Adaptive Policies for Selecting Groupon Style Chunked Reward Ads in a Stochastic Knapsack Framework

Michael Grabchak[*]
Cornell University
Ithaca, U.S.A.
mg323@cornell.edu

Narayan Bhamidipati
Yahoo! Labs
Bangalore, India
narayanb@yahoo-inc.com

Rushi Bhatt
Yahoo! Labs
Bangalore, India
rushi@yahoo-inc.com

Dinesh Garg
Yahoo! Labs
Bangalore, India
dineshg@yahoo-inc.com

ABSTRACT

Stochastic knapsack problems deal with selecting items with potentially random sizes and rewards so as to maximize the total reward while satisfying certain capacity constraints. A novel variant of this problem, where items are worthless unless collected in bundles, is introduced here. This setup is similar to the Groupon model, where a deal is off unless a minimum number of users sign up for it. Since the optimal algorithm to solve this problem is not practical, several adaptive greedy approaches with reasonable time and memory requirements are studied in detail – theoretically, as well as, experimentally. Worst case performance guarantees are provided for some of these greedy algorithms, while results of experimental evaluation demonstrate that they are much closer to optimal than what the theoretical bounds suggest. Applications include optimizing for online advertising pricing models where advertisers pay only when certain goals, in terms of clicks or conversions, are met. We perform extensive experiments for the situation where there are between two and five ads. For typical ad conversion rates, the greedy policy of selecting items having the highest individual expected reward obtains a value within 5% of optimal over 95% of the time for a wide selection of parameters.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY

General Terms

Algorithms

Keywords

Groupon, Chunked Rewards, Ad Selection, Revenue Maximization

[*]Most of the research work described herein was done when Michael was a summer intern at Yahoo! Labs, Bangalore

1. INTRODUCTION

How should a publisher optimally select and display advertisements or offers when advertisers agree to pay for *chunks* of clicks or conversions? A real world example of such *chunked reward* mechanism is Groupon (www.groupon.com). At Groupon, purchase vouchers are sold at heavy discount but they come into effect only when a fixed, pre-determined, number of individuals sign up within a fixed time. Once this threshold number of sign-ups is met, the whole group of buyers receives a discount. In the presence of multiple concurrently active offers, the Groupon selection problem, when transformed to an online and realtime setting, is to show a small number, possibly one, of the available offers to Groupon users. While we use the Groupon model as an example, it is easy to generalize this scenario as an online advertising scheme where the advertiser pays the publisher only upon receiving a pre-determined number of item purchases (or conversions). Similar models already exist, most notably the dynamic cost per impression (dCPM) model offered by RightMedia Exchange (www.rightmedia.com), where the publisher is paid per-impression, as long as a fixed conversion goal is maintained. Here, we generalize the dCPM model where the publisher gets paid when a fixed number of conversions happen within a given time period. Our case is different in that, instead of guaranteeing a *minimum number* of conversions, the publisher agrees to a *fixed number of conversions and within a specified time limit.* Such a payout model is more desirable for the advertiser because, assuming a successful campaign, the inventory, marketing costs, and profits may be estimated on a per-week or a per-quarter basis and for known quantities. While desirable for the advertiser or the seller, our problem formulation presents significant algorithmic challenges for the publisher or the ad serving exchange. As we shall see in this paper, such a chunked reward format poses unique challenges for revenue maximization.

The goal of this paper is to investigate how revenue for the *publisher* may be optimized under a chunked reward ad pricing model. We will show that the chunked reward model proves to be a computationally hard problem. Theoretical performance guarantees will be developed for greedy or sensible heuristic policies for our model, and extensive experiments will show how certain policies enable good revenue generation with explicit campaign run-time constraints. More formally, we consider a scenario with k concurrent ads or coupon offers. Exactly one of these ads must be shown to a user. Let the probability of a user signing up for the offer i be p_i, and upon n_i sign ups (i.e., n_i successes), the publisher (or the ad exchange) receives a fixed, known, reward r_i. We also assume, for the sake of simplicity in analysis, that once the reward is collected, the publisher has no interest in showing the offer anymore. With this latter assumption, we depart slightly from the Groupon.com model where the ad campaign may run for a fixed duration rather than until thresholds are met.

The goal of the publisher is now to maximize reward within a given, fixed, time frame by presenting the best sequence of ads, out of the currently active k ads. We will show how this maximization problem is NP-hard. We will then devise a number of heuristic selection policies and show formal approximation bounds for these. Through extensive experimentation, we will also show how some of the adaptive greedy policies devised here work quite well in practice.

To summarize the key contributions of this paper:

1. We formally study a chunked reward ad selection problem suitable for online advertisements. To the best of our knowledge, this problem of optimizing revenue under chunked probabilistic reward and fixed time horizon has not been studied earlier.

2. We show that our optimization problem is a variant of the stochastic knapsack problem [2] and, therefore, intractable to exactly maximize. We devise a number of heuristic ad-selection policies and provide worst-case performance bounds on expected revenues for those policies. We also prove 1/3- and 1/4-approximation bounds for some policies that are easily computable.

3. Using extensive experiments we compare a number of ad selection policies and demonstrate their strengths and weaknesses. We also identify a policy that works quite well empirically and, at the same time, is fast to compute.

The remainder of this paper is organized as follows. In Section 2, we give a formal setup of the problem. We then describe some related work in Section 3. In Section 4, we describe several greedy and modified greedy policies, and for the modified greedy policies we give worst case performance bounds.

After discussing the complexity of these policies in Section 5, experimental results are given in Section 6. These results show that a simple greedy policy is often very good. Finally, in Section 7, we conclude and give some directions for future work.

2. FORMAL SETUP

Assume that we have k ads that we wish to display on a particular website. Assume that the ith ad has a CTR of p_i, and if it gets at least n_i clicks on the web site before a fixed time, then we get a reward of r_i. The number of users who will visit the site by that time is some random variable T. When the tth user arrives, we must choose an ad to display in order to maximize our revenue. We assume that we can only display one ad at a time.

We will think of the problem in the following slightly more abstract setting. There is a machine with k arms. Pulling the ith arm results in a success with probability p_i and a failure with probability $1 - p_i$. The arm has an associated goal of obtaining n_i successes, only on accomplishment of which, a reward of r_i is given by the machine. This reward may be obtained only once for any arm. We are allowed a maximum of T arm pulls, where T may be random or fixed and known. The objective is to pull a sequence of arms such that the total reward is maximized. Note that, when $p_i = 1$ for all i, and T is fixed and known, the problem reduces to the standard (deterministic) knapsack problem. Since that problem is known to be NP-hard (see [7]), our problem must, in general, be NP-hard as well.

Let $\underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}$, and $\underset{\sim}{\mathbf{n}}$ denote the vectors of probabilities, rewards, and the goals for a set of arms. We assume that these vectors are known beforehand, along with F_T (the distribution of T). In practice, of course, we would not know $\underset{\sim}{\mathbf{p}}$ and F_T, but we assume that we have good estimates of these. Further, we assume that whether we get a success or a failure for any arm on any trial is independent of T.

For each arm i, let W_i be the random number of times that we must pull arm i to get the n_ith success. Clearly, W_i has a negative binomial distribution. Since there are different parametrizations of the negative binomial, we will specify what we mean. We will write $W_i \sim NB(n_i, p_i)$, where $p_i \in [0,1]$ and $n_i \in \mathbb{N} \setminus \{0\}$, to mean that, for $t \in \mathbb{N}$,

$$P(W_i = t) = \begin{cases} 0 & \text{if } t < n_i \\ \begin{pmatrix} t-1 \\ n_i - 1 \end{pmatrix} p_i^{n_i}(1-p_i)^{t-n_i} & o.w. \end{cases} \tag{1}$$

Note that $\mathrm{E}W_i = n_i/p_i$. We assume that the W_is are mutually independent both of each other and of T.

For every time t, let the arm pulled at t be θ_t and the result of this pull be δ_t. Once the arm to be pulled is determined, the probability of success in that pull is known and given by

$$P(\delta_t = 1|\theta_t = i) = p_i \text{ and } P(\delta_t = 0|\theta_t = i) = 1 - p_i.$$

For each t, let d_t be a realization of the random variable δ_t, and let $\underset{\sim}{\mathbf{d}}$, $\underset{\sim}{\delta}$, and $\underset{\sim}{\theta}$ be the corresponding vector forms.

A policy π is either a random or a deterministic function that chooses the arm to be pulled at time $t+1$ given all the available information at time t. Thus, the arm chosen by policy π at time $t+1$ may be written as

$$\theta_{t+1} = \pi(\underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}}, F_T|\{(\theta_i, d_i)\}_{i=1,\ldots,t}). \tag{2}$$

Note that a policy only knows T through its distribution. In the important special case when T is deterministic, its distribution is a point mass and it is thus known exactly. We will use the notation Π to denote the class of all policies. Without loss of generality, assume that $P(W_i \leq T) > 0$ for all i.

Let $S_i(t)$ and $s_i(t)$ denote the random number of successes and the observed number of successes, respectively, of arm i in the first t pulls. They are defined as follows:

$$S_i(t) = \sum_{j=1}^{t} \delta_j 1_{[\theta_j = i]}, (i = 1, 2, \ldots, k), \tag{3}$$

$$s_i(t) = \sum_{j=1}^{t} d_j 1_{[\theta_j = i]}, (i = 1, 2, \ldots, k), \tag{4}$$

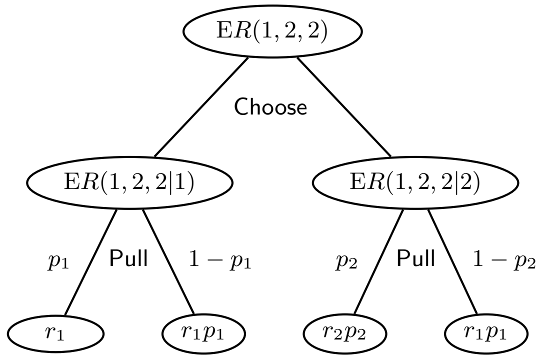


Figure 1: Example of how π^* computes the expected reward for $T = 2$, given the values at $T = 1$. Here, $n_1 = 1$, $n_2 = 2$, and $r_1p_1 < r_2p_2$

and let $\underset{\sim}{\mathbf{S}}(t)$ and $\underset{\sim}{\mathbf{s}}(t)$ be the corresponding vector forms. Thus, given $\underset{\sim}{\mathbf{d}}$, the reward obtained by π is

$$R(\pi, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}}, F_T|\underset{\sim}{\theta}, \underset{\sim}{\delta} = \underset{\sim}{\mathbf{d}}) = \sum_{i=1}^{k} r_i 1_{[s_i(T) \geq n_i]}, \quad (5)$$

and the expected reward is

$$\mathrm{E}R(\pi, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}}, F_T|\underset{\sim}{\theta}, \underset{\sim}{\delta} = \underset{\sim}{\mathbf{d}}) = \sum_{i=1}^{k} r_i P(S_i(T) \geq n_i; \pi).$$

We will call a policy optimal if it attains the largest reward. Thus a policy π^* is optimal if

$$\pi^* = \operatorname*{argsup}_{\pi \in \Pi} \mathrm{E}R(\pi, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}}, F_T|\underset{\sim}{\theta}, \underset{\sim}{\delta} = \underset{\sim}{\mathbf{d}}). \quad (6)$$

Since the problem is NP-hard we cannot hope to find an efficient optimal algorithm. Never-the-less, we will give an exponential time optimal algorithm. Although it is of little practical use, it guarantees that an optimal policy exists, and moreover, in small scale experiments, it allows us to compare other policies with the optimal one. When T has a bounded support, the optimal algorithm exists. It can be defined recursively as the algorithm which at time t chooses the arm

$$\begin{aligned} i^* = \operatorname*{argmax}_{i=1,\dots,k} \Big\{ & p_i \big[r_i 1\{n_i - s_i(t) = 1\} \\ & + \mathrm{E}R(\pi^*, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}} - \underset{\sim}{\mathbf{s}}(t) - \underset{\sim}{\mathbf{e_i}}, F_{T-t-1}|\emptyset) \big] \quad (7) \\ & + (1-p_i)\mathrm{E}R(\pi^*, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}} - \underset{\sim}{\mathbf{s}}(t), F_{T-t-1}|\emptyset) \Big\}. \end{aligned}$$

This can be easily shown by induction on the time until T_0, where T_0 is the largest value that T can take with positive probability. Note that this policy is Markovian in the sense that

$$\begin{aligned} & \mathrm{E}R(\pi^*, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}}, F_T|(\theta_s, d_s)_{s=1,\dots,t}) \\ & \qquad = \mathrm{E}R(\pi^*, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}} - \underset{\sim}{\mathbf{s}}(t), F_{T-t}|\emptyset). \quad (8) \end{aligned}$$

To simplify the notation, we will often write

$$R(\pi, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}}, F_T|\underset{\sim}{\theta}, \underset{\sim}{\delta} = \underset{\sim}{\mathbf{d}}) \text{ as } R(\pi)$$

when the other parameters are understood from context.

An example demonstrating how the expected reward is computed is now presented. Let T be known and fixed, $k = 2$, and let the parameters $\underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}$ be fixed, and so chosen that $r_1p_1 < r_2p_2$. Using E$R(n_1, n_2, T)$ as shorthand notation for E$R(\pi^*, \underset{\sim}{\mathbf{p}}, \underset{\sim}{\mathbf{r}}, \underset{\sim}{\mathbf{n}}, F_T)$, it is easy to see that (a) E$R(1, 2, 1) = r_1p_1$, (b) E$R(0, 2, 1) = 0$, and (c) E$R(1, 1, 1) = r_2p_2$. Fig. 1 shows a portion of the policy tree to be constructed by π^*. The levels of the tree are alternately associated with the actions *choose* and *pull*, the former where π^* decides on which arm to pull, and the latter where the result of the arm pulled would be observed. Thus, when $T = 2$, pulling arms 1 and 2 yield the expected rewards E$R(1, 2, 2|1) = p_1(r_1) + (1-p_1)(r_1p_1)$, and E$R(1, 2, 2|2) = p_2(r_2p_2) + (1-p_2)(r_1p_1)$, respectively, and π^* chooses the more valuable arm.

3. RELATED WORK AND BACKGROUND

The general framework of our problem is very similar to that of the multi-armed bandit (MAB) problem. For details on MAB see, for instance, [5, 6] and the references therein. However, that is an inference problem, where the goal is to learn the value of $\underset{\sim}{\mathbf{p}}$. We assume that $\underset{\sim}{\mathbf{p}}$ is known and are interested in optimizing the total reward by time T when the rewards are chunked in the manner described in the previous section.

Our problem is actually a variant of the stochastic knapsack problem. An overview of the standard (deterministic) knapsack problem can be found in [7]. There are a number of stochastic extensions, see the references in [2]. Our version is most similar to the version considered in [2].

They assume that there are k items. Each item i has a fixed and known value r_i and a random weight W_i, with $W_i \sim F_i$ for some distribution function satisfying $F_i(0) = P(W_i \leq 0) = 0$. One-by-one, items are placed into a knapsack that can hold at most a weight of T, where T is fixed and known. Once an item has been inserted we find out how big it is. If it fits, then we collect the corresponding reward and we can place another item in the knapsack. If it does not fit, then we do not get a reward and we are done. A number of approximation algorithms are given in [2].

Our version allows for T to be random (but independent of all of the W_is) and it assumes that $W_i \sim NB(n_i, p_i)$ for each i. The most important difference, however, is that in the setting of [2], once we begin inserting an item, we must insert all of it, while in our setting, at each time point we have the option to switch to a different item (or, in our terminology, arm).

In general, the stochastic knapsack problem (as considered in [2]) is NP-hard. However, there are situations where the optimal policy is a simple greedy algorithm. When all of the sizes follow an exponential distribution, such a solution is given in [3]. A different proof of this result is in [4]. The proof in [3] only uses the fact that the exponential distribution has the memoryless property. Therefore, it is easily extended to the geometric distribution, which is a special case of the negative binomial. Moreover, because the result does not depend on T, it immediately extends to our setting, where T is random and we can change arms at any time. We now state this result.

THEOREM 1. *In the context of Section 2, if for every arm i, $W_i \sim NB(1, p_i)$ then the optimal arm to pull at time t is the one with the largest r_ip_i from which we have not yet received a reward.*

4. PRACTICAL POLICIES FOR CHUNKED REWARDS

For the optimal policy π^* given in Eq. (7), the choice of the arm to be pulled at time t relies on all of the possibilities at time $t+1$, each of which in turn relies on all of those at time $t+2$, and so on. In this section, we will study several policies that can be implemented without requiring such a lookahead. We will restrict our attention to policies that satisfy the following *feasibility criteria*:

1. arm i would be considered at time $t+1$ only if
$$s_i(t) < n_i \text{ and } P\left(n_i - s_i(t) \leq T - t\right) > 0,$$
i.e., only if its goal is not attained yet, and there is a non-zero probability of attaining it,
2. all other parameters being equal, arm i would be chosen over arm j if $p_i > p_j$ or $r_i > r_j$ or $n_i < n_j$, and
3. if the rewards of all of the arms are multiplied by the same constant, the choice of arm will not change.

The reason for considering only policies that satisfy these criteria is that any policy that does not satisfy these, can be easily replaced by one that does and is uniformly better than the given policy. The third criterion ensures that the policy does not depend on the units of the reward.

4.1 Greedy Policies

In this section, we will list several adaptive greedy policies. These are policies that at time $t+1$ compute a specified index for each arm. They then choose the arm which maximizes this index. In light of the second feasibility criterion, we will only consider policies where the index for arm i is a non-decreasing function of p_i and r_i, and a non-increasing function of $n_i - s_i(t)$, and involves all of them. Moreover, we will assume that the index is linear in r_i in order to satisfy the third feasibility criterion. The greedy policies considered are:

- π_1 : Index $\frac{r_i p_i}{n_i - s_i(t)} 1_{[s_i(t) < n_i]} 1_{[n_i - s_i(t) \leq T - t]}$. This is inspired by the fractional knapsack problem (see [7]).
- π_2 : Index $\frac{r_i p_i}{n_i - s_i(t)} P(W_i \leq T | S_i(t) = s_i(t))$. This is a variation of π_1 that decreases the index of i if the chances of attaining its goal are lower. For large T, π_2 behaves similar to π_1.
- π_3 : Index $r_i P(W_i \leq T | S_i(t) = s_i(t))$. The term $P(W_i \leq T | S_i(t) = s_i(t))$ is already a monotone decreasing function of $n_i - s_i(t)$ and a monotone increasing function of p_i.
- π_4 : Index $\frac{r_i}{\mathrm{E}[W_i \wedge T | S_i(t) = s_i(t)]} P(W_i \leq T | S_i(t) = s_i(t))$. This is inspired by the simplified greedy algorithm given in [2].

In evaluating the indices above, it is useful to note that $[W_i - t | S_i(t) = s_i(t)] \sim NB(n_i - s_i(t), p_i)$.

An important point to make is that these indices need to be recomputed at every time point. One can instead compute the indices only once, at the beginning. Then, at each time t, pull the arm with the largest index (without updating the indices) for which we have not yet received a reward. We will call such policies the non-adaptive versions and denote the non-adaptive version of π_i by γ_i, $i = 1, 2, 3, 4$. Clearly these policies do not satisfy the feasibility criteria and can be uniformly improved. However, we should mention that they are not immediately comparable with the adaptive versions, and may, in some situations, be better.

4.2 Greedy Policies May Be Arbitrarily Bad

Although the greedy policies that we defined are easy to compute, they may be arbitrarily bad relative to the optimal algorithm. It is easiest to see this in the case when $p_i = 1$ for each arm i. In this case the problem reduces to the deterministic knapsack and the policies π_1, π_2, π_4, γ_1, γ_2, and γ_4 all reduce to the standard greedy algorithm for that problem. To see that these can be arbitrarily bad see Chapter 2 of [7]. For policy π_3 (and γ_3) consider the following setup. There are $k = T+1$ arms. For arm 1, $r_1 = 2$ and $n_1 = T$. For arm i with $i = 2, \dots, T+1$, $r_i = 1$ and $n_i = 1$. Following π_3, we would only pull arm 1 and at the end get a reward of 2, however the optimal algorithm is to never pull arm 1 and to instead pull each of the other arms once to get a reward of T, which may be arbitrarily larger than 2. This behavior is not limited to situations where p is large. For all of these policies, it is not difficult to construct similar situations when $p \in (0,1)$.

However, although the greedy algorithms may be arbitrarily bad, algorithm π_1 is, in fact, the optimal algorithm in the following cases:
1. when $n_i = 1$ for all arms $i = 1, \dots, k$ (this follows by Theorem 1),
2. when any two of the parameters are identical for all arms and only one parameter changes.

4.3 Policies With Worst Case Performance Bounds

Since greedy policies can be arbitrarily bad, we cannot provide worst case performance bounds for them. However, we can provide bounds for certain modified versions. First we introduce a policy that is not very useful in practice, but important for the theory:

- γ_0 : always pull the arm with $\max_i r_i P(W_i \leq T)$ even after we have received the reward for this arm

Modified greedy policies choose between two greedy policies at the beginning, then they stick with the same greedy policy until time T. We will consider the following modified greedy policies:

- γ_5 : w.p .5, choose γ_0, otherwise choose γ_1
- γ_6 : w.p .5, choose γ_0, otherwise choose γ_2
- γ_7 : choose γ_1 if $\max_i r_i P(W_i \leq T) < \mathrm{E}R(\gamma_1)$, otherwise choose γ_0
- γ_8 : choose γ_2 if $\max_i r_i P(W_i \leq T) < \mathrm{E}R(\gamma_2)$, otherwise choose γ_0
- γ_9 : choose γ_4 if $\max_i r_i P(W_i \leq T) < \Psi$, otherwise choose γ_0. Here
$$\Psi = \frac{1}{2} \sum_{i=1}^{k} r_i P(W_i \leq T) \prod_{j=1}^{i-1} \left(1 - \mathrm{E}\left[\left(\frac{W_j}{T}\right) \wedge 1\right]\right),$$
where the items are ordered as in γ_4.

Note that γ_9 is a version of the simplified greedy algorithm given in [2]. It is immediate that, in expectation, policy γ_7 is uniformly better then γ_0 and γ_1 and that policy γ_8 is uniformly better then γ_0 and γ_2.

When T is random, it may be difficult to evaluate $P(W_i \le T)$, but this can be approximated through simulation. Even when T is non-random, it may be difficult to implement policies γ_7 and γ_8. The difficulty lies in evaluating

$$\mathrm{E}R(\gamma_j) = \sum_{i=1}^{k} r_i P\left(\sum_{\ell=1}^{i} W_\ell \le T\right)$$

for $j = 1, 2$, where the W_is are assumed to be ordered as in γ_j. Again, this can be approximated through simulation.

The following theorem gives bounds on the worst case performance of policies γ_5-γ_8.

THEOREM 2. *Assume that the W_is are mutually independent of themselves and of T. We have*

$$\sup_{\pi\in\Pi} \mathrm{E}R(\pi) \le \frac{2}{\min_i P(W_i \le T)} \mathrm{E}R(\gamma_5), \tag{9}$$

$$\sup_{\pi\in\Pi} \mathrm{E}R(\pi) \le \frac{2}{\min_i P(W_i \le T)} \mathrm{E}R(\gamma_6), \tag{10}$$

$$\sup_{\pi\in\Pi} \mathrm{E}R(\pi) \le \left(1 + \frac{1}{\min_i P(W_i \le T)}\right) \mathrm{E}R(\gamma_7), \tag{11}$$

$$\sup_{\pi\in\Pi} \mathrm{E}R(\pi) \le \frac{1 + \max_i P(W_i \le T)}{\min_i P(W_i \le T)} \mathrm{E}R(\gamma_8). \tag{12}$$

Note that, by Eq. (7), when T has a bounded support, the supremum is attained. Clearly, after we have chosen a policy in the prescribed way, if we then use a policy that is uniformly better, then the result will still hold. In particular, it is not difficult to come up with policies that are uniformly better than γ_0. When $p_i = 1$ for all i and T is non-random, the problem reduces to the deterministic knapsack and both π_7 and π_8 reduce to the usual modified greedy algorithm for the deterministic knapsack, and our bounds reduce to the bound in that case.

PROOF. In the interest of space, we will only prove the bounds in Eq. (9) and Eq. (11). The proof of the rest is similar but slightly more involved. First, consider a fractional version of our problem where instead of arm i, there are n_i arms with reward r_i/n_i attainable after one success. Thus we have arms (i, j) for $i = 1, \ldots, k$ and $j = 1, \ldots, n_i$ with $n_{i,j} = 1$, $p_{i,j} = p_i$, $r_{i,j} = r_i/n_i$, and $W_{i,j} \sim NB(1, p_i)$. Clearly, for any policy $\pi \in \Pi$, if we pull the exact same sequence of arms in the fractional case that the policy π tells us to pull for the original case, the expected reward in the fractional case will be greater than or equal to the expected reward in the original case. By Theorem 1, the optimal policy in the fractional case is one that is greedy with index $r_i p_i/n_i$. Thus, in the original problem, for any $\pi \in \Pi$ we have $R(\pi) \le R(\gamma_1) + \max_i r_i$. Hence

$$\begin{aligned}
\mathrm{E}R(\pi) &\le \mathrm{E}R(\gamma_1) + \max_i r_i \\
&\le \mathrm{E}R(\gamma_1) + \max_i r_i \frac{P(W_i \le T)}{P(W_i \le T)} \\
&\le \mathrm{E}R(\gamma_1) + \frac{\max_i r_i P(W_i \le T)}{\min_i P(W_i \le T)}.
\end{aligned}$$

This is bounded by

$$\left(1 + \frac{1}{\min_i P(W_i \le T)}\right) \max\left\{\mathrm{E}R(\gamma_1), \max_i r_i P(W_i \le T)\right\}$$
$$= \left(1 + \frac{1}{\min_i P(W_i \le T)}\right) \mathrm{E}R(\gamma_7)$$

and

$$\frac{2}{\min_i P(W_i \le T)} \left[.5\mathrm{E}R(\gamma_1) + .5 \max_i r_i P(W_i \le T)\right]$$
$$= \frac{2}{\min_i P(W_i \le T)} \mathrm{E}R(\gamma_5).$$

Thus Eq. (9) and Eq. (11) hold. □

In the context of ads, except in extreme situations, we would not consider ads that have very low probability of attaining their goals. In particular, it is reasonable to assume that $\min_i P(W_i \le T) \ge .5$. Under this condition, the bounds reduce to

$$\frac{1}{3} \sup_{\pi\in\Pi} \mathrm{E}R(\pi) \le \mathrm{E}R(\gamma_7) \tag{13}$$

and for $m = 5, 6, 8$

$$\frac{1}{4} \sup_{\pi\in\Pi} \mathrm{E}R(\pi) \le \mathrm{E}R(\gamma_m). \tag{14}$$

In Section 6 of [2], a similar bound is given for γ_9. It is shown that when T is fixed and known

$$\frac{1}{4} \mathrm{E}R(\pi_0^*) \le \mathrm{E}R(\gamma_9), \tag{15}$$

where π_0^* is the best policy among those that keep pulling a chosen arm until either the reward is given or T is reached. Thus, these policies will keep pulling the same arm even if the probability of getting a reward with the arm becomes zero. This is a slightly weaker result, but it holds regardless of what $\min_i P(W_i \le T)$ is.

REMARK 3. *Since we are interested in maximizing the expected reward, nothing that we have done would change if we assume that the rewards are random so long as they have a finite expectation and are mutually independent, both of each other and of the W_is and T. In this case we just replace r_i by $\mathrm{E}[r_i]$ in the discussion above. In the context of ads, this takes into account the credit risk of the advertiser.*

5. COMPUTATIONAL COMPLEXITY

In this section, we will analyze the time and memory requirements of the greedy policies and the optimal policy when T is fixed and known. For simplicity we set $n = \max_i n_i$. If we want more detailed complexity bounds, in the following discussion n^k may be replaced by $\prod_{i=1}^k n_i$.

It is clear that π_1 can be implemented with time complexity of $\mathcal{O}(kT)$ and space complexity $\mathcal{O}(1)$. However, note that the order of the indices does not change over time, except that sometimes an arm may stop being feasible. Thus we can evaluate and order all of the indices at the beginning, this takes $\mathcal{O}(k)$ space and $\mathcal{O}(k \log k)$ time. Then at each t we only need to check if the arm is still feasible, thus the overall time complexity is $\mathcal{O}(k \log k + T)$.

The policies π_2 and π_3 have, essentially, the same complexity. At time $t + 1$, π_3 involves computing, for each arm i, $P(W_i \le T | S_i(t) = s_i(t))$, which is the same as $1 - P(\text{at most } n_i - s_i(t) - 1 \text{ successes in } T - t)$, requiring $\mathcal{O}(n)$ time. Thus, the overall time complexity for π_2 and π_3 is $\mathcal{O}(knT)$.

The policy π_4 is similar to π_2 and π_3 but we also need to compute $\mathrm{E}\left[W_i \wedge T | S_i(t) = s_i(t)\right]$, where

$$\mathrm{E}[W_i \wedge T] = \sum_{w=n}^{T} wP(W_i = w) + T(1 - P(W_i \le T)),$$

thereby requiring a total of $\mathcal{O}(knT^2)$ time.

Now, we will discuss implementing the optimal policy. As mentioned in Section 2, this policy is exponential in T. With additional memory, however, one may come up with pseudo-polynomial time dynamic programming algorithms similar to those used for solving deterministic knapsack problems (Section 2.6 in [7]). With $\mathcal{O}(n^k)$ memory, all the distinct nodes in level $t+1$ of the policy tree may be precomputed and stored, and for each node at level t, just k comparisons suffice to choose the optimal arm. To precompute these values, however, one needs to start from level $t = T$, and move up the policy tree, retaining the nodes from only the previous level. Thus, each level in the policy tree requires $\mathcal{O}(kn^kT)$ time, and the total time complexity for T levels is $\mathcal{O}(kn^kT^2)$. With $\mathcal{O}(n^kT)$ memory, every node of the policy tree can be stored in memory, and so requires $\mathcal{O}(n^kT)$ time.

6. EXPERIMENTAL EVALUATION

In this section, we will compare the performance of the greedy policies given in Section 4.1. We will show that although in certain situations these policies may be bad, in practice they are often very good. For simplicity we assume that T is fixed and known. Nevertheless, there are many possible values for $\underset{\sim}{\mathbf{p}}$, $\underset{\sim}{\mathbf{r}}$, $\underset{\sim}{\mathbf{n}}$, and T, all of which may result in different expected rewards for the different policies. In order to compare the policies, we ran extensive experiments for a wide variety of parameter combinations.

To evaluate the expected reward of a policy for a particular combination of parameters, we used an approach similar to that of evaluating the optimal algorithm as described in Section 2 and Fig. 1, but here the arm to be pulled was chosen based on the index of the policy. Another approach would be to simulate paths and then to average over these, however we would need many runs and only get approximations, while this way we get the true values.

First, we present the results for the case when there are only two arms ($k = 2$). We enumerate the expected reward for various scenarios by performing a parameter sweep. The success probabilities are chosen from $\{\frac{1}{256}, \frac{1}{64}, \frac{1}{16}, \frac{1}{4}, 1\}$, and r_2 is set to one of $\{\frac{1}{16}, \frac{1}{4}, 1, 4, 16\}$. The units of the reward are so chosen that r_1 is fixed at 1. In addition, n_1 and n_2 are both varied from 1 to T, for various values of T. When $T = 300$, a total of 11.25M distinct parameter combinations are covered. Although the expected reward is guaranteed to be positive for every such combination, it can be extremely small. All figures are rounded off to 8 decimal places before being used for reporting.

We begin by depicting the behavior of various policies on a small portion of the parameter space. Fig. 2 plots the expected reward obtained by π^* and π_i, $i = 1, 2, 3, 4$ with only n_2 varying while all the other parameters are fixed. In both Figs. 2a and 2b, $p_1 = \frac{1}{4}$, $p_2 = \frac{1}{16}$, $r_2 = 4$, and $T = 100$. Fig. 2a has n_1 set to 10, while Fig. 2b has n_1 set to 20. For arm 1 we expect to need about 4 trials per success, while for arm 2, we expect to need about 16 trials per success.

When $n_2 = 3$, all the policies start off by pulling arm 2 which has the larger reward. However, as n_2 increases, obtaining a reward from arm 2 becomes less probable, and all the policies pull arm 1 first, ensuring the reward of 1.

In Fig. 2b, we also see that π_1 may be significantly worse than the rest. Whenever $n_2 < n_1$, π_1 persists with arm 2 as long as the feasibility criterion is satisfied, and obtains a zero reward with a high probability. On the other hand, π_2 and π_3 appear to be consistently close to optimal in both the scenarios in Fig. 2.

We now discuss how we compare various policies and the differences between them. The performance of each greedy policy is measured in terms of how similar its expected reward is to that of π^*. We quantify this similarity using the following three measures:

- Agreement: For each parameter combination, agreement is a binary indicator of the expected reward of π_i matching that of π^*.
- Efficiency: This is the ratio $\frac{ER(\pi_i)}{ER(\pi^*)}$. This lies in $[0, 1]$, and the higher the better.
- Regret: This is the difference $ER(\pi^*) - ER(\pi_i)$. By definition of π^*, regret is non-negative, and the lower the better.

We are now confronted with the problem of presenting the performance measures. Clearly, showing individual values as in Fig. 2 is not feasible. For, $T = 300$, even if all the 300 values of n_2 are covered in a single plot, 375K such plots would be required! We will now explain our methodology for reporting these.

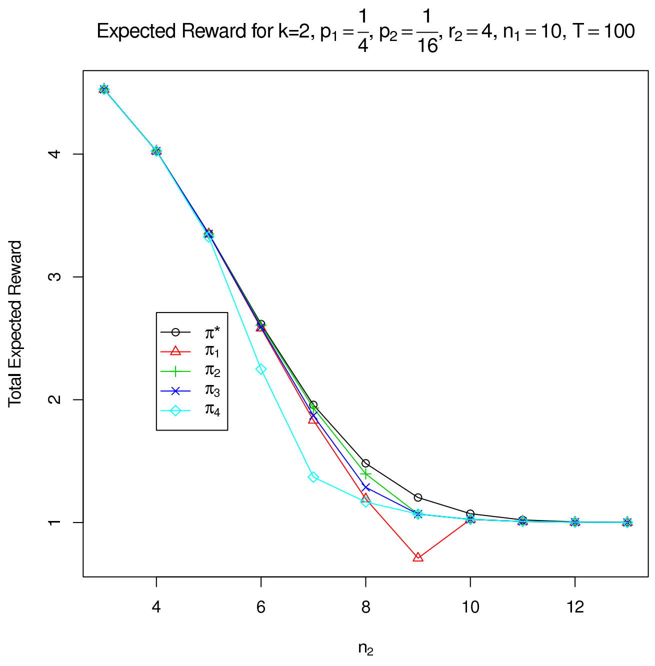
The problem being considered is the following. For $k = 2$ (say), and a fixed set of values p_1, p_2, r_1, r_2, and T, one may choose any n_1 and n_2 between 1 and T. If n_1 and n_2 are both very small, no matter which order the arms are pulled in, there is a high probability of obtaining a reward of $r_1 + r_2$. On the other hand, if they are both very large, there is very little chance of obtaining a non-zero reward, even for the optimal algorithm. Naturally, in both of these cases, a greedy policy can easily perform about the same as the optimal one, and so these can be considered trivial cases. Intermediate choices of parameters, however, make for a challenging situation where choosing the wrong arm has a greater impact.

While the trivial cases may be abundant in number, they are likely to be rare in practice, not being valuable to either the advertiser or the publisher. Consequently, clubbing the trivial cases together with the challenging ones would mean that the aggregate performance measures would appear to be far better than those for cases typically encountered in the real world. For this reason, we propose a systematic way of identifying (and excluding) the trivial cases. Observing that the expectation and variance of the number of trials required for obtaining n_i successes are $\mu_i = \frac{n_i}{p_i}$ and $\sigma_i^2 = \frac{n_i(1-p_i)}{p_i^2}$, respectively, we identify the difficulty level based on the interval $\mu_i \pm 2\sigma_i$. For each arm i, μ_i, σ_i, and T are used to identify the difficulty level of the arm as follows:

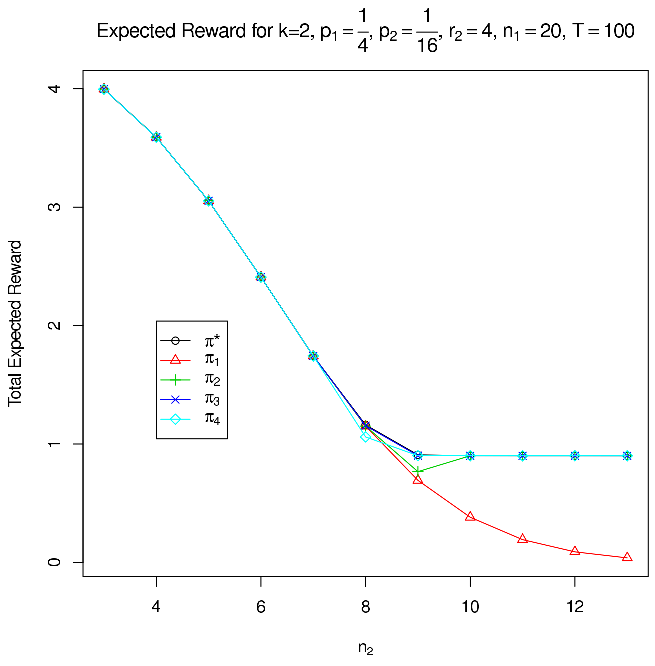
- V (very easy): $T - \sum_{j \neq i}(\mu_j + 2\sigma_j) > \mu_i + 2\sigma_i$
- E (easy): $T > \mu_i + 2\sigma_i$ and not V
- M (medium): $\mu_i - 2\sigma_i < T \leq \mu_i + 2\sigma_i$
- D (difficult): $T \leq \mu_i - 2\sigma_i$

Note that the label V applies to either all the arms or none. Other such definitions, with intervals of different sizes and more gradation, have been tried in our experiments, but the symmetric intervals for individual arms offer simplicity and ease of interpretation.

The box-and-whisker plots in Fig. 3 show how the optimal policy fares in various scenarios with $T = 300$. We observe the following:

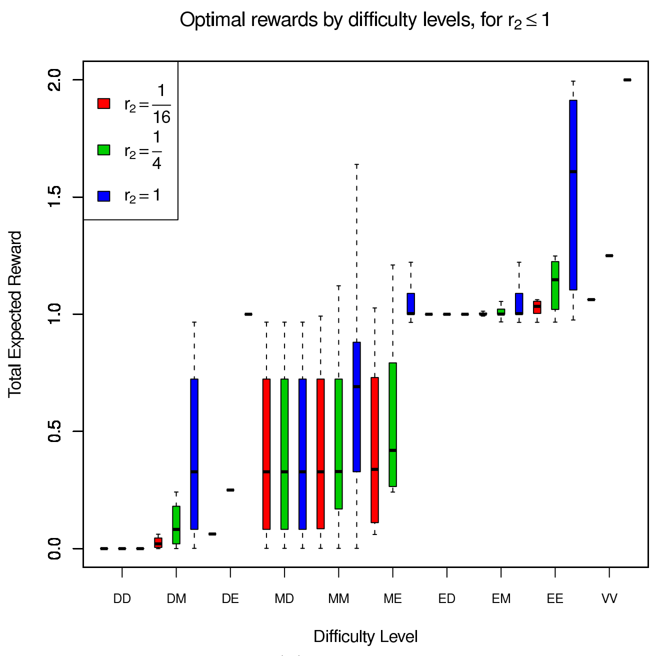


(a) $n_1 = 10$

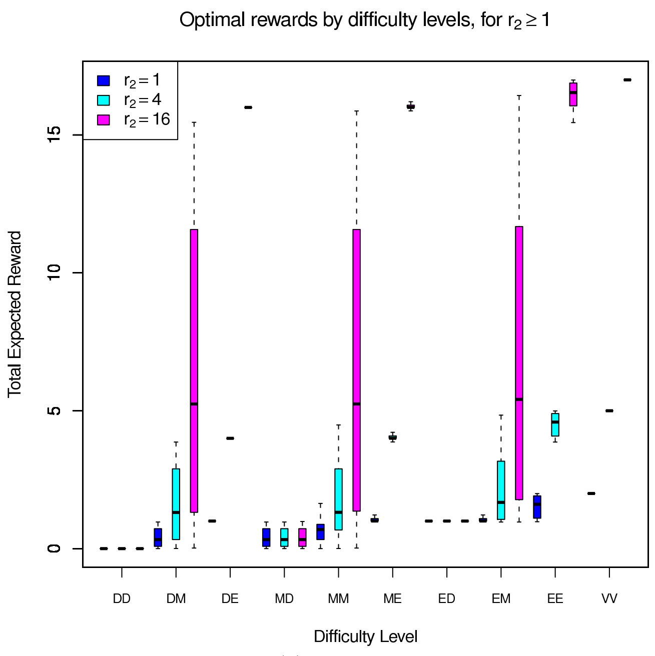


(b) $n_1 = 20$

Figure 2: Expected Reward of various policies as n_2 increases while all other parameters are fixed, with n_1 set to (a) 10, and (b) 20



(a) $r_2 \leq 1$



(b) $r_2 \geq 1$

Figure 3: Total Expected Reward of π^* for various rewards and difficulty levels, D: difficult, M: medium, E: easy, V: very easy, k=2, T=300

- If one arm's level is fixed, the expected reward increases as the difficulty decreases for the other arm.
- DD (difficult for both the arms), covers about 5.5M cases with near zero expected reward for each of them.
- If one arm has level D, π^* obtains reward only from the other arm.
- Level VV fetches almost the maximum possible reward of $r_1 + r_2$ for most cases.
- The variance in the reward is highest when arms with level M are involved. This agrees with our reasoning that there is lower uncertainty for levels D and E.

Fig. 4 shows the performance of greedy policies π_1 and π_3 relative to π^*. Note the difference in the scales of the two plots. We also ran the experiments for π_2 and π_4 but their plots are not shown due to space constraints. In general π_2 and π_4 did much better than π_1 but slightly worse than π_3.

- All the policies, except π_1, match π^* for any level involving D for one of the arms. This is because in these cases, the greedy policies, just like π^*, realize that pulling arm with level D is futile and allocate these pulls to the other arm instead.
- π_1 fares poorly in all levels (except VV), and even in case DD, where the optimal expected reward is near 0. For levels MD and DM, when the arm with D has an extremely high reward, π_1 may have an arbitrarily bad performance. This is a consequence of not taking available time T into consideration.
- The only levels where π_2, π_3, and π_4 fare significantly worse than optimal are ME, EM and MM, that too when the reward ratios are far from 1. When both arms look valuable enough to be pulled, these policies' simple indices cannot match the information that π^* uses. It may be noted that the lowest points in the plot represent the actual minimum for that level. So, over all the 11.25M cases considered, π_3 has a worst case efficiency of 0.6, and the Avg. Efficiency in the corresponding categories, EM and ME, is over 0.98. Avg. Efficiency of π_3 is over 0.99 in each of the remaining categories.

Although Fig. 4 shows the comparison only in terms of Efficiency, the plots for Agreement and Regret lead to similar conclusions. For example, the only cases with Avg. Regret significantly above 0 are ME (for $r_2 = \frac{1}{16}$ and $r_2 = \frac{1}{4}$, only), EM (for $r_2 = 16$ and $r_2 = 4$, only), and MM. Unlike efficiency, regret is sensitive to the value of r_2, and hence, is more difficult to interpret. For EM, when $r_2 = 16$, Avg. Regret is 0.117, whereas for its symmetric counterpart, ME with $r_2 = \frac{1}{16}$, it is 0.007.

One may also note that the parameter combinations in Figs. 2a and 2b correspond to difficulty levels EE and ME, respectively, and therefore show how well the greedy policies perform even in the challenging portions of the parameter space.

Our next set of experiments deal with checking if the greedy policies maintain the high efficiency levels as the number of arms increases. With $T = 1000$ and k varied over 2, 3, 4, and 5, a complete enumeration is no longer feasible. Instead, we selected 125 random combinations of p_i's and r_i's and varied the goal for each arm from 1 to 20. Unlike the previous experiments, p_i's were chosen from $\{\frac{1}{256}, \frac{1}{64}, \frac{1}{16}\}$, the reasons being that (a) CTR values typically lie in this range [1], and (b) for higher p_i's, even an n_i value of 20 would lead to the difficulty level V and is not challenging enough.

We did not include π_4 because of its extremely high computational cost. Although other greedy policies scale linearly with the n_i's and T, given that the ground truth used for measuring their performance is based on the optimal policy, and since our implementation of π^* assumes availability of $\mathcal{O}(n^k)$ memory, larger values of k, n_i's and T have not been considered.

Based on our experiences for the case of $k = 2$, we treated the challenging cases with at least one arm having level M and at least another having level E separate from the rest. To be able to compare across the different k values, Fig. 5 reports the proportion of cases having efficiency over a threshold, as the threshold is varied from 0.75 to 0.99. From Fig. 5, we see that π_3 continues to outperform the rest as k increases, and maintains an efficiency of about 0.95 for nearly 95% of the challenging cases.

7. CONCLUSIONS AND FUTURE WORK

The present work introduces an interesting variant of the stochastic knapsack problem that may be used for goal based all-or-none pricing for online ads. It provides feasible alternatives to the optimal but prohibitively expensive algorithm to maximize the total expected reward in this setting. We also introduce a methodology for reporting these results, by accounting for the inherent complexity of each problem.

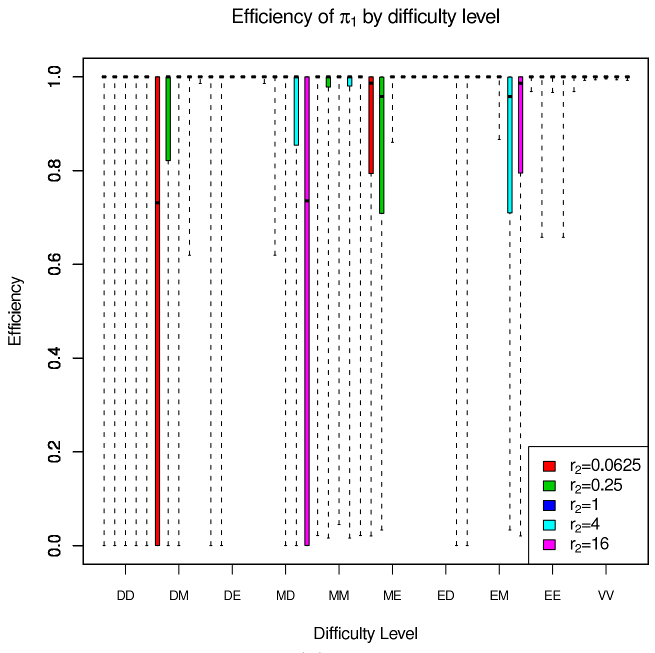
We showed that certain policies are assured a fraction of the optimal reward, while others, for which we have no theoretical guarantees, perform close to optimal for a wide variety of situations. We also showed the importance of adaptivity. The non-adaptive policy π_1 (it only ensures that a feasibility condition is satisfied) was consistently beaten, in experiments, by fully adaptive modifications ($\pi_2 - \pi_4$).

There are a number of directions for future work. Since the probabilities may be unknown, it would be interesting to combine this with the MAB problem [5, 6], to try to estimate the probabilities even as we optimize. Also, in practice, there may be a waiting time between when the ad is shown and when we find out if a click or a conversion happened, yet during this time a new user may have appeared, which ad should we display? Another direction is estimating the fair price r_i for an ad, given n_i and p_i as well as the other ads. This would be useful, especially when p_i and T are not known exactly.

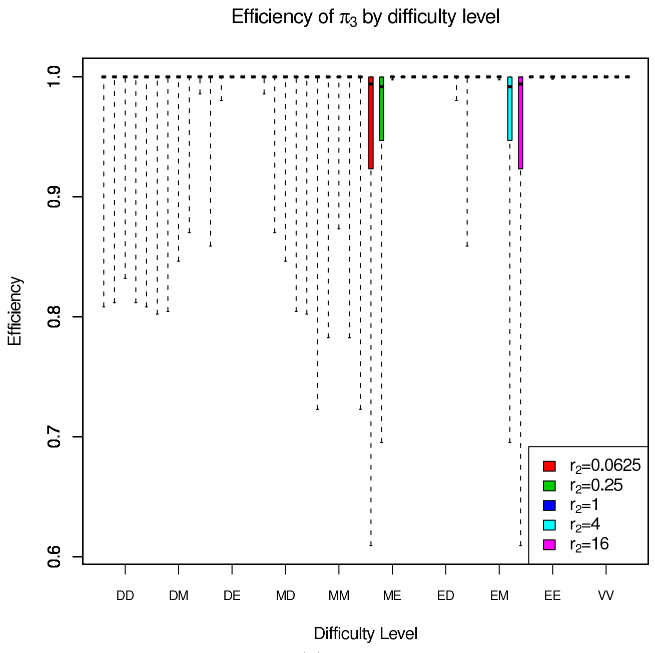
8. REFERENCES

[1] Chakrabarti, D., Agarwal, D., and Josifovski, V. Contextual advertising by combining relevance with click feedback. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web* (2008), pp. 417–426.

[2] Dean, B. C., Goemans, M. X., and Vondrák, J. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research 33*, 4 (Nov 2008), 945–964.



(a) π_1



(b) π_3

Figure 4: Efficiency of π_1 and π_3 by difficulty levels, with $k = 2$, $T = 300$

[3] Derman, C., Lieberman, G. J., and Ross, S. M. A renewal decision problem.l *Management Science 24*, 5 (1978), 554–561.

[4] Kan, A. H. G. R., and Stougie, L. On the relation between complexity and uncertainty. *Annals of Operations Research 18* (1989), 17–24.

[5] Langford, J., and Zhang, T. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. MIT Press, Cambridge, MA, 2008, pp. 817–824.

[6] Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *WWW '10: Proceedings of the 19th International Conference on World Wide Web* (2010), pp. 661–670.

[7] Martello, S., and Toth, P. *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley and Sons, Chichester, 1990.

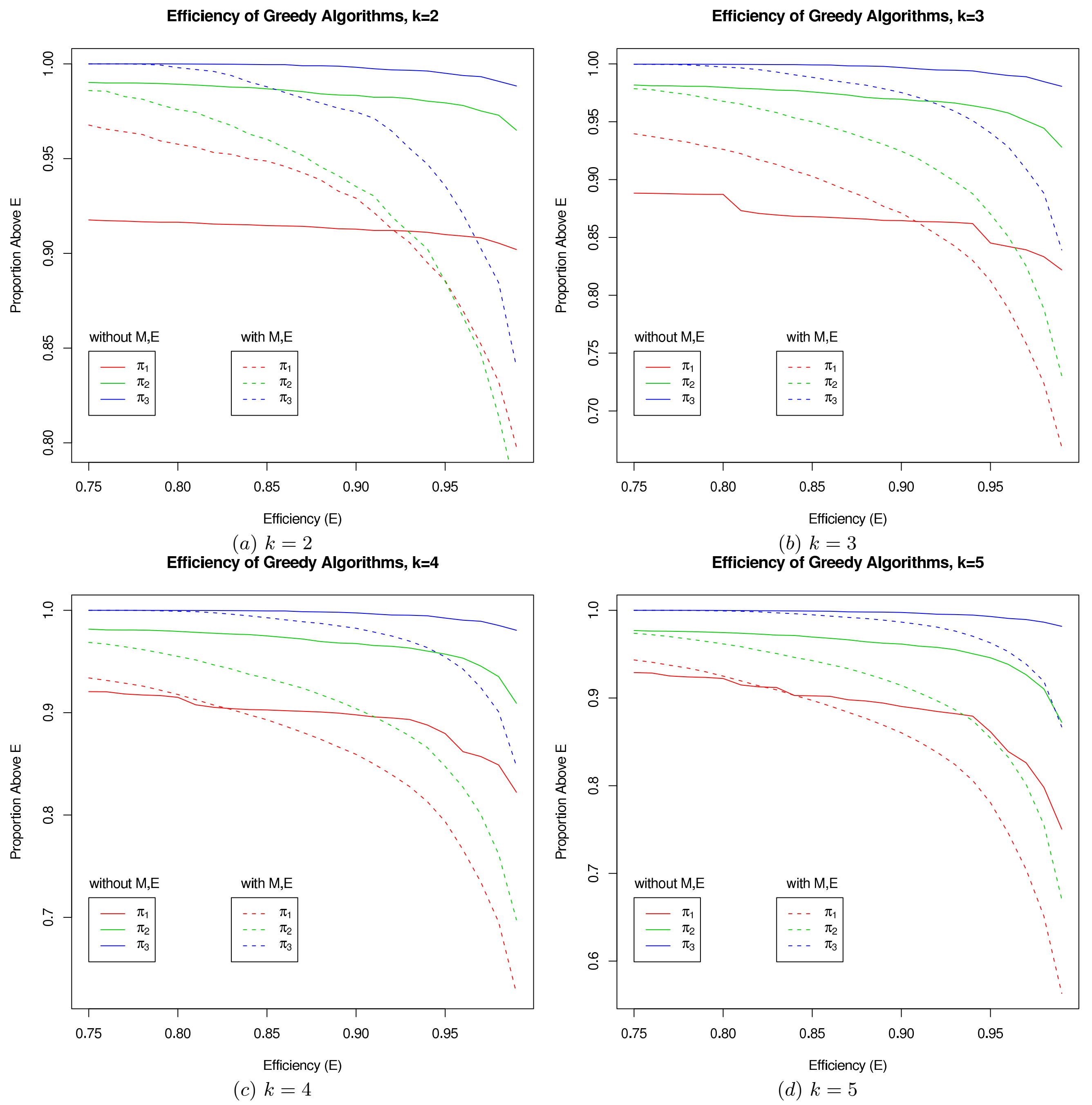


Figure 5: Efficiency of greedy policies with $1 \leq n_i \leq 20$ and $T = 1000$, for (a) $k = 2$, (b) $k = 3$, (c) $k = 4$, and (d) $k = 5$. Cases involving both difficulty levels M and E have been separated from the rest.