Knapsack based Optimal Policies for Budget–Limited Multi–Armed Bandits

Long Tran–Thanh
Faculty of Physical and Applied Sciences
University of Southampton, UK
ltt08r@ecs.soton.ac.uk

Archie Chapman
The University of Sydney Business School
Sydney, Australia
a.chapman@econ.usyd.edu.au

Alex Rogers
Faculty of Physical and Applied Sciences
University of Southampton, UK
acr@ecs.soton.ac.uk

Nicholas R. Jennings
Faculty of Physical and Applied Sciences
University of Southampton, UK
nrj@ecs.soton.ac.uk

Abstract

In budget–limited multi–armed bandit (MAB) problems, the learner's actions are costly and constrained by a fixed budget. Consequently, an optimal exploitation policy may not be to pull the optimal arm repeatedly, as is the case in other variants of MAB, but rather to pull the sequence of different arms that maximises the agent's total reward within the budget. This difference from existing MABs means that new approaches to maximising the total reward are required. Given this, we develop two pulling policies, namely: (i) KUBE; and (ii) fractional KUBE. Whereas the former provides better performance up to 40% in our experimental settings, the latter is computationally less expensive. We also prove logarithmic upper bounds for the regret of both policies, and show that these bounds are asymptotically optimal (i.e. they only differ from the best possible regret by a constant factor).

1 Introduction

The standard multi–armed bandit (MAB) problem was originally proposed by Robbins (1952), and presents one of the clearest examples of the trade–off between *exploration* and *exploitation* in reinforcement learning. In the standard MAB problem, there are K arms of a single machine, each of which delivers rewards that are independently drawn from an unknown distribution when an arm of the machine is pulled. Given this, an agent must choose which of these arms to pull. At each time step, it pulls one of the machine's arms and receives a reward or payoff. The agent's goal is to maximise its return; that is, the expected sum of the rewards its receives over a sequence of pulls. As the reward distributions differ from arm to arm, the goal is to find the arm with the highest expected payoff as early as possible, and then to keep playing using that best arm. However, the agent does not know the rewards for the arms, so it must sample them in order to learn which is the optimal one. In other words, in order to choose the optimal arm (exploitation) the agent first has to estimate the mean rewards of all of the arms (exploration). In the standard MAB, this trade–off has been effectively balanced by decision–making policies such as *upper confidence bound* (UCB) and ϵ_n*–greedy* (Auer, Cesa-Bianchi, and Fischer 2002).

However, this MAB model gives an incomplete description of the sequential decision–making problem facing an agent in many real–world scenarios. To this end, a variety of other related models have been studied recently, and, in particular, a number of researchers have focused on MABs with budget constraints, where arm–pulling is costly and is limited by a fixed budget (Bubeck, Munos, and Stoltz 2009; Guha and Munagala 2007; Antos, Grover, and Szepesvári 2008). In these models, the agent's exploration budget limits the number of times it can sample the arms in order to estimate their rewards, which defines an initial exploration phase. In the subsequent cost–free exploitation phase, an agent's policy is then simply to pull the arm with the highest expected reward. However, in many settings, it is not only the exploration phase, but the exploitation phase that is also limited by a cost budget. To address this limitation, a new bandit model, the *budget–limited MAB*, was introduced (Tran-Thanh *et al.* 2010). In this model, pulling an arm is again costly, but crucially *both* the exploration and exploitation phases are limited by a *single budget*. This type of limitation is well motivated by several real–world applications. For example, in many wireless sensor network applications, a sensor node's actions, such as sampling or data forwarding, consume energy, and therefore the number of actions is limited by the capacity of the sensor's batteries (Padhy *et al.* 2010). Furthermore, many of these scenarios require that sensors learn the optimal sequence of actions that can be performed, with the goal of maximising the long term value of the actions they take (Tran-Thanh, Rogers, and Jennings 2011). In such settings, each action can be considered as an arm, with a cost equal to the amount of energy needed to perform that task. Now, because the battery is limited, both the exploration (i.e. learning the rewards tasks) and exploitation (i.e. taking the optimal actions given reward estimates) phases are budget limited.

Against this background, Tran-Thanh *et al.* (2010) showed that the budget–limited MAB cannot be derived from any other existing MAB model, and therefore, previous MAB learning methods are not suitable to efficiently deal with this problem. Thus, they proposed a simple budget–limited ε*–first* approach for the budget–limited MAB. This splits the overall budget B into two portions, the first εB of which is used for exploration, and the remaining $(1-\varepsilon)B$ for exploitation. However, this budget–limited ε–first method suffers from a number of drawbacks. First, the performance of ε–first approaches depend on the value of ε chosen. In particular, high values guarantee accurate exploration but

inefficient exploitation, and *vice versa*. Given this, finding a suitable ε for a particular problem instance is a challenge, since settings with different budget limits or arm costs (which are not known beforehand) will typically require different values of ε. In addition, even with a good ε value, the method typically provides poor efficiency in terms of minimising its performance regret (defined as the difference between its performance and that of the optimal policy), which is a standard measure of performance. In particular, the regret bound that ε–first provides is $O\left(B^{\frac{2}{3}}\right)$, where B is the budget limit, whereas the theoretical best possible regret bound is typically a logarithmic function of the number of pulls[1] (Lai and Robbins 1985).

To address this shortcoming, in this paper we propose two new learning algorithms, called KUBE (for knapsack–based upper confidence bound exploration and exploitation) and *fractional* KUBE, that do not explicitly separate exploration from exploitation. Instead, they explore and exploit at the same time by adaptively choosing which arm to pull next, based on the current estimates of the arms' rewards. In more detail, at each time step, KUBE calculates the best set of arms that provides the highest total upper confidence bound of the estimated expected reward, and still fits into the residual budget, using an unbounded knapsack model to determine this best set (Kellerer, Pferschy, and Pisinger 2004). However, since unbounded knapsack problems are known to be NP–hard, the algorithm uses an efficient approximation method taken from the knapsack literature, called the *density–ordered greedy* approach, in order to estimate the best set (Kohli, Krishnamurti, and Mirchandani 2004). Following this, KUBE then uses the frequency that each arm occurs within this approximated best set as a *probability* with which to randomly choose an arm to pull in the next time step. The reward that is received is then used to update the estimate of the upper confidence bound of the pulled arm's expected reward, and the unbounded knapsack problem is solved again. The intuition behind this algorithm is that if we know the real value of the arms, then the budget–limited MAB can be reduced to an unbounded knapsack problem, where the optimal solution is to subsequently pull from the set of arms that forms the solution of the knapsack problem. Given this, by randomly choosing the next arm from the current best set at each time step, the agent generates an accurate estimate of the true optimal solution (i.e. real best set of arms), and, accordingly, the sequence of pulled arms will converge to this optimal set. In a similar vein, fractional KUBE also estimates the best set of arms that provides the highest total upper confidence bound of the estimated expected reward at each time step, and uses the frequency that each arm occurs within this approximated best set as a probability to randomly pull the arms. However, instead of using the density–ordered greedy to solve the underlying unbounded knapsack problem, fractional KUBE relies on a computationally less expensive approach, namely the *fractional relaxation based* algorithm (Kellerer, Pferschy, and Pisinger 2004). Given this, fractional KUBE requires less computation than KUBE.

To analyse the performance of KUBE and its fractional counterpart in terms of minimising the regret, we devise proveably asymptotically optimal upper bounds on their *performance regret*. That is, our proposed upper bounds differ from the best possible one only with a constant factor. Following this, we numerically evaluate the performance of the proposed algorithms against a state–of–the–art method, namely the buget–limited ε–first approach, in order to demonstrate that our algorithms are the first that can achieve this optimal bound. In addition, we show that KUBE typically outperforms its fractional counterpart by up to 40%, however, this results in an increased computational cost (from $O(K)$ to $O(K \ln K)$). Given this, the main contributions of this paper are:

- We introduce KUBE and fractional KUBE, the first budget–limited MAB learning algorithms that proveably achieve a $O(\ln B)$ theoretical upper bound on the regret, where B is the budget limit.
- We demonstrate that with an increased computational cost, KUBE outperforms fractional KUBE in the experiments. We also show that while both algorithms achieve logarithmic regret bounds, the buget–limited ε–first approaches fail to do so.

The paper is organised as follows: Next we describe the budget–limited MAB. We then introduce our two learning algorithms in Section 3. In Section 4 we provide regret bounds on the performance of the proposed algorithms. Following this, Section 5 presents an empirical comparison of KUBE and its fractional counterpart with the ε–first approach. Section 6 concludes.

2 Model Description

The budget–limited MAB model consists of a machine with K arms, one of which must be pulled by the agent at each time step. By pulling arm i, the agent has to pay a pulling cost, denoted with c_i, and receives a non–negative reward drawn from a distribution associated with that specific arm. The agent has a cost budget B, which it cannot exceed during its operation time (i.e. the total cost of pulling arms cannot exceed this budget limit). Now, since reward values are typically bounded in real–world applications, we assume that each arm's reward distribution has bounded supports. Let μ_i denote the mean value of the rewards that the agent receives from pulling arm i. Within our model, the agent's goal is to maximise the sum of rewards it earns from pulling the arms of the machine, with respect to the budget B. However, the agent has no initial knowledge of the μ_i of each arm i, so it must learn these values in order to deduce a policy that maximises its sum of rewards. Given this, our objective is to find the optimal pulling algorithm, which maximises the expectation of the total reward that the agent can achieve, without exceeding the cost budget B.

Formally, let A be an arm–pulling algorithm, giving a finite sequence of pulls. Let $N_i^A(B)$ be the random variable that represents the number of pulls of arm i by A, with re-

[1] Note that in the budget–limited MAB, the budget B determines the number of pulls. Thus, a logarithmic function of the number of pulls is also a logarithmic function of the budget.

spect to the budget limit B. Since the total cost of the sequence A cannot exceed B, we have:

$$P\left(\sum_{i}^{K} N_i^A(B)\, c_i \leq B\right) = 1. \quad (1)$$

Let $G(A)$ be the total reward earned by using A to pull the arms. The expectation of $G(A)$ is:

$$\mathbb{E}[G(A)] = \sum_{i}^{K} \mathbb{E}\left[N_i^A(B)\right]\mu_i. \quad (2)$$

Then, let A^* denote an optimal solution that maximises the expected total reward, that is:

$$A^* = \operatorname*{argmax}_A \sum_{i}^{K} \mathbb{E}\left[N_i^A(B)\right]\mu_i. \quad (3)$$

Note that in order to determine A^*, we have to know the value of μ_i in advance, which does not hold in our case. Thus, A^* represents a theoretical optimum value, which is unachievable in general.

Nevertheless, for any algorithm A, we can define the regret for A as the difference between the expected cumulative reward for A and that of the theoretical optimum A^*. More precisely, letting $R(A)$ denote the regret, we have:

$$R(A) = \mathbb{E}[G(A^*)] - \mathbb{E}[G(A)]. \quad (4)$$

Given this, our objective is to derive a method of generating a sequence of arm pulls that minimises this regret for the class of MAB problems defined above.

3 The Algorithms

Given the model described in the previous section, we now introduce two learning methods, KUBE and fractional KUBE, that efficiently deal with the challenges discussed in Section 1. Recall that at each time step of the algorithms, we determine the optimal set of arms that provides the best total estimated expected reward. Due to the similarities of our MAB to unbounded knapsack problems when the rewards are known, we use techniques taken from the unbounded knapsack domain. Thus, in this section, we first introduce the unbounded knapsack problem, and then show how to use knapsack methods in our algorithms.

3.1 The Unbounded Knapsack Problem

The unbounded knapsack problem is formulated as follows. A knapsack of weight capacity C is to be filled with some set of K different types of items. Each item type $i \in K$ has a corresponding value v_i and weight w_i, and the problem is to select a set that maximises the total value of items in the knapsack, such that their total weight does not exceed the knapsack capacity C. That is, the goal is to find the non–negative integers $\{x_i\}_{i=1}^{K}$ that maximise:

$$\sum_{i=1}^{K} x_i v_i, \quad (5)$$

$$\text{s.t.} \quad \sum_{i=1}^{K} x_i w_i \leq C,$$

$$\forall i \in \{1, \ldots, K\} : x_i \text{ integer}.$$

Note that this problem is a generalisation of the standard knapsack problem, in which $x_i \in \{0, 1\}$; that is, each item type contains only one item, and we can either choose it or not. The unbounded knapsack problem is *NP*–hard. However, near–optimal approximation methods have been proposed to solve it (a detailed survey can be found in (Kellerer, Pferschy, and Pisinger 2004)). Among these approximation methods, a simple, but efficient approach is the *density–ordered greedy* algorithm, and here we make use of this method. In more detail, the density–ordered greedy algorithm has $O(K \log K)$ computational complexity, where K is the number of item types (Kohli, Krishnamurti, and Mirchandani 2004). This algorithm works as follows. Let v_i/w_i denote the *density* of type i. To begin, the item types are sorted in order of their density, which is an operation of $O(K \log K)$ computational complexity. Next, in the first round of this algorithm, as many units of the highest density item are selected as is feasible without exceeding the knapsack capacity. Then, in the second round, the densest item of the remaining feasible items is identified, and as many units of it as possible are selected. This step is repeated until there are no feasible items left (i.e. at most K rounds).

Another way to approximate the optimal solution of the unbounded knapsack problem is the *fractional relaxation based* algorithm. This relaxes the original problem to its fractional version. In particular, within the *fractional unbounded knapsack problem* we allow x_i to be fractional. Now, it is easy to show that the optimal solution of the fractional unbounded knapsack is to solely choose $I^* = \arg\max_i v_i/w_i$ (i.e. I^* is the item type with the highest density) (Kellerer, Pferschy, and Pisinger 2004). That is, if $\mathbf{x}^* = \langle x_1^*, \ldots, x_1^* \rangle$ denotes the optimal solution of the fractional unbounded knapsack, then $x_{I^*}^* = C/w_{I^*}$, while $\forall j \neq I^*$, $x_j = 0$. Given this, within the original unbounded knapsack problem (where x_i are integers), the fractional relaxation based algorithm chooses $x_{I^*} = \lfloor C/w_{I^*} \rfloor$, and $x_j = 0$, $\forall j \neq I^*$. It can easily shown that the complexity of this algorithm is $O(K)$, which is the cost of determining the highest density type.

3.2 KUBE

The KUBE algorithm is depicted in Algorithm 1. Here, let t denote the time step, and B_t denote the residual budget at time $t \geq 1$, respectively. Note that at the start (i.e. $t = 1$), $B_1 = B$, where B is the total budget limit. At each subsequent time step, t, KUBE first checks that arm pulling is feasible. That is, it is feasible only if at least one of the arms can be pulled with the remaining budget. Specifically, if $B_t < \min_j c_j$ (i.e. the residual budget is smaller than the lowest pulling cost), then KUBE stops (steps $3-4$).

If arm pulling is still feasible, KUBE first pulls each arm once in the initial phase (steps $6-7$). Following this, at each time step $t > K$, it estimates the best set of arms according to their upper confidence bound using the density–ordered greedy approximation method applied to the following prob-

lem:

$$\max \sum_{i=1}^{K} m_{i,t} \left(\hat{\mu}_{i,n_{i,t}} + \sqrt{\frac{2 \ln t}{n_{i,t}}} \right) \quad (6)$$

$$\text{s.t.} \quad \sum_{i=1}^{K} m_{i,t} c_i \leq B_t, \ \forall i, t : m_{i,t} \text{ integer.}$$

In the above expression, $\hat{\mu}_{i,n_{i,t}}$ is the current estimate of arm i's expected reward (calculated as the average reward received so far from pulling arm i), $n_{i,t}$ is the number of pulls of arm i until time step t, and $\sqrt{\frac{2 \ln t}{n_{i,t}}}$ is the size of the upper confidence interval. The goal, then, is to find integers $\{m_{i,t}\}_{i \in K}$ such that Equation 6 is maximised, with respect to the residual budget limit B_t (n.b. from here on, we drop the subscript $i \in K$ on this set). Since this problem is NP–hard, we use the density–ordered greedy method to find a near–optimal set of arms (step 9). Note that the upper confidence bound on arm i's density is:

$$\frac{\hat{\mu}_{i,n_{i,t}}}{c_i} + \frac{\sqrt{\frac{2 \ln t}{n_{i,t}}}}{c_i}. \quad (7)$$

Let $M^*(B_t) = \{m^*_{i,t}\}$ be this method's solution to the problem in Equation 6, giving us the desired set of arms, where $m^*_{i,t}$ is an index of arm i that indicates how many times arm i is taken into account within the set. Using $\{m^*_{i,t}\}$, KUBE *randomly* chooses the next arm to pull, $i(t)$, by selecting arm i with probability (step 10):

$$P(i(t) = i) = \frac{m^*_{i,t}}{\sum_{k=1}^{K} m^*_{k,t}}. \quad (8)$$

After the pull, it then updates the estimated upper bound of the chosen arm, and the residual budget limit B_t (steps 12 – 13).

The intuition behind KUBE is the following. By repeatedly drawing the next arm to pull from a distribution formed by the current estimated approximate best set, the expected reward of KUBE equals the average reward for following the optimal solution to the corresponding unbounded knapsack problem, given the current reward estimates. If the true values of the arms were known, then this would imply that the average performance of KUBE efficiently converges to the optimal solution of the unbounded knapsack problem reduced from the budget–limited MAB model. It is easy to show that the optimal solution of this knapsack model forms the theoretical optimal policy of the budget–limited MAB in case of having full information. Put differently, if the mean reward value of each arm is known, then the budget–limited problem can be reduced to the unbounded knapsack problem, and thus, the optimal solution of the knapsack problem is the optimal solution of the budget–limited MAB as well. In addition, by combining the upper confidence bound with the estimated mean values of the arms, we guarantee that an arm that is not yet sampled many times may be pulled more frequently, since its upper confidence interval is large. Thus, we explore and exploit at the same time (for more details, see (Auer, Cesa-Bianchi, and Fischer 2002; Audibert, Munos, and Szepesvári 2009)). Note that, by using the density–ordered greedy method, KUBE achieves a $O(K \ln K)$ computational cost per time step.

Algorithm 1 The KUBEAlgorithm

1: $t = 1$; $B_t = B$; $\gamma > 0$;
2: **while** pulling is feasible **do**
3: **if** $B_t < \min_i c_i$ **then**
4: STOP! {pulling is not feasible}
5: **end if**
6: **if** $t \leq K$ **then**
7: Initial phase: play arm $i(t) = t$;
8: **else**
9: use density–ordered greedy to calculate $M^*(B_t) = \{m^*_{i,t}\}$, the solution of Equation 6;
10: randomly pull $i(t)$ with $P(i(t) = i) = \frac{m^*_{i,t}}{\sum_{k=1}^{K} m^*_{k,t}}$;
11: **end if**
12: update the estimated upper bound of arm $i(t)$;
13: $B_{t+1} = B_t - c_{i(t)}$; $t = t + 1$;
14: **end while**

3.3 Fractional KUBE

We now turn to the fractional version of KUBE, which follows the underlying concept of KUBE. It also approximates the underlying unbounded knapsack problem at each time step t in order to determine the frequency of arms within the estimated best set of arms. However, it differs from KUBE by using the fractional relaxation based method to approximate the unbounded knapsack in Step 9 of Algorithm 1. Crucially, fractional KUBE uses the fractional relaxation based algorithm to solve the following fractional unbounded knapsack problem at each t:

$$\max \sum_{i=1}^{K} m_{i,t} \left(\hat{\mu}_{i,n_{i,t}} + \sqrt{\frac{2 \ln t}{n_{i,t}}} \right) \quad \text{s.t.} \quad \sum_{i=1}^{K} m_{i,t} c_i \leq B_t. \quad (9)$$

Recall that within KUBE, the frequency of arms within the approximated solution of the unbounded knapsack forms a probability distribution from which the agent randomly pulls the next arm. Now, since the fractional relaxation based algorithm solely chooses the arm (i.e. item type) with the highest estimated confidence bound–cost ratio (i.e. item density), fractional KUBE does not need to randomly choose an arm. Instead, at each time step t, it pulls the arm that maximises $\left(\hat{\mu}_{i,n_{i,t}}/c_i + \sqrt{\frac{2 \ln t}{n_{i,t}}}/c_i \right)$. That is, fractional KUBE can also be seen as the budget–limited version of UCB (see (Auer, Cesa-Bianchi, and Fischer 2002) for more details of UCB).

Computation–wise, by replacing the density–ordered greedy with the fractional relaxation based algorithm, fractional KUBE decreases the computational cost to $O(K)$ per time step. In what follows, we show that both KUBE and its fractional counterpart achieve asymptotically optimal regret bounds.

4 Performance Analysis

We now focus on the analysis of the expected regret of KUBE and fractional KUBE, defined by Equation 4. To this end, we: (i) derive an upper bound on the regret, and (ii) show that these bounds are asymptotically optimal.

To begin, let us state some simplifying assumptions and define some useful terms. Without loss of generality, for ease of exposition we assume that the reward distribution of each arm has support in $[0, 1]$, and that the pulling cost $c_i \geq 1$ for each i (our result can be scaled for different size supports and costs as appropriate). Let $I^* = \arg\max_i \mu_i/c_i$ be the arm with the highest true mean value density. For the sake of simplicity, we assume that I^* is unique (however, our proofs do not exploit this fact). Let $d_{\min} = \min_{j \neq I^*} \{\mu_{I^*}/c_{I^*} - \mu_j/c_j\}$ denote the minimal true mean value density difference of arm I^* and that of any other arm j. In addition, let $c_{\min} = \min_j c_j$ and $c_{\max} = \max_j c_j$ denote the smallest and largest pulling costs, respectively. Then let $\delta_j = c_j - c_{I^*}$ be the difference of arm j's pulling cost and the minimal pulling cost. Similarly, let $\Delta_j = \mu_{I^*} - \mu_j$ denote the difference of the highest true mean value and that of arm j. Note that both δ_j and Δ_j could be negative values, since I^* does not necessarily have the highest true mean value, nor the smallest pulling cost. In addition, let T denote the finite–time operating time of the agent.

Now, we first analyse the performance of KUBE. In what follows, we first estimate the number of times we pull arm $j \neq I^*$, instead of I^*. Based on this result, we estimate $\mathbb{E}[T]$, the average number of pulls of KUBE. This bound guarantees that KUBE always pulls "enough" arms so that the difference of the number of pulls in the theoretical optimal solution and that of KUBE is small, compared to the size of the budget. By using the estimated value of $\mathbb{E}[T]$, we then show that KUBE achieves a $O(\ln(B))$ worst case regret on average. In more detail, we get:

Theorem 1 (Main result 1) *For any budget size $B > 0$, the performance regret of* KUBE *is at most*

$$\left(\frac{8}{d_{\min}^2} + \left(\frac{c_{\max}}{c_{\min}}\right)^2\right)\left(\sum_{\Delta_j>0}\Delta_j + \sum_{\delta_j>0}\frac{\delta_j}{c_{I^*}}\right)\ln\left(\frac{B}{c_{\min}}\right) + \left(\sum_{\Delta_j>0}\Delta_j + \sum_{\delta_j>0}\frac{\delta_j}{c_{I^*}}\right)\left(\frac{\pi^2}{3}+1\right) + 1\,.$$

It is easy to show that for each $j \neq I^*$, at least one between δ_j and Δ_j has to be positive. This implies that $\left(\sum_{\Delta_j>0}\Delta_j + \sum_{\delta_j>0}\frac{\delta_j}{c_{I^*}}\right) > 0$. That is, the performance regret of KUBE (i.e. $R(\text{KUBE})$) is upper–bounded by $O(\ln B)$.

In a similar vein, we can show that the regret of fractional KUBE is bounded as follows:

Theorem 2 (Main result 2) *For any budget size $B > 0$, the performance regret of fractional* KUBE *is at most*

$$\frac{8}{d_{\min}^2}\left(\sum_{\Delta_j>0}\Delta_j + \sum_{\delta_j>0}\frac{\delta_j}{c_{I^*}}\right)\ln\left(\frac{B}{c_{\min}}\right) + \left(\sum_{\Delta_j>0}\Delta_j + \sum_{\delta_j>0}\frac{\delta_j}{c_{I^*}}\right)\left(\frac{\pi^2}{3}+1\right) + 1\,.$$

Note that the regret bound of fractional KUBE is better (i.e. the constant factor within the regret bound of fractional KUBE is smaller than that of KUBE). However, this does not indicate that fractional KUBE has better performance in practice. This implies that these bounds are not tight. In fact, as we will demonstrate in Section 5, KUBE typically outperforms its fractional counterpart by up to 40%.

Having established a regret bound for the two algorithms, we now move on to show that they produce optimal behaviour, in terms of minimising the regret.

Theorem 3 (Main result 3) *For any arm pulling algorithm, there exists a constant $C \geq 0$, and a particular instance of the budget–limited MAB problem, such that the regret of that algorithm within that particular problem is at least $C \ln B$.*

Now, since the performance regret of both algorithms is $O(\ln(B))$, Theorem 3 indicates that their performance is asymptotically optimal (i.e. their performance differs from that of the optimal policy by a constant factor).

Intuitively, to prove Theorems 1 and 2, we first estimate the expected value of the total number of pulls T, which is a random variable (unlike in the case of standard MABs, where T is fixed). In addition, for each value of T, we estimate the conditional probability of pulling suboptimal arms (conditional to the value of T). Based on these results, we then estimate the regret of the algorithms. To prove Theorem 3, we reduce the standard MAB into an instance of the budget–limited bandit model. Due to lack of space, the proof of the abovementioned theorems are omitted, but they are available in (Tran-Thanh et al. 2012).

The results in Theorem 1 and 2 can be interpreted to the standard MAB domain as follows. The standard MAB can be reduced to a budget–limited MAB by setting all the pulling costs to be the same. Given this, $B/c_{\min} = T$ in any sequence of pulls. This implies that both KUBE and fractional KUBE achieve $O(\ln T)$ regret within the standard MAB domain, which is optimal (Lai and Robbins 1985; Auer, Cesa-Bianchi, and Fischer 2002).

5 Performance Evaluation

In the previous section, we showed that the two algorithms provide asymptotically optimal regret bounds, and that the theoretical regret bound of fractional KUBE is tighter than that of KUBE. In addition, we also demonstrated that fractional KUBE outperforms KUBE in terms of computational complexity. However, it might be the case that these bounds are not tight, and thus, fractional KUBE is less practical than KUBE in real–world applications, as is the case with the standard MAB algorithm, where simple but not optimal methods (e.g. ε–first, or ε–greedy) typically outperform more advanced, theoretically optimal, algorithms (e.g. POKER (Vermorel and Mohri 2005), or UCB). Given this, we now evaluate the performance of both algorithms through extensive simulations, in order to determine their efficiency in practice. We also compare the performance of the proposed algorithms against that of different budget–limited ε–first approaches. In particular, we show that both of our algorithms outperform the budget–limited ε–first algorithms. In addition, we also demonstrate that KUBE typically achieves lower regret than its fractional counterpart.

Now, note that if the pulling costs are homogeneous — that is, the pulling cost of the arms do not significantly differ from each other — then the performance of the density–ordered greedy algorithm does not significantly differ from that of the fractional relaxation based (Kellerer, Pferschy,

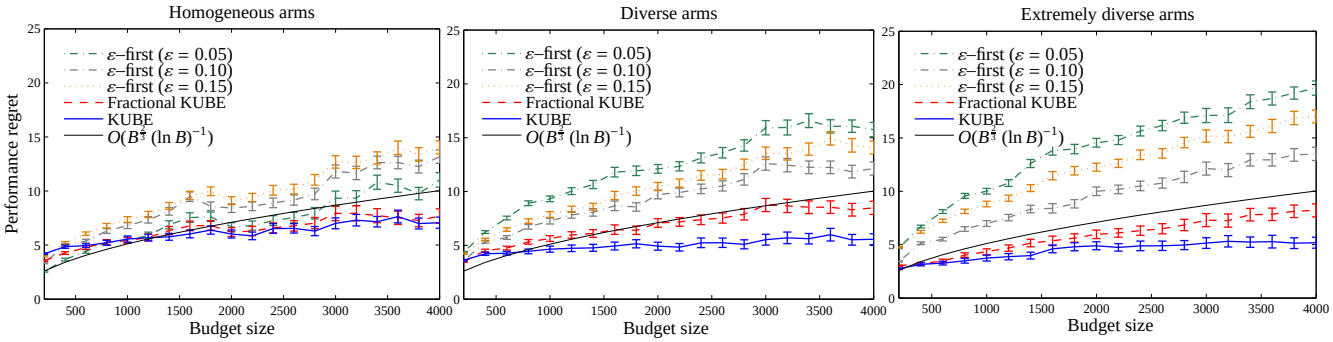

Figure 1: *Performance regret of the algorithms, divided by* $\ln\left(\frac{B}{c_{min}}\right)$*, for a* 100*–armed bandit machine with homogeneous arms, moderately diverse arms, or extremely diverse arms (left to right).*

and Pisinger 2004). Indeed, since the pulling costs are similar, it is easy to show that the density–ordered greedy approach typically stops after one round, and thus, results in similar behaviour to the fractional relaxation based method. On the other hand, if the pulling costs are more diverse (i.e. the pulling costs of the arms differ from each other), then the performance of the density–ordered greedy algorithm becomes more efficient than that of the fractional relaxation based algorithm. Given this, in order to compare the performance of KUBE and its fractional counterpart, we set three test cases, namely: bandits with (i) homogeneous pulling costs; (ii) moderately diverse pulling costs; and (iii) extremely diverse costs. In particular, within the homogeneous case, the pulling costs are randomly and independently chosen from the interval $[5, 10]$. In addition, the pulling costs are set to be between $[1, 10]$ within the moderately diverse case, and between $[1, 20]$ in the extremely diverse case, respectively. The reward distribution of each arm i is set to be a truncated Gaussian, with mean μ_i, randomly taken from interval $[10, 20]$, variance $\sigma_i^2 = \frac{\mu_i}{2}$, and with supports $[0, 2\mu_i]$. In addition, we set number of arms K to be 100.

Our results are shown in Figure 1. These plots show the performance of each algorithm divided by $\ln \frac{B}{c_{\min}}$, and the error bars represent the 95% confidence intervals. By doing this, we can see that the performance regret of both algorithms is $O\left(\ln \frac{B}{c_{\min}}\right)$, since in each test case, their performance converges to $C \ln \frac{B}{c_{\min}}$(after it is divided by $\ln \frac{B}{c_{\min}}$), where C is some constant factor. From the numerical results, we can see that both KUBE and fractional KUBE differ from the best possible solution by small constant factors (i.e. C), since the limit of their convergence is typically low (i.e. it varies between 4 and 7 in the test cases), compared to the regret value of the algorithm. In addition, we can also see that fractional KUBE algorithm is typically outperformed by KUBE. The reason is that the density–ordered greedy algorithm provides a better approximation than the fractional relaxation based approach to the underlying unbounded knapsack problem. This implies that KUBE converges to the optimal pulling policy faster than its fractional counterpart. In particular, as expected, the performance of the algorithms are similar to each other in the homogeneous case, where the density–ordered greedy method shows similar behaviour to the fractional relaxation based approach. In contrast, KUBE clearly achieves better performance (i.e. lower regret) within the diverse cases. Specifically, within the moderately diverse case, KUBE outperforms its fractional counterpart by up to 40% (i.e. the regret of KUBE is 40% lower than that of the fractional KUBE algorithm). In addition, the performance improvement of KUBE is typically around 30% in the extremely diverse case. This implies that, although the current theoretical regret bounds are asymptotically optimal, they are not tight.

Apart from this, we can also observe that both of our algorithms outperform the budget–limited ε–first approaches. In particular, KUBE and its fractional counterpart typically achieves less regret by up to 70% and 50% than the budget–limited ε–first approaches, respectively. Note that the performance of the proposed algorithms are typically under the line $O(B^{\frac{2}{3}} (\ln B)^{-1})$, while the budget–limited ε–first approaches achieve larger regrets. This implies that our proposed algorithms are the first methods that achieve logarithmic regret bounds.

6 Conclusions

In this paper, we introduced two new algorithms, KUBE and fractional KUBE, for the budget–limited MAB problem. These algorithms sample each arm in an initial phase. Then, at each subsequent time step, they determine a best set of arms, according to the agent's current reward estimates plus a confidence interval based on the number of samples taken of each arm. In particular, KUBE uses the density–ordered greedy algorithm to determine this best set of arms. In contrast, fractional KUBE relies on the fractional relaxation based algorithm. KUBE and its fractional counterpart then use this best set as a probability distribution with which to randomly choose the next arm to pull. As such, both algorithms do not explicitly separate exploration from exploitation. We have also provided a $O \ln(B)$ theoretical upper bound for the performance regret of both algorithms, where B is the budget limit. In addition, we proved that the provided bounds are asymptotically optimal, that is, they differ from the best possible regret by only a constant factor. Finally, through simulation, we have demonstrated that KUBE typically outperforms its fractional counterpart up to 40%, however, with an increased computational cost. In particular, the average computational complexity of KUBE per time step is $O(K \ln K)$, while this value is $O(K)$ for fractional KUBE.

One of the implications of the numerical results is that although fractional KUBE has a better bound on its performance regret than KUBE, the latter typically ourperforms the former in practice. Given this, our future work consists of improving the results of Theorems 1 and 2 to determine tighter upper bounds can be found. In addition, we aim to extend the budget–limited MAB model to settings where the reward distributions are dynamically changing, as is the case in a numer of real–world problems. This, however, is not trivial, since both of our algorithms rely on the assumption that the expected value of the rewards is static, and thus, the estimates converge to their real value.

Acknowledgements

We would like to thank Csaba Szepesvári for the useful advices. We also would like to thank the anonymous reviewers of ALT 2011 and NIPS 2011 for pointing out the unclear parts and flaws within the previous version of the paper.

References

Antos, A.; Grover, V.; and Szepesvári, C. 2008. Active learning in multi–armed bandits. *In Proceedings of the Nineteenth International Conference on Algorithmic Learning Theory* 287–302.

Audibert, J.-Y.; Munos, R.; and Szepesvári, C. 2009. Exploration–exploitation trade-off using variance estimates in multi-armed bandits. *Theoretical Computer Science* 410:1876–1902.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite–time analysis of the multiarmed bandit problem. *Machine Learning* 47:235–256.

Bubeck, S.; Munos, R.; and Stoltz, G. 2009. Pure exploration for multi–armed bandit problems. *In Proceedings of the Twentieth International Conference on Algorithmic Learning Theory* 23–37.

Guha, S., and Munagala, K. 2007. Approximation algorithms for budgeted learning problems. *In Proceedings of the Thirty-Nineth ACM Symposium on Theory of Computing* 104–113.

Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. *Knapsack Problems*. Springer.

Kohli, R.; Krishnamurti, R.; and Mirchandani, P. 2004. Average performance of greedy heuristics for the integer knapsack problem. *European Journal of Operational Research* 154(1):36–45.

Lai, T. L., and Robbins, H. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathemathics* 6(1):4–22.

Padhy, P.; Dash, R. K.; Martinez, K.; and Jennings, N. R. 2010. A utility-based adaptive sensing and multihop communication protocol for wireless sensor networks. *ACM Transactions on Sensor Networks* 6(3):1–39.

Robbins, H. 1952. Some aspects of the sequential design of experiments. *Bulletin of the AMS* 55:527–535.

Tran-Thanh, L.; Chapman, A.; de Cote, J. E. M.; Rogers, A.; and Jennings, N. R. 2010. Epsilon–first policies for budget–limited multi–armed bandits. *In Proceedings of the Twenti-Fourth National Conference on Artificial Intelligence* 1211–1216.

Tran-Thanh, L.; Chapman, A.; Rogers, A.; and Jennings, N. R. 2012. Knapsack based optimal policies for budget–limited multi–armed bandits. *Technical report* `http://arxiv.org/abs/1204.1909`.

Tran-Thanh, L.; Rogers, A.; and Jennings, N. R. 2011. Long–term information collection with energy harvesting wireless sensors: A multi–armed bandit based approach. *Journal of Autonomous Agents and Multi-Agent Systems*.

Vermorel, J., and Mohri, M. 2005. Multi-armed bandit algorithms and empirical evaluation. *ECML'05* 437–448.