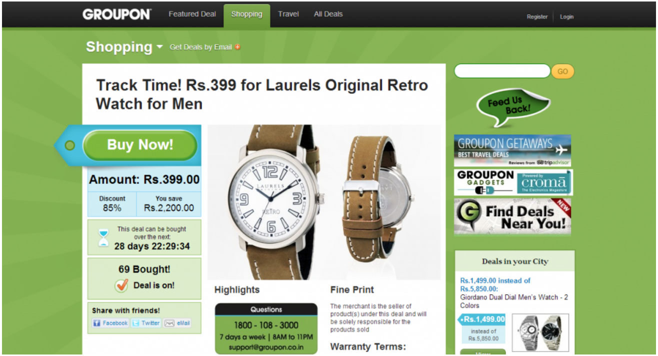
Adaptive Policies for Online Ad Selection Under Chunked Reward Pricing Model

Dinesh Garg
IBM India Research Lab, Bangalore
garg.dinesh@in.ibm.com

(Joint Work with **Michael Grabchak, Narayan Bhamidipati**, and **Rushi Bhatt**)

September 8, 2014

Motivation: Group Discount



- Groupon sells purchase vouchers at heavy discounts
- Only when a certain number of people sign up, the deal becomes effective
- If the predetermined minimum is not met, no one gets the deal
- # of customers and discount is jointly agreed by Groupon and merchant
- The revenue is split between merchant and Group (usually 50 : 50)

The Problem of Groupon: *How to Maximize Reward?*

T = # of user requests received within a fixed time interval

$T \sim F_T =$ Distribution of T (assumed to be known)

p_i = Probability of a user subscribing for the deal i

n_i = Minimum # of subscriptions required for the deal i to be ON

r_i = Reward of Groupon if at least n_i users subscribe

k = # of active deals in the fray

$\tilde{\mathbf{p}}, \tilde{\mathbf{r}}, \tilde{\mathbf{n}}$ = Vectors of p_i, r_i, and n_i, resp.

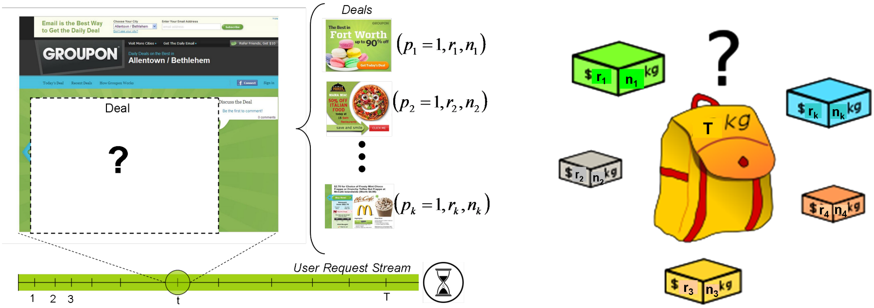
Key Assumptions

- We need to pick only one deal for display against every user request
- The reward r_i can be received at most once during the interval
- T is unknown but its distribution F_T is known
- (p_i, r_i, n_i) is known for each active deal
- A user subscribing for a deal is independent of T

Connection to Multi-Armed Bandit (MAB) Problem

- Our framework is similar to MAB problem except that we assume probabilities $\widetilde{\mathbf{p}}$ to be known
- Our goal is not to infer probabilities $\widetilde{\mathbf{p}}$ but to maximize the total reward
- We will stick to following convention
  - Refer to deals as arms
  - Pulling an arm i means displaying the deal i
  - A pull being successful means the user subscribing for that deal
  - Getting a reward from an arm means attaining the minimum # of subscriptions

Knapsack Connection and Hardness of the Problem



- Suppose T is known and $p_i = 1\ \forall i = 1 \rightarrow k$
- The problem reduces to standard $0-1$ knapsack problem
- Because Knapsack is NP-Hard [1], our problem must, in general, be NP-hard as well!

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

Connection to Stochastic Knapsack Problem

Stochastic Knapsack (SK) Problem

- Several versions of Stochastic Knapsack exist. Our setting is similar to [1]
- In SK problem, item i has a fixed and known value r_i but a random weight W_i, where $W_i \sim F_i$, satisfying $P(W_i \leq 0) = 0$
- One-by-one, items are placed into a fixed and known size (T) Knapsack
- Once an item has been inserted, we find out how big it is
- If item fits then we collect the reward otherwise not
- Even if item doesnot fit, it exhausts the remaining capacity of the knapsack

Our Problem v/s Stochastic Knapsack (SK) Problem

- Our problem allows T to be random
- In our setting, weight W_i of an arm i corresponds to a random number of times we need to pull this arm to get n_i subscriptions.
- $W_i \sim NB(n_i, p_i)$, where $p_i \in [0, 1]$ and $n_i \in \mathbb{N} \setminus \{0\}$
- In our setting, inserting an item is equivalent to keep showing a deal until we get reward

---

[1] B.C. Dean, M. Goemans, and J. Vondrák, *Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity*. Mathematics of Operations Research, 33 (4), pp. 945-964, 2008.

Optimality of Simple Greedy Scheme for Stochastic Knapsack

- In general, the Stochastic Knapsack problem (as defined earlier) is NP-hard
- There are situations where simple greedy algorithm is the optimal policy
  - Whenever, $W_i \sim Exp(\lambda_i)$ or $W_i \sim Geom(p_i) = NB(1, p_i)$
- The above result does not depend on T and hence we can extend it for our setting

Theorem 1: If for every deal i, we have $W_i \sim NB(1, p_i)$ then the optimal deal to show at time t is the one with the largest $r_i p_i$ from which we have not yet received a reward.

Policy Definition

- Notations

$$\begin{aligned}
\theta_t &= \text{The id of the deal that is shown at time } t \\
\delta_t &= \text{A } \{0,1\} \text{ random variable capturing the user's action at time } t \\
d_t &= \text{Realization of the random variable} \\
\widetilde{\boldsymbol{\theta}}, \widetilde{\boldsymbol{\delta}}, \widetilde{\mathbf{d}} &= \text{Vectors of } \theta_t,\ \delta_t, \text{ and } d_t, \text{ resp.} \\
S_i(t) &= \#\text{ of subscriptions for deal } i \text{ in the first } t \text{ impressions (R.V.)} \\
s_i(t) &= \text{Realization of } S_i(t) \\
\widetilde{\mathbf{S}}(\mathbf{t}), \widetilde{\mathbf{s}}(\mathbf{t}) &= \text{Vectors of } S_i(t) \text{ and } s_i(t), \text{ resp.}
\end{aligned}$$

- Policy (π)
  - 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
  - $$\theta_{t+1} = \pi(\widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{n}}, F_T \mid \{\theta_i, d_i\}_{i=1\to t})$$

(Expected) Reward and Optimal Policy

- (Expected) Reward
  - $$R(\pi, \widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{n}}, F_T \mid \widetilde{\boldsymbol{\theta}}, \widetilde{\boldsymbol{\delta}} = \widetilde{\mathbf{d}}) = \sum_{i=1}^{k} r_i \mathbf{1}_{[s_i(T) \geq n_i; \pi]}$$
  - $$ER(\pi, \widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{n}}, F_T \mid \widetilde{\boldsymbol{\theta}}, \widetilde{\boldsymbol{\delta}} = \widetilde{\mathbf{d}}) = \sum_{i=1}^{k} r_i P(S_i(T) \geq n_i; \pi)$$
- Optimal Policy (π^*)
  - $$\pi^* = \underset{\pi \in \Pi}{\operatorname{argsup}} \; ER(\pi, \widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{n}}, F_T \mid \widetilde{\boldsymbol{\theta}}, \widetilde{\boldsymbol{\delta}} = \widetilde{\mathbf{d}})$$

A Lookahead Procedure to Compute Optimal Policy

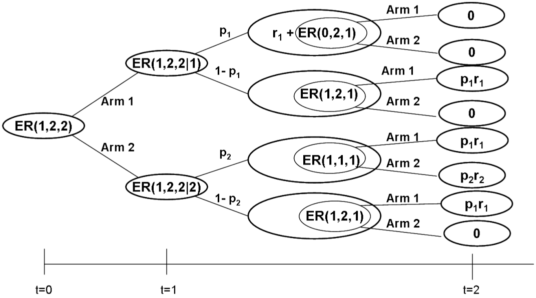
- The best arm to pull at time t, can be given by

$$
\begin{aligned}
i_t^* &= \underset{i=1\to k}{\text{argmax}} \left[p_i \left(r_i \mathbf{1}_{[n_i - s_i(t)=1]} + ER(\pi^*, \widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{n}} - \widetilde{\mathbf{s}}(\mathbf{t}) - \widetilde{\mathbf{e}}_{\mathbf{i}}, F_{T-t} \mid \phi) \right) \right. \\
&\quad \left. + (1 - p_i) ER(\pi^*, \widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{n}} - \widetilde{\mathbf{s}}(\mathbf{t}), F_{T-t} \mid \phi) \right]
\end{aligned}
$$

- The above policy can be shown to be optimal by induction technique whenever T has a bounded support, say T_0
- Since the problem is NP-hard, we can't hope to compute above policy efficiently for large scale problems
- Never-the-less, this trick could be useful for small scale problems

An Example of a Policy Tree

- Let us consider a scenario, where
  - k=2 (i.e. two arms to be pulled)
  - T is known and fixed, say $T = 2$
  - $n_1 = 1; n_2 = 2$ and $r_i p_i < r_2 p_2$
  - $ER(n_1, n_2, T)$ is a shorthand notation for $ER(\pi^*, \widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{n}}, F_T)$
- The Policy Tree for this example will look like this.



- From this example, it is clear that this policy computation is exponential in T

Practical Policies for Chunked Reward

We restrict our attention on the class of policies $\Pi_p \subset \Pi$ that satisfy the following *feasibility* criteria

- Arm i would be considered at time $t + 1$, only if $s_i(t) < n_i$ and $P(n_i - s_i(t) < T - t) > 0$
- All other parameters being equal, arm i would be chosen over arm j if $p_i > p_j$ or $r_i > r_j$, or $n_j > n_i$
- If r_i is multiplied by a constant for all the arms, the choice of the arm does not change

Remark: Any policy π which does not satisfy above criteria can be easily replaced by some policy $\pi' \in \Pi_p$ such that π' is uniformly better than π

(Adaptive) Greedy Policies

- We consider greedy policies that compute an index for each arm at any time and then choose the arm which maximizes this index.
- In light of criteria discussed earlier, we will consider only those greedy policies where the index for arm i is
  - non-decreasing function of p_i and r_i, and
  - non-increasing function of $n_i - s_i(t)$, and
  - involves all of p_i, r_i, and $n_i - s_i(t)$
  - linear in r_i so as to satisfy scale invariance criteria
- We consider the following 3 greedy policies

$$
\begin{aligned}
\text{Index}(\pi_1) &:= \frac{r_i p_i}{n_i - s_i(t)} \mathbf{1}_{[s_i(t) < n_i]} \mathbf{1}_{[n_i - s_i(t) \leq T - t]} \\
\text{Index}(\pi_2) &:= \frac{r_i p_i}{n_i - s_i(t)} P\left(W_i \leq T \mid S_i(t) = s_i(t)\right) \\
\text{Index}(\pi_3) &:= r_i P(W_i \leq T \mid S_i(t) = s_i(t))
\end{aligned}
$$

(Non-Adaptive) Greedy Policies

- Note, for greedy policies are adaptive in the sense that we need to compute the index at every time point
- For a non-adaptive greedy policy,
  - We compute indices only once (at the beginning), and
  - At each time t, pull the arm with the largest index for which we have not yet received a reward
- We denote the corresponding non-adaptive policy for π_i as γ_i
- Non-adaptive policies do not satisfy the feasibility criteria and hence can be uniformly improved

Greedy Policies May Be Arbitrarily Bad

- Consider a scenario having $p_i = 1 \ \forall i$ (i.e. $0 - 1$ Knapsack)
- Greedy policies $\pi_1, \pi_2, \gamma_1, \gamma_2$ reduce to standard greedy algorithm for this problem
- For greedy policy π_3 (and γ_3) consider the following scenario:
  - There are $k = T + 1$ arms
  - For arm 1, $r_1 = 2$ and $n_1 = T$
  - For all other arms, $r_i = 1$, $n_i = T$
- Following π_3 (and γ_3), we will only pull arm 1 and at the end get a reward of 2
- The optimal algorithm, however, is to never pull arm 1 and instead pull each of the other arms once to get a reward of T

Policies with Worst case Performance Guarantees

- Greedy policies can be arbitrary bad and hence we can't provide worst case performance bounds
- However, we can provide bounds for certain modified versions
- We introduce the following non-adaptive policy which will be used for analysis

$$\gamma_0 := \text{Always pull the arm with } \left[\max_i r_i P(W_i \le T)\right]$$

(even after we have received the reward from this arm)

$$\gamma_4 := \text{Choose } \gamma_0 \text{ and } \gamma_1 \text{ each with probability 0.5}$$

$$\gamma_5 := \text{Choose } \gamma_0 \text{ and } \gamma_2 \text{ each with probability 0.5}$$

$$\gamma_6 := \text{Choose } \gamma_1 \text{ if } \left[\max_i r_i P(W_i \le T) < ER(\gamma_1)\right], \text{ o/w choose } \gamma_0$$

$$\gamma_7 := \text{Choose } \gamma_2 \text{ if } \left[\max_i r_i P(W_i \le T) < ER(\gamma_2)\right], \text{ o/w choose } \gamma_0$$

Remark: Computing the quantities $P(W_i \le T), ER(\gamma_1)$, and $ER(\gamma_2)$ may not be straightforward and one may resort to simulation approaches for this.

Policies with Worst case Performance Guarantees

Theorem 2: Assume that the W_i's are mutually independent of themselves and of T. Then, we have

$$\sup_{\pi\in\Pi} ER(\pi) \leq \frac{2}{\min_i P(W_i \leq T)} ER(\gamma_4)$$

$$\sup_{\pi\in\Pi} ER(\pi) \leq \frac{2}{\min_i P(W_i \leq T)} ER(\gamma_5)$$

$$\sup_{\pi\in\Pi} ER(\pi) \leq \left(1 + \frac{1}{\min_i P(W_i \leq T)}\right) ER(\gamma_6)$$

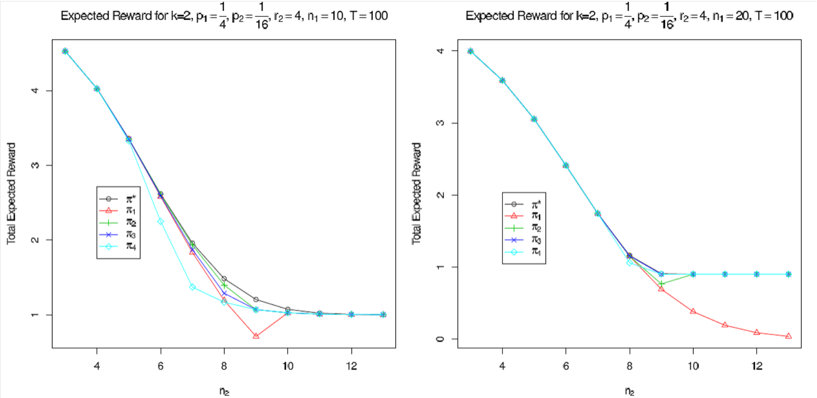
$$\sup_{\pi\in\Pi} ER(\pi) \leq \left(\frac{1 + \max_i P(W_i \leq T)}{\min_i P(W_i \leq T)}\right) ER(\gamma_7)$$

Policies with Worst case Performance Guarantees

Proof Sketch:

- Replace each arm i with n_i arms each yielding a reward of n_i/r_i after every success with success probability being as p_i. Call this as *fractional case*.
- For any policy $\pi \in \Pi$, if we pull the exact same sequence of arms in the fractional case as suggested by policy π for the original case, $ER(\text{Fractional Case}) \geq ER(\text{Original Case})$
- For fractional case, the optimal policy (as per earlier Theorem 1) is greedy policy with index $r_i p_i / n_i$.

Experimental Evaluation



(a) $n_1 = 10$ (b) $n_1 = 20$

Summary and Future Directions

- Introduced a variant of stochastic knapsack problem that can be used for goal based all-or-none pricing for online ads
- Provided feasible alternatives to the optimal policy
- 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
- A number of avenues for future directions, crucial one being the following
  - Combine this with MAB for situations where probabilities p_i need to be learned

Thank You!