \;\; \leq \;\; & \sum_{j=1}^{m} G(j)\Delta_j \\ + \;\; & \sum_{t=m+1}^{n} G(t) \sum_{j=1}^{m} \Delta_j \left(\varepsilon_t \frac{1}{m} + (1-\varepsilon_t)\beta_j^S(t)\right). \end{aligned}$$

Appendix C. Regret bound for the UCB algorithm with hard threshold

Theorem 8 *The bound on the mean regret* $\mathbb{E}[R_n]$ *at time* n *is given by*

$$\begin{aligned}\mathbb{E}[R_n] &\leq \sum_{j=1}^{m} G(j)\Delta_j \\ &+ z\left[8\sum_{j:\mu_j<\mu_*}\left(\frac{\log\tilde{n}}{\Delta_j}\right)+\left(1+\frac{\pi^2}{3}\right)\left(\sum_{j=1}^{m}\Delta_j\right)\right] \\ &+ \sum_{j=1}^{m}\Delta_j\sum_{k=1}^{|B|}\sum_{t\in Y_k} G(t)2\beta_j^U(t).\end{aligned}$$

where

$$\beta_j^U(t)=\frac{2}{\Delta_j^2}e^{-\frac{\Delta_j^2}{2}(x_t-1)}$$

and x_t *is the minimum amount of pulls for each arm at time* t *(see Lemma 6.)*

First step: Decomposition of $\mathbb{E}[R_n]$.

The total mean regret of a game $\mathcal{I}=\{I_t\}_{t=1}^{n}$ at round n is given by

$$R_n=\sum_{t=1}^{n}\sum_{j=1}^{m}\Delta_j G(t)\mathbb{1}_{\{I_t=j\}}, \tag{49}$$

where $G(t)$ is the greed function evaluated at time t, $\mathbb{1}_{\{I_t=j\}}$ is an indicator function equal to 1 if arm j is played at time t (otherwise its value is 0) and $\Delta_j=\mu^*-\mu_j$ is the difference between the mean of the best arm reward distribution and the mean of the j's arm reward distribution. By considering the threshold z which determines which rule is applied to decide what arm to play, we can rewrite the regret as

$$\begin{aligned}R_n &\leq \sum_{j=1}^{m}G(j)\Delta_j \\ &+ \sum_{t=m+1}^{n}\sum_{j=1}^{m}\Delta_j G(t)\mathbb{1}_{\{G(t)<z\}}\mathbb{1}_{\{I_t=j\}} \\ &+ \sum_{t=m+1}^{n}\sum_{j=1}^{m}\Delta_j G(t)\mathbb{1}_{\{G(t)\geq z\}}\mathbb{1}_{\{I_t=j\}}.\end{aligned}$$

Let $B=\{t:G(t-1)<z,G(t)\geq z\}$ be the set of rounds where the high-reward zone is entered. Let us call $y_1,y_2,\cdots,y_B$ the elements of B and order them in increasing order such that $y_1<y_2<\cdots<y_B$. Let us also define for every $k\in\{1,\cdots,|B|\}$ the set $Y_k=\{t:t\geq y_k,G(t)\geq z,t<y_{k+1}\}$ (where $y_{B+1}=n$) of

times in the high-reward period entered at time y_k. By taking the expectation over the policy we have that

$$\begin{aligned}\mathbb{E}[R_n] &\leq \sum_{j=1}^{m} G(j)\Delta_j \\ &+ z\sum_{j=1}^{m}\Delta_j \mathbb{E}\left[T_j(\tilde{n})\right] \\ &+ \sum_{j=1}^{m}\Delta_j \sum_{k=1}^{|B|}\sum_{t\in Y_k} G(t)\mathbb{P}(I_t=j), \end{aligned} \tag{50}$$

where $\tilde{n}$ is the number of turns played when $G(t)$ is under the threshold z at the end of the game, and $\mathbb{E}\left[T_j(\tilde{n})\right] = \mathbb{E}\left[\sum_{t=m+1}^{n} \mathbb{1}_{\{G(t)<z\}}\mathbb{1}_{\{I_t=j\}}\right]$ is the expected number of time arm j is played when $G(t)$ is under the threshold z. For the rounds of the algorithm where $G(t) < z$, we are in the standard setting, so for those times, we follow the standard proof of Auer et al. (2002). For the times that $G(t)$ is over the threshold, we need to create a separate bound. Let us now bound the probability of playing the sub-optimal arm j at time t when the greed function is above the threshold z.

Second step: Upper bound for $\mathbb{P}(\{I_t = j\})$ when $G(t) \geq z$.

From Proposition 14, we have that

$$\begin{aligned}\mathbb{P}(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)}\ \forall i) &\leq \mathbb{P}(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{*,T_*(t-1)}) \\ &\leq \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j}{2}\right) + \mathbb{P}\left(\widehat{X}_{*,T_*(t-1)} < \mu_* - \frac{\Delta_j}{2}\right). \end{aligned} \tag{51}$$

Let us consider the first term of (51) (the computations for the second term are similar), and let us call x_t the minimum number of times an arm is pulled at turn t when $G(t)$ is under the threshold (see Lemma 6). We have that

$$\begin{aligned}\mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j}{2}\right) &= \sum_{s=1}^{t-1}\mathbb{P}\left(T_j(t-1)=s, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right) \\ &= \sum_{s=1}^{t-1}\mathbb{P}\left(T_j(t-1)=s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right)\mathbb{P}\left(\widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right) \\ &\leq \sum_{s=1}^{t-1}\mathbb{P}\left(T_j(t-1)=s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right) e^{-\frac{\Delta_j^2}{2}s} &(52) \\ &= \sum_{s=x_t}^{t-1}\mathbb{P}\left(T_j(t-1)=s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j}{2}\right) e^{-\frac{\Delta_j^2}{2}s} &(53) \\ &\leq \frac{2}{\Delta_j^2}e^{-\frac{\Delta_j^2}{2}(x_t-1)} := \beta_j^U(t), &(54)\end{aligned}$$

where, from (52) to (53), we used the fact that, for $s < x_t$, $T_j(t-1)$ can not be equal to s since x_t is the minimum number of pulls, and for the remaining terms we used the fact that $\sum_{s=x}^{\infty} e^{-ks} \leq \frac{1}{k}e^{-k(x-1)}$, where in our case $k = \frac{\Delta_j^2}{2}$. By symmetric argument, we can bound $\mathbb{P}(\{I_t = j\})$ by $2\beta_j^U(t)$.

Third step: Upper bound for $\mathbb{E}[T_j(\tilde{n})]$ when $G(t) < z$.

When $G(t) < z$, we balance exploration and exploitation by classic UCB. The bound of $\mathbb{E}\left[T_j(\tilde{n})\right]$ follows from the usual bound on the UCB algorithm: for n rounds the UCB algorithm has a mean regret bounded by

$$\sum_{j=1}^{m} \frac{8 \log n}{\Delta_j} + \left(1 + \frac{\pi^2}{3}\right) \sum_{j=1}^{m} \Delta_j.$$

Fourth step: Bringing together all bounding quantities.

The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by (recall that the first m turns are used in the initialization phase, each yielding a regret of $G(j)\Delta_j$):

$$\begin{aligned} \mathbb{E}[R_n] &\leq \sum_{j=1}^{m} G(j)\Delta_j \\ &+ z\left[8 \sum_{j:\mu_j<\mu_*} \left(\frac{\log \tilde{n}}{\Delta_j}\right) + \left(1+\frac{\pi^2}{3}\right)\left(\sum_{j=1}^{m} \Delta_j\right)\right] \\ &+ \sum_{j=1}^{m} \Delta_j \sum_{k=1}^{|B|} \sum_{t \in Y_k} G(t) 2\beta_j^U(t). \end{aligned}$$

Lemma 6 *Suppose the rewards of the arms are bounded in $[a, b]$ and let us define $r = b - a$ the range of the possible rewards. When using the UCB policy for a game consisting of n turns, each arm will be pulled at least x_n times, where*

$$\begin{aligned} x_n &= \max\left\{y \in \mathbb{N} : (m-1)\phi(r, y) \leq n\right\} + 1, \\ \phi(r, y) &= \min\left\{t \geq \tau(r_{ji}, y),\, t \in \mathbb{N} : t > \frac{2\,\log(t)}{r^2 + \frac{2\,\log(t)}{y} - 2r\sqrt{\frac{2\,\log(t)}{y}}}\right\}, \\ \tau(r, y) &= \min\left\{t \in \mathbb{N} : \sqrt{\frac{2\log(t)}{y}} \geq r\right\}. \end{aligned}$$

During the initialization phase, each arm is pulled once, so by turn $t = m$ we have that $T_i(m) = 1\ \forall i$. Let us consider arm j, and $t > m$. After arm j has been played $T_j(t-1)$ times, the algorithm will play arm j for the $(T_j(t-1)+1)$-th time when the following condition is met:

$$\widehat{X}_j + \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} > \widehat{X}_i + \sqrt{\frac{2\,\log(t)}{T_i(t-1)}}\ \ \forall i \neq j.$$

Since rewards are bounded in $[a, b]$, we have that $\widehat{X}_j \geq a$ and $\widehat{X}_i \leq b$. Therefore,

$$a + \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} > b + \sqrt{\frac{2\,\log(t)}{T_i(t-1)}}\ \ \forall i \neq j \ \Rightarrow\ \widehat{X}_j + \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} > \widehat{X}_i + \sqrt{\frac{2\,\log(t)}{T_i(t-1)}}\ \ \forall i \neq j.$$

The algorithm is forced to play arm j at time t when each of the $T_i(t-1)$'s are large enough so that the following holds:

$$a + \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} > b + \sqrt{\frac{2\,\log(t)}{T_i(t-1)}}\ \ \forall i \neq j. \tag{55}$$

We want to solve (55) for $T_i(t-1)$, to see how many times the algorithm is required to pull each arm $i \neq j$ to satisfy (55), which can be rewritten as:

$$\sqrt{\frac{2\,\log(t)}{T_i(t-1)}} < \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} - r \;\; \forall i \neq j. \tag{56}$$

Let us define

$$\tau(r, T_j(t-1)) = \min\left\{ t \in \mathbb{N} : \sqrt{\frac{2\log(t)}{T_j(t-1)}} \geq r \right\}.$$

For $t < \tau(r, T_j(t-1))$, there is no solution to inequality (56) for $T_i(t-1)$ (because the right hand side of (56) is negative), and therefore arm j is not necessarily being pulled in turns $t < \tau(r, T_j(t-1))$.
For $t \geq \tau(r, T_j(t-1))$, for (56) to hold, we need that $\forall i \neq j$

$$\frac{2\,\log(t)}{T_i(t-1)} < r^2 + \frac{2\,\log(t)}{T_j(t-1)} - 2r\sqrt{\frac{2\,\log(t)}{T_j(t-1)}}$$
$$\Leftrightarrow \quad T_i(t-1) > \frac{2\,\log(t)}{r^2 + \frac{2\,\log(t)}{T_j(t-1)} - 2r\sqrt{\frac{2\,\log(t)}{T_j(t-1)}}}.$$

Let us define

$$\phi(r, T_j(t-1)) = \min\left\{ t \geq \tau(r, T_j(t-1)),\, t \in \mathbb{N} : t > \frac{2\,\log(t)}{r^2 + \frac{2\,\log(t)}{T_j(t-1)} - 2r\sqrt{\frac{2\,\log(t)}{T_j(t-1)}}} \right\},$$

which can be interpreted as the minimum number of times to pull arm i such that (56) is satisfied if there were only two arms in the game (arm j and arm i).
We will show that the algorithm will have pulled arm j for the $(T_j(t-1)+1)$-th time before or at round

$$(m-1)\phi(r, T_j(t-1)).$$

In order to prove this, assume for the sake of contradiction that the algorithm has played the j-th arm $T_j(t-1)$ times at turn t, and will not play arm j thereafter. Then, there will eventually be a time s_1 such that for some arm i_1 we have

$$T_{i_1}(s_1) = \phi(r, T_j(s_1)).$$

In other words, we will play arm i_1 at least $T_{i_1}(s_1)$ times. By definition of $\phi(r, T_j(s_1))$, this means that

$$\widehat{X}_{i_1} + \sqrt{\frac{2\,\log(s_1)}{T_{i_1}(s_1)}} < \widehat{X}_j + \sqrt{\frac{2\,\log(s_1)}{T_j(s_1)}}.$$

Thus, between arm i_1 and j, the algorithm will always prefer arm j. Since by assumption the algorithm does not play arm j, it will not choose either arm j nor arm i_1 for all the following turns $t \geq s_1$:

$$\widehat{X}_{i_1} + \sqrt{\frac{2\,\log(t)}{T_{i_1}(t)}} < \widehat{X}_j + \sqrt{\frac{2\,\log(t)}{T_j(t)}} \quad \forall t \geq s_1.$$

Therefore, there will be a time s_2 such that, for some arm $i_2 \neq i_1, j$, we have

$$T_{i_2}(s_2) = \phi(r, T_j(s_2)).$$

By the same argument, neither arm i_2 nor i_1 (nor j) can be pulled in the following turns $t \geq s_2$. Since there are $m-1$ arms different from j, this argument repeats $m-1$ times, until we have that

$$T_i(s_i) = \phi(r, T_j(s_i)) \quad \forall i \neq j.$$

Since by assumption the number of times the algorithm pulls arm j never changes, at turn t we have $T_j(s_i) = T_j(t-1)\ \forall i$. Thus, after $(m-1)\phi(r, T_j(t-1))$ pulls, the algorithm will prefer arm j to all the other arms. This means arm j will be pulled the next turn, leading to a contradiction. Thus after at most $(m-1)\phi(r, T_j(t-1))$ turns, the algorithm must have pulled arm j once more.
Now that we can compute the turns at which the algorithm will have pulled arm j for at least the $[T_j(t-1)+1]$th time, we can define the minimum number of times x_n that the algorithm will play arm j during a game of n rounds:

$$x_n = \max\{T_j(t-1) \in \mathbb{N} : (m-1)\phi(r, T_j(t-1)) \leq n\} + 1.$$

Intuitively, the term $\max\{T_j(t-1) \in \{1, \cdots, n\} : (m-1)\phi(r, T_j(t-1)) \leq n\}$ is the maximum number of pulls to arm j such that the next pull is still possible before the game ends. Therefore, by adding one more pull, x_n counts the minimum number of pulls for arm j. Note that this proof holds for each arm, so the lower bound on the number of times the algorithm plays an arm is the same for each arm and depends on the number of arms m, the range r, and the maximum number of turns in the game, n.

Corollary 17 is generalization of this result.

Corollary 17 *Suppose the rewards of arm i are bounded in $[a_i, b_i]$, and let us define $r_{ji} = b_i - a_j$. When using the UCB policy for a game consisting of n turns, each arm j will be pulled at least $x_n(j)$ times, where*

$$\begin{aligned} x_n(j) &= \max\left\{ y \in \mathbb{N} : \sum_{i \neq j} \phi(r_{ji}, y) \leq n \right\} + 1, \\ \phi(r_{ji}, y) &= \min\left\{ t \geq \tau(r_{ji}, y),\, t \in \mathbb{N} : t > \frac{2\,\log(t)}{r_{ji}^2 + \frac{2\,\log(t)}{y} - 2r_{ji}\sqrt{\frac{2\,\log(t)}{y}}} \right\}, \\ \tau(r_{ji}, y) &= \min\left\{ t \in \mathbb{N} : \sqrt{\frac{2\log(t)}{y}} \geq r_{ji} \right\}. \end{aligned}$$

The proof of this corollary closely follows the proof of Lemma 6.
During the initialization phase, each arm is pulled once, so by turn $t = m$ we have that $T_i(m) = 1\ \forall i$. Let us consider arm j, and $t > m$. After arm j has been played $T_j(t-1)$ times, the algorithm will play arm j for the $(T_j(t-1)+1)$-th time when the following condition is met:

$$\widehat{X}_j + \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} > \widehat{X}_i + \sqrt{\frac{2\,\log(t)}{T_i(t-1)}} \ \ \forall i \neq j.$$

Since rewards are bounded in $[a_i, b_i]\ \forall i$, we have that $\widehat{X}_j \geq a_j$ and $\widehat{X}_i \leq b_i$. Therefore,

$$a_j + \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} > b_i + \sqrt{\frac{2\,\log(t)}{T_i(t-1)}} \ \ \forall i \neq j \ \Rightarrow \ \widehat{X}_j + \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} > \widehat{X}_i + \sqrt{\frac{2\,\log(t)}{T_i(t-1)}} \ \ \forall i \neq j.$$

The algorithm is forced to play arm j at time t when each of the $T_i(t-1)$'s are large enough so that the following holds:

$$a_j + \sqrt{\frac{2\,\log(t)}{T_j(t-1)}} > b_i + \sqrt{\frac{2\,\log(t)}{T_i(t-1)}} \ \ \forall i \neq j. \tag{57}$$

We want to solve (57) for $T_i(t-1)$, to see how many times the algorithm is required to pull each arm $i \neq j$ to satisfy (57), which can be rewritten as:

$$\sqrt{\frac{2\ \log(t)}{T_i(t-1)}} < \sqrt{\frac{2\ \log(t)}{T_j(t-1)}} - r_{ji} \ \ \forall i \neq j. \tag{58}$$

Let us define

$$\tau(r_{ji}, T_j(t-1)) = \min\left\{ t \in \mathbb{N} : \sqrt{\frac{2\log(t)}{T_j(t-1)}} \geq r_{ji} \right\}.$$

For $t < \tau(r_{ji}, T_j(t-1))$, there is no solution to inequality (58) for $T_i(t-1)$ (because the right hand side of (58) is negative), and therefore arm j is not necessarily being pulled in turns $t < \tau(r_{ji}, T_j(t-1))$.

For $t \geq \tau(r_{ji}, T_j(t-1))$, for (58) to hold, we need that $\forall i \neq j$

$$\begin{aligned} \frac{2\ \log(t)}{T_i(t-1)} &< r_{ji}^2 + \frac{2\ \log(t)}{T_j(t-1)} - 2r_{ji}\sqrt{\frac{2\ \log(t)}{T_j(t-1)}} \\ \Leftrightarrow \quad T_i(t-1) &> \frac{2\ \log(t)}{r_{ji}^2 + \frac{2\ \log(t)}{T_j(t-1)} - 2r_{ji}\sqrt{\frac{2\ \log(t)}{T_j(t-1)}}}. \end{aligned}$$

Let us define

$$\phi(r_{ji}, T_j(t-1)) = \min\left\{ t \geq \tau(r_{ji}, T_j(t-1)),\ t \in \mathbb{N}\ : t > \frac{2\ \log(t)}{r_{ji}^2 + \frac{2\ \log(t)}{T_j(t-1)} - 2r_{ji}\sqrt{\frac{2\ \log(t)}{T_j(t-1)}}} \right\}.$$

We will show that the algorithm will have pulled arm j for the $(T_j(t-1)+1)$-th time before or at round

$$\sum_{i \neq j} \phi(r_{ji}, T_j(t-1)).$$

In order to prove this, assume for the sake of contradiction that the algorithm has played the j-th arm $T_j(t-1)$ times at turn t, and will not play arm j thereafter. Then, there will eventually be a time s_1 such that for some arm i_1 we have

$$T_{i_1}(s_1) = \phi(r_{ji_1}, T_j(s_1)).$$

In other words, we will play arm i_1 at least $T_{i_1}(s_1)$ times. By definition of $\phi(r_{ji_1}, T_j(s_1))$, this means that

$$\widehat{X}_{i_1} + \sqrt{\frac{2\ \log(s_1)}{T_{i_1}(s_1)}} < \widehat{X}_j + \sqrt{\frac{2\ \log(s_1)}{T_j(s_1)}}.$$

Thus, between arm i_1 and j, the algorithm will always prefer arm j. Since by assumption the algorithm does not play arm j, it will not choose either arm j nor arm i_1 for all the following turns $t \geq s_1$:

$$\widehat{X}_{i_1} + \sqrt{\frac{2\ \log(t)}{T_{i_1}(t)}} < \widehat{X}_j + \sqrt{\frac{2\ \log(t)}{T_j(t)}} \quad \forall t \geq s_1.$$

Therefore, there will be a time s_2 such that, for some arm $i_2 \neq i_1, j$, we have

$$T_{i_2}(s_2) = \phi(r_{ji_2}, T_j(s_2)).$$

By the same argument, neither arm i_2 nor i_1 (nor j) can be pulled in the following turns $t \geq s_2$. This argument repeats for all the arms, until we have that

$$T_i(s_i) = \phi(r_{ji}, T_j(s_i)) \quad \forall i \neq j.$$

Since by assumption the number of times the algorithm pulls arm j never changes, at turn t we have $T_j(s_i) = T_j(t-1)\ \forall i$. Thus, after $\sum_{i\neq j} T_i(s_i) = \sum_{i\neq j} \phi(r_{ji}, T_j(t-1))$ pulls, the algorithm will prefer arm j to all the other arms. This means arm j will be pulled the next turn, leading to a contradiction. Thus after at most $\sum_{i\neq j} \phi(r_{ji}, T_j(t-1))$ turns, the algorithm must have pulled arm j once more.

Now that we can compute the turns at which the algorithm will have pulled arm j for at least the $[T_j(t-1)+1]$th time, we can define the minimum number of times $x_n(j)$ that the algorithm will play arm j during a game of n rounds:

$$x_n(j) = \max\left\{T_j(t-1) \in \mathbb{N} : \sum_{i\neq j} \phi(r_{ji}, T_j(t-1)) \leq n\right\} + 1.$$

Intuitively, the term $\max\left\{T_j(t-1) \in \mathbb{N} : \sum_{i\neq j} \phi(r_{ji}, T_j(t-1)) \leq n\right\}$ is the maximum number of pulls to arm j such that the next pull is still possible before the game ends. Therefore, by adding one more pull, $x_n(j)$ counts the minimum number of pulls for arm j.

Theorem 7 *Suppose the rewards of the arms are bounded in $[a, b]$ and let us define $r = b - a$ the range of the possible rewards. Let x_n be the minimum number of times each arm will be pulled when using the UCB policy for a game consisting of n turns ($n > m$). Then,*

$$x_n \in \Omega(\log(n)).$$

In other words, the minimum number of times the algorithm pulls an arm in a game of n turns grows at least logarithmically[4] *in n.*

Recall that

$$x_n = \max\{y \geq 1,\ y \in \mathbb{N} : (m-1)\phi(r, y) \leq n\} + 1,$$

$$\phi(r, y) = \min\left\{t \geq \tau(r, y),\ t \in \mathbb{N} : t > \frac{2\log(t)}{r^2 + \frac{2\log(t)}{y} - 2r\sqrt{\frac{2\log(t)}{y}}}\right\},$$

and

$$\tau(r, y) = \min\left\{t \in \mathbb{N} : \sqrt{\frac{2\log(t)}{y}} \geq r\right\}.$$

Let us define the following:

$$\begin{aligned} c &> \left(\frac{\sqrt{2}}{r} + 1\right)^2, \quad c \in \mathbb{R}^+, \\ q_n &= \frac{1}{n}\left\lfloor \frac{n}{m-1} \right\rfloor, \quad q_n \in \mathbb{Q}^+ \\ \alpha_n &= \frac{c\, r^2}{\beta \log_n(q_n) + 2\beta}, \quad \alpha_n \in \mathbb{R}^+, \quad 0 < \beta < \frac{1}{2}. \end{aligned}$$

To show that $x_n \in \Omega(\log(n))$, let us choose

$$y = \left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor$$

4. $f(n) \in \Omega(g(n))$ if $\exists k > 0\ \exists n_0$: $\forall n > n_0\ f(n) \geq k g(n)$. In our case, $k = \frac{2\beta}{c\, r^2}$ and $n_0 = m + 1$.

and prove that $(m-1)\phi(r,y) \leq n$ (first, second, and third step). This implies that $x_n \geq y+1 \geq \frac{2\log(n)}{\alpha_n}$. Then, by finding an upper bound on α_n that does not depend on n (fourth step), we conclude that $x_n \in \Omega(\log(n))$.

First step: $\tau(r,y) \leq \left\lceil n^{\frac{1}{\alpha_n} r^2} \right\rceil$

By definition,

$$\tau(r,y) = \tau\left(r, \left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor\right) = \min\left\{ t \in \mathbb{N} : \sqrt{\frac{2\log(t)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor}} \geq r \right\} \leq \min\left\{ t \in \mathbb{N} : \sqrt{\frac{\alpha_n \log(t)}{\log(n)}} \geq r \right\}.$$

For $s \in \mathbb{R}^+$, we have that

$$\begin{aligned} & \sqrt{\frac{\alpha_n \log(s)}{\log(n)}} \geq r \\ \Rightarrow \quad & \log(s) \geq \frac{1}{\alpha_n} r^2 \log(n) \\ \Rightarrow \quad & s \geq n^{\frac{1}{\alpha_n} r^2} \\ \Rightarrow \quad & \min\left\{ t \in \mathbb{N} : \sqrt{\frac{\alpha_n \log(t)}{\log(n)}} \geq r \right\} = \left\lceil n^{\frac{1}{\alpha_n} r^2} \right\rceil \\ \Rightarrow \quad & \tau(r,y) \leq \left\lceil n^{\frac{1}{\alpha_n} r^2} \right\rceil. \end{aligned}$$

Second step: $\phi(r,y) \leq \left\lceil n^{\frac{1}{\alpha_n} c r^2} \right\rceil$

By definition,

$$\phi(r,y) = \phi\left(r, \left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor\right) \leq \min\left\{ t \geq \left\lceil n^{\frac{1}{\alpha_n} r^2} \right\rceil, t \in \mathbb{N} : t > \frac{2\log(t)}{r^2 + \frac{2\log(t)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor} - 2r\sqrt{\frac{2\log(t)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor}}} \right\},$$

where the inequality follows from the fact that are using an upper bound on $\tau(r,y)$.
Let us prove that

$$\min\left\{ t \geq \left\lceil n^{\frac{1}{\alpha_n} r^2} \right\rceil, t \in \mathbb{N} : t > \frac{2\log(t)}{r^2 + \frac{2\log(t)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor} - 2r\sqrt{\frac{2\log(t)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor}}} \right\} \leq \left\lceil n^{\frac{1}{\alpha_n} c r^2} \right\rceil. \tag{59}$$

Let us call

$$t > \frac{2\log(t)}{r^2 + \frac{2\log(t)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor} - 2r\sqrt{\frac{2\log(t)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor}}} \tag{60}$$

the "inequality constraint". Let us set $t = \left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil$ and prove that the inequality constraint is satisfied. We have that

$$\frac{2\,\log\left(\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil\right)}{r^2 + \frac{2\,\log\left(\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil\right)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor} - 2r\sqrt{\frac{2\,\log\left(\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil\right)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor}}} = \frac{2\,\log\left(\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil\right)}{\left(\sqrt{\frac{2\,\log\left(\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil\right)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor}} - r\right)^2}. \tag{61}$$

Since

$$\sqrt{\frac{2\,\log\left(\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil\right)}{\left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor}} - r \geq \sqrt{\frac{2\,\log\left(n^{\frac{1}{\alpha_n} c\, r^2}\right)}{\frac{2\log(n)}{\alpha_n}}} - r = r\sqrt{c} - r > r\left(\frac{\sqrt{2}}{r} + 1\right) - r > \sqrt{2},$$

then,

$$(61) < \log\left(\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil\right) \leq \left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil.$$

Thus, $\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil$ is an integer that satisfies the inequality constraint (60). Therefore, the minimal integer $t \geq \left\lceil n^{\frac{1}{\alpha_n} r^2} \right\rceil$ satisfying the inequality constraint (60) must be less or equal to $\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil$, proving (59) and concluding the proof that $\phi(r, y) \leq \left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil$, where $y = \left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor$.

Third step: $(m-1)\phi(r, y) < n$

Using the upper bound on $\phi(r, y)$ from the previous step, we have that, for $y = \left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor$,

$$\begin{aligned}
(m-1)\phi(r,y) &\leq (m-1)\left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil \\
&= (m-1)\left\lceil n^{\frac{\beta \log_n(q_n) + 2\beta}{c\, r^2} c\, r^2} \right\rceil \\
&= (m-1)\left\lceil n^{\frac{\beta \log_n\left(\frac{1}{n}\left\lfloor \frac{n}{m-1} \right\rfloor\right) + 2\beta}{c\, r^2} c\, r^2} \right\rceil \\
&= (m-1)\left\lceil \left(\frac{1}{n}\left\lfloor \frac{n}{m-1} \right\rfloor\right)^{\beta} n^{2\beta} \right\rceil \\
&= (m-1)\left\lceil n^{2\beta-1}\left\lfloor \frac{n}{m-1} \right\rfloor^{\beta} \right\rceil \\
&< (m-1)\left\lceil \left\lfloor \frac{n}{m-1} \right\rfloor \right\rceil \qquad (62)\\
&= (m-1)\left\lfloor \frac{n}{m-1} \right\rfloor \\
&\leq n,
\end{aligned}$$

where in (62) we used the fact that $2\beta - 1 < 0$ (therefore, $n^{2\beta-1} < 1$), and that $n/(m-1) > 1$ (because $n > m$, and therefore $\lfloor n/(m-1) \rfloor^{\beta} < \lfloor n/(m-1) \rfloor$).

Fourth step: $-1 \leq \log_n(q_n) < 0$

Since $n > m$,

$$q_n = \frac{1}{n}\left\lfloor \frac{n}{m-1} \right\rfloor \geq \frac{1}{n}$$
$$\Rightarrow \quad \log_n(q_n) \geq \log_n\left(\frac{1}{n}\right) = -1.$$

We also have that

$$q_n = \frac{1}{n}\left\lfloor \frac{n}{m-1} \right\rfloor \leq \frac{n}{n(m-1)} = \frac{1}{m-1}$$
$$\Rightarrow \quad \log_n(q_n) < \log_n\left(\frac{1}{m-1}\right) < 0.$$

Fifth step: $\frac{2\log(n)}{\alpha_n} \geq k\log(n), \quad k = \frac{2\beta}{c\,r^2}$

Using $-1 \leq \log_n(q_n) < 0$,

$$\alpha_n = \frac{c\,r^2}{\beta\log_n(q_n) + 2\beta} \leq \frac{c\,r^2}{-\beta + 2\beta} = \frac{c\,r^2}{\beta}$$
$$\Rightarrow \quad \frac{2\log(n)}{\alpha_n} \geq \frac{2\beta}{c\,r^2}\log(n).$$

In conclusion, we showed that

$$x_n \geq y + 1 = \left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor + 1 \geq \frac{2\log(n)}{\alpha_n} \geq \frac{2\beta}{c\,r^2}\log(n).$$

This means that x_n is lower bounded by a quantity that increases logarithmically in n: $x_n \in \Omega(\log(n))$.

Corollary 18 *Suppose the rewards of arm i are bounded in $[a_i, b_i]$ and let us define $r_{ji} = b_i - a_j$ the range of the possible rewards. Let $x_n(j)$ be the minimum number of times each arm will be pulled when using the UCB policy for a game consisting of n turns ($n > m$). Then,*

$$x_n(j) \in \Omega(\log(n)).$$

In other words, the minimum number of times the algorithm pulls arm j in a game of n turns grows at least logarithmically[5] *in n.*

The proof closely follows the proof of Theorem 7, with some adjustments to account for the different r_{ji}.

Recall that

$$x_n(j) = \max\left\{ y \geq 1,\, y \in \mathbb{N} : \sum_{i \neq j} \phi(r_{ji}, y) \leq n \right\} + 1,$$

$$\phi(r_{ji}, y) = \min\left\{ t \geq \tau(r_{ji}, y),\, t \in \mathbb{N} : t > \frac{2\,\log(t)}{r_{ji}^2 + \frac{2\,\log(t)}{y} - 2r_{ji}\sqrt{\frac{2\,\log(t)}{y}}} \right\},$$

$$\tau(r_{ji}, y) = \min\left\{ t \in \mathbb{N} : \sqrt{\frac{2\log(t)}{y}} \geq r_{ji} \right\}.$$

5. $f(n) \in \Omega(g(n))$ if $\exists k > 0\ \exists n_0$: $\forall n > n_0\ f(n) \geq kg(n)$. In our case, $k = (2\beta)/(c\,r^2)$ and $n_0 = m + 1$.

Let us define the following quantities:

$$
\begin{aligned}
r &= \max_{i \neq j} r_{ji} \\
c_i &= \left(\frac{\sqrt{2}}{r_{ji}} + 1 \right)^2, && c_i \in \mathbb{R}^+ \;\; \forall i \neq j \\
c &= \max_{i \neq j} \; c_i \\
q_n &= \frac{1}{n} \left\lfloor \frac{n}{m-1} \right\rfloor, && q_n \in \mathbb{Q}^+ \\
\alpha_{n,i} &= \frac{c\, r_{ji}^2}{\beta \log_n (q_n) + 2\beta}, && \alpha_{n,i} \in \mathbb{R}^+, \;\; \forall i \neq j, \quad 0 < \beta < \frac{1}{2} \\
\alpha_n &= \min_{i \neq j} \; \alpha_{n,i}.
\end{aligned}
$$

To show that $x_n(j) \in \Omega(\log(n))$, let us choose

$$
y = \left\lfloor \frac{2 \log(n)}{\alpha_n} \right\rfloor
$$

and prove that $\sum_{i \neq j} \phi(r_{ji}, y) \leq n$ (first, second, and third step). This implies that $x_n(j) \geq y + 1 \geq \frac{2\log(n)}{\alpha_n}$. Then, by finding an upper bound on α_n that does not depend on n (fourth and fifth step), we conclude that $x_n \in \Omega(\log(n))$.

First step: $\tau(r_{ji}, y) \leq \left\lceil n^{\frac{1}{\alpha_n} r_{ji}^2} \right\rceil \quad \forall i \neq j$

Same proof as in Theorem 7 for each $i \neq j$, with r_{ji}.

Second step: $\phi(r_{ji}, y) \leq \left\lceil n^{\frac{1}{\alpha_n} c\, r_{ji}^2} \right\rceil \quad \forall i \neq j$

Same proof as in Theorem 7 for each $i \neq j$, with r_{ji}, c_i, and $\alpha_{n,i}$ to prove that

$$
\phi(r_{ji}, y) \leq \left\lceil n^{\frac{1}{\alpha_{n,i}} c_i\, r_{ji}^2} \right\rceil.
$$

Then, since $c \geq c_i$, $r \geq r_{ji}$, and $\alpha_n \leq \alpha_{n,i}$ $\forall i \neq j$, it follows that

$$
\phi(r_{ji}, y) \leq \left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil, \;\; \forall i \neq j.
$$

Third step: $\sum_{i \neq j} \phi(r_{ji}, y) < n$

Same proof as in Theorem 7 since

$$
\phi(r_{ji}, y) \leq \left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil \quad \forall i \neq j \quad \Rightarrow \sum_{i \neq j} \phi(r_{ji}, y) \leq (m-1) \left\lceil n^{\frac{1}{\alpha_n} c\, r^2} \right\rceil < n.
$$

Fourth step: $-1 \leq \log_n(q_n) < 0$.

Same proof as in Theorem 7.

Fifth step: $\frac{2\log(n)}{\alpha_n} \geq k\log(n), \quad k = \frac{2\beta}{c\,r^2}$

Using $-1 \leq \log_n(q_n) < 0$, $r \geq r_{ji}$ $\forall j \neq i$

$$\alpha_n = \min_{i\neq j}\{\alpha_{n,i}\} = \min_{i\neq j}\left\{\frac{c\,r_{ji}^2}{\beta\log_n(q_n) + 2\beta}\right\} \leq \frac{c\,r^2}{-\beta+2\beta} = \frac{c\,r^2}{\beta}$$
$$\Rightarrow \quad \frac{2\log(n)}{\alpha_n} \geq \frac{2\beta}{c\,r^2}\log(n).$$

In conclusion, we showed that

$$x_n \geq y + 1 = \left\lfloor \frac{2\log(n)}{\alpha_n} \right\rfloor + 1 \geq \frac{2\log(n)}{\alpha_n} \geq \frac{2\beta}{c\,r^2}\log(n).$$

This means that x_n is lower bounded by a quantity that increases logarithmically in n: $x_n \in \Omega(\log(n))$.

Corollary 19 *Suppose we used a P-UCB policy that plays at turn t arm j with highest upper confidence bound $\widehat{X}_j + \sqrt{\frac{P\ \log(t)}{T_j(t-1)}}$ (the classic UCB is a P-UCB policy with $P = 2$). Then, $x_n \geq \frac{P\beta}{c\,r^2}\log(n)$.*

To prove this define

$$\phi(r_{ji}, y) = \min\left\{t \geq \tau(r_{ji}, y),\, t \in \mathbb{N} : t > \frac{P\ \log(t)}{r_{ji}^2 + \frac{P\ \log(t)}{y} - 2r_{ji}\sqrt{\frac{P\ \log(t)}{y}}}\right\},$$
$$\tau(r_{ji}, y) = \min\left\{t \in \mathbb{N} : \sqrt{\frac{P\log(t)}{y}} \geq r_{ji}\right\},$$
$$c_i = \left(\frac{\sqrt{P}}{r_{ji}} + 1\right)^2, \quad c_i \in \mathbb{R}^+ \ \ \forall i \neq j,$$
$$y = \left\lfloor \frac{P\log(n)}{\alpha_n} \right\rfloor.$$

Appendix D. The regret bound of the Soft UCB algorithm

Proposition 20 *When*

$$\mathbb{1}\left\{\widehat{X}_j+\sqrt{\frac{2\,\log\xi(t)}{T_j(t-1)}}\geq\widehat{X}_*+\sqrt{\frac{2\,\log\xi(t)}{T_*(t-1)}}\right\}\tag{63}$$

is equal to one, at least one of the following has to be true:

$$\begin{aligned}\widehat{X}_* &\leq \mu_*-\sqrt{\frac{2\,\log\xi(t)}{T_*(t-1)}}, &(64)\\ \widehat{X}_j &\geq \mu_j+\sqrt{\frac{2\,\log\xi(t)}{T_j(t-1)}}, &(65)\\ \mu_* &< \mu_j+2\sqrt{\frac{2\,\log\xi(t)}{T_j(t-1)}}. &(66)\end{aligned}$$

Assume for the sake of contradiction that none of them hold simultaneously. Then from (64) we would have that

$$\widehat{X}_*>\mu_*-\sqrt{\frac{2\log\xi(t)}{T_*(t-1)}};$$

then, by applying (66) (with opposite verse since we are assuming it does not hold) we have that

$$\widehat{X}_*>\mu_j+2\sqrt{\frac{2\log\xi(t)}{T_j(t-1)}}-\sqrt{\frac{2\log\xi(t)}{T_*(t-1)}}$$

and then from (65) (again, with opposite verse) follows that

$$\widehat{X}_*>\widehat{X}_j+\sqrt{\frac{2\log\xi(t)}{T_j(t-1)}}-\sqrt{\frac{2\log\xi(t)}{T_*(t-1)}}$$

which is in contradiction with (63).

Proposition 21

$$\begin{aligned}\mathbb{1}\left\{\widehat{X}_j+\sqrt{\frac{2\,\log\xi(t)}{T_j(t-1)}}\geq\widehat{X}_*+\sqrt{\frac{2\,\log\xi(t)}{T_*(t-1)}}\right\} &\leq \mathbb{1}\left\{\widehat{X}_*\leq\mu_*-\sqrt{\frac{2\,\log\xi(t)}{T_*(t-1)}}\right\}\\ &+ \mathbb{1}\left\{\widehat{X}_j\geq\mu_j+\sqrt{\frac{2\,\log\xi(t)}{T_j(t-1)}}\right\} &(67)\end{aligned}$$

We have that

$$\begin{aligned} & \mu_* - \mu_j - 2\sqrt{\frac{2\log\xi(t)}{T_j(t-1)}} \\ \geq\ & \mu_* - \mu_j - 2\sqrt{\frac{2\log\xi(t)}{u}} \\ =\ & \mu_* - \mu_j - \Delta_j \sqrt{\frac{\log\xi(t)}{\left\lceil \log\left(\max_{t\in\{m+1,\cdots,n\}} \xi(t)\right)\right\rceil}} \\ \geq\ & \mu_* - \mu_j - \Delta_j = 0, \end{aligned}$$

therefore, with this choice of u, (66) can not hold. Using Preposition 20 we have the result.

Theorem 9 *Let $S = \{m+1, \ldots, n\}$. The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\begin{aligned} \mathbb{E}[R_n] \quad \leq \quad & \sum_{j=1}^{m} G(j)\Delta_j \\ + \quad & \max_{t\in S} G(t) \left[\sum_{j:\mu_j<\mu_*} \frac{8}{\Delta_j} \log\left(\max_{t\in S}\xi(t)\right) + \sum_{j=1}^{m} \Delta_j \left(1 + \sum_{t=m+1}^{n} \frac{2(t-1-m)^2}{\xi(t)^4}\right)\right]. \end{aligned}$$

First step: Derivation of $\mathbb{E}[R_n]$.

The total mean regret of a game $\mathcal{I} = \{I_t\}_{t=1}^n$ at round n is given by

$$\begin{aligned} R_n \quad = \quad & \sum_{j=1}^{m} G(j)\Delta_j + \sum_{t=m+1}^{n} \sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{I_t=j\}} \\ \leq \quad & \sum_{j=1}^{m} G(j)\Delta_j + \left(\max_{t\in\{m+1,\cdots,n\}} G(t)\right) \sum_{j=1}^{m} \Delta_j \sum_{t=m+1}^{n} \mathbb{1}_{\{I_t=j\}}. \end{aligned}$$

The expected regret $\mathbb{E}[R_n]$ (expectation taken over the policy) at round n is bounded by

$$\mathbb{E}[R_n] \leq \sum_{j=1}^{m} G(j)\Delta_j + \left(\max_{t\in\{m+1,\cdots,n\}} G(t)\right) \sum_{j=1}^{m} \Delta_j \mathbb{E}[T_j(n)]. \tag{68}$$

where $T_j(n) = \sum_{t=1}^n \mathbb{1}_{\{I_t=j\}}$ is the number of times the sub-optimal arm j has been chosen up to round n.

Second step: Upper bound on the probability of overestimating or underestimating the expected reward of arm j by more than $\sqrt{2\log\xi(t)/s_j}$ when $T_j(t-1) = s_j$.

Recall from (1) that

$$\widehat{X}_j = \frac{1}{T_j(t-1)} \sum_{s=1}^{T_j(t-1)} X_j(s).$$

Suppose rewards are bounded[6] in $[0,1]$. From Hoeffding's inequality we have that

$$\mathbb{P}\left(\frac{1}{s_j}\sum_{i=1}^{s_j} X_{j,i} - \mu_j \leq -\varepsilon \,\middle|\, T_j(t-1) = s_j\right) \leq \exp\{-2s_j\varepsilon^2\},$$

6. If rewards are bounded in $[a,b]$, with $r = b-a$, in the following choose $\varepsilon = \sqrt{\frac{2r\,\log\xi(t)}{s_j}}$

and

$$\mathbb{P}\left(\frac{1}{s_j}\sum_{i=1}^{s_j} X_{j,i} - \mu_j \geq \varepsilon \,\middle|\, T_j(t-1) = s_j\right) \leq \exp\{-2s_j\varepsilon^2\}.$$

Let us define the following function:

$$\xi(t) = \left(1 + \frac{t}{G(t)}\right),$$

by selecting $\varepsilon = \sqrt{\frac{2\,\log\xi(t)}{s_j}}$ we have

$$\mathbb{P}\left(\widehat{X}_j + \sqrt{\frac{2\,\log\xi(t)}{s_j}} \leq \mu_j \,\middle|\, T_j(t-1) = s_j\right) \leq \xi(t)^{-4}, \tag{69}$$

and

$$\mathbb{P}\left(\widehat{X}_j - \sqrt{\frac{2\,\log\xi(t)}{s_j}} \geq \mu_j \,\middle|\, T_j(t-1) = s_j\right) \leq \xi(t)^{-4}. \tag{70}$$

Third step: Upper bound on $T_j(n)$ (the number of times that arm j is played by the end of the game).

In order to emphasize the dependence of $\widehat{X}_j$ from $T_j(t-1)$ we will sometimes write $\widehat{X}_{j,T_j(t-1)}$. In the following, notice that in (72) the summation starts from $m+1$ because in the first m initialization rounds each arm is played once. Moreover, step (73) follows from (72) by assuming that arm j has already been played u times. Then, for each t,

$$\left\{\widehat{X}_{j,T_j(t-1)} + \sqrt{\frac{2\log\xi(t)}{T_j(t-1)}} \geq \widehat{X}_{*,T_*(t-1)} + \sqrt{\frac{2\log\xi(t)}{T_*(t-1)}}, T_j(t-1) \geq u\right\} \subset$$

$$\left\{\max_{s_j\in\{u,\dots,T_j(t-1)\}} \widehat{X}_{j,s_j} + \sqrt{\frac{2\log\xi(t)}{s_j}} \geq \min_{s_*\in\{1,\dots,T_*(t-1)\}} \widehat{X}_{*,s_*} + \sqrt{\frac{2\log\xi(t)}{s_*}}\right\} \tag{71}$$

which justifies (75). We also have that (71) is included in

$$\bigcup_{s_*=1}^{T_*(t-1)} \bigcup_{s_j=u}^{T_j(t-1)} \left\{\widehat{X}_{j,s_j} + \sqrt{\frac{2\log\xi(t)}{s_j}} \geq \widehat{X}_{*,s_*} + \sqrt{\frac{2\log\xi(t)}{s_*}}\right\}.$$

Thus, for any integer u, we may write

$$T_j(n) = 1 + \sum_{t=m+1}^{n} \mathbb{1}\{I_t = j\} \tag{72}$$

$$= u + \sum_{t=m+1}^{n} \mathbb{1}\{I_t = j, T_j(t-1) \geq u\} \tag{73}$$

$$= u + \sum_{t=m+1}^{n} \mathbb{1}\left\{\widehat{X}_{j,T_j(t-1)} + \sqrt{\frac{2\log\xi(t)}{T_j(t-1)}} \geq \widehat{X}_{*,T_*(t-1)} + \sqrt{\frac{2\log\xi(t)}{T_*(t-1)}}, T_j(t-1) \geq u\right\} \tag{74}$$

$$\leq u + \sum_{t=m+1}^{n} \mathbb{1}\left\{\max_{s_j\in\{u,\dots,T_j(t-1)\}} \widehat{X}_{j,s_j} + \sqrt{\frac{2\log\xi(t)}{s_j}} \geq \min_{s_*\in\{1,\dots,T_*(t-1)\}} \widehat{X}_{*,s_*} + \sqrt{\frac{2\log\xi(t)}{s_*}}\right\} \tag{75}$$

$$\leq u + \sum_{t=m+1}^{n} \sum_{s_*=1}^{T_*(t-1)} \sum_{s_j=u}^{T_j(t-1)} \mathbb{1}\left\{\widehat{X}_{j,s_j} + \sqrt{\frac{2\log\xi(t)}{s_j}} \geq \widehat{X}_{*,s_*} + \sqrt{\frac{2\log\xi(t)}{s_*}}\right\}. \tag{76}$$

Now, by setting $u = \left\lceil \frac{8}{\Delta_j^2} \log\left(\max_{t \in \{m+1,\cdots,n\}} \xi(t) \right) \right\rceil$ and using Proposition 21, for $T_j(t-1) \geq u$ we have that Thus, using this choice of u and (76), we have that

$$\begin{aligned} T_j(n) &\leq \left\lceil \frac{8}{\Delta_j^2} \log\left(\max_{t \in \{m+1,\cdots,n\}} \xi(t) \right) \right\rceil \\ &+ \sum_{t=m+1}^{n} \sum_{s_*=1}^{T_*(t-1)} \sum_{s_j=u}^{T_j(t-1)} \mathbb{1}\left\{ \widehat{X}_{*,s_*} \leq \mu_* - \sqrt{\frac{2\log \xi(t)}{s_*}} \right\} \\ &+ \sum_{t=m+1}^{n} \sum_{s_*=1}^{T_*(t-1)} \sum_{s_j=u}^{T_j(t-1)} \mathbb{1}\left\{ \widehat{X}_{j,s_j} \geq \mu_j + \sqrt{\frac{2\log \xi(t)}{s_j}} \right\} \end{aligned}$$

Fourth step: Upper bound on $\mathbb{E}[T_j(n)]$ (the expected number of times that arm j is played by the end of the game).

By taking the expected value of $T_j(n)$ and using the result from the **Second step**, we have that

$$\begin{aligned} \mathbb{E}[T_j(n)] &\leq \left\lceil \frac{8}{\Delta_j^2} \log\left(\max_{t \in \{m+1,\cdots,n\}} \xi(t) \right) \right\rceil \\ &+ \sum_{t=m+1}^{n} \sum_{s_*=1}^{T_*(t-1)} \sum_{s_j=u}^{T_j(t-1)} \mathbb{P}\left\{ \widehat{X}_{*,s_*} \leq \mu_* - \sqrt{\frac{2\log \xi(t)}{s_*}} \right\} \\ &+ \sum_{t=m+1}^{n} \sum_{s_*=1}^{T_*(t-1)} \sum_{s_j=u}^{T_j(t-1)} \mathbb{P}\left\{ \widehat{X}_{j,s_j} \geq \mu_j + \sqrt{\frac{2\log \xi(t)}{s_j}} \right\} \\ &\leq \frac{8}{\Delta_j^2} \log\left(\max_{t \in \{m+1,\cdots,n\}} \xi(t) \right) + 1 + 2 \sum_{t=m+1}^{n} \xi(t)^{-4} (t-1-m)^2. \end{aligned}$$

where in the last step we use $(t-1-m)$ as upper bound for $T_*(t-1)$ and $T_j(t-1)$ (cases where we have only played the best arm or arm j).

Fifth step: Determine the upper bound on $\mathbb{E}[R_n]$.

Using (68) and the result from the previous step, we have that

$$\begin{aligned} \mathbb{E}[R_n] &\leq \sum_{j=1}^{m} G(j)\Delta_j \\ &+ \max_{t \in \{m+1,\cdots,n\}} G(t) \left(\sum_{j:\mu_j<\mu_*} \frac{8}{\Delta_j} \log\left(\max_{t \in \{m+1,\cdots,n\}} \xi(t) \right) + \sum_{j=1}^{m} \Delta_j \left[1 + \sum_{t=m+1}^{n} \frac{2(t-1-m)^2}{\xi(t)^4} \right] \right). \end{aligned}$$

Lemma 22 *Suppose the rewards of the arms are bounded in $[a, b]$ and let us define $r = b - a$ the range of the possible rewards. When using the UCB soft policy for a game consisting of n turns and a bounded multiplier function $G(t)$, each arm will be pulled at least x_n times, where*

$$\begin{aligned} x_n &= \max\left\{y \in \mathbb{N} : (m-1)\phi(r, y, G) \le n\right\} + 1, \\ \phi(r, y, G) &= \min\left\{ t \ge \tau(r, y),\, t \in \mathbb{N} : t > \frac{2\,\log(\overline{\xi(t)})}{r^2 + \frac{2\,\log(\underline{\xi(t)})}{y} - 2r\sqrt{\frac{2\,\log(\underline{\xi(t)})}{y}}} \right\}, \\ \tau(r, y, G) &= \min\left\{ t \in \mathbb{N} : \sqrt{\frac{2\log(\underline{\xi(t)})}{y}} \ge r \right\}, \\ \overline{\xi(t)} &= \log\left(1 + \frac{t}{\min_{s>m} G(s)}\right) \quad \textit{and} \quad \underline{\xi(t)} = \log\left(1 + \frac{t}{\max_{s>m} G(s)}\right). \end{aligned}$$

During the initialization phase, each arm is pulled once, so by turn $t = m$ we have that $T_i(m) = 1\ \forall i$. Let us consider arm j, and $t > m$. After arm j has been played $T_j(t-1)$ times, the algorithm will play arm j for the $(T_j(t-1)+1)$-th time when the following condition is met:

$$\widehat{X}_j + \sqrt{\frac{2\,\log(\xi(t))}{T_j(t-1)}} > \widehat{X}_i + \sqrt{\frac{2\,\log(\xi(t))}{T_i(t-1)}}\ \ \forall i \ne j.$$

Since rewards are bounded in $[a, b]$, we have that $\widehat{X}_j \ge a$ and $\widehat{X}_i \le b$. Let us define

$$\overline{\xi(t)} = \log\left(1 + \frac{t}{\min_{s>m} G(s)}\right) \quad \text{and} \quad \underline{\xi(t)} = \log\left(1 + \frac{t}{\max_{s>m} G(s)}\right).$$

Then,

$$a + \sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}} > b + \sqrt{\frac{2\,\log(\overline{\xi(t)})}{T_i(t-1)}}\ \ \forall i \ne j \ \Rightarrow\ \widehat{X}_j + \sqrt{\frac{2\,\log(\xi(t))}{T_j(t-1)}} > \widehat{X}_i + \sqrt{\frac{2\,\log(\xi(t))}{T_i(t-1)}}\ \ \forall i \ne j.$$

The algorithm is forced to play arm j at time t when each of the $T_i(t-1)$'s are large enough so that the following holds:

$$a + \sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}} > b + \sqrt{\frac{2\,\log(\overline{\xi(t)})}{T_i(t-1)}}\ \ \forall i \ne j. \tag{77}$$

We want to solve (77) for $T_i(t-1)$, to see how many times the algorithm is required to pull each arm $i \ne j$ to satisfy (77), which can be rewritten as:

$$\sqrt{\frac{2\,\log(\overline{\xi(t)})}{T_i(t-1)}} < \sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}} - r\ \ \forall i \ne j. \tag{78}$$

Let us define

$$\tau(r, T_j(t-1)) = \min\left\{ t > m : \sqrt{\frac{2\log(\underline{\xi(t)})}{T_j(t-1)}} \ge r \right\}.$$

For $t < \tau(r, T_j(t-1))$, there is no solution to inequality (78) for $T_i(t-1)$ (because the right hand side of (78) is negative), and therefore arm j is not necessarily being pulled in turns $t < \tau(r, T_j(t-1))$.

For $t \geq \tau(r, T_j(t-1))$, for (78) to hold, we need that $\forall i \neq j$

$$\frac{2\,\log(\overline{\xi(t)})}{T_i(t-1)} < r^2 + \frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)} - 2r\sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}}$$

$$\Leftrightarrow \quad T_i(t-1) > \frac{2\,\log(\overline{\xi(t)})}{r^2 + \frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)} - 2r\sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}}}.$$

Let us define

$$\phi(r, T_j(t-1), G) = \min\left\{ t \in \{\tau(r, T_j(t-1)), \cdots, n\} : t > \frac{2\,\log(\overline{\xi(t)})}{r^2 + \frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)} - 2r\sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}}} \right\},$$

which can be interpreted as the minimum number of times to pull arm i such that (78) is satisfied if there were only two arms in the game (arm j and arm i).
We will show that the algorithm will have pulled arm j for the $(T_j(t-1)+1)$-th time before or at round

$$(m-1)\phi(r, T_j(t-1), G).$$

In order to prove this, assume for the sake of contradiction that the algorithm has played the j-th arm $T_j(t-1)$ times at turn t, and will not play arm j thereafter. Then, there will eventually be a time s_1 such that for some arm i_1 we have

$$T_{i_1}(s_1) = \phi(r, T_j(s_1), G).$$

In other words, we will play arm i_1 at least $T_{i_1}(s_1)$ times. By definition of $\phi(r, T_j(s_1), G)$, this means that

$$\widehat{X}_{i_1} + \sqrt{\frac{2\,\log(\overline{\xi(s_1)})}{T_{i_1}(s_1)}} < \widehat{X}_j + \sqrt{\frac{2\,\log(\underline{\xi(s_1)})}{T_j(s_1)}}.$$

Since by assumption the algorithm does not play arm j, it will not choose either arm j nor arm i_1 for all the following turns $t \geq s_1$:

$$\widehat{X}_{i_1} + \sqrt{\frac{2\,\log(\overline{\xi(t)})}{T_{i_1}(t)}} < \widehat{X}_j + \sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t)}} \quad \forall t \geq s_1.$$

Thus, between arm i_1 and j, the algorithm will always prefer arm j. Therefore, there will be a time s_2 such that, for some arm $i_2 \neq i_1, j$, we have

$$T_{i_2}(s_2) = \phi(r, T_j(s_2), G).$$

By the same argument, neither arm i_2 nor i_1 (nor j) can be pulled in the following turns $t \geq s_2$. Since there are $m-1$ arms different from j, this argument repeats $m-1$ times, until we have that

$$T_i(s_i) = \phi(r, T_j(s_i), G) \quad \forall i \neq j.$$

Since by assumption the number of times the algorithm pulls arm j never changes, at turn t we have $T_j(s_i) = T_j(t-1)$ $\forall i$. Thus, after $(m-1)\phi(r, T_j(t-1), G)$ pulls, the algorithm will prefer arm j to all the other arms. This means arm j will be pulled the next turn, leading to a contradiction. Thus after at most $(m-1)\phi(r, T_j(t-1), G)$ turns, the algorithm must have pulled arm j once more.
Now that we can compute the turns at which the algorithm will have pulled arm j for at least the $[T_j(t-1)+1]$th

time, we can define the minimum number of times x_n that the algorithm will play arm j during a game of n rounds:

$$x_n = \max\{T_j(t-1) \in \mathbb{N} : (m-1)\phi(r, T_j(t-1), G) \le n\} + 1.$$

Intuitively, the term $\max\{T_j(t-1) \in \mathbb{N} : (m-1)\phi(r, T_j(t-1), G) \le n\}$ is the maximum number of pulls to arm j such that the next pull is still possible before the game ends. Therefore, by adding one more pull, x_n counts the minimum number of pulls for arm j. Note that this proof holds for each arm, so the lower bound on the number of times the algorithm plays an arm is the same for each arm and depends on the number of arms m, the range r, and the maximum number of turns in the game, n.
Lemma 23 is generalization of this result.

Lemma 23 *Suppose the rewards of arm i are bounded in $[a_i, b_i]$, and let us define $r_{ji} = b_i - a_j$. When using the UCB soft policy for a game consisting of n turns and a bounded multiplier function $G(t)$, each arm will be pulled at least $x_n(j)$ times, where*

$$\begin{aligned}
x_n(j) &= \max\left\{ y \ge 1,\, y \in \mathbb{N} : \sum_{i \ne j} \phi(r_{ji}, y, G) \le n \right\} + 1, \\
\phi(r_{ji}, y, G) &= \min\left\{ t \ge \tau(r_{ji}, y),\, t \in \mathbb{N} : t > \frac{2\log(\overline{\xi(t)})}{r_{ji}^2 + \frac{2\log(\underline{\xi(t)})}{y} - 2r_{ji}\sqrt{\frac{2\log(\underline{\xi(t)})}{y}}} \right\}, \\
\tau(r_{ji}, y, G) &= \min\left\{ t \in \mathbb{N} : \sqrt{\frac{2\log(\underline{\xi(t)})}{y}} \ge r_{ji} \right\}, \\
\overline{\xi(t)} &= \log\left(1 + \frac{t}{\min_{s>m} G(s)}\right) \quad \text{and} \quad \underline{\xi(t)} = \log\left(1 + \frac{t}{\max_{s>m} G(s)}\right).
\end{aligned}$$

The proof of this Lemma closely follows the proof of Lemma 22.
During the initialization phase, each arm is pulled once, so by turn $t = m$ we have that $T_i(m) = 1\ \forall i$. Let us consider arm j, and $t > m$. After arm j has been played $T_j(t-1)$ times, the algorithm will play arm j for the $(T_j(t-1)+1)$-th time when the following condition is met:

$$\widehat{X}_j + \sqrt{\frac{2\log(t)}{T_j(t-1)}} > \widehat{X}_i + \sqrt{\frac{2\log(t)}{T_i(t-1)}}\quad \forall i \ne j.$$

Since rewards are bounded in $[a_i, b_i]\ \forall i$, we have that $\widehat{X}_j \ge a_j$ and $\widehat{X}_i \le b_i$. Let us define

$$\overline{\xi(t)} = \log\left(1 + \frac{1}{\min_{t>m} G(t)}\right) \quad \text{and} \quad \underline{\xi(t)} = \log\left(1 + \frac{1}{\max_{t>m} G(t)}\right).$$

Then,

$$a_j + \sqrt{\frac{2\log(\underline{\xi(t)})}{T_j(t-1)}} > b_i + \sqrt{\frac{2\log(\overline{\xi(t)})}{T_i(t-1)}}\ \forall i \ne j \ \Rightarrow\ \widehat{X}_j + \sqrt{\frac{2\log(\xi(t))}{T_j(t-1)}} > \widehat{X}_i + \sqrt{\frac{2\log(\xi(t))}{T_i(t-1)}}\ \forall i \ne j.$$

The algorithm is forced to play arm j at time t when each of the $T_i(t-1)$'s are large enough so that the following holds:

$$a_j + \sqrt{\frac{2\log(\underline{\xi(t)})}{T_j(t-1)}} > b_i + \sqrt{\frac{2\log(\overline{\xi(t)})}{T_i(t-1)}}\ \forall i \ne j. \tag{79}$$

We want to solve (79) for $T_i(t-1)$, to see how many times the algorithm is required to pull each arm $i \neq j$ to satisfy (79), which can be rewritten as:

$$\sqrt{\frac{2\,\log(\overline{\xi(t)})}{T_i(t-1)}} < \sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}} - r_{ji} \;\; \forall i \neq j. \tag{80}$$

Let us define

$$\tau(r_{ji}, T_j(t-1)) = \min\left\{ t > m : \sqrt{\frac{2\log(\underline{\xi(t)})}{T_j(t-1)}} \geq r_{ji} \right\}.$$

For $t < \tau(r_{ji}, T_j(t-1))$, there is no solution to inequality (80) for $T_i(t-1)$ (because the right hand side of (80) is negative), and therefore arm j is not necessarily being pulled in turns $t < \tau(r_{ji}, T_j(t-1))$.
For $t \geq \tau(r_{ji}, T_j(t-1))$, for (80) to hold, we need that $\forall i \neq j$

$$\begin{aligned} &\frac{2\,\log(\overline{\xi(t)})}{T_i(t-1)} < r_{ji}^2 + \frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)} - 2r_{ji}\sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}} \\ \Leftrightarrow \quad & T_i(t-1) > \frac{2\,\log(\overline{\xi(t)})}{r_{ji}^2 + \frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)} - 2r_{ji}\sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}}}. \end{aligned}$$

Let us define

$$\phi(r_{ji}, T_j(t-1)) = \min\left\{ t \in \{\tau(r_{ji}, T_j(t-1)), \cdots, n\} : t > \frac{2\,\log(\overline{\xi(t)})}{r_{ji}^2 + \frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)} - 2r_{ji}\sqrt{\frac{2\,\log(\underline{\xi(t)})}{T_j(t-1)}}} \right\}.$$

We will show that the algorithm will have pulled arm j for the $(T_j(t-1)+1)$-th time before or at round

$$\sum_{i \neq j} \phi(r_{ji}, T_j(t-1)).$$

In order to prove this, assume for the sake of contradiction that the algorithm has played the j-th arm $T_j(t-1)$ times at turn t, and will not play arm j thereafter. Then, there will eventually be a time s_1 such that for some arm i_1 we have

$$T_{i_1}(s_1) = \phi(r_{ji_1}, T_j(s_1)).$$

In other words, we will play arm i_1 at least $T_{i_1}(s_1)$ times. By definition of $\phi(r_{ji_1}, T_j(s_1))$, this means that

$$\widehat{X}_{i_1} + \sqrt{\frac{2\,\log(\overline{\xi(s_1)})}{T_{i_1}(s_1)}} < \widehat{X}_j + \sqrt{\frac{2\,\log(\underline{\xi(s_1)})}{T_j(s_1)}}.$$

Thus, between arm i_1 and j, the algorithm will always prefer arm j. Since by assumption the algorithm does not play arm j, it will not choose either arm j nor arm i_1 for all the following turns $t \geq s_1$:

$$\widehat{X}_{i_1} + \sqrt{\frac{2\,\log(t)}{T_{i_1}(t)}} < \widehat{X}_j + \sqrt{\frac{2\,\log(t)}{T_j(t)}} \quad \forall t \geq s_1.$$

Therefore, there will be a time s_2 such that, for some arm $i_2 \neq i_1, j$, we have

$$T_{i_2}(s_2) = \phi(r_{ji_2}, T_j(s_2)).$$

By the same argument, neither arm i_2 nor i_1 (nor j) can be pulled in the following turns $t \geq s_2$. This argument repeats for all the arms, until we have that

$$T_i(s_i) = \phi(r_{ji}, T_j(s_i)) \quad \forall i \neq j.$$

Since by assumption the number of times the algorithm pulls arm j never changes, at turn t we have $T_j(s_i) = T_j(t-1)$ $\forall i$. Thus, after $\sum_{i \neq j} T_i(s_i) = \sum_{i \neq j} \phi(r_{ji}, T_j(t-1))$ pulls, the algorithm will prefer arm j to all the other arms. This means arm j will be pulled the next turn, leading to a contradiction. Thus after at most $\sum_{i \neq j} \phi(r_{ji}, T_j(t-1))$ turns, the algorithm must have pulled arm j once more.
Now that we can compute the turns at which the algorithm will have pulled arm j for at least the $[T_j(t-1)+1]$th time, we can define the minimum number of times $x_n(j)$ that the algorithm will play arm j during a game of n rounds:

$$x_n(j) = \max\left\{ T_j(t-1) \in \mathbb{N} : \sum_{i \neq j} \phi(r_{ji}, T_j(t-1)) \leq n \right\} + 1.$$

Intuitively, the term $\max\left\{T_j(t-1) \in \mathbb{N} : \sum_{i \neq j} \phi(r_{ji}, T_j(t-1)) \leq n\right\}$ is the maximum number of pulls to arm j such that the next pull is still possible before the game ends. Therefore, by adding one more pull, $x_n(j)$ counts the minimum number of pulls for arm j.

Appendix E. Regret bound proof for regulating greed with variable arm pool size

Proposition 24 *Let us call $\Delta_j^{*-m_t} = \mu^{*-m_t} - \mu_j$ the difference between the mean reward of the m_tth-best arm and the mean reward of arm j (following definition, $\Delta_j = \Delta_j^{*-1}$ and $\mu^* = \mu^{*-1}$). Let us define the following events:*

$$\begin{aligned} A &= \left\{\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{*-m_t,T_{*-m_t}(t-1)}\right\}, \\ B &= \left\{\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right\}, \\ C &= \left\{\widehat{X}_{*,T_*(t-1)} < \mu^{*-m_t} - \frac{\Delta_j^{*-m_t}}{2}\right\}. \end{aligned}$$

Then,

$$A \subset (B \cup C)\,. \tag{81}$$

Intuitively, inclusion (81) means that we play arm j when we underestimate the mean reward of the m_tth best arm, or when we overestimate that of arm j. Assume for the sake of contradiction that there exists an element $\omega \in A$ that does not belong to $B \cup C$. Then, we have that $\omega \in (B \cup C)^{\mathcal{C}}$

$$\Rightarrow \omega \in \left(\left\{\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right\} \cup \left\{\widehat{X}_{*-m_t,T_{*-m_t}(t-1)} < \mu^{*-m_t} - \frac{\Delta_j^{*-m_t}}{2}\right\}\right)^{\mathcal{C}} \tag{82}$$

$$\Rightarrow \omega \in \left\{\widehat{X}_{j,T_j(t-1)} \leq \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right\} \cap \left\{\widehat{X}_{*-m_t,T_{*-m_t}(t-1)} \geq \mu^{*-m_t} - \frac{\Delta_j^{*-m_t}}{2}\right\}. \tag{83}$$

By definition we have $\mu^{*-m_t} - \frac{\Delta_j^{*-m_t}}{2} = \mu_* - \frac{\mu^{*-m_t} - \mu_j}{2} = \frac{\mu^{*-m_t} + \mu_j}{2} = \mu_j + \frac{\Delta_j^{*-m_t}}{2}$. From the inequalities given in (83) it follows that

$$\widehat{X}_{*-m_t,T_{*-m_t}(t-1)} \geq \mu^{*-m_t} - \frac{\Delta_j^{*-m_t}}{2} = \mu_j + \frac{\Delta_j^{*-m_t}}{2} \geq \widehat{X}_{j,T_j(t-1)},$$

but this contradicts our assumption that $\omega \in A = \left\{\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{*-m_t,T_{*-m_t}(t-1)}\right\}$.
Therefore, all elements of A belong to $B \cup C$.

Theorem 12 *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\mathbb{E}[R_n] \leq \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \frac{2}{m_t}\beta_j,$$

where

$$\beta_j(t) = \gamma \log(t)(t)^{-\gamma/5} + \frac{2}{(\Delta_j^{*-m_t})^2} t^{-\frac{\gamma(\Delta_j^{*-m_t})^2}{2}}.$$

First step: Derivation of $\mathbb{E}[R_n]$.

The total mean regret of a game $\mathcal{I} = \{I_t\}_{t=1}^n$ at round n is given by

$$R_n = \sum_{t=1}^{n}\sum_{j=1}^{m} \Delta_j G(t) \mathbb{1}_{\{I_t = j\}}, \tag{84}$$

where $G(t)$ is the greed function evaluated at time t, $\mathbb{1}_{\{I_t=j\}}$ is an indicator function equal to 1 if arm j is played at time t (otherwise its value is 0) and $\Delta_j = \mu^* - \mu_j$ is the difference between the mean of the best arm reward distribution and the mean of the j's arm reward distribution. Similarly, $\Delta_j^{*-m_t} = \mu^{*-m_t} - \mu_j$ is the difference between the mean reward of the m_tth-best arm and the mean reward of arm j (with this definition we can also write $\Delta_j = \Delta_j^{*-1}$ and $\mu^* = \mu^{*-1}$). By taking the expectation with respect to the policy of (84) we get

$$\mathbb{E}[R_n] = \sum_{t=1}^{n} \frac{G(t)}{m_t} \sum_{j=1}^{m} \Delta_j \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \quad \text{for at least } \; m - m_t \; \text{ indexes } \; i\right), \tag{85}$$

where m_t is the size of the pool of arms at time t, defined by $m_t = \min\left(m, \max\left(1, \left\lfloor \frac{cm}{tG(t)} \right\rfloor\right)\right)$.

Second step: Upper bound on $\mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \quad \text{for at least } \; m - m_t \; \text{ indexes } \; i\right)$.

We have that

$$\begin{aligned}
&\mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{i,T_i(t-1)} \quad \text{for at least } \; m - m_t \; \text{ indexes } \; i\right) \\
&\leq \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \widehat{X}_{*-m_t,T_{*-m_t}(t-1)}\right) \\
&\leq \mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) + \mathbb{P}\left(\widehat{X}_{*,T_*(t-1)} < \mu^{*-m_t} - \frac{\Delta_j^{*-m_t}}{2}\right)
\end{aligned} \tag{86}$$

the last inequality follows from Proposition 24.

Third step: Upper bound for (86).

Let us consider the first term of (86) (the computations for the second term are similar),

$$\begin{aligned}
\mathbb{P}\left(\widehat{X}_{j,T_j(t-1)} > \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) &= \sum_{s=1}^{t-1} \mathbb{P}\left(T_j(t-1) = s, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) \\
&= \sum_{s=1}^{t-1} \mathbb{P}\left(T_j(t-1) = s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) \mathbb{P}\left(\widehat{X}_{j,s} > \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) \\
&\leq \sum_{s=1}^{t-1} \mathbb{P}\left(T_j(t-1) = s \,\middle|\, \widehat{X}_{j,s} > \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) e^{-\frac{s(\Delta_j^{*-m_t})^2}{2}},
\end{aligned} \tag{87}$$

where in the last inequality we used the Hoeffding's bound[7].

Fourth step: Upper bound for (87).

Let us define $T_j^R(t-1)$ as the number of times arm j is played at random when the pool size is full before round t starts, and let us define

$$\lambda_t = \frac{1}{2m} \sum_{s=1}^{t} \mathbb{1}\left\{\min\left(m, \max\left(1, \left\lfloor \frac{cm}{sG(s)} \right\rfloor\right)\right) = m\right\}.$$

7. **Hoeffding's bound:** Let $X_1, \cdots, X_n$ be r.v. bounded in $[a_i, b_i]$ $\forall i$. Let $\widehat{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[\widehat{X}]$. Then, $\mathbb{P}\left(\widehat{X} - \mu \geq \varepsilon\right) \leq \exp\left\{-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right\}$.

Then,

$$\begin{aligned}
(87) \quad &\leq \sum_{s=1}^{\lfloor \lambda_t \rfloor} \mathbb{P}\left(T_j(t-1) = s \middle| \widehat{X}_{j,s} \geq \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) + \sum_{s=\lfloor \lambda_t \rfloor + 1}^{t-1} e^{-\frac{(\Delta_j^{*-m_t})^2}{2} s} \\
&\leq \sum_{s=1}^{\lfloor \lambda_t \rfloor} \mathbb{P}\left(T_j(t-1) = s \middle| \widehat{X}_{j,s} \geq \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) + \frac{2}{\Delta_j^2} e^{-\frac{(\Delta_j^{*-m_t})^2}{2} \lfloor \lambda_t \rfloor} \\
&\leq \sum_{s=1}^{\lfloor \lambda_t \rfloor} \mathbb{P}\left(T_j^R(t-1) \leq s \middle| \widehat{X}_{j,s} \geq \mu_j + \frac{\Delta_j^{*-m_t}}{2}\right) + \frac{2}{\Delta_j^2} e^{-\frac{(\Delta_j^{*-m_t})^2}{2} \lfloor \lambda_t \rfloor} \\
&\leq \lfloor \lambda_t \rfloor \mathbb{P}\left(T_j^R(t-1) \leq \lambda_t\right) + \frac{2}{(\Delta_j^{*-m_t})^2} e^{-\frac{(\Delta_j^{*-m_t})^2}{2} \lfloor \lambda_t \rfloor} \qquad (88)
\end{aligned}$$

where for the first $\lfloor \lambda_t \rfloor$ terms of the sum we upper-bounded $e^{-\frac{\Delta_j^2}{2} s}$ by 1, and for the remaining terms we used the fact that $\sum_{\lfloor \lambda_t \rfloor + 1}^{\infty} e^{-ks} \leq \frac{1}{k} e^{-kx}$, where in our case $k = \frac{\Delta_j^2}{2}$.

Fifth step: Upper bound for $\mathbb{P}\left(T_j^R(t-1) \leq \lambda_t\right)$.

Since $T_j^R(t-1)$ is a sum of $\lambda = \sum_{s=1}^{t} \mathbb{1}\{m_s = m\}$ independent Bernoulli r.v. with parameter $1/m_s$, we have that

$$\begin{aligned}
\mathbb{E}[T_j^R(t-1)] &= \frac{1}{m} \sum_{s=1}^{t} \mathbb{1}\left\{\min\left(m, \max\left(1, \left\lfloor \frac{cm}{sG(s)} \right\rfloor\right)\right) = m\right\} = 2\lambda_t, \\
Var(T_j^R(t-1)) &= \frac{1}{m}\left(1 - \frac{1}{m}\right) \sum_{s=1}^{t} \mathbb{1}\left\{\min\left(m, \max\left(1, \left\lfloor \frac{cm}{sG(s)} \right\rfloor\right)\right) = m\right\} \leq \mathbb{E}[T_j^R(t-1)],
\end{aligned}$$

and, using the Bernstein inequality $\mathbb{P}(S_n \leq \mathbb{E}[S_n] - a) \leq \exp\{-\frac{a^2/2}{\sigma^2 + a/2}\}$ with $S_n = T_j^R(t-1)$ and $a = \frac{1}{2}\mathbb{E}[T_j^R(t-1)]$,

$$\begin{aligned}
\mathbb{P}(T_j^R(t-1) \leq \lambda_t) &= \mathbb{P}\left(T_j^R(t-1) \leq \mathbb{E}[T_j^R(t-1)] - \frac{1}{2}\mathbb{E}[T_j^R(t-1)]\right) \\
&\leq \exp\left\{-\frac{\frac{1}{8}(\mathbb{E}[T_j^R(t-1)])^2}{\mathbb{E}[T_j^R(t-1)] + \frac{1}{4}\mathbb{E}[T_j^R(t-1)]}\right\} \\
&= \exp\left\{-\frac{4}{5}\frac{1}{8}\mathbb{E}[T_j^R(t-1)]\right\} = \exp\left\{-\frac{1}{5}\lambda_t\right\}. \qquad (89)
\end{aligned}$$

In order to bound (89) we need $\lambda_t \geq \gamma \log(t)$ with $\gamma > 5$ so that $\mathbb{P}(T_j^R(t-1) \leq \lambda_t) < t^{-\frac{\gamma}{5}}$.

Remark 11. If $G(t)$ does not satisfy the requirement that $\lambda_t \geq \gamma \log(t)$ it is easy to construct a new multiplier function $G'(t)$ by first finding the set $S = \{t : \lambda_t > \lceil \gamma \log(t-1) \rceil\}$ and then by defining

$$G'(s) = \begin{cases} (c-1)/s & \text{for } s \in \{t, t+1, \cdots, t+2m\} \text{ if } s \in S \\ G(s) & \text{otherwise.} \end{cases}$$

If we do use $G'(t)$ instead of $G(t)$ in the algorithm, the bound holds for $G'(t)$ instead of $G(t)$.

Sixth step: Bringing together all bounding quantities.

We have that (88) is bounded by

$$\beta_j^{VP}(t) = \gamma \log(t)(t)^{-\gamma/5} + \frac{2}{(\Delta_j^{*-m_t})^2} t^{-\frac{\gamma(\Delta_j^{*-m_t})^2}{2}}.$$

The computations for the second term in (87) are similar, therefore

$$\mathbb{E}[R_n] \leq \sum_{t=1}^{n} \sum_{j=1}^{m} \Delta_j G(t) \frac{2}{m_t} \beta_j^{VP}(t).$$

Appendix F. Regret bound proof for the UCB soft mortal algorithm

Proposition 25 *The event*

$$A = \left\{ \widehat{X}_j + \psi_{\textit{future}}(j,t)\sqrt{\frac{2\log \xi_{\textit{present}}(t-s_j)}{T_j(t-1)}} > \widehat{X}_{i_t^*} + \psi_{\textit{future}}(i_t^*,t)\sqrt{\frac{2\log \xi_{\textit{present}}(t-s_{i_t^*})}{T_{i_t^*}(t-1)}} \right\}$$

is included in $B \cup C \cup D$, where

$$\begin{aligned} B &= \left\{ \widehat{X}_{i_t^*} < \mu_{i_t^*} - \psi_{\textit{future}}(i_t^*,t)\sqrt{\frac{2\log \xi_{\textit{present}}(t-s_{i_t^*})}{T_{i_t^*}(t-1)}} \right\} \\ C &= \left\{ \widehat{X}_j > \mu_j + \psi_{\textit{future}}(j,t)\sqrt{\frac{2\log \xi_{\textit{present}}(t-s_j)}{T_j(t-1)}} \right\} \\ D &= \left\{ \mu_{i_t^*} - \mu_j < 2\psi_{\textit{future}}(j,t)\sqrt{\frac{2\log \xi_{\textit{present}}(t-s_j)}{T_j(t-1)}} \right\}. \end{aligned}$$

The inclusion $A \subset (B \cup C \cup D)$ intuitively means that if the algorithm is choosing to play suboptimal arm j at turn t, then it is underestimating the best arm available (event B), or it is overestimating arm j (event C), or it has not pulled enough times arm j to distinguish its performance from the one of arm i_t^* (event D).

For the sake of contradiction let us assume there exists $\omega \in A$ such that $\omega \in (B \cup C \cup D)^{\mathcal{C}}$. Then, for that ω, none of the inequalities that define the events B, C, and D would hold, i.e. (using, in order, the inequality in B, then the one in D, then the one in C):

$$\begin{aligned} \widehat{X}_{i_t^*} &\geq \mu_{i_t^*} - \psi_{\text{future}}(i_t^*,t)\sqrt{\frac{2\log \xi_{\text{present}}(t-s_{i_t^*})}{T_{i_t^*}(t-1)}} \\ &\geq \mu_j + 2\psi_{\text{future}}(j,t)\sqrt{\frac{2\log \xi_{\text{present}}(t-s_j)}{T_j(t-1)}} - \psi_{\text{future}}(i_t^*,t)\sqrt{\frac{2\log \xi_{\text{present}}(t-s_{i_t^*})}{T_{i_t^*}(t-1)}} \\ &\geq \widehat{X}_j + \psi_{\text{future}}(j,t)\sqrt{\frac{2\log \xi_{\text{present}}(t-s_j)}{T_j(t-1)}} - \psi_{\text{future}}(i_t^*,t)\sqrt{\frac{2\log \xi_{\text{present}}(t-s_{i_t^*})}{T_{i_t^*}(t-1)}}, \end{aligned}$$

which contradicts $\omega \in A$.

Proposition 26 *When*

$$T_j(t-1) \geq \left\lceil \frac{8\psi^2_{\textit{future}}(j,t)\log \xi_{\textit{present}}(t-s_j)}{\Delta^2_{j,i_t^*}} \right\rceil$$

event D in Preposition 25 can not happen.

In fact,

$$\begin{aligned}
& \mu_{i_t^*} - \mu_j - 2\psi_{\text{future}}(j,t)\sqrt{\frac{2\log \xi_{\text{present}}(t-s_j)}{T_j(t-1)}} \\
\geq\ & \mu_{i_t^*} - \mu_j - 2\psi_{\text{future}}(j,t)\sqrt{\frac{2\log \xi_{\text{present}}(t-s_j)}{\left\lceil \frac{8\psi^2_{\text{future}}(j,t)\log \xi_{\text{present}}(t-s_j)}{\Delta^2_{j,i_t^*}} \right\rceil}} \\
\geq\ & \mu_{i_t^*} - \mu_j - 2\psi_{\text{future}}(j,t)\sqrt{\frac{\log \xi_{\text{present}}(t-s_j)\Delta^2_{j,i_t^*}}{4\psi^2_{\text{future}}(j,t)\log \xi_{\text{present}}(t-s_j)}} \\
=\ & \mu_{i_t^*} - \mu_j - \Delta_{j,i_t^*} = 0.
\end{aligned}$$

The proof of the following theorem follows steps similar to the proof of the UCB-L algorithm presented in Tracà et al. (2019).

Theorem 13 *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*

$$u_{j,z} = \max_{t\in\{s_j^z,\cdots,l_j^z\}} \left\lceil \frac{8\psi_{\textit{future}}(j,t)\log \xi_{\textit{present}}(t-s_j)}{\Delta^2_{j,z}} \right\rceil, \tag{90}$$

where

$$\Delta_{j,i_t^*} = \Delta_{j,z} \ \ \textit{for } t \in L_j^z. \tag{91}$$

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

$$\mathbb{E}[R_n] \leq \sum_{j\in M_I} G(j)\Delta_{j,i_t^*} + \sum_{j\in M}\sum_{z=1}^{E_j}\left(\max_{t\in E_j} G(t)\right)\Delta_{j,z}\min\left(l_j^z - s_j^z\ ,\ \beta_j^M\right),$$

where

$$\beta_j^M = u_{j,z} + \sum_{\substack{t\in L_j^z\\ t>m_I}} (t-s_{i_t^*})(t-s_j-u_{j,z}+1)\left[\xi_{\textit{present}}(t-s_j)^{-\frac{4}{r^2}\psi_{\textit{future}}(j,t)} + \xi_{\textit{present}}(t-s_{i_t^*})^{-\frac{4}{r^2}\psi_{\textit{future}}(i_t^*,t)}\right]. \tag{92}$$

First step: Decomposition of $\mathbb{E}[R_n]$.

Let us partition the set of steps L_j during which arm j is available into E_j epochs L_j^z, such that

- $\bigcup_{z=1}^{E_j} L_j^z = L_j$,
- $L_j^{z_1} \cap L_j^{z_2} = \emptyset$ if $z_1 \neq z_2$,
- $i_t^* \neq i_s^*$ if $t \in L_j^{z_1}$ and $s \in L_j^{z_2}$ (i.e., if different epochs have different best arm available).

Since during the same epoch the best arm available does not change, let us define

$$\Delta_{j,i_t^*} = \Delta_{j,z} \ \ \text{for } t \in L_j^z, \tag{93}$$

and $s_j^z = \min L_j^z$, $l_j^z = \max L_j^z$ the first and last step of epoch L_j^z.
We have that

$$R_n = \sum_{j\in M_I} G(j)\Delta_{j,i_t^*} + \sum_{j\in M} \sum_{\substack{t\in L_j\\ t>m_I}} G(t)\Delta_{j,i_t^*}\mathbb{1}\{t\in I(j)\} \tag{94}$$

$$\leq \sum_{j\in M_I} \Delta_{j,i_t^*} + \sum_{j\in M} \sum_{z=1}^{E_j} \left(\max_{t\in E_j} G(t)\right) \Delta_{j,z} \sum_{\substack{t\in L_j^z\\ t>m_I}} \mathbb{1}\{t\in I(j)\} \tag{95}$$

Let us call

$$T_j^z(l_j^z) = \sum_{\substack{t\in L_j^z\\ t>m_I}} \mathbb{1}\{t\in I(j)\}$$

the total number of times we choose arm j in epoch z during the game (after initialization). Then, by taking the expectation with respect to the policy of (95) we get

$$\mathbb{E}[R_n] = \sum_{j\in M_I} G(j)\Delta_{j,i_t^*} + \sum_{j\in M} \sum_{z=1}^{E_j} \left(\max_{t\in E_j} G(t)\right) \Delta_{j,z}\, \mathbb{E}\left[T_j^z(l_j^z)\right]. \tag{96}$$

Therefore, finding an upper bound for the expected value of (94) can be accomplished by bounding the expected value of $T_j^z(l_j^z)$.

Second step: Decomposition of $T_j^z(l_j^z)$.

Recall that with $T_j(t-1)$ we indicate the number of times we played arm j before turn t starts. For any integer $u_{j,z}$, we can write

$$\begin{aligned}
T_j^z(l_j^z) &= u_{j,z} + \sum_{\substack{t\in L_j^z\\ t>m_I}} \mathbb{1}\{t\in I(j), T_j(t-1)\geq u_{j,z}\}\\
&= u_{j,z}\\
&+ \sum_{\substack{t\in L_j^z\\ t>m_I}} \mathbb{1}\left\{\widehat{X}_j + \psi_{\text{fut.}}(j,t)\sqrt{\frac{2\log\xi_{\text{pres.}}(t-s_j)}{T_j(t-1)}} > \widehat{X}_{i_t^*} + \psi_{\text{fut.}}(i_t^*,t)\sqrt{\frac{2\log\xi_{\text{pres.}}(t-s_{i_t^*})}{T_{i_t^*}(t-1)}}, T_j(t-1)\geq u_{j,z}\right\}\\
&\leq u_{j,z}\\
&+ \sum_{\substack{t\in L_j^z\\ t>m_I}} \sum_{k_j=u_{j,z}}^{t-s_j} \sum_{k_{i_t^*}=1}^{t-s_{i_t^*}} \mathbb{1}\left\{\widehat{X}_j + \psi_{\text{fut.}}(j,t)\sqrt{\frac{2\log\xi_{\text{pres.}}(t-s_j)}{k_j}} > \widehat{X}_{i_t^*} + \psi_{\text{fut.}}(i_t^*,t)\sqrt{\frac{2\log\xi_{\text{pres.}}(t-s_{i_t^*})}{k_{i_t^*}}}\right\}.
\end{aligned}$$

Therefore we can find an upper bound for the expectation of $T_j^z(l_j^z)$ by finding an upper bound for the probability of the event

$$A = \left\{\widehat{X}_j + \psi_{\text{future}}(j,t)\sqrt{\frac{2\log\xi_{\text{present}}(t-s_j)}{k_j}} > \widehat{X}_{i_t^*} + \psi_{\text{future}}(i_t^*,t)\sqrt{\frac{2\log\xi_{\text{present}}(t-s_{i_t^*})}{k_{i_t^*}}}\right\}.$$

Third step: Upper bound for $\mathbb{E}[T_j^z(l_j^z)]$.

By choosing $u_{j,z} = \max_{t\in\{s_j^z,\cdots,l_j^z\}} \left\lceil \frac{8\psi^2_{\text{future}}(j,t)\log(t-s_j^z)}{\Delta^2_{j,z}} \right\rceil$ and using Proposition 25 and Proposition 26, we have that,

$$A \subset \left(\left\{\widehat{X}_{i_t^*} < \mu_{i_t^*} - \psi_{\text{future}}(i_t^*,t)\sqrt{\frac{2\log\xi_{\text{present}}(t-s_{i_t^*})}{k_{i_t^*}}}\right\} \cup \left\{\widehat{X}_j > \mu_j + \psi_{\text{future}}(j,t)\sqrt{\frac{2\log\xi_{\text{present}}(t-s_j)}{k_j}}\right\}\right). \tag{97}$$

Using Hoeffding's[8] bound we have that

$$\mathbb{P}\left(\widehat{X}_{i_t^*} < \mu_{i_t^*} - \psi_{\text{fut.}}(i_t^*,t)\sqrt{\frac{2\log \xi_{\text{pres.}}(t-s_{i_t^*})}{T_{i_t^*}(t-1)}}\right) \leq \exp\left\{-\frac{2k_{i_t^*}^2\psi_{\text{fut.}}^2(i_t^*,t)\frac{2\log\xi_{\text{pres.}}(t-s_{i_t^*})}{k_{i_t^*}}}{k_{i_t^*}r^2}\right\}$$
$$= \xi_{\text{pres.}}(t-s_{i_t^*})^{-\frac{4}{r^2}\psi_{\text{fut.}}(i_t^*,t)}$$

$$\mathbb{P}\left(\widehat{X}_j > \mu_j + \psi_{\text{fut.}}(j,t)\sqrt{\frac{2\log \xi_{\text{pres.}}(t-s_j)}{T_j(t-1)}}\right) \leq \exp\left\{-\frac{2k_j^2\psi_{\text{fut.}}^2(j,t)\frac{2\log\xi_{\text{pres.}}(t-s_j)}{k_j}}{k_j r^2}\right\}$$
$$= \xi_{\text{pres.}}(t-s_j)^{-\frac{4}{r^2}\psi_{\text{fut.}}(j,t)}.$$

Using the inclusion in (97) in combination with Hoeffding's bounds, we have that

$$\begin{aligned}
&\mathbb{E}\left[T_j^z(l_j^z)\right] \leq u_{j,z} \\
&+ \sum_{\substack{t\in L_j^z\\ t>m_I}} \sum_{k_j=u_{j,z}}^{l_j} \sum_{k_{i_t^*}=1}^{t-s_{i_t^*}} \mathbb{P}\left\{\widehat{X}_j + \psi_{\text{fut.}}(j,t)\sqrt{\frac{2\log\xi_{\text{pres.}}(t-s_j)}{k_j}} > \widehat{X}_{i_t^*} + \psi_{\text{fut.}}(i_t^*,t)\sqrt{\frac{2\log\xi_{\text{pres.}}(t-s_{i_t^*})}{k_{i_t^*}}}\right\} \\
&\leq u_{j,z} + \sum_{\substack{t\in L_j^z\\ t>m_I}} \sum_{k_j=u_{j,z}}^{t-s_j} \sum_{k_{i_t^*}=1}^{t-s_{i_t^*}} \left[\xi_{\text{pres.}}(t-s_{i_t^*})^{-\frac{4}{r^2}\psi_{\text{fut.}}(i_t^*,t)} + \xi_{\text{pres.}}(t-s_j)^{-\frac{4}{r^2}\psi_{\text{fut.}}(j,t)}\right] \\
&= u_{j,z} + \sum_{\substack{t\in L_j^z\\ t>m_I}} (t-s_{i_t^*})(t-s_j-u_{j,z}+1)\left[\xi_{\text{pres.}}(t-s_j)^{-\frac{4}{r^2}\psi_{\text{fut.}}(j,t)} + \xi_{\text{pres.}}(t-s_{i_t^*})^{-\frac{4}{r^2}\psi_{\text{fut.}}(i_t^*,t)}\right]. \qquad (98)
\end{aligned}$$

Fourth step: Upper bound for $\mathbb{E}[R_n]$.

Combining (98) with (96) we get that the bound on the cumulative regret is given by

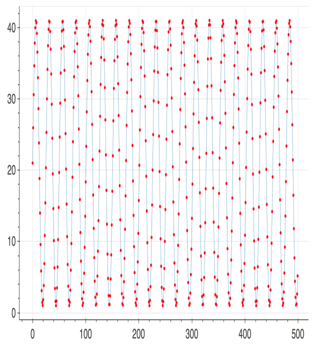
$$\begin{aligned}
\mathbb{E}[R_n] \leq\ & \sum_{j\in M_I} G(j)\Delta_{j,i_t^*} + \sum_{j\in M}\sum_{z=1}^{E_j}\left(\max_{t\in E_j} G(t)\right)\Delta_{j,z}\times\min\Big(l_j^z - s_j^z\ , \\
& u_{j,z} + \sum_{\substack{t\in L_j^z\\ t>m_I}} (t-s_{i_t^*})(t-s_j-u_{j,z}+1)\left[\xi_{\text{pres.}}(t-s_j)^{-\frac{4}{r^2}\psi_{\text{fut.}}(j,t)}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

Notice that if $\psi_{\text{future}}(j,t) = 1$, $G(t) = 1$, $s_j = 0$ and $l_j > n\ \forall j,t$, you can recover the bound of the standard UCB algorithm used in the stochastic case. (Note that you should use $P > 2$ instead of 2 when r is not 1 to create the UCB.)

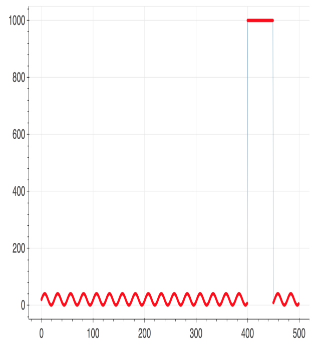
8. **Hoeffding's bound:** Let $X_1,\cdots,X_n$ be r.v. bounded in $[a_i,b_i]\ \forall i$. Let $\widehat{X} = \frac{1}{n}\sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[\widehat{X}]$.
Then, $\mathbb{P}\left(\widehat{X}-\mu\geq\varepsilon\right) \leq \exp\left\{-\frac{2n^2\varepsilon^2}{\sum_{i=1}^n(b_i-a_i)^2}\right\}$.
In our case, n is k_j or $k_{i_t^*}$, $b_i - a_i$ is r, μ is μ_j or $\mu_{i_t^*}$, and ε is $\psi_{\text{future}}(j,t)\sqrt{\frac{2\log\xi_{\text{present}}(t-s_j)}{T_j(t-1)}}$ or $\psi_{\text{future}}(i_t^*,t)\sqrt{\frac{2\log\xi_{\text{present}}(t-s_{i_t^*})}{T_{i_t^*}(t-1)}}$.

Figure 16: Shapes of the multiplier functions used in the experiments.

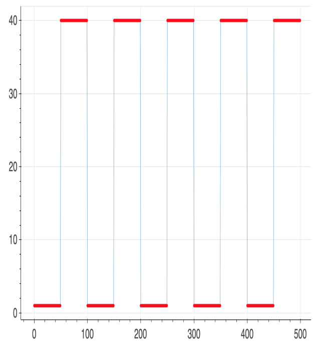
(a) The *Wave Greed*



(b) The *Christmas Greed*



(c) The *Step Greed*



Appendix G. Experiments in a simulated environment

We present plots on the average final rewards in different settings for each type of the greed functions introduced in Section 4 (Wave, Step, and Christmas) and shown again in Figure 16.

The red part of the bars indicate the portion of the final cumulative rewards coming from "pure exploitation". The definition of "pure exploitation" depends on the algorithm used:

- pure exploitation in ε-greedy algorithms: when the algorithms decide to exploit. This is forced in algorithms with threshold when the greed function is above that threshold;
- pure exploitation in UCB algorithms: when the arm played has the highest estimated mean reward. This is forced in algorithms with threshold when the greed function is above that threshold;
- pure exploitation in the variable-pool algorithm: when the pool size is 1.

The rewards are generated from Bernoulli distributions (each arm has probability of success drawn from a Uniform distribution in $[0, 1]$) or from Truncated-Normal distributions (each arm has mean drawn from a Uniform distribution in $[0, 1]$ and standard deviation equal to 1, and the rewards are bounded in $[0, 1]$). The list of figures regarding experiments in various settings that used a Wave-type greed function is reported in Table 7, the list of the ones using a Step-type greed function is reported in Table 8, and the list of the ones using a Christmas-type greed function is reported in Table 9.

We also report the average increase in rewards with respect to the (smarter) versions of the standard ε-greedy algorithm (Algorithm 6) and the standard UCB algorithm (Algorithm 7). The list of figures that show the average percentage increase in rewards is reported in Table 10. In these plots, on the x-axis there are different combinations of number of arms and number of turns played in each game (for example, $100a, 1500t$ means that there were 100 arms and the game had 1500 turns.) On the y-axis is reported the average increase in rewards for each of the algorithms that regulate greed over time. For example, each dot in Figure 29 represents the percentage increase of rewards for a regulating greed over time algorithm compared with the standard ε-greedy algorithm. The reward multiplier function is Wave-type greed function. For instance, the green circle in the upper left indicates that there was a 68% increase in rewards of Soft ε-greedy as compared with smarter ε-greedy for a simulation with 25 arms and 500 rounds. The plot shows that across different numbers of arms, across different rounds, the algorithms that regulate greed over time generally increase rewards by anywhere between 10 and 80%.

Table 7: List of the figures showing the final cumulative rewards when using a Wave-type greed function.

Figure	**Number of arms**	**Number of turns**	**Reward distribution**
17a	25	500	Bernoulli
17b	25	500	Truncated Normal
18a	50	500	Bernoulli
18b	50	500	Truncated Normal
19a	100	500	Bernoulli
19b	100	500	Truncated Normal
20a	200	500	Bernoulli
20b	200	500	Truncated Normal
21a	25	1000	Bernoulli
21b	25	1000	Truncated Normal
22a	50	1000	Bernoulli
22b	50	1000	Truncated Normal
23a	100	1000	Bernoulli
23b	100	1000	Truncated Normal
24a	200	1000	Bernoulli
24b	200	1000	Truncated Normal
25a	25	1500	Bernoulli
25b	25	1500	Truncated Normal
26a	50	1500	Bernoulli
26b	50	1500	Truncated Normal
27a	100	1500	Bernoulli
27b	100	1500	Truncated Normal
28a	200	1500	Bernoulli
28b	200	1500	Truncated Normal

Table 8: List of the figures showing the final cumulative rewards when using a Step-type greed function.

Figure	**Number of arms**	**Number of turns**	**Reward distribution**
33a	25	500	Bernoulli
33b	25	500	Truncated Normal
34a	50	500	Bernoulli
34b	50	500	Truncated Normal
35a	100	500	Bernoulli
35b	100	500	Truncated Normal
36a	200	500	Bernoulli
36b	200	500	Truncated Normal
37a	25	1000	Bernoulli
37b	25	1000	Truncated Normal
38a	50	1000	Bernoulli
38b	50	1000	Truncated Normal
39a	100	1000	Bernoulli
39b	100	1000	Truncated Normal
40a	200	1000	Bernoulli
40b	200	1000	Truncated Normal
41a	25	1500	Bernoulli
41b	25	1500	Truncated Normal
42a	50	1500	Bernoulli
42b	50	1500	Truncated Normal
43a	100	1500	Bernoulli
43b	100	1500	Truncated Normal
44a	200	1500	Bernoulli
44b	200	1500	Truncated Normal

Table 9: List of the figures showing the final cumulative rewards when using a Christmas-type greed function.

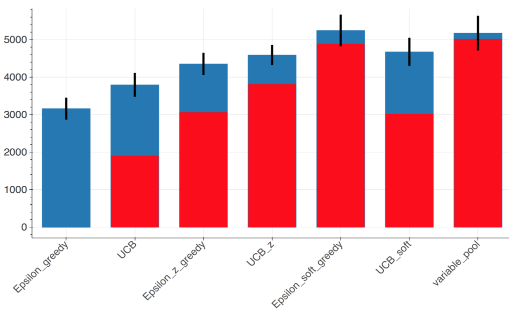
Figure	**Number of arms**	**Number of turns**	**Reward distribution**
49a	25	500	Bernoulli
49b	25	500	Truncated Normal
50a	50	500	Bernoulli
50b	50	500	Truncated Normal
51a	100	500	Bernoulli
51b	100	500	Truncated Normal
52a	200	500	Bernoulli
52b	200	500	Truncated Normal
53a	25	1000	Bernoulli
53b	25	1000	Truncated Normal
54a	50	1000	Bernoulli
54b	50	1000	Truncated Normal
55a	100	1000	Bernoulli
55b	100	1000	Truncated Normal
56a	200	1000	Bernoulli
56b	200	1000	Truncated Normal
57a	25	1500	Bernoulli
57b	25	1500	Truncated Normal
58a	50	1500	Bernoulli
58b	50	1500	Truncated Normal
59a	100	1500	Bernoulli
59b	100	1500	Truncated Normal
60a	200	1500	Bernoulli
60b	200	1500	Truncated Normal

Table 10: List of the figures showing the average increase in rewards with respect to the (smarter) versions of the standard ε-greedy algorithm (Algorithm 6) and the standard UCB algorithm (Algorithm 7).

Figure	**Greed function**	**Baseline**	**Reward distribution**
29	Wave	ε-greedy	Bernoulli
30	Wave	UCB	Bernoulli
31	Wave	ε-greedy	Truncated Normal
32	Wave	UCB	Truncated Normal
45	Step	ε-greedy	Bernoulli
46	Step	UCB	Bernoulli
47	Step	ε-greedy	Truncated Normal
48	Step	UCB	Truncated Normal
61	Christmas	ε-greedy	Bernoulli
62	Christmas	UCB	Bernoulli
63	Christmas	ε-greedy	Truncated Normal
64	Christmas	UCB	Truncated Normal

Figure 17: Comparison of average final rewards in games with 25 arms, 500 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

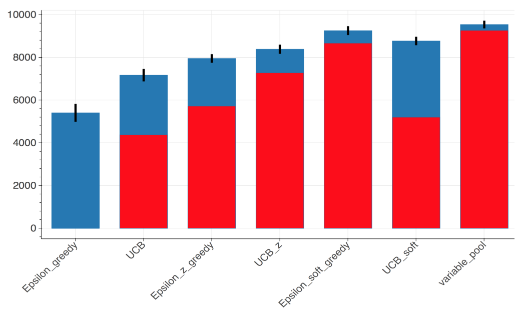
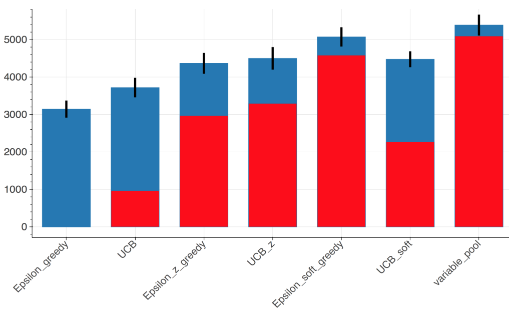


Figure 18: Comparison of average final rewards in games with 50 arms, 500 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

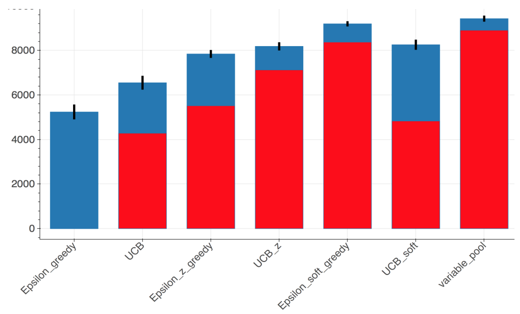
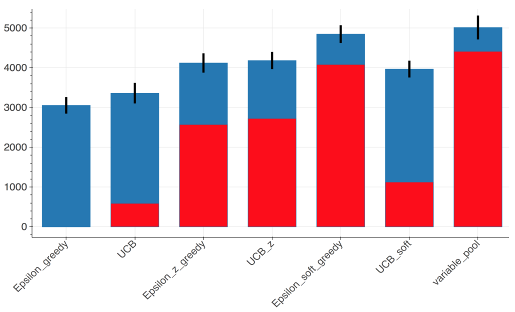


Figure 19: Comparison of average final rewards in games with 100 arms, 500 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

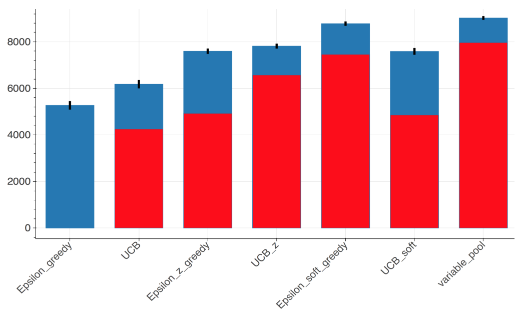
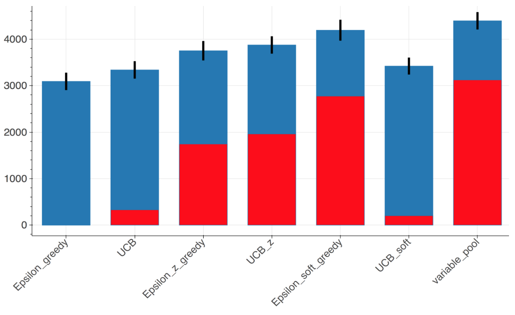


Figure 20: Comparison of average final rewards in games with 200 arms, 500 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

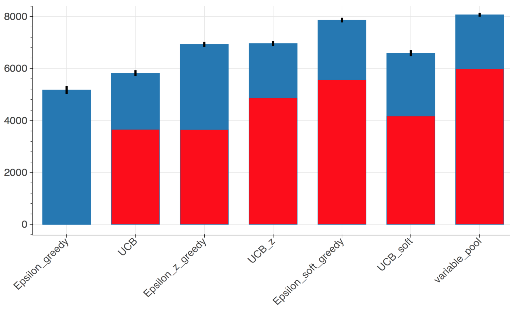
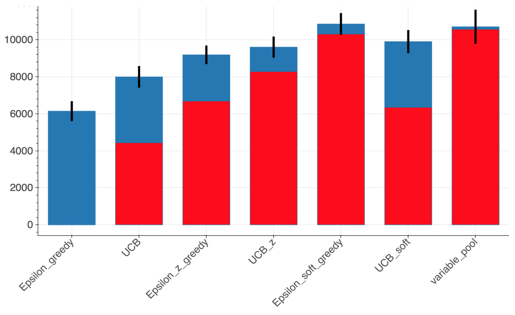


Figure 21: Comparison of average final rewards in games with 25 arms, 1000 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

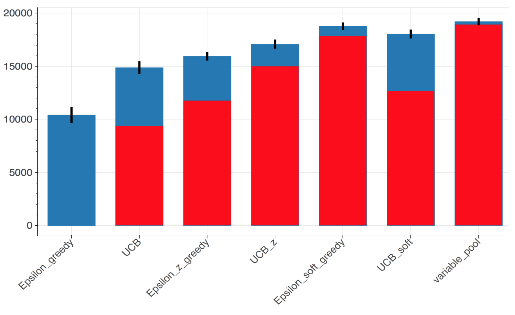
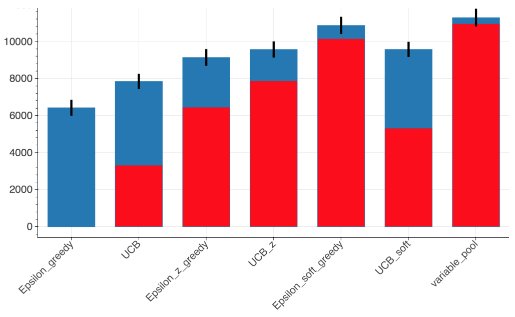


Figure 22: Comparison of average final rewards in games with 50 arms, 1000 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

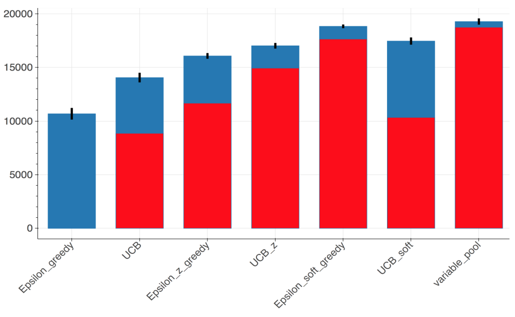
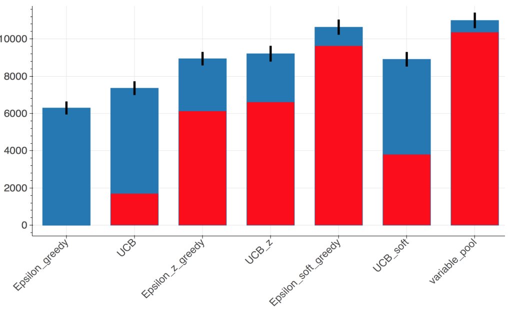


Figure 23: Comparison of average final rewards in games with 100 arms, 1000 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

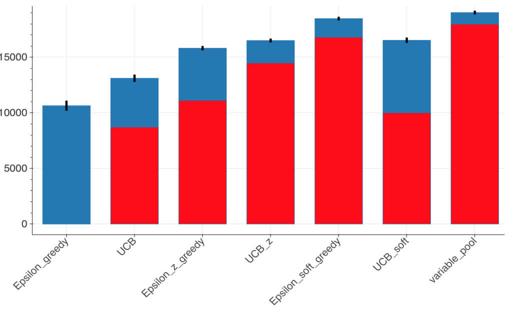
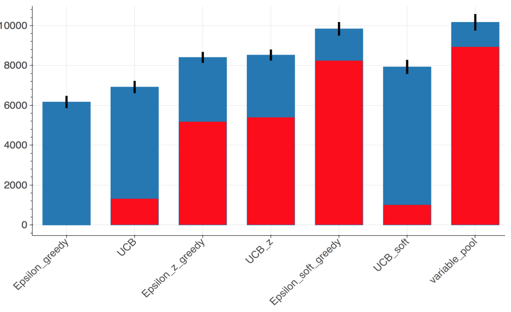


Figure 24: Comparison of average final rewards in games with 200 arms, 1000 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

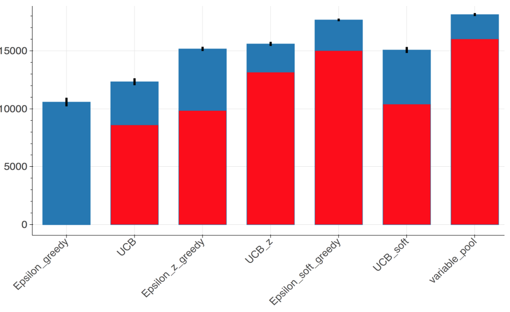
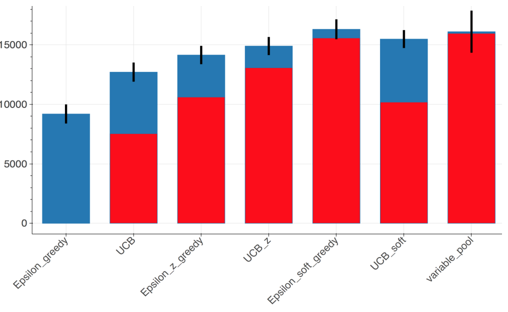


Figure 25: Comparison of average final rewards in games with 25 arms, 1500 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

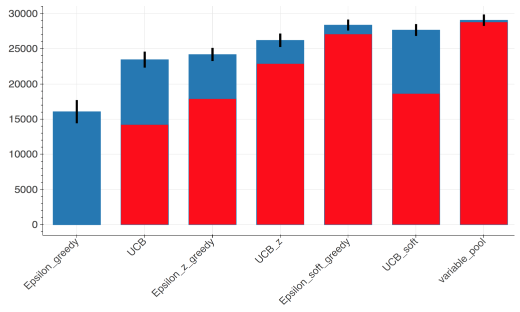
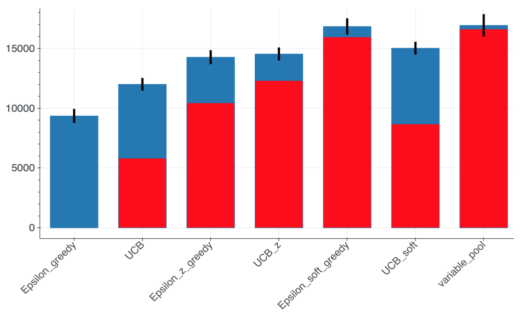


Figure 26: Comparison of average final rewards in games with 50 arms, 1500 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

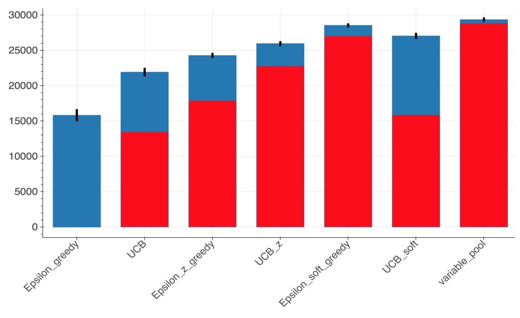
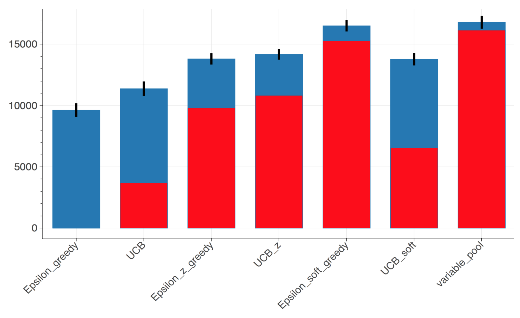


Figure 27: Comparison of average final rewards in games with 100 arms, 1500 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

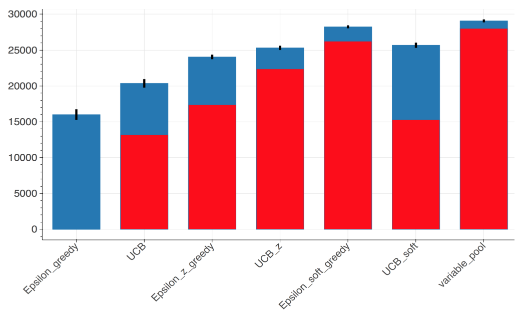
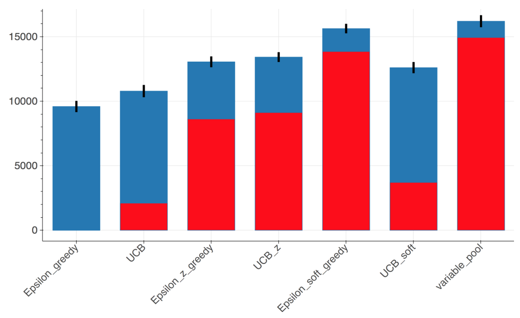


Figure 28: Comparison of average final rewards in games with 200 arms, 1500 turns, and a Wave-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

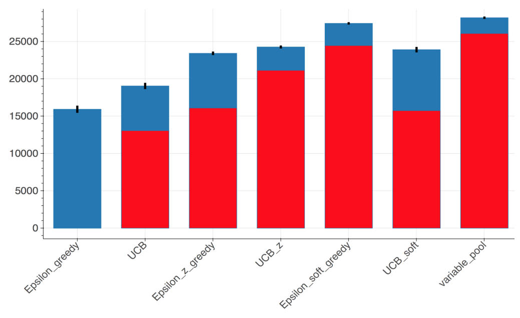


Figure 29: Average increase in rewards (coming from Bernoulli distributions) compared to the (smarter) version of the ε-greedy algorithm (Algorithm 7) with a Wave-type greed function.

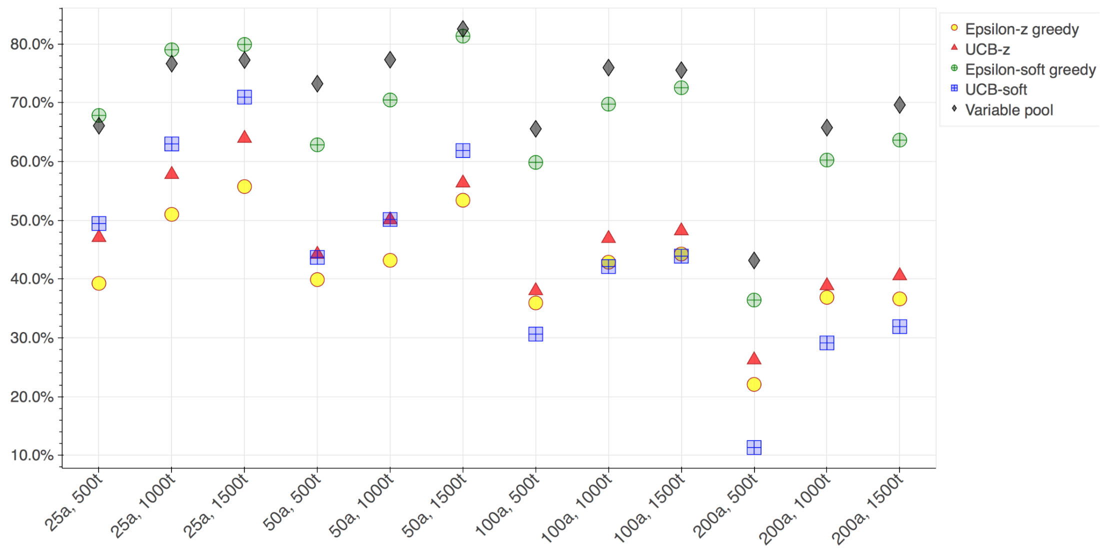


Figure 30: Average increase in rewards (coming from Bernoulli distributions) compared to the (smarter) version of the UCB algorithm (Algorithm 7) with a Wave-type greed function.

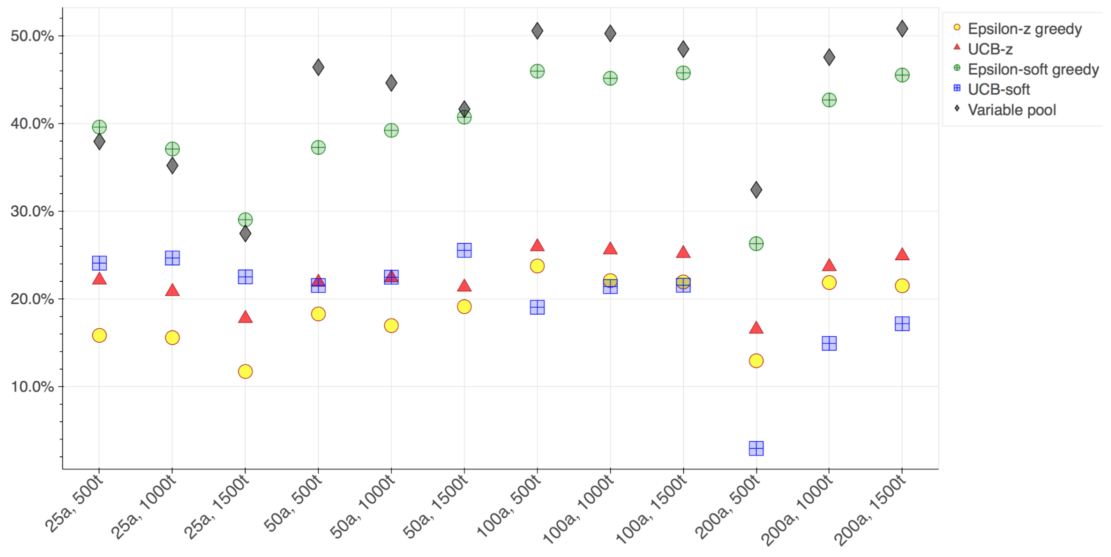


Figure 31: Average increase in rewards (coming from Truncated-Normal distributions) compared to the (smarter) version of the ε-greedy algorithm (Algorithm 6) with a Wave-type greed function.

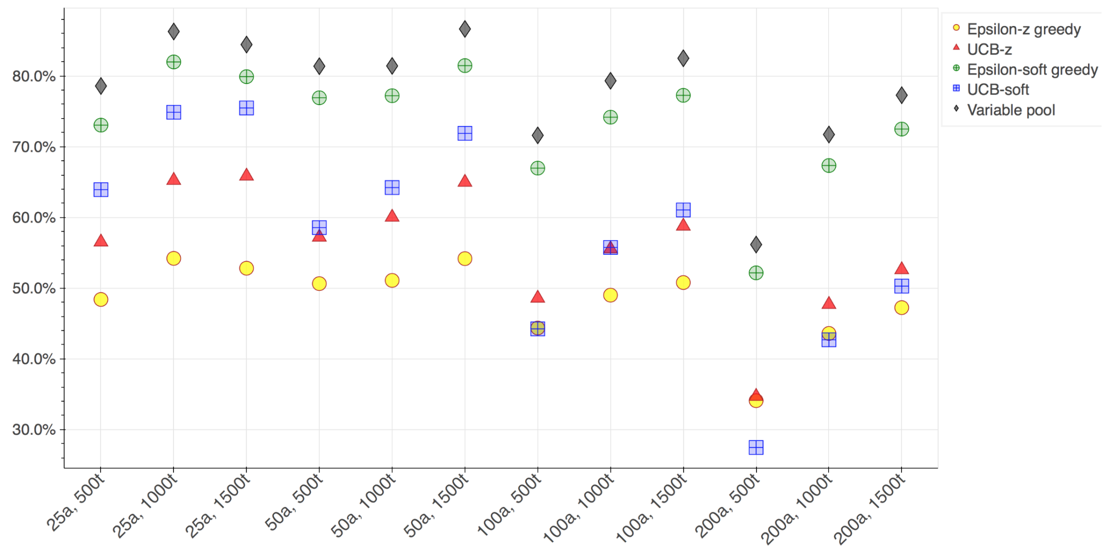


Figure 32: Average increase in rewards (coming from Truncated-Normal distributions) compared to the (smarter) version of the UCB algorithm (Algorithm 7) with a Wave-type greed function.

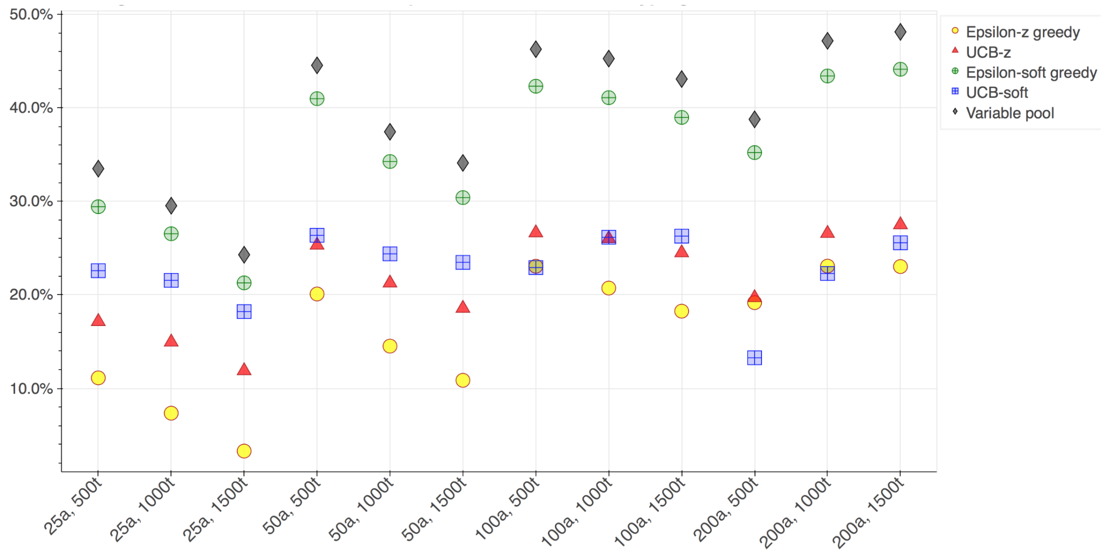
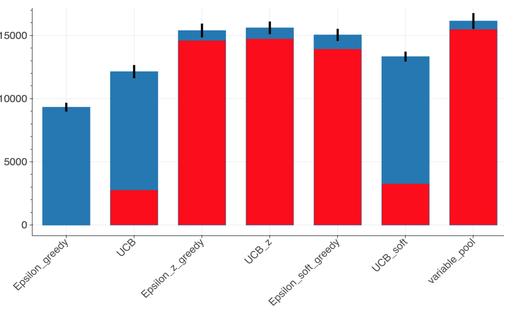


Figure 33: Comparison of average final rewards in games with 25 arms, 500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

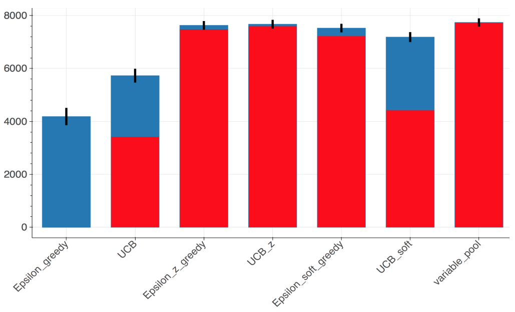
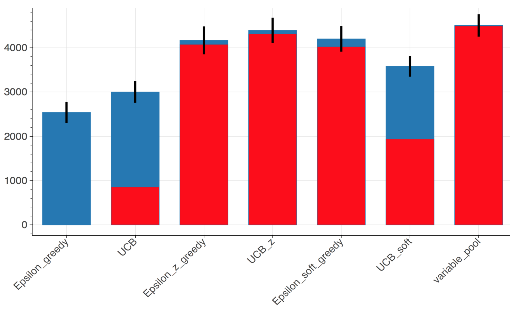


Figure 34: Comparison of average final rewards in games with 50 arms, 500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

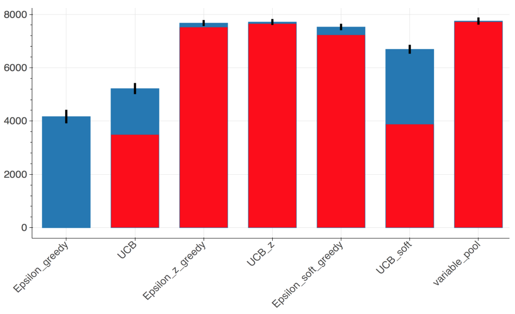
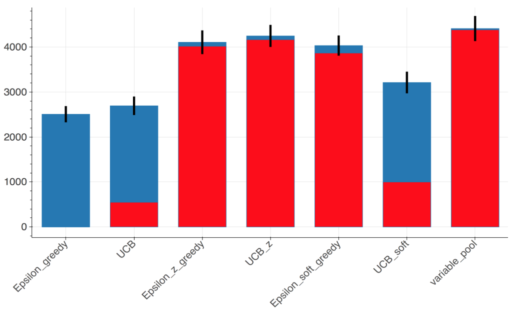


Figure 35: Comparison of average final rewards in games with 100 arms, 500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

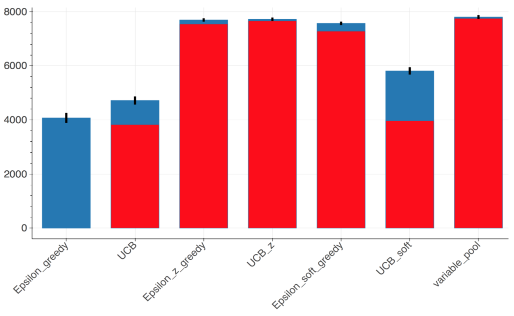
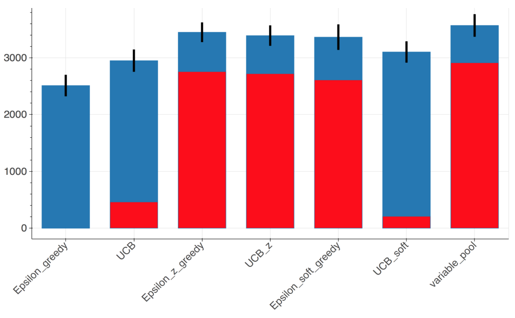


Figure 36: Comparison of average final rewards in games with 200 arms, 500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

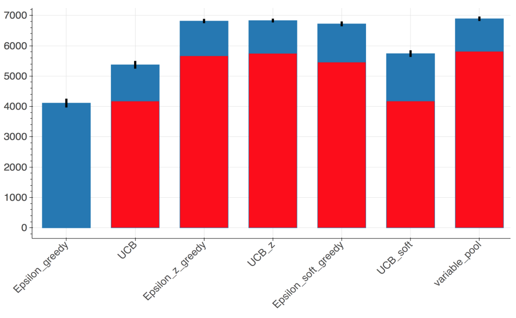
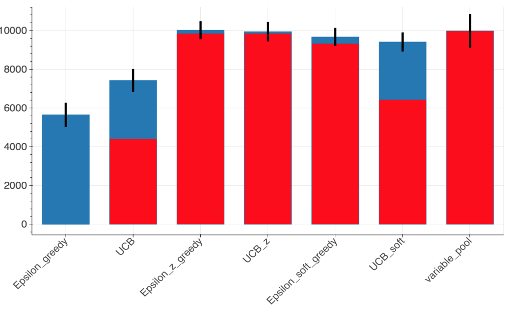


Figure 37: Comparison of average final rewards in games with 25 arms, 1000 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

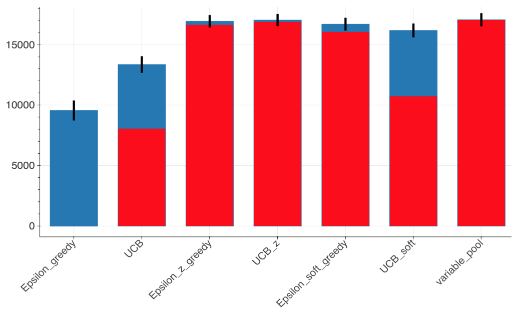
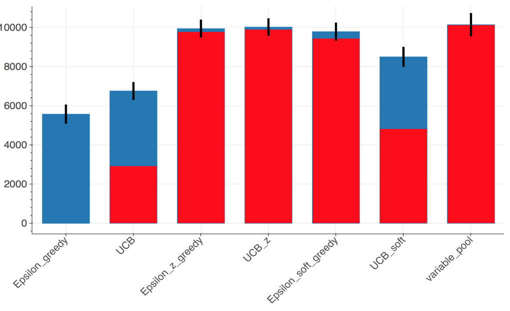


Figure 38: Comparison of average final rewards in games with 50 arms, 1000 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

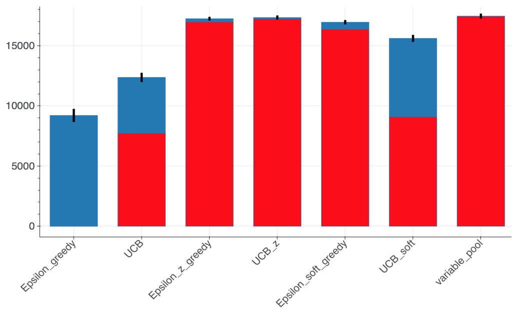
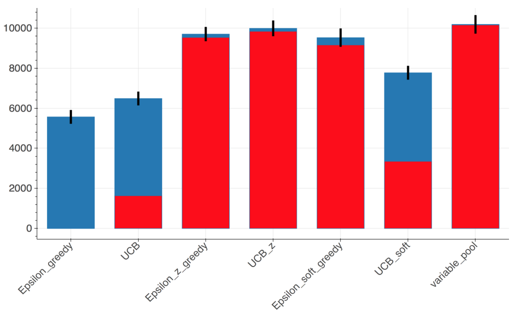


Figure 39: Comparison of average final rewards in games with 100 arms, 1000 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

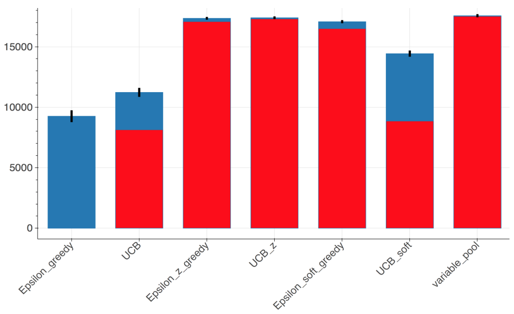
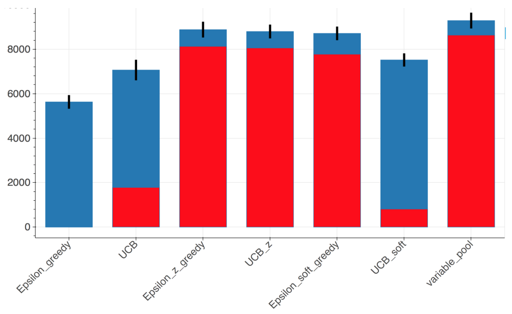


Figure 40: Comparison of average final rewards in games with 200 arms, 1000 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

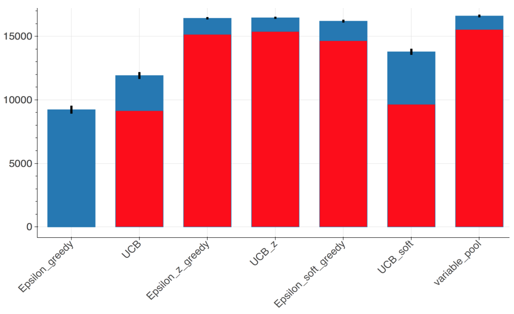
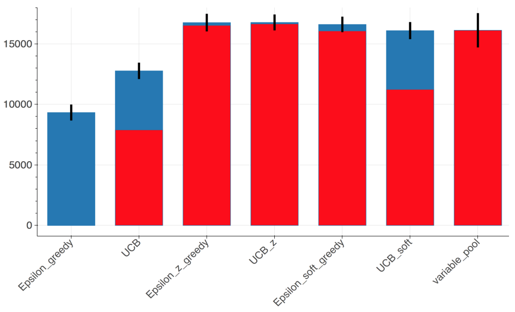


Figure 41: Comparison of average final rewards in games with 25 arms, 1500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

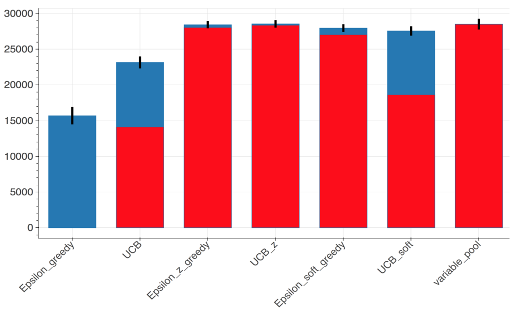
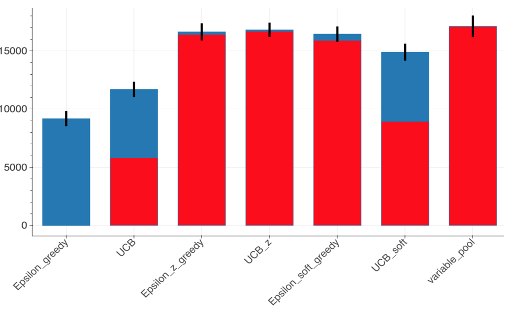


Figure 42: Comparison of average final rewards in games with 50 arms, 1500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

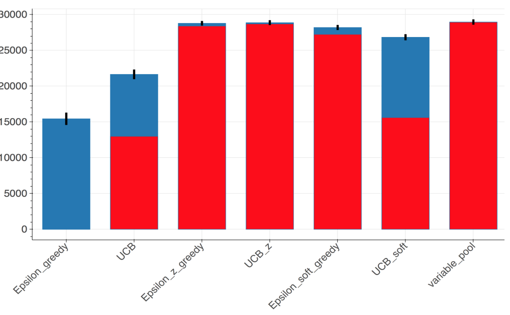
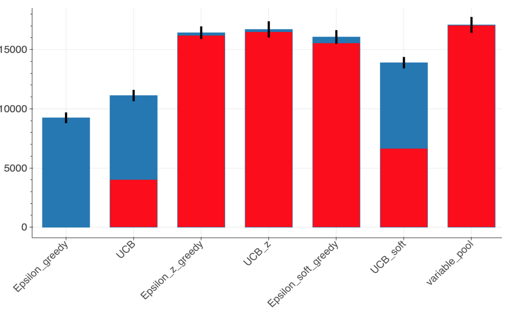


Figure 43: Comparison of average final rewards in games with 100 arms, 1500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

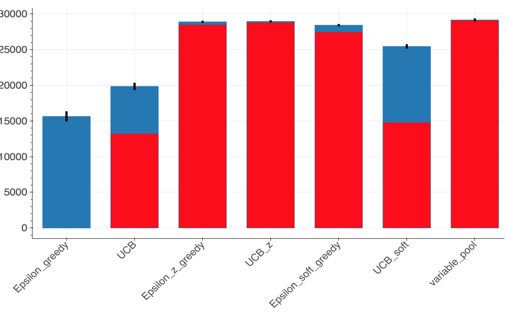
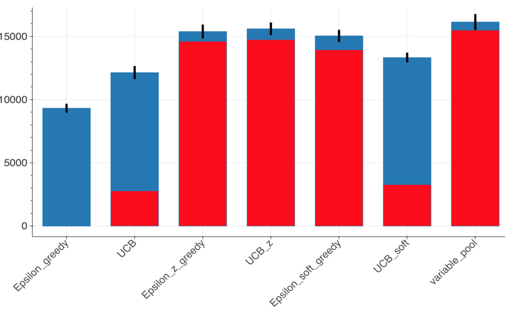


Figure 44: Comparison of average final rewards in games with 200 arms, 1500 turns, and a Step-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

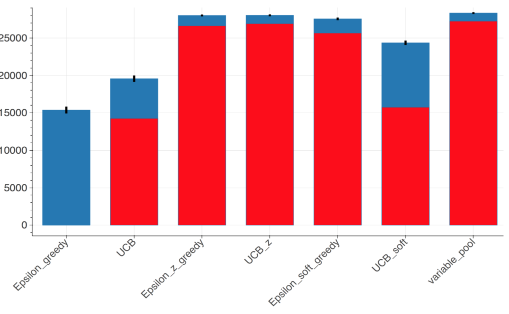


Figure 45: Average increase in rewards (coming from Bernoulli distributions) compared to the (smarter) version of the ε-greedy algorithm (Algorithm 6) with a Step-type greed function.

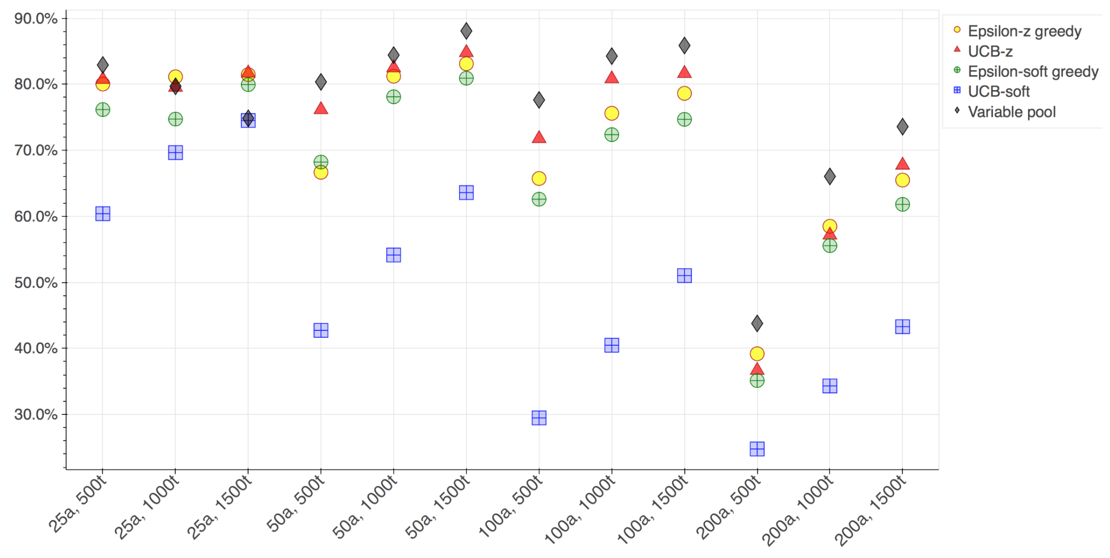


Figure 46: Average increase in rewards (coming from Bernoulli distributions) compared to the (smarter) version of the UCB algorithm (Algorithm 7) with a Step-type greed function.

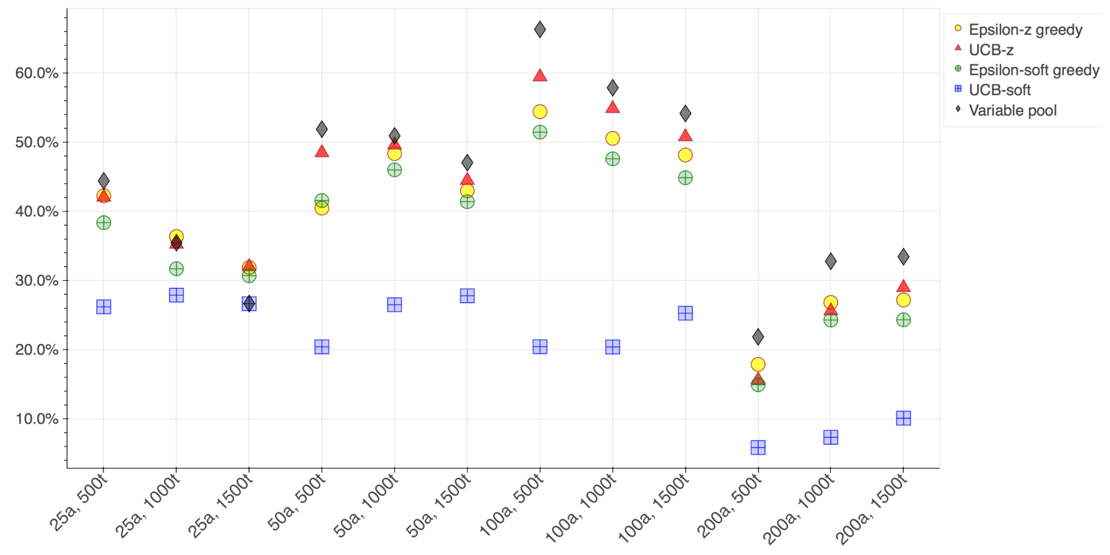


Figure 47: Average increase in rewards (coming from Truncated-Normal distributions) compared to the (smarter) version of the ε-greedy algorithm (Algorithm 6) with a Step-type greed function.

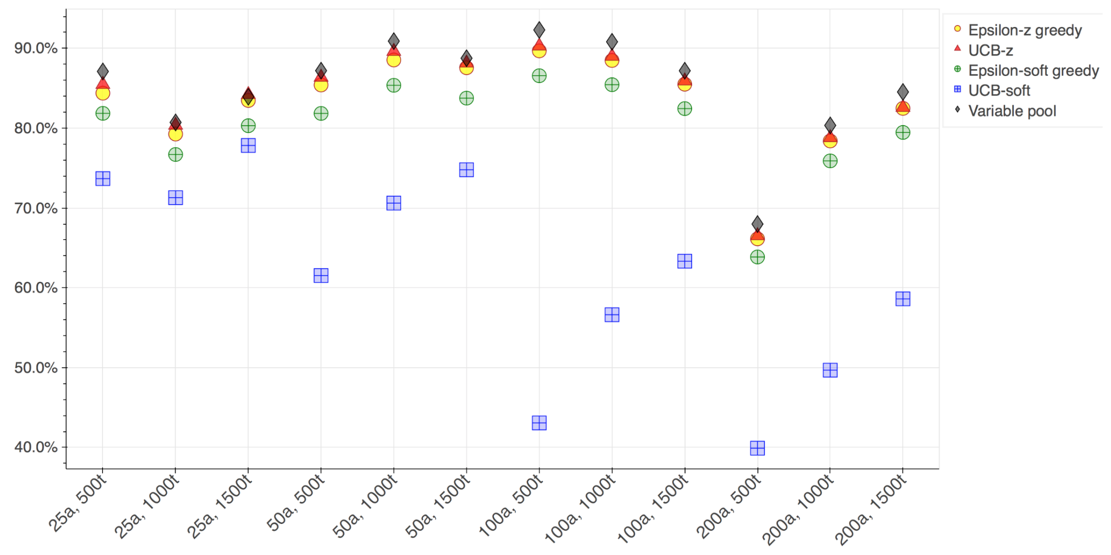


Figure 48: Average increase in rewards (coming from Truncated-Normal distributions) compared to the (smarter) version of the UCB algorithm (Algorithm 7) with a Step-type greed function.

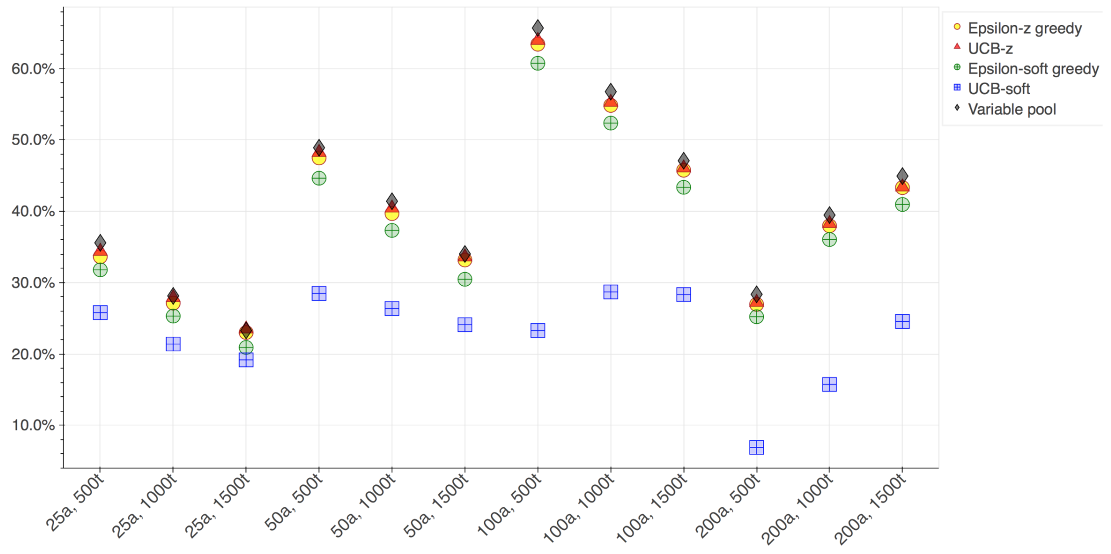
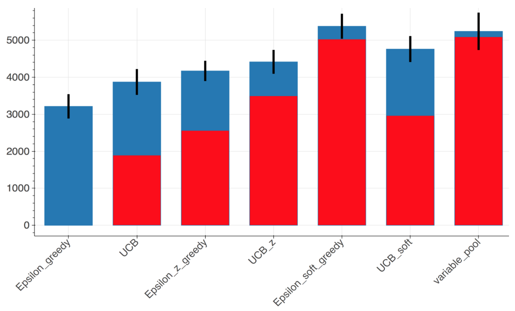


Figure 49: Comparison of average final rewards in games with 25 arms, 500 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

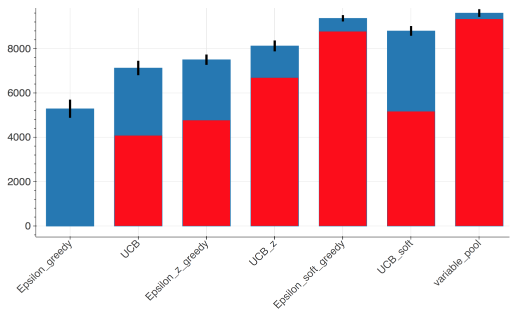
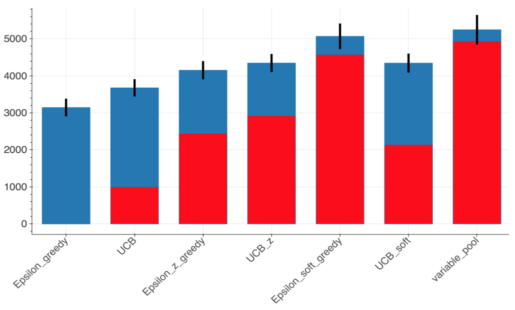


Figure 50: Comparison of average final rewards in games with 50 arms, 500 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

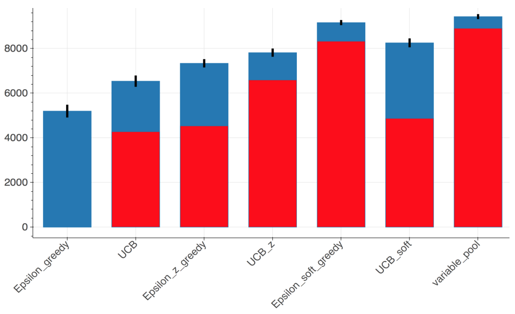
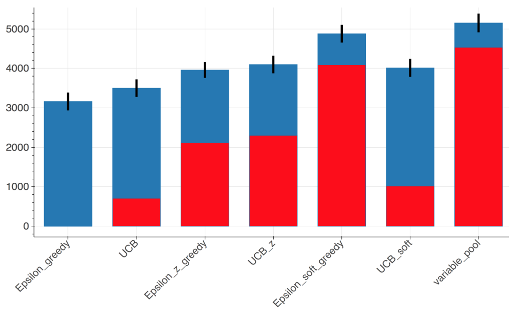


Figure 51: Comparison of average final rewards in games with 100 arms, 500 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

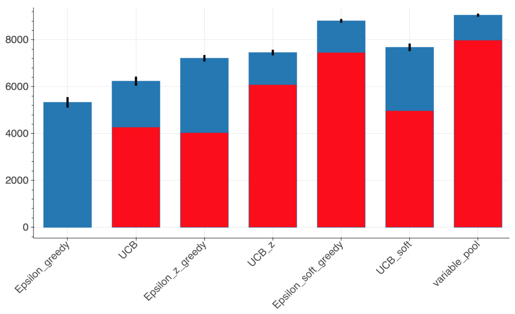
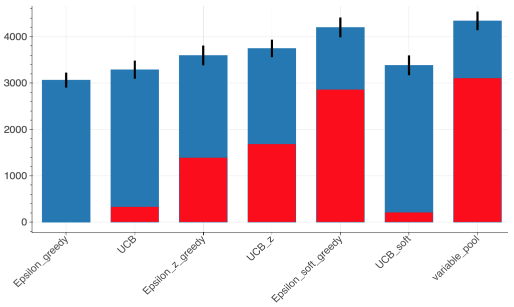


Figure 52: Comparison of average final rewards in games with 200 arms, 500 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

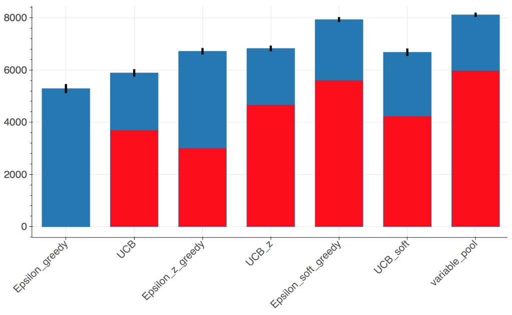
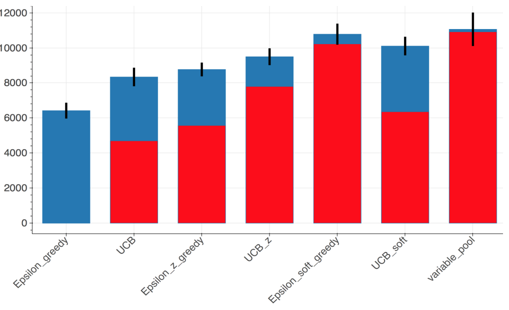


Figure 53: Comparison of average final rewards in games with 25 arms, 1000 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

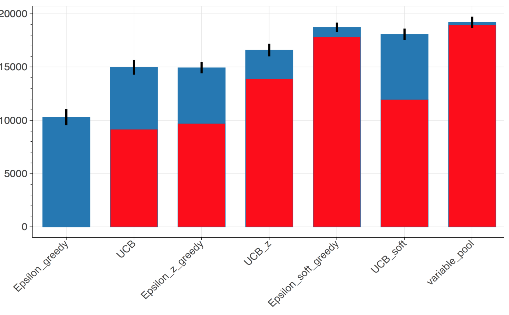
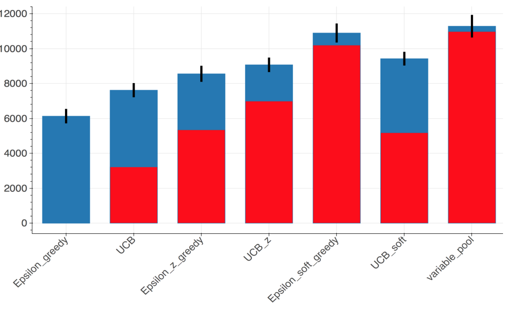


Figure 54: Comparison of average final rewards in games with 50 arms, 1000 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

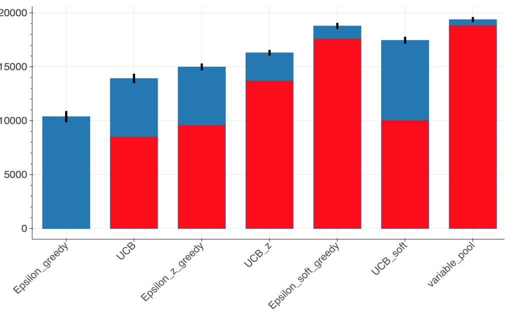
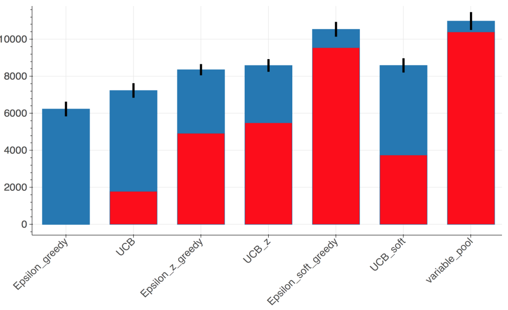


Figure 55: Comparison of average final rewards in games with 100 arms, 1000 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

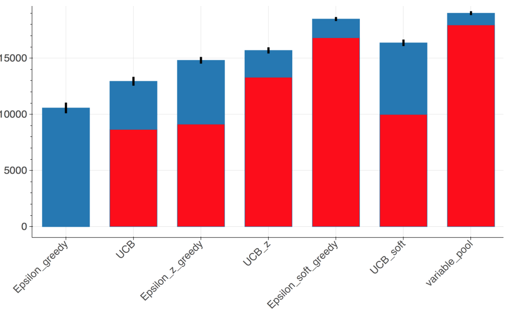
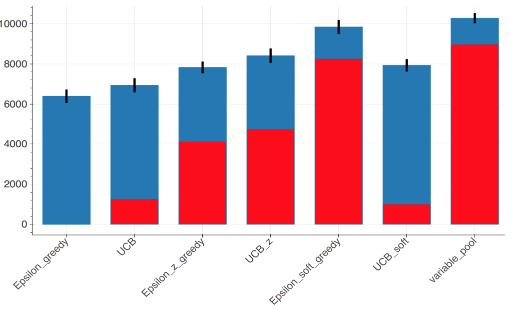


Figure 56: Comparison of average final rewards in games with 200 arms, 1000 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

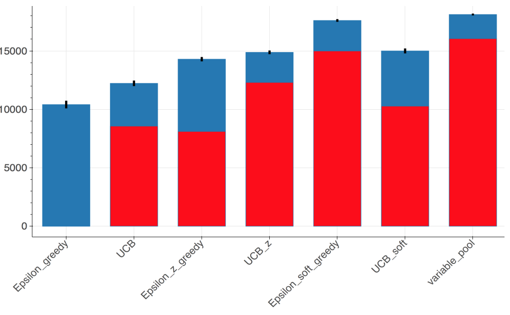
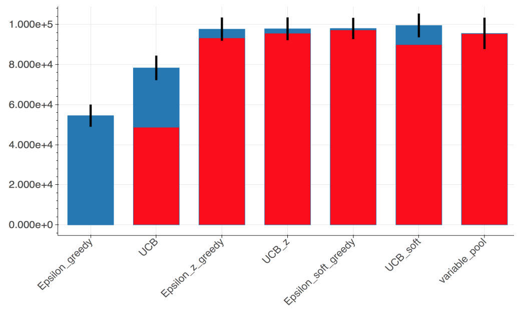


Figure 57: Comparison of average final rewards in games with 25 arms, 1500 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

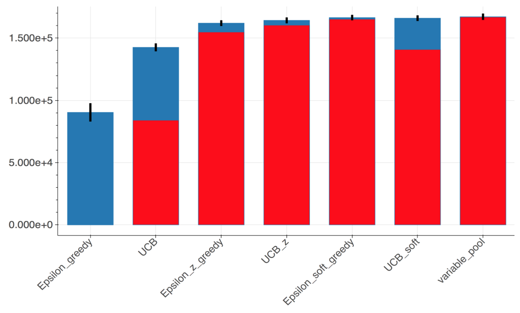
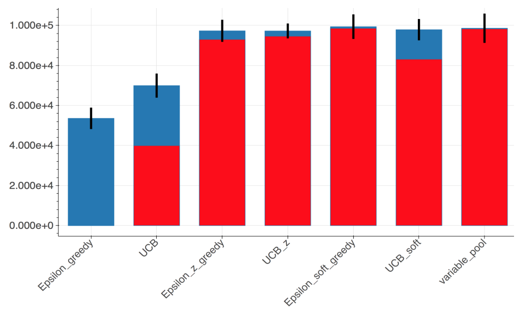


Figure 58: Comparison of average final rewards in games with 50 arms, 1500 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

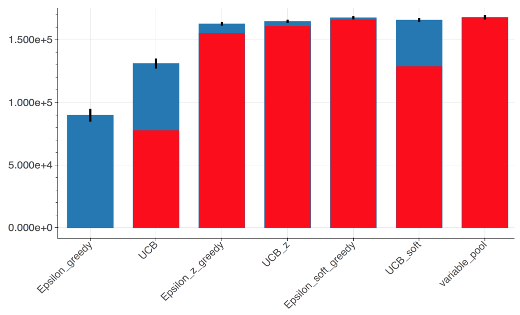
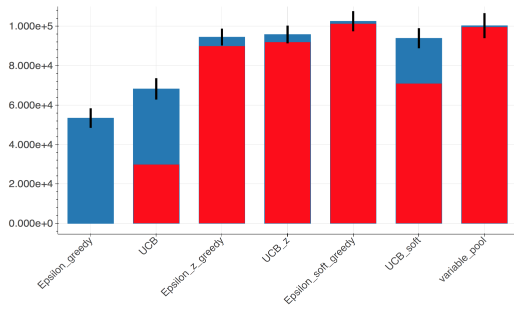


Figure 59: Comparison of average final rewards in games with 100 arms, 1500 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

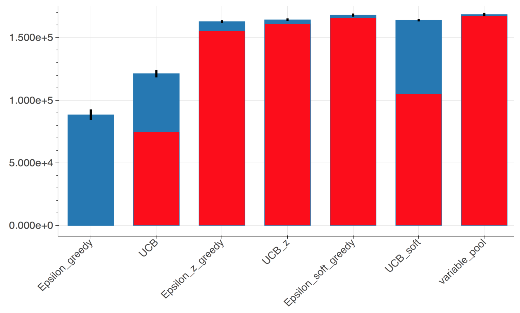
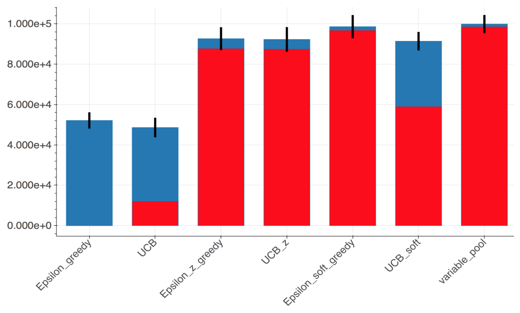


Figure 60: Comparison of average final rewards in games with 200 arms, 1500 turns, and a Christmas-type greed function.

(a) Rewards from Bernoulli distributions.



(b) Rewards from truncated Normal distributions.

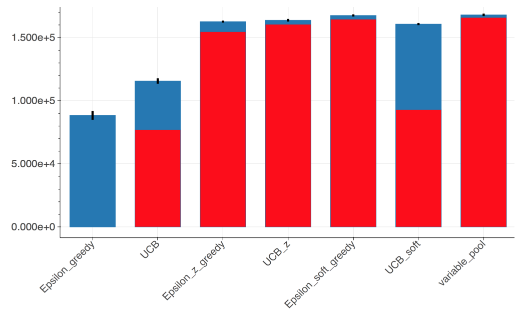


Figure 61: Average increase in rewards (coming from Bernoulli distributions) compared to the (smarter) version of the ε-greedy algorithm (Algorithm 6) with a Christmas-type greed function.

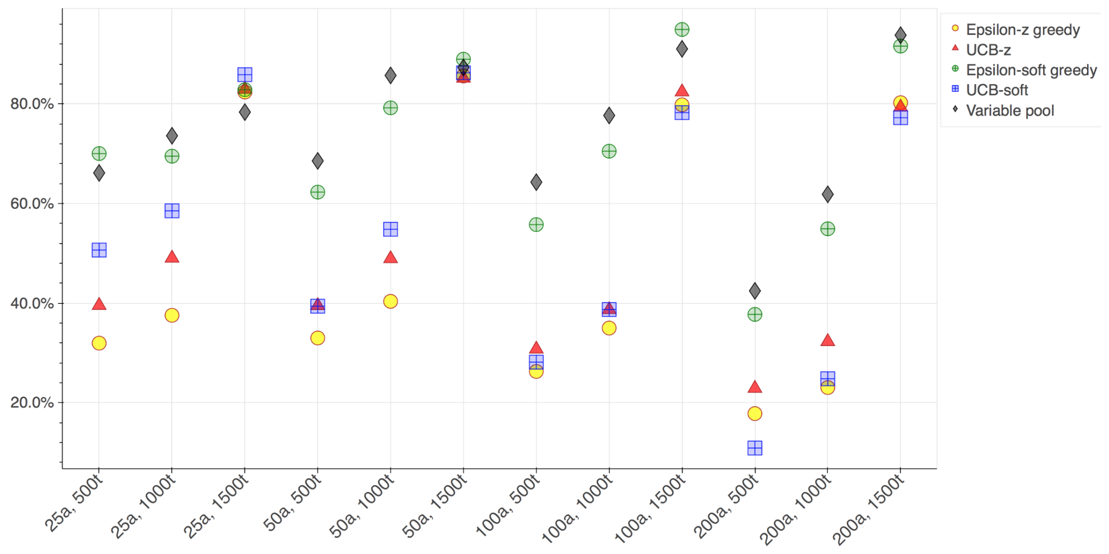


Figure 62: Average increase in rewards (coming from Bernoulli distributions) compared to the (smarter) version of the UCB algorithm (Algorithm 7) with a Christmas-type greed function.

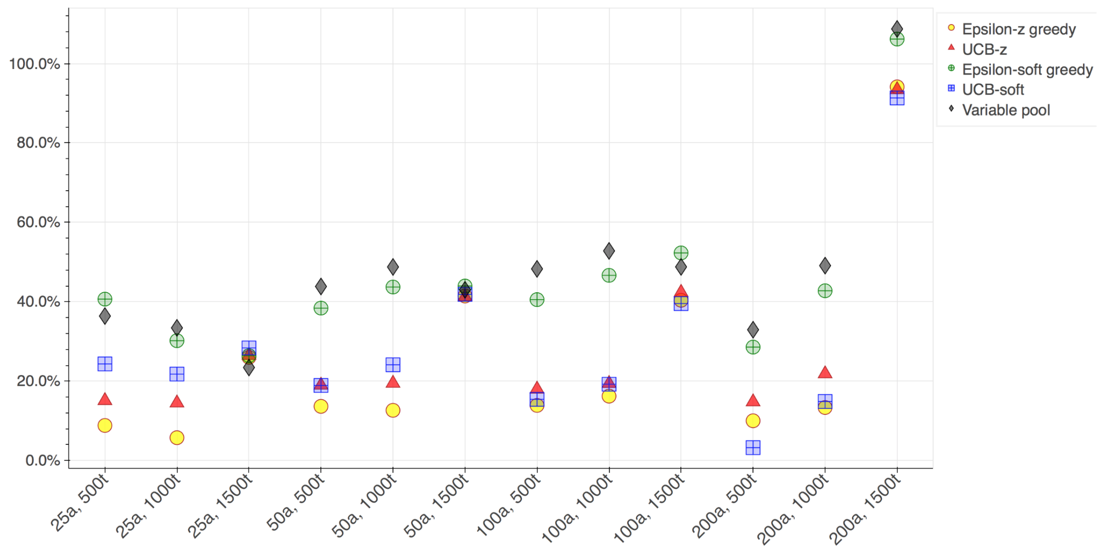


Figure 63: Average increase in rewards (coming from Truncated-Normal distributions) compared to the (smarter) version of the ε-greedy algorithm (Algorithm 6) with a Christmas-type greed function.

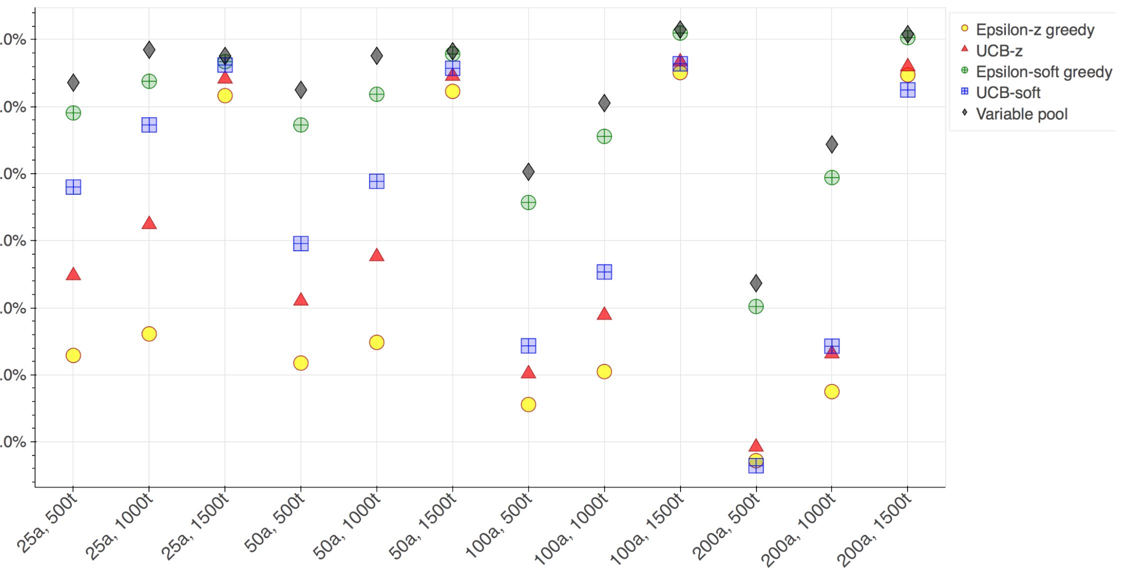


Figure 64: Average increase in rewards (coming from Truncated-Normal distributions) compared to the (smarter) version of the UCB algorithm (Algorithm 7) with a Christmas-type greed function.

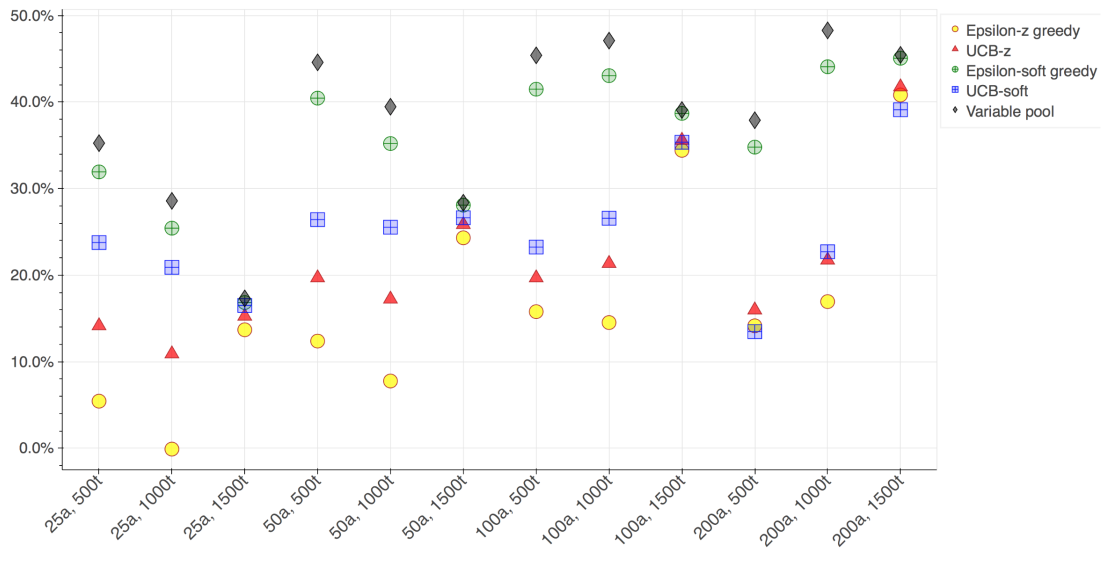


Table 11: Raw structure of the Yahoo! dataset. Each user receives a personalized recommendation. The value of $G(t)$ is computed by binning records that appear within a second.

turn	timestamp	user	displayed article	cliced?	article pool	G(t)
1	1317513291	user1	id-560620	0	id-560620, ...	32
2	1317513291	user2	id-565648	0	id-560620, ...	32
3	1317513291	user3	id-563115	1	id-560620, ...	32
...	...	...	...	...	...	...
32	1317513291	user32	id-563115	0	id-560620, ...	32
33	1317513292	user33	id-552077	1	id-552077, ...	13
34	1317513292	user34	id-564335	0	id-552077, ...	13
...	...	...	...	0	...	...
45	1317513292	user45	id-564335	1	id-552077, ...	13
...	...	...	...	...	...	...

Appendix H. Experiment details

H.1 Offline evaluation methodology

Each entry of the Yahoo! Webscope dataset contains information on:

- the arm pulled (which is the article shown to the human viewing articles on Yahoo!);
- the outcome (whether the article was clicked or not);
- the pool of arms (articles) available at that time and the associated timestamp.

A unique property of this dataset is that the displayed article was chosen uniformly at random from a pool of available articles. Therefore, it is possible to use an unbiased offline evaluation method (see Li et al., 2010) to compare bandit algorithms in a reliable way. The key point of the method is to ignore the events in which the article chosen at random does not match the article chosen by the algorithm. This means that a large portion of the dataset gets discarded because it is very common that the random choice and the algorithm choice do not match. In the initialization phase, we made the simulation faster and simpler: by pre-scanning the dataset, we can detect the articles that never match the recommendations chosen by the algorithm, and therefore we do not initialize such articles.

A difference that the Yahoo! dataset has from our setting is that each visitor receives a different article recommendation (as opposed to customers receiving the same recommendation as is common in application like retail). Table 11 shows how the data is structured in the Yahoo! dataset, while Table 12 shows how the data is expected by the setting described Section 3.1. In order to reconcile this difference, we follow the structure of the event stream, treating each turn associated with one user, but we use $G(t)$ to decide at each turn how to balance exploration and exploitation according to the strategy of our algorithms.

H.2 Choice of the parameters

The algorithms have some tunable parameters. Table 13 lists the parameters chosen for each algorithm. They were chosen by tuning them on a small sample of the data. In practice, it is always useful to have data that resembles the one that the algorithm will use so that parameters can be tuned. The value of $z = 31$ corresponds to the 75th quantile of $G(t)$.

Table 12: Expected structure of the dataset following the setting described in Section 3.1. Visitors are binned in groups of size $G(t)$ and they all receive the same recommendation.

turn	timestamp	user	displayed article	clicked?	article pool	$G(t)$
1	1317513291	user1 user2 user3 ... user32	id-560620	0	id-560620, ...	32
2	1317513292	user33 user34 ... user45	id-552077	1	id-552077, ...	13
...	...	...	...	...	...	...

Table 13: Parameters of the algorithms used in the simulation.

algorithm	hyperparameters
ε-z Greedy	$z = 31$
Variable Pool	$c = 10$, $z = 31$
UCB-L	$c = 0.011$
UCB-z	$z = 31$

Appendix I. When $G(t)$ is not known exactly

In this Appendix we show how to change the theorems when $G(t)$ is not known. We also show that the final regret did not change considerably in the simulations when using some standard predictive methods to estimate $G(t+1)$ by using the information available in turn t.

I.1 Adapting regret bound theorems for the case of estimated $G(t)$

The theorems regarding the regret bound on the final cumulative reward are similar to the case when $G(t)$ is known. Suppose we estimate $G(t)$ with $H(t)$, where $H(t)$ is estimated using any method. Then, for example, Theorem 1 becomes:

Theorem 27 (ε-greedy algorithm with hard threshold and estimated multiplier function) *The bound on the mean regret $\mathbb{E}[R_n]$ at time n is given by*

$$\begin{aligned}
\mathbb{E}[R_n] \quad \leq \quad & \sum_{j=1}^{m} G(j)\Delta_j \\
+ \quad & \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)<z \textit{ and } H(t)<z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \left(\varepsilon_t \frac{1}{m} + (1-\varepsilon_t)\beta_j(\tilde{t})\right) \\
+ \quad & \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)\geq z \textit{ and } H(t)<z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \left(\varepsilon_t \frac{1}{m} + (1-\varepsilon_t)\beta_j(\tilde{t})\right) \qquad (99) \\
+ \quad & \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)<z \textit{ and } H(t)\geq z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \beta_j(\tilde{t}), \qquad (100) \\
+ \quad & \sum_{t=m+1}^{n} G(t)\mathbb{1}_{\{G(t)\geq z \textit{ and } H(t)\geq z\}} \sum_{j:\mu_j<\mu_*} \Delta_j \beta_j(\tilde{t}), \\
& \textit{where} \quad \beta_j(\tilde{t}) = k\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k}{10}} \log\left(\frac{\tilde{t}}{mke}\right) + \frac{4}{\Delta_j^2}\left(\frac{\tilde{t}}{mke}\right)^{-\frac{k\Delta_j^2}{4}}
\end{aligned}$$

and $\tilde{t}$ is the number of times the estimated multiplier $H(t)$ has been under the threshold.

Summations (99) and (100) are the extra addends that appear when the estimator and the true multiplier function are not both either under or above the threshold z:

- the term in (99) represents the regret incurred when the algorithm balances exploration and exploitation (because the estimated $H(t)$ is below the threshold z) when it should have been exploiting (because $G(t)$ is actually above the threshold z);
- the term in (100) represents the regret incurred when the algorithm is exploiting (because the estimated $H(t)$ is above the threshold z) when it should have been balancing exploration and exploitation (because $G(t)$ is actually below the threshold z).

Similarly, it is possible to rewrite the regret bound theorems of all the other algorithms for the case when $G(t)$ is estimated by $H(t)$:

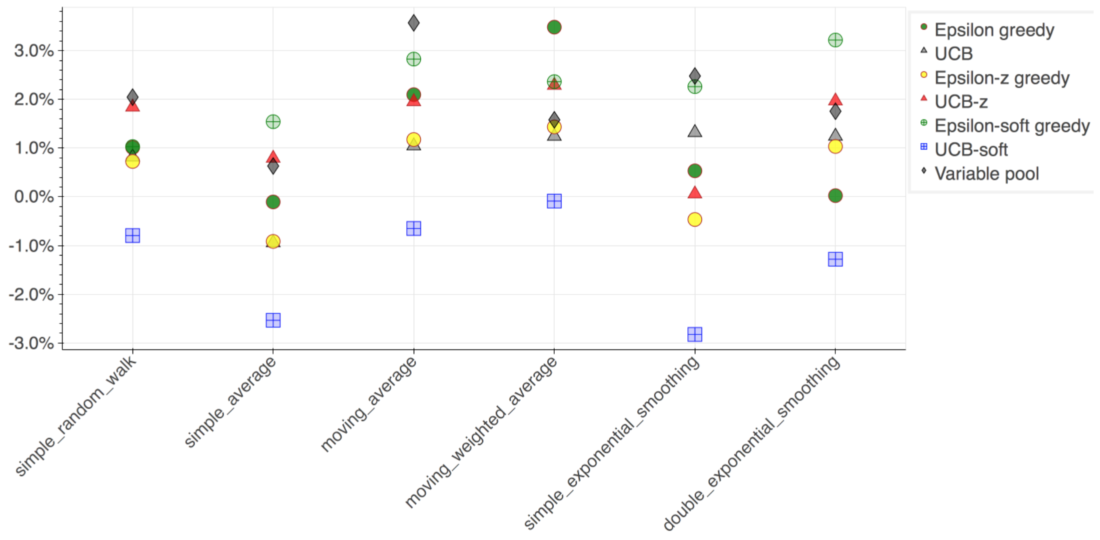
- for the UCB-z algorithm, the regret bound would have two extra terms for when $G(t)$ and $H(t)$ are not above or below the threshold z at the same time;
- for the Soft ε-greedy algorithm, the regret bound still depends on $G(t)$, but $\psi(t)$ depends on $H(t)$;
- for the Soft UCB algorithm, the regret bound still depends on $G(t)$, but $\xi(t)$ depends on $H(t)$;
- for the variable pool algorithm, the regret bound still depends on $G(t)$, but λ_t depends on $H(t)$;
- for the Soft UCB mortal algorithm, the regret bound still depends on $G(t)$, but $\psi_{\text{future}}(j,t)$ and $\xi_{\text{present}}(t)$ depend on $H(t)$.

I.2 Rewards change when estimating $G(t+1)$ in turn t

We used some standard methods for predicting $G(t+1)$ by using the information available in turn t:

- simple random walk;
- simple average;

Figure 65: Final rewards when predicting $G(t)$ step by step compared to when knowing $G(t)$, with Bernoulli rewards and Wave-type greed function.



- moving average;
- moving weighted average;
- simple exponential smoothing;
- double exponential smoothing.

The difference in final rewards is not significantly different from when $G(t)$ is known. Figures 65-70 show the percentage change in final regret from the case of known $G(t)$ in the simulation.

Figure 66: Final rewards when predicting $G(t)$ step by step compared to when knowing $G(t)$, with Bernoulli rewards and Christmas-type greed function.

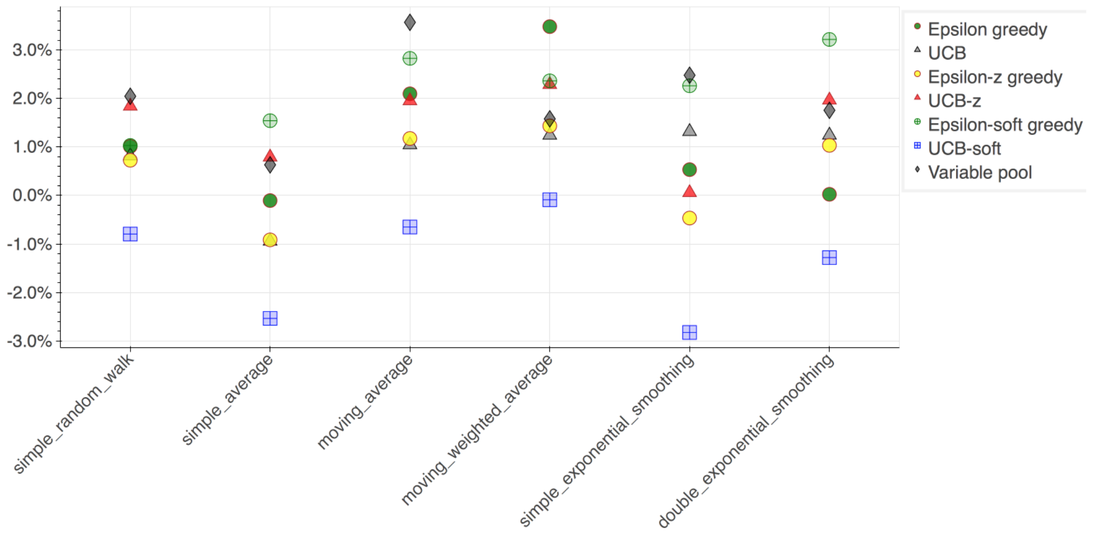


Figure 67: Final rewards when predicting $G(t)$ step by step compared to when knowing $G(t)$, with Bernoulli rewards and Step-type greed function.

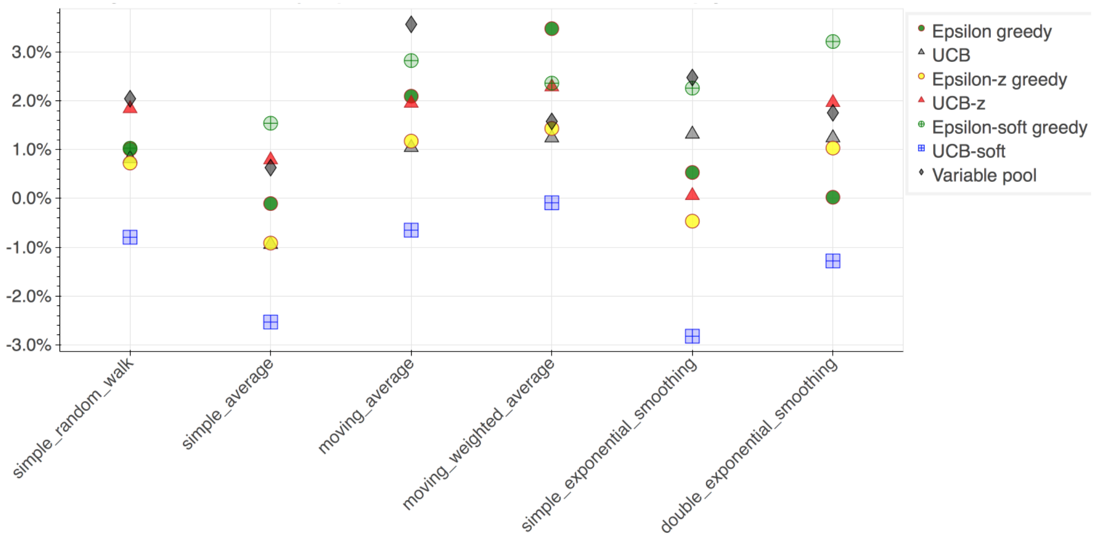


Figure 68: Final rewards when predicting $G(t)$ step by step compared to when knowing $G(t)$, with Truncated-Normal rewards and Wave-type greed function.

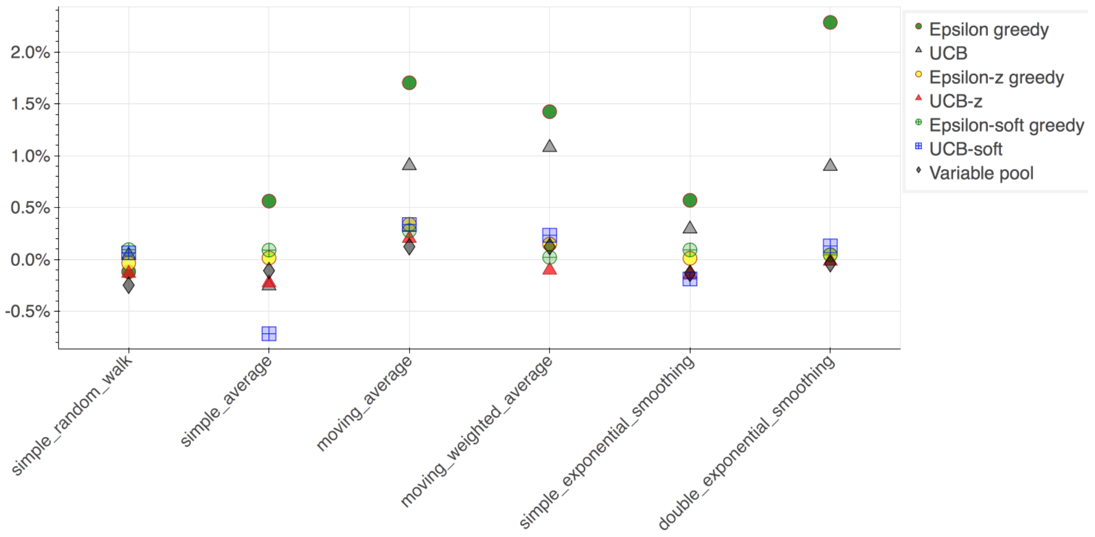


Figure 69: Final rewards when predicting $G(t)$ step by step compared to when knowing $G(t)$, with Truncated-Normal rewards and Christmas-type greed function.

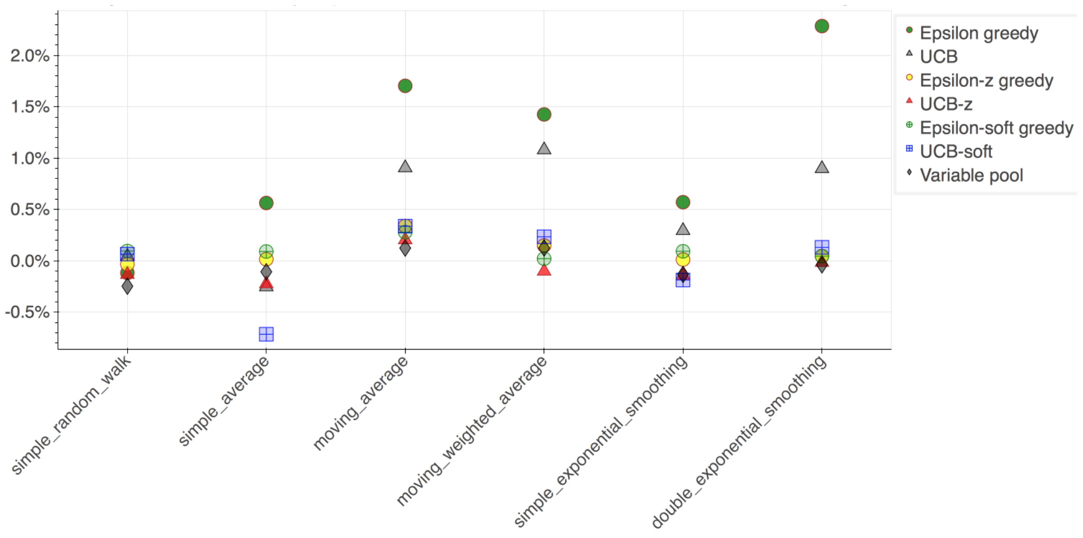
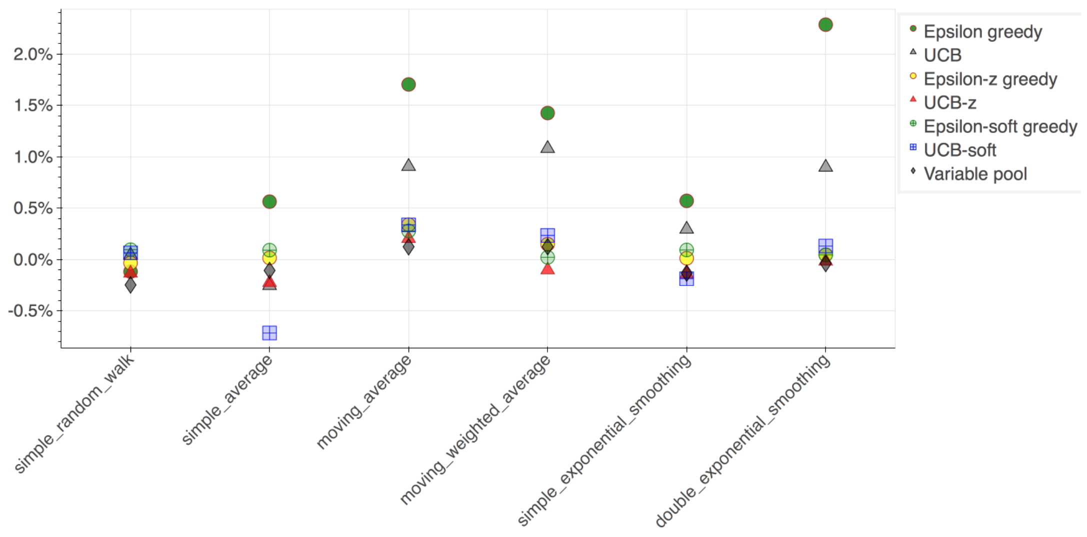


Figure 70: Final rewards when predicting $G(t)$ step by step compared to when knowing $G(t)$, with Truncated-Normal rewards and Step-type greed function.



Appendix J. Notation summary

- $\beta_j(\tilde{t})$: upper bound on the probability of considering arm j being the best arm at round $\tilde{t}$ when using Algorithm 1;
- $\beta_j^S(t)$: upper bound on the probability of considering arm j being the best arm at round t when using Algorithm 2;
- $\beta_j^{\text{old}}(t)$: upper bound on the probability of considering arm j being the best arm at round t when using Algorithm 6;
- $\beta_j^U(t)$: upper bound on the probability of considering arm j being the best arm at round t when using Algorithm 3;
- $\beta_j^{VP}(t)$: upper bound on the probability of considering arm j being the best arm at round t when using Algorithm 5;
- β_j^M: upper bound on the expected number of times suboptimal arm j is pulled when using Algorithm 8;
- B: set of rounds when the "high reward" zone is entered in Algorithm 3 ($B = \{t : G(t-1) < z, G(t) > z\}$);
- c: a constant greater than 10 in Algorithm 6;
- Δ_j: difference between the mean reward of the optimal arm and the mean reward of arm j ($\Delta_j = \mu_* - \mu_j$);
- $\Delta_{j,z}$: difference between the mean reward of the optimal arm in epoch z and the mean reward of arm j;
- d: a constant such that $d < \min_j \Delta_j$ and $0 < d < 1$ in Algorithm 6;
- ε_t: probability of exploration at turn t (used in Algorithm 1 and Algorithm 2);
- E_j: number of epochs in which L_j is partitioned depending on i_t^*;

- γ: lowest value of $\psi(t)$ ($\gamma = \min_{s\in\{m+1,\cdots,n\}} \psi(s)$);
- $G : \{1, \cdots, n\} \to \mathbb{R}^+$: known multiplier function;
- I_t: arm played at turn t;
- i_t^*: best arm in turn t (mortal case);
- k: a constant greater than 10 such that $k > 4/(\min_j \Delta_j^2)$ in Algorithm 1 and Algorithm 2;
- L_j: set of turns during which arm j is available;
- L_j^z: epoch z of the set L_j depending on i_t^*;
- l_j: last turn in L_j;
- l_j^z: last turn in epoch L_j^z;
- μ_*: mean reward of the optimal arm ($\mu_* = \max_{1\le j\le m} \mu_j$);
- m: number of arms;
- m_t: size of the arms pool in Algorithm 5;
- M_t: set of arms available in turn t (mortal case);
- n: number of rounds;
- $\tilde{n}$: number of rounds under the threshold z by the end of the game;
- n': particular time defined as km in the comparison between Algorithm 2 and Algorithm 6 in Section 3.3;
- $\psi(t)$: smoothing function used to define the probabilities of exploration ε_t in Algorithm 2 (see Figure 4);
- $\psi_{\text{future}}(j, t)$: function that compares the scores of available arms in Algorithm 8;
- R_n: total regret at round n;
- score$_j$: score given by the sum of the values of $G(t)$ over the lifespan of arm j (mortal case);
- s_j: first turn in L_j;
- s_j^z: first turn in epoch L_j^z;
- $T_j(t-1)$: number of times arm j has been played before round t starts;
- $\tilde{t}$: number of rounds under the threshold z up to time t;
- w: first round such that $f(s) = km/s$ is less than γ (i.e., $w = \min\{s : f(s) < \gamma\}$);
- $\xi(t)$: smoothing function used to define the decision rule in Algorithm 4;
- $\xi_{\text{present}}(t)$: smoothing function used to define the decision rule in Algorithm 8;
- $X_j(t)$: unscaled random reward for playing arm j in turn t;
- $\widehat{X}_j$: current estimate of μ_j;
- $Y_k = \{t : t \ge y_k, G(t) > z, t < y_{k+1}\}$: set of rounds in the high-reward period entered at time y_k;
- z: regulating threshold (used in Algorithm 1 and Algorithm 3);

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

Oren Anava, Elad Hazan, Shie Mannor, and Ohad Shamir. Online learning for time series prediction. In *Conference on Learning Theory*, pages 172–184, 2013.

Raman Arora, Ofer Dekel, and Ambuj Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. *arXiv preprint arXiv:1206.6400*, 2012.

Jean-Yves Audibert, Sébastien Bubeck, et al. Best arm identification in multi-armed bandits. In *Proceedings of the Conference on Learning Theory*, 2010.

Peter Auer and Chao-Kai Chiang. An algorithm with nearly optimal pseudo-regret for both stochastic and adversarial bandits. In *Proceedings of the Conference on Learning Theory*, pages 116–120, 2016.

Peter Auer and Ronald Ortner. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

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

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 207–216. IEEE, 2013.

Dirk Bergemann and Juuso Valimaki. Bandit problems. *Cowles Foundation discussion paper, No. 1551*, 2006.

Donald A Berry and Bert Fristedt. *Bandit Problems: Sequential Allocation of Experiments (Monographs on Statistics and Applied Probability)*. Springer, 1985.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Optimal exploration-exploitation in a multi-armed-bandit problem with non-stationary rewards. *Available at SSRN 2436629*, 2014.

Djallel Bouneffouf and Raphael Feraud. Multi-armed bandit problem with known trend. *Neurocomputing*, 205:16–21, 2016.

Olivier Bousquet. New approaches to statistical learning theory. *Annals of the Institute of Statistical Mathematics*, 55(2):371–389, 2003.

Sébastien Bubeck and Aleksandrs Slivkins. The best of both worlds: stochastic and adversarial bandits. In *Conference on Learning Theory*, pages 42–1, 2012.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet. Bounded regret in stochastic multi-armed bandits. In *Conference on Learning Theory*, pages 122–134, 2013.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

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

Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual conference on Learning Theory*, pages 359–376, 2011.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

Elad Hazan and Satyen Kale. Better algorithms for benign bandits. *Journal of Machine Learning Research*, 12 (Apr):1287–1311, 2011.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On bayesian upper confidence bounds for bandit problems. In *International Conference on Artificial Intelligence and Statistics*, pages 592–600, 2012a.

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Proceedings of Algorithmic Learning Theory*, volume 12, pages 199–213. Springer, 2012b.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 681–690. ACM, 2008.

Junpei Komiyama, Issei Sato, and Hiroshi Nakagawa. Multi-armed bandit problem with lock-up periods. In *Asian Conference on Machine Learning*, pages 116–132, 2013.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670. ACM, 2010.

Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 297–306, 2011.

Haoyang Liu, Keqin Liu, and Qing Zhao. Learning in a changing world: Restless multiarmed bandit with unknown dynamics. *IEEE Transactions on Information Theory*, 59(3):1902–1916, 2013.

Odalric-Ambrym Maillard and Shie Mannor. Latent bandits. In *International Conference on Machine Learning*, pages 136–144, 2014.

Shie Mannor and Ohad Shamir. From bandits to experts: On the value of side-observations. In *Advances in Neural Information Processing Systems*, pages 684–692, 2011.

Herbert Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.

Cynthia Rudin, Virot Ta Chiraphadhanakul, and Edward Su. Regulating greed over time for Yahoo! front page news article recommendations, from the Exploration and Exploitation 3 Challenge. Lecture slides, 2012.

Yevgeny Seldin and Gábor Lugosi. A lower bound for multi-armed bandits with expert advice. In *13th European Workshop on Reinforcement Learning (EWRL)*, 2016.

Aleksandrs Slivkins and Eli Upfal. Adapting to a stochastically changing environment: The dynamic multi-armed bandits problem. Technical Report CS-07-05, Brown University, 2007.

Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert Schapire. Efficient algorithms for adversarial contextual learning. In *International Conference on Machine Learning*, pages 2159–2168, 2016.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

Stefano Tracà, Cynthia Rudin, and Weiyu Yan. Reducing exploration of dying arms in mortal bandits. In *Proceedings of Uncertainty in Artificial Intelligence*, 2019.