CEMAB: A Cross-Entropy-based Method for Large-Scale Multi-Armed Bandits

Erli Wang[1] **, Hanna Kurniawati[2], and Dirk P. Kroese[1]

[1] School of Mathematics and Physics, The University of Queensland Brisbane, QLD 4072, AUSTRALIA
[2] School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, QLD 4072, AUSTRALIA
`e.wang2@uq.edu.au, hannakur@uq.edu.au, kroese@maths.uq.edu.au`

Abstract. The multi-armed bandit (MAB) problem is an important model for studying the exploration–exploitation tradeoff in sequential decision making. In this problem, a gambler has to repeatedly choose between a number of slot machine arms to maximize the total payout, where the total number of plays is fixed. Although many methods have been proposed to solve the MAB problem, most have been designed for problems with a small number of arms. To ensure convergence to the optimal arm, many of these methods, including state-of-the-art methods such as UCB [2], require sweeping over the entire set of arms. As a result, such methods perform poorly in problems with a large number of arms. This paper proposes a new method for solving such large-scale MAB problems. The method, called Cross-Entropy-based Multi Armed Bandit (CEMAB), uses the Cross-Entropy method as a noisy optimizer to find the optimal arm with as little cost as possible. Experimental results indicate that CEMAB outperforms state-of-the-art methods for solving MABs with a large number of arms.

Keywords: Cross-Entropy method, Sequential decision making, Multi-armed bandit

1 Introduction

A fundamental question in sequential decision making is how to select the best action sequence even if the consequence of each action may not be exactly known. In its simplest form, this question can be studied as a Multi-Armed Bandit (MAB) [9] problem. Under this framework, selecting an action is akin to selecting which slot machine to play from a number of such machines. The question becomes how to balance between playing machines that have been giving good rewards in the past (often called exploitation) and machines that have not been tried before (often called exploration), such that the total reward received is as close as possible to the total reward that would have been received if the player had always played the highest-paying machine.

** Corresponding author

Many methods, such as ε-greedy [13], softmax [11], and UCB [2], have been proposed to solve the above problem of balancing exploration and exploitation. In fact, many of such methods have become the foundation of today's Reinforcement Learning [11]. However, except for a few [4, 6], most methods [5] try and estimate the reward of each and every action, to ensure that the best action is not missed. Therefore, their effectiveness is limited to problems with a relatively small number of arms (e.g., fewer than 20). Unfortunately, this assumption is quickly becoming unrealistic in a growing number of applications. For instance, one can now choose from hundreds of drug cocktails —combinations of various types of drugs at various dosages— in personalized medicine, choose one of hundreds of different combinations of investment portfolios, and select a subset of tens of millions of possible combinations of data and sensors that can be used to analyze consumer preferences. As a result, most of today's methods for solving MABs [5] are no longer effective for solving the more recent large-scale problems.

To alleviate the difficulty of solving MABs with a large number of discrete actions, we propose a novel method called Cross-Entropy-based Multi-Armed Bandit (CEMAB). Key to CEMAB is the use of the Cross-Entropy (CE) method [10] as a stochastic optimization method to identify the best action. By doing so, CEMAB can significantly reduce the number of actions to test before identifying the best action, assuming that the reward for pulling an arm is retrieved from an unknown fixed distribution. Preliminary results on standard test cases for MAB indicate that the number of arms to pull before CEMAB identifies the (close to) optimal arms is not directly dependent on the number of arms in the problem, which indicates that CEMAB is able to scale up well. This observation is supported by our simulation results, where tests on various MAB problems with up to 10,000 arms indicate that CEMAB outperforms state-of-the-art MAB solvers on large problems.

2 Background and Related Work

2.1 Multi-armed Bandit Problem

The MAB problem was first described in [9]. In this problem, a gambler has to decide which of several slot machines (often called arms) to play, where each machine gives a different reward according to some unknown distribution. The goal is to maximize the total reward of all the plays. Ideally, the player should only pull the machine that yields, on average, the highest reward.

More formally, let the set of arms be denoted by $\mathcal{K} = \{1, \ldots, |\mathcal{K}|\}$. Each arm $k \in \mathcal{K}$ corresponds to an unknown reward distribution D_k with support $[0, 1]$ and expectation μ_k. In this paper, we assume that the reward distributions are fixed (that is, they do not change over time) and independent of each other. At each time step t, an arm $k_t \in \mathcal{K}$ is pulled and a reward r_{k_t}, drawn from D_{k_t}, is received. Many objective functions have been proposed for MAB [5]. In this paper, we use the simple objective [9] to maximize the expected total reward received within a fixed number T of plays, i.e.,

$$\max_{(k_1,k_2,\ldots,k_T)\in\mathcal{K}^T} \mathbb{E}\left[\sum_{t=1}^{T} r_{k_t}\right] = \max_{(k_1,k_2,\ldots,k_T)\in\mathcal{K}^T} \sum_{t=1}^{T} \mu_{k_t}. \tag{1}$$

An equivalent goal is to minimize the total regret [2], which is:

$$\min_{(k_1,k_2,\ldots,k_T)\in\mathcal{K}^T}\left(T\max_{k\in\mathcal{K}}\mu_k-\sum_{t=1}^{T}\mu_{k_t}\right). \quad (2)$$

Various methods for solving MAB have been proposed. The rest of this subsection provides a brief overview of the most commonly-used methods, which we will use for comparison later on.

ε-greedy [13]. The ε-greedy method is the simplest and most widespread way to solve the MAB problem. At each time step, the algorithm has a probability ε to select an arm uniformly at random (exploration) and a probability $1-\varepsilon$ to choose the arm with the highest estimated reward so far (exploitation). In general, this strategy does not converge to the optimal arm.

Softmax [11]. Softmax picks each arm with a probability according to its empirical performance. The probability of each arm in Softmax can be based on the Boltzmann distribution $p_k = \mathrm{e}^{\hat{\mu}_k/\mathcal{T}}/\sum_{k=1}^{|\mathcal{K}|}\mathrm{e}^{\hat{\mu}_k/\mathcal{T}}$, where $\hat{\mu}_k$ is an estimate of the expected reward μ_k and $\mathcal{T}$ is the temperature. If $\mathcal{T}$ is very small, the arm with the highest estimated reward will have a large probability of being chosen (exploitation). In contrast, when $\mathcal{T}$ is very large, all $\{p_k\}$ are approximately equal, so that in this case Softmax is purely exploring.

Exp3 [3]. Exp3 (*exponential weight algorithm for exploration and exploitation*) is a famous variant of Softmax. The probability of choosing arm k is defined by $p_k = (1-\gamma)w_k/\sum_{j=1}^{|\mathcal{K}|}w_j+\gamma/|\mathcal{K}|$. The weights $\{w_j\}$ are updated after each step. In particular, after arm k is chosen (yielding reward r_k), the weight w_k is updated as $w_k \leftarrow w_k \exp^{\gamma r_k/p_k|\mathcal{K}|}$. It can be shown that the "weak regret", defined as $\left(T\max_{k\in\mathcal{K}}\mu_k-\sum_{t=1}^{T}r_t\right)$, is bounded under Exp3.

UCB[2]. The UCB is a family of algorithms for which optimal logarithmic regret can be achieved uniformly over time, assuming that all reward distributions have bounded support [2]. The simplest member of this family is UCB1. It records the number of times that each arm has been played, $\texttt{visits}(k)$, and after each choice k updates its current estimate of μ_k via $\hat{\mu}_k \leftarrow \hat{\mu}_k + (r_k-\hat{\mu}_k)/\texttt{visits}(k)$. At the beginning, each arm is played once (full sweep). Subsequently, at each time t arm k is chosen that satisfies $k = \mathrm{argmax}_{k=1,\ldots,|\mathcal{K}|}\,\hat{\mu}_k+\sqrt{C\log t/\texttt{visits}(k)}$.

Thompson Sampling (TS)[1] Thompson sampling is a Bayesian sampling algorithm based on [12]. For each arm k, the knowledge of the expected reward μ_k is described by a $\mathsf{Beta}(\alpha_k,\beta_k)$ distribution. At time t, random variables $\theta_k, k=1,\ldots,|\mathcal{K}|$ are generated from each of these distributions. The index k^* corresponding to the largest of the $\{\theta_k\}$ is the arm to play. If r is the corresponding reward, then a Bernoulli trial B with probability r is generated. If $B=1$, then α_{k^*} is increased by 1, otherwise β_{k^*} is increased by 1.

All of the above methods have been designed for various types of reward function, e.g., stochastic and deterministic, and have various ways to commit to arms that have performed well so far. However, they have not been designed for problems with a large number of arms. In fact, the state-of-the art method, UCB, explicitly requires a sweep of the entire set of arms, which will be problematic when the MAB problem has hundreds or thousands of arms. This problem is exactly the focus of our paper.

2.2 Cross-Entropy (CE) Method for Noisy Optimization

The CE method [10] is a randomized optimization method that has proved to be very useful for solving noisy optimization problems; i.e., optimization problems in which the objective function is contaminated by noise. As the MAB problem can be viewed as a type of noisy optimization problem, the CE can be viable.

To introduce the CE idea, suppose the goal is to find the optimum of a function $S(x)$ on a set $\mathcal{X}$, where $S(x)$ is not known, but estimates $\hat{S}(x)$ can be obtained, e.g., by simulation. The CE method consists of the following steps:

1. Generate independent samples $X_1, \ldots, X_N$ from some probability distribution on $\mathcal{X}$, parameterized by a vector $\mathbf{v}$. For every $x \in \mathcal{X}$, the corresponding family of distributions should contain the "degenerate distribution" at x, which assigns all its probability mass to the point x.
2. Obtain estimates of the corresponding function values $\hat{S}(X_1), \ldots, \hat{S}(X_N)$, and identify the worst of the best $N_e = \rho N$ samples — the so-called *elite* samples. Typically, $\rho \in (0.01, 0.1)$.
3. Update the parameter $\mathbf{v}$ based on the elite samples. This involves the minimization of the Kullback-Leibler divergence (cross-entropy distance). In practice this often means that the parameters are updated according to their maximum likelihood estimates, using only the elite samples.

The method thus produces a sequence of parameters $\mathbf{v}_1, \mathbf{v}_2, \ldots$ that converges to (approximately) the parameter value that corresponds to the degenerate distribution at the maximizer x^*.

3 Cross-Entropy-based Multi-Armed Bandit (CEMAB)

3.1 The Method

The key idea of CEMAB is to transform the MAB problem into a simpler stochastic optimization problem, and then solve this simpler problem using CE. To this end, notice that, under the assumption that the reward distributions of the arms do not change over time, the maximum reward of MAB (i.e., (1)) can be simplified as follows:

$$\max_{(k_1,k_2,\ldots,k_T)\in\mathcal{K}^T} \sum_{t=1}^{T} \mu_{k_t} = T \cdot \max_{k\in\mathcal{K}} \mu_k \tag{3}$$

Solving the right hand side of (3) is in general simpler than solving the original MAB problem (left hand side of (3)) because, the solutions of the simplified problem lies in a space of size $|\mathcal{K}|$, while the solution of the original problem lies in a space of size $|\mathcal{K}|^T$. This difference in computational complexity becomes more pronounced as the number of arms increases. Therefore, to be effective in solving MABs with a large number of arms, CEMAB finds the best sequence of arms by searching the optimal arm to play.

Despite this simplification, the stochastic nature of MAB remains, as the expected reward μ_k of any arm $k \in \mathcal{K}$ is not known a priori and can only be estimated by playing the arm. Therefore, to keep the total reward high, CEMAB strives to avoid using arms with low rewards as much as possible when searching for the best arm. To this end, CEMAB adopts CE for noisy optimization and modifies it to suit the nature of the MAB problem. It uses the quantile statistics in CE and carefully adapts it to degrade the probability of selecting the bad arms gracefully, such that it can find the best arm quickly, while avoiding pulling arms with low reward as much as possible but without starving the arms that might be optimal. The CEMAB algorithm is presented in Algorithm 1, which is followed by a discussion of the algorithm.

Algorithm 1: CEMAB Method

```
Input: The number of arms $|\mathcal{K}|$, a (black box) function reward() to sample
       random rewards. CE parameters: sample size $N$, elite sample size
       $N_e = \rho N$ and learning rate $\alpha \in (0,1)$. Maximum number of plays $T$ (for
       simplicity, we assume $T = MN$).
Output: Total reward $G$.
1  Set $\boldsymbol{\mu} \leftarrow \mathbf{0}_{1\times|\mathcal{K}|}$, visits $\leftarrow \mathbf{0}_{1\times|\mathcal{K}|}$ and $\boldsymbol{p} \leftarrow (1/|\mathcal{K}|)_{1\times|\mathcal{K}|}$.
2  for $\tau \leftarrow 1$ to $M$ do
3      $A \leftarrow [\,]$                                        // empty matrix
4      for $i \leftarrow 1$ to $N$ do
5          Draw an arm $k$ from the discrete distribution parameterized by $\boldsymbol{p}$.
6          $r \leftarrow$ reward($k$)                      // Draw an immediate reward
7          $G \leftarrow G + r$.
8          visits($k$) $\leftarrow$ visits($k$) + 1.
9          $\boldsymbol{\mu}(k) \leftarrow \boldsymbol{\mu}(k) + (r - \boldsymbol{\mu}(k))/$visits($k$).
10         $A \leftarrow [X; [k, \boldsymbol{\mu}(k)]]$.           // Append row $[k, \boldsymbol{\mu}(k)]$ to $A$
11     $\tilde{\boldsymbol{p}} \leftarrow$ update($|\mathcal{K}|, N_e, A$)
12     Using the learning rate $\alpha$, update $\boldsymbol{p}$ as
                   $\boldsymbol{p} \leftarrow (1-\alpha)\,\boldsymbol{p} + \alpha\,\tilde{\boldsymbol{p}}$.          (4)
13 return $G$
```

To find the optimal arm k^* (i.e., the arm that solves the right-hand-side of (3)), CE starts by initializing the probability $\boldsymbol{p}$ of pulling a particular arm uniformly (Line 1). It iteratively chooses an arm (say $k \in \mathcal{K}$) to play, based on the probability $\boldsymbol{p}$, receives a reward r, which is drawn randomly from the unknown distribution D_k, and updates the estimated expected reward μ_k (Line 9). This sampling and estimation process repeats until one is confident that updating the selection probability $\boldsymbol{p}$ will benefit the optimization procedure. Once the probability is updated, the iterative sampling and estimation procedure repeats using the new selection probability.

Algorithm 2: `update`$(|\mathcal{K}|, N_e, A)$ for CEMAB-truncated

```
1 for $k \leftarrow 1$ to $|\mathcal{K}|$ do
2     Rearrange $A$ by sorting its rows according to the second column, from
        largest to smallest.
3     $\tilde{\boldsymbol{p}}(k) = \frac{1}{N_e} \sum_{j=1}^{N_e} I_{\{A(j,1)=k\}}$
4 return $\tilde{\boldsymbol{p}}$
```

Algorithm 3: `update`$(|\mathcal{K}|, -, A)$ for CEMAB-proportional

```
1 for $k \leftarrow 1$ to $|\mathcal{K}|$ do
2     For each arm $k$ sampled in $A$, get the latest estimate $\boldsymbol{\mu}(k)$.
3     $\tilde{\boldsymbol{p}}(k) = \frac{p_k \boldsymbol{\mu}(k)}{\sum_{j=1}^{K} p_j \boldsymbol{\mu}(j)}$.
4 return $\tilde{\boldsymbol{p}}$
```

Key to the performance of CE is how it updates its selection probability (Lines 11–12). A straightforward application of CE for noisy optimization would estimate the expected reward of all of the arms, and only after all estimates are improved, the probability $\boldsymbol{p}$ is updated. However, in the MAB problem, an estimate of the expected reward of any arm can only be improved by playing an arm, and each play incurs a reward. Therefore, CEMAB updates the probability $\boldsymbol{p}$ in an asynchronous manner: It clusters a sequence of N samples of the arm into a single batch and updates the probability $\boldsymbol{p}$ after each batch ends. Note that at the end of each batch the estimated expected reward of some of the arms may not have improved at all. Therefore, a smoothing mechanism (Line 12) is needed, to avoid being overcommitted to the new estimate of the different arms and also to guarantee that each arm has a non-zero probability of being visited.

Similar to most CE-based algorithms, CEMAB updates the probability $\boldsymbol{p}$ on the basis of the estimate $\hat{S}$ of the samples. The question is how the probability $\boldsymbol{p}$ should be updated based on the set of samples (Line 11). To this end, we propose two strategies: CEMAB-truncated and CEMAB-proportional. In CEMAB-truncated, we use the traditional CE updating formula, ignoring any arm that does not make it to the elite sample set. Specifically, the probability update rule for CEMAB-truncated is in Algorithm 2. In CEMAB-proportional, we assign the probability based on the estimated values $\hat{S}$ of each arm after the batch ends, and never set the probability of selecting an arm to be zero. The description of this update strategy is given in Algorithm 3.

3.2 Time Complexity and Convergence Properties

Similar to many state-of-the-art methods for solving MABs, such as Exp3 [3] and UCB [2], the most time-consuming part of CEMAB is its update step, i.e., Line 11 of Algorithm 1. For CEMAB-truncated, each update will take $O(N \log(N) + \max(|\mathcal{K}|, N_e))$, where the first component is due to sorting the samples within a batch (Line 2 of Algorithm 2). For CEMAB-proportional, each update will take $O(|\mathcal{K}|)$. Although Thompson sampling, Exp3 and UCB (current state-of-the-art methods) require $O(|\mathcal{K}|)$ for each update too, the number of updates for CEMAB is much less than for these two methods. Exp3 and UCB update their probability for selecting an arm at each step, but CEMAB updates its probability for selecting the arms only once per batch, i.e., $M = T/N$ times for a total of T plays.

CEMAB-proportional is guaranteed to converge to the optimal expected reward, assuming that the cumulative distribution function of the reward of the optimal arm is strictly increasing. The proof is a straightforward application of the proof of the CE-proportional algorithm for noisy optimization [7]. We do not have a theoretical proof that CEMAB-truncated will converge to the optimal expected reward. However, under the aforementioned assumption on the cumulative distribution function, CEMAB-truncated converges to the quantile of the total reward function. This proof is a straightforward application of the proof of the commonly used CE algorithm for noisy optimization in [8]. CEMAB-truncated is more aggressive in its distribution update compared to CEMAB-proportional, and therefore we can expect that CEMAB-truncated tends to converge to a particular arm faster than CEMAB-proportional, which is good if the quantile function of the total reward is equivalent to the expected total reward.

4 Experimental Results

The goal of our experiments are two-fold: First is to test the proposed methods against existing MAB methods on well-known benchmarks and understand the properties of the proposed methods better (Section 4.1), so as to also help us in setting the parameters for tests on large MAB problems. The second and ultimate goal is to test the performance of our proposed methods on large MAB problems (Section 4.2).

We compare the empirical performance of ε-greedy (with 0 initialization), ε-greedy (play once), Softmax, Exp3, and UCB1, with our proposed CE-based methods on discrete (Bernoulli) and continuous (truncated Gaussian) reward distributions. Note that we use two types of ε-greedy: One initializes the estimate of the expected reward to zero (denoted as E1), while the other initializes the estimate of the expected reward based on the reward received when playing the arm once (denoted as E2). The reason for these two versions is that we found significant performance differences between ε-greedy with these two different initializations, as will be seen later on.

4.1 Small-Scale MABs

Experimental Setup

We test our methods and comparators on 10 small-scale MAB problems, with up to 10 arms. Table 1 details the reward distributions of these problems.

The first 6 problems (i.e., B1–B6) are MABs with discrete reward distributions, which is the benchmark used in [2]. The reward of each arm in each of these problems is sampled from a Bernoulli distribution, where the success probability corresponds to the probability of generating a reward of 1 and the failure probability corresponds to the probability of generating a reward of 0.

For example, B1 defines an MAB with 2 arms, where the reward of arm 1 follows a Bernoulli distribution with success probability 0.9, while the reward of arm 2 follows a Bernoulli distribution with success probability 0.6. B1–B3 specify MABs with 2 arms and B4–B6 define MABs with 10 arms, where the reward of each arm is Bernoulli distributed. Note that B3 and B6 are relatively "difficult", because the reward of the optimal arm has a higher variance and the gaps $\mu^* - \mu_k$, $k = 1, \dots, 10$ are small.

The last 4 problems (i.e., G1–G4) are MABs with continuous reward distributions, in particular truncated normal distributions with support $[0, 1]$. Table 1 specifies the mean and standard deviation of the Gaussian distribution of each arm in each MAB problem. In this set of problems, G2 and G4 are quite challenging. The standard deviations of the reward distributions in these two problems are large and the support of these distributions also overlap significantly, which makes it difficult to distinguish between the best arm and bad arms.

Table 1: Bx refers to Bernoulli distributions and Gx to truncated Gaussian distributions.

	1	2	3	4	5	6	7	8	9	10
Mean of B1	0.9	0.6								
Mean of B2	0.9	0.8								
Mean of B3	0.55	0.45								
Mean of B4	0.9	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Mean of B5	0.9	0.8	0.8	0.8	0.7	0.7	0.7	0.6	0.6	0.6
Mean of B6	0.55	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
Mean of G1	0.3	0.6								
Std of G1	0.2	0.2								
Mean of G2	0.3	0.6								
Std of G2	0.6	0.2								
Mean of G3	0.5	0.2	0.4	0.3	0.8	0.1	0.7	0.8	0.3	0.9
Std of G3	0.3	0.2	0.3	0.1	0.1	0.2	0.5	0.4	0.2	0.1
Mean of G4	0.5	0.2	0.4	0.3	0.8	0.1	0.7	0.8	0.3	0.9
Std of G4	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

To set the parameters for testing, we first run a set of preliminary tests for each algorithm on each problem in Table 1 with a wide range of parameters. The parameters are summarized in Table 2. For each algorithm, the best parameters are those that maximize the most problems across the 10 MAB problems described above.

Results

Fig. 1 presents the performance of CEMAB and the comparator methods in B5, B6, G3, and G4, which are the more difficult problems among the 10 small problems defined in Table 1. The trend for the performance of the other MAB problems is similar, and hence we do not present them due to space constraints.

Table 2: Parameter Range

Method	Parameters Tested	Best Parameter
CEMAB-truncated	$N \in [50, 100], \rho \in [0.1, 0.5], \alpha \in [0.6, 1]$	$N = 50, \rho = 0.5, \alpha = 0.8$
CEMAB-proportional	$N \in [50, 100], \alpha \in [0.7, 1]$	$N = 50, \alpha = 0.7$
ε-greedy (E1 and E2)	$\varepsilon \in [0.01, 0.4]$	$\varepsilon = 0.1$ (E1), $\varepsilon = 0.05$ (E2)
Softmax	$t \in [0.01, 0.5]$	$t = 0.1$
Exp3	$\gamma \in [0.1, 0.7]$	$\gamma = 0.2$
UCB	$c \in [0.05, 3]$	$C = 0.1$

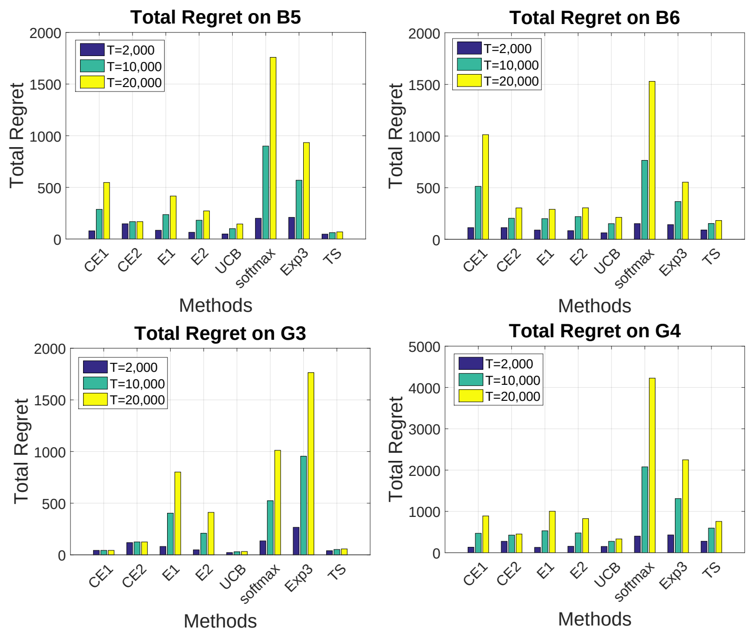

Fig. 1: The average total regret on 4 problem sets, B5, B6, G3 and G4. All algorithms use the best parameters and all experiments are repeated 50 times.

In this set of problems, TS achieves the lowest total regret in B5 and B6, followed by UCB and both CEMABs. In G3 and G4, UCB takes first place, while one of the CEMABs is second. The reason is that in B6 the best arm (i.e., arm 1) does not show significant performance difference at the beginning, and it is quite easy for CE-truncated to underestimate this arm and set a probability zero during updating step. Once this happens, CEMAB-truncated fails to identify the best arm, and as time progresses the difference in the total regret will become more apparent. However, by avoiding this aggressive update, CEMAB-proportional perform well and is similar to UCB in G4. It is important to note that the best empirical parameter here is $C = 0.1$, rather than the default value $C = 2$.

Softmax and Exp3, have the worst performance. In Softmax, the use of the Boltzmann distribution is likely to exaggerate an arm with a "good" estimate. For Exp3, it is important to note that this method is designed for non-stochastic MAB problems. a particular arm is highly influenced by only the current reward rather than the current estimate of the reward for each arm, which is a downside for stochastic problems, which we address in this paper.

It is also interesting to note that the performance of the simplest algorithm, ε-greedy, differs significantly when applying different initializations. Variant E1 initializes the reward estimate of each arm with 0, while variant E2 initializes the reward estimate based on the reward received when playing the arm once. The performance of E2 is better in this set of small-scale problems. However, in the next section we will see that E1 is better for large-scale problems.

4.2 Large-scale MABs

To assess CEMAB's performance for large problems, we test the algorithms on problems with an increasing number of arms. For each number of arms, we test the algorithms on four different MAB problems, as shown in Table 3, which consists of two LB (Large Bernoulli) that represent MABs whose reward distributions are Bernoulli distributed and two LG (Large Gaussian) that represent MABs whose reward distributions are truncated Gaussian with support $[0, 1]$.

Table 3: Large-scale MAB Settings

LB1	$\mu_k \sim \mathsf{U}(0, 1)$
LB2	10% of $\mu_k \sim \mathsf{U}(0.75, 1)$ and the rest of $\mu_k \sim \mathsf{U}(0, 0.25)$
LG1	$\mu_k \sim \mathsf{U}(0, 1), \sigma_k \sim \mathsf{U}(0, 0.25)$
LG2	10% of $\mu_k \sim \mathsf{U}(0.75, 1)$ and the rest of $\mu_k \sim \mathsf{U}(0, 0.25), \sigma_k \sim \mathsf{U}(0, 0.25)$

For LB1, the success probability (i.e., the probability of sampling a reward of 1) of each reward distribution is uniformly sampled from $(0, 1)$. For LB2, 10% of the arms have rewards drawn from a Bernoulli distribution whose success probability is sampled from $(0.75, 1)$ and 90% have rewards drawn from a Bernoulli distribution whose success probability is sampled from $(0, 0.25)$. For LG1, the means are uniformly sampled from interval $(0, 1)$, while for LG2, 10% of the means are sampled from $(0.75, 1)$ uniformly at random and 90% are sampled from $(0, 0.25)$ uniformly at random. The standard deviations for both LG1 and LG2 are sampled uniformly at random from $(0, 0.25)$ for each arm. All of these parameters for the reward distributions are sampled independently for each arm. It is not hard to see that LB2 and LG2 is harder than LB1 and LG1, since it requires a strategy that has a good capability of exploring, rather than keep playing the best arm so far.

For these tests, each algorithm uses the best parameters as found in Table 2. The results of these tests for $|\mathcal{K}| = 100, 1000$, and 10000 are summarized in Table 4. The results indicate that, as the number of arms increases, CEMAB outperforms all other methods, including UCB. The reason for the significantly decreasing performance of UCB is that it must play each arm at least once to estimate the performance of each arm, so that it can converge to the optimal solution. However, exactly because of this, its performance becomes impractical as the number of arms increases. On the other hand, CEMAB incrementally improves its estimate on the performance of the arms based on sampling, without ever requiring to play the entire set of arms at first. This causes the convergence property of CEMAB to be weaker than UCB, but its empirical performance to be significantly better in large problems.

It is also interesting to note that the simple ε-greedy with zero initial estimate (E1) is a relatively strong competitor. In fact, for problems with a large number of arms, this simple methods is a stronger competitor than the state-of-the-art UCB. Note that for the Gaussian reward case, the gap between rewards is much less than for the Bernoulli case. As a result, even if an arm that is played is not very good, the reward obtained by playing a better arm will not be much higher. This could be a reason why the performance of E1 is comparable to CEMAB's in the Gaussian, while it loses in the Bernoulli case.

Table 4: The average total reward for large MABs. All algorithms use the best parameters (as per Table 2) and all experiments are repeated 200 times. The best method is highlighted in boldface. If the difference between the best and second best method is not statistically significant (meaning that one method lies in the 95% confidence interval of the other), we highlight both of them.

		LB1				LB2			
		The number of plays T				The number of plays T			
$\|\mathcal{K}\|$	Method	$1,000$	$5,000$	$10,000$	$20,000$	$1,000$	$5,000$	$10,000$	$20,000$
100	CE1	**893**	**4749**	**9569**	19207	718	4106	8342	16817
	CE2	806	4612	9463	19184	741	4192	8571	17341
	E1	864	4572	9230	18547	**820**	4139	8288	16597
	E2	868	4686	9460	19010	767	4167	8437	17022
	UCB	833	4674	9538	19225	787	**4311**	**8788**	**17796**
	softmax	859	4407	8866	17798	801	4127	8305	16686
	Exp3	580	3564	7702	16285	241	2610	6202	13550
	TS	856	4695	9557	**19323**	678	4168	8634	17615
$1,000$	CE1	**896**	**4788**	**9651**	**19380**	784	4493	9130	18402
	CE2	810	4655	9559	**19409**	792	**4538**	**9297**	18850
	E1	868	4510	9131	18496	771	4352	8885	18019
	E2	500	4129	8802	18532	197	3852	8635	18217
	UCB	499	3812	8358	17906	197	3730	8568	18361
	softmax	875	4470	8989	18043	**853**	4445	8976	18074
	Exp3	507	2666	5643	12378	202	1122	2601	7114
	TS	580	4027	8957	18886	305	3949	8925	**18897**
$10,000$	CE1	**895**	**4784**	**9644**	**19367**	795	4554	9251	18649
	CE2	809	4655	9554	**19393**	**800**	**4574**	**9358**	**18969**
	E1	863	4479	9025	18158	761	4322	8812	17818
	E2	515	2492	5013	14126	181	1004	2007	11022
	UCB	515	2486	5007	12135	182	1004	2007	9880
	softmax	832	4341	8786	17734	776	4264	8725	17734
	Exp3	500	2510	5040	10154	201	1006	2026	4109
	TS	511	2707	5811	13091	207	1191	3081	10433

		LG1				LG2			
		The number of plays T				The number of plays T			
$\|\mathcal{K}\|$	Method	$1,000$	$5,000$	$10,000$	$20,000$	$1,000$	$5,000$	$10,000$	$20,000$
100	CE1	**885**	**4624**	**9297**	**18644**	772	4378	8884	17897
	CE2	800	4484	9150	18512	767	4435	9068	18341
	E1	868	4463	8973	17995	783	4411	8977	18109
	E2	877	4562	9177	18415	**879**	4685	9443	18956
	UCB	804	4450	9110	18506	863	**4777**	**9705**	**19578**
	softmax	833	4242	8513	17067	813	4282	8682	17551
	Exp3	559	3449	7457	15737	281	2752	6678	14878
	TS	785	4406	9071	18477	851	**4779**	**9706**	**19580**
$1,000$	CE1	**876**	**4599**	**9254**	18564	778	**4405**	8937	18004
	CE2	782	4441	9109	18499	763	4381	**8990**	18236
	E1	865	4532	9180	18542	768	4340	**8895**	18043
	E2	499	4358	9183	**18835**	238	4051	8818	**18351**
	UCB	500	3560	7750	16852	238	3440	8071	17711
	softmax	811	4125	8281	16609	**816**	4235	8535	17165
	Exp3	504	2624	5493	11865	242	1287	2831	7095
	TS	563	3713	8186	17729	312	3670	8378	18004
$10,000$	CE1	**877**	**4612**	**9281**	18618	**779**	**4394**	8913	17951
	CE2	788	4470	9153	18572	770	4369	**8934**	**18113**
	E1	876	4568	**9247**	**18656**	**774**	4313	**8815**	17894
	E2	488	2475	4997	14736	241	1207	2394	11942
	UCB	488	2477	4997	12390	240	1207	2393	9131
	softmax	794	4102	8275	16648	740	4030	8227	16690
	Exp3	501	2506	5027	10108	239	1200	2412	4865
	TS	507	2661	5637	12469	244	1339	3164	9783

5 Conclusion

We proposed a new approach, CEMAB, for solving MABs with a large number of discrete arms. It uses the Cross-Entropy method as a noisy optimization method to search for the best arm with as little regret as possible. We presented and evaluated the CEMAB algorithm with two variants for the updating procedure. Using results on CE for noisy optimization, one of the variants is guaranteed to converge to the optimal arm, under certain conditions on the reward function. Empirical results on a number of MAB problems with an increasing number of arms indicate that CEMAB outperforms state-of-the-art methods.

Acknowledgments

This work was supported by the Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS) under grant number CE140100049. Erli Wang would also like to acknowledge the support from the University of Queensland through the UQ International Scholarships scheme.

References

1. Agrawal, S., Goyal, N.: Analysis of thompson sampling for the multi-armed bandit problem. In: COLT. pp. 39–1 (2012)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning 47(2-3), 235–256 (2002)
3. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The non-stochastic multi-armed bandit problem. SIAM Journal on Computing 32(1), 48–77 (2002)
4. Bubeck, S., Munos, R., Stoltz, G., Szepesvari, C.: X-armed bandits. Journal of Machine Learning Research 12(May), 1655–1695 (2011)
5. Burtini, G., Loeppky, J., Lawrence, R.: A survey of online experiment design with the stochastic multi-armed bandit. arXiv preprint arXiv:1510.00757 (2015)
6. Coquelin, P.A., Munos, R.: Bandit algorithms for tree search. In: UAI. pp. 67–74 (2007)
7. Goschin, S., Littman, M.L., Ackley, D.H.: The effects of selection on noisy fitness optimization. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. pp. 2059–2066. ACM (2011)
8. Goschin, S., Weinstein, A., Littman, M.L.: The cross-entropy method optimizes for quantiles. In: ICML (3). pp. 1193–1201 (2013)
9. Robbins, H.: Some aspects of the sequential design of experiments. Bulletin of the American Mathematical Society 58(5), 527–535 (1952)
10. Rubinstein, R.Y., Kroese, D.P.: The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning. Springer (2004)
11. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (1998)
12. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25(3/4), 285–294 (1933)
13. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, University of Cambridge England (1989)