A Multi-Arm Bandit Approach To Subset Selection Under Constraints

Ayush Deva
IIIT Hyderabad
ayushdeva97@gmail.com

Kumar Abhishek
IIIT Hyderabad
kumar.abhishek@research.iiit.ac.in

Sujit Gujar
IIIT Hyderabad
sujit.gujar@iiit.ac.in

ABSTRACT

We explore the class of problems where a central planner needs to select a subset of agents, each with different quality and cost. The planner wants to maximize its utility while ensuring that the average quality of the selected agents is above a certain threshold. When the agents' quality is known, we formulate our problem as an integer linear program (ILP) and propose a deterministic algorithm, namely DPSS that provides an exact solution to our ILP.

We then consider the setting when the qualities of the agents are unknown. We model this as a Multi-Arm Bandit (MAB) problem and propose DPSS-UCB to learn the qualities over multiple rounds. We show that after a certain number of rounds, τ, DPSS-UCB outputs a subset of agents that satisfy the average quality constraint with a high probability. Next, we provide bounds on τ and prove that after τ rounds, the algorithm incurs a regret of $O(\ln T)$, where T is the total number of rounds. We further illustrate the efficacy of DPSS-UCB through simulations. To overcome the computational limitations of DPSS, we propose a polynomial-time greedy algorithm, namely GSS, that provides an approximate solution to our ILP. We also compare the performance of DPSS and GSS through experiments.

KEYWORDS

Multi-Arm Bandits, Online Learning

1 INTRODUCTION

Almost all countries have cooperative societies that cater to developing sectors such as agriculture and handicrafts. We observed that some cooperatives, especially those that are consumer-oriented, such as Coop (Switzerland) or artisan cooperatives who operate their stores, lack a well-defined system to procure products from its many members (manufacturers, artisans, or farmers). Since the production is highly decentralized and usually not standardized, each producer has a different quality and cost of produce depending on various factors such as workmanship and the scale at which it operates. The central planner (say, the cooperative manager) has to carefully trade-off between each producer's qualities and cost to decide the quantity to procure from each producer so that it is most beneficial for the society as a whole.

This problem is not limited to cooperatives, but it is also faced in other familiar marketplaces. E-commerce platforms, like Amazon and Alibaba, have several sellers registered on their platform. For each product, the platform needs to select a subset of sellers to display on its page while ensuring that it avoids low-quality sellers and does not display only the searched product's high-cost variants. Similarly, a supermarket chain may need to decide the number of apples to procure from the regional apple farmers, each with a different quality of produce, to maximize profits while ensuring that the quality standards are met.

We formulate this as a subset selection problem where a central planner needs to select a subset of these sellers/producers, whom we refer to as agents. In this paper, we associate each agent with its quality and cost of production. The agent's quality refers to the average quality of the units produced by it; however, the quality of an individual unit of its product could be stochastic, especially in artistic and farm products. Thus, it becomes difficult to design an algorithm that guarantees constraint satisfaction on the realized qualities of the individual units procured. Towards this, we show that we achieve probably approximately correct (PAC) results by satisfying our constraint on the expected average quality of the units procured. Every unit procured from these agents generates revenue that is a function of its quality. The planner aims to maximize its utility (i.e., revenue - cost) while ensuring that the procured units' average quality is above a certain threshold to guarantee customer satisfaction and retention [1, 29]. When the agents' quality is known, we model our problem as an Integer Linear Program (ILP) and propose a novel algorithm, DPSS that provides an exact solution to our ILP.

Often, the quality of the agents is unknown to the planner beforehand. An E-commerce platform may not know its sellers' quality at the time of registration, and an artisan's quality of work may be hard to estimate until its products are procured and sold in the market. Thus, the planner needs to carefully learn the qualities by procuring units from the agents across multiple rounds while minimizing its utility loss. Towards this, we model our setting as a Multi-Arm Bandit (MAB) problem, where each agent represents an independent arm with an unknown parameter (here, quality). To model our subset selection problem, we consider the variant of the classical MAB setting where we may select more than one agent in a single round. This setting is popularly referred to as a Combinatorial MAB (CMAB) problem [12, 17, 24]. In studying CMAB, we consider the semi-bandit feedback model where the algorithm observes the quality realizations corresponding to each of the selected arms and the overall utility for selecting the subset of arms. The problem becomes more interesting when we also need to ensure our quality constraint in a CMAB problem. We position our work with respect to the existing literature in Section 2.

Typically, in a CMAB problem, the planner's goal is to minimize the *expected regret*, i.e., the difference between the expected cumulative utility of the best offline algorithm with known distributions of an agent's quality and the expected cumulative reward of the algorithm. However, the traditional definition of regret is not suitable in our setting as an optimal subset of agents (in terms of utility) may violate the quality constraint. Thus, we modify the regret definition to make it compatible with our setting. We propose a novel, UCB-inspired algorithm, DPSS-UCB, that addresses the subset selection problem when the agents' quality is unknown. We show that after

a certain threshold number of rounds, τ, the algorithm satisfies the quality constraint with a high probability for every subsequent round, and under the revised regret definition, it incurs a regret of $O(\ln T)$, where T is the total number of rounds.

To address the computational challenges of DPSS which has a time complexity of $O(2^n)$, we propose a greedy-based algorithm, GSS that runs in polynomial time $O(n \ln n)$, where n is the number of agents. We show that while the approximation ratio of the utility achieved by GSS to that of DPSS can be arbitrarily small in the worst case, it achieves almost the same utility as DPSS in practice, which makes GSS a practical alternative to DPSS especially when n is large.

In summary, our contributions are:

- We propose a framework, SS-UCB, to model subset selection problem under constraints when the properties (here, qualities) of the agents are unknown to the central planner. In our setting, both the objective function and the constraint depends on the unknown parameter.
- We first formulate our problem as an ILP assuming the agents' quality to be known and propose a novel, deterministic algorithm, namely DPSS(Algorithm 1) to solve the ILP.
- Using DPSS, we design DPSS-UCB which addresses the setting where the agents' quality is unknown. We prove that after a certain number of rounds, $\tau = O(\ln T)$, DPSS-UCB satisfies quality constraint with high probability. We also prove that it achieves a regret of $O(\ln T)$ (Theorem 1).
- To address the computational limitation of DPSS, we propose an alternative greedy approach, GSS and GSS-UCB, that solves the known and the unknown settings, respectively. We show that while the greedy approach may not be optimal, it performs well in practice with a huge computational gain that allows our framework to scale to settings with a large number of agents.

The remaining of the paper is organized as follows: In Section 2, we discuss the related works. In Section 3, we define our model and solve for the setting when the quality of the agents are known. In Section 4, we address the problem when the quality of the agents is unknown. In Section 5, we propose a greedy approach to our problem. In Section 6, we discuss our simulation-based analysis and conclude the paper in Section 7.

2 RELATED WORK

Subset selection is a well-studied class of problems that finds its applications in many fields, for example, in retail, vehicle routing, and network theory. Usually, these problems are modeled as knapsack problems where a central planner needs to select a subset of agents that maximizes its utility under budgetary constraints [33]. There are several variations to the knapsack, such as robustness [26], dynamic knapsacks [25], and knapsack with multiple constraints [27] studied in the literature. In this paper, we consider a variant where the constraint is not additive, i.e., adding another agent to a subset doesn't always increase the average quality.

When online learning is involved, the stochastic multi-armed bandit (MAB) problem captures the exploration vs. exploitation trade-off effectively [6, 19, 21–23, 28, 31, 32]. The classical MAB problem involves learning the optimal agent from a set of agents with a fixed but unknown reward distribution [3, 7, 28, 30]. Combinatorial MAB (CMAB) [8–10, 13, 16] is an extension to the classical MAB problem where multiple agents can be selected in any round. In [10, 11, 17], the authors have considered a CMAB setting where they assume the availability of a feasible set of subsets to select from. The key difference with our setting is that our constraint itself depends on the unknown parameter (quality) that we are learning through MAB. Thus, the feasible subsets that satisfy the constraint need to be learned, unlike the previous works. [10, 11, 17] also assumes the availability of an oracle that outputs an optimal subset given the estimates of the parameter as input, whereas we design such an oracle for our problem. Bandits with Knapsacks (BwK) is another interesting extension that introduces constraints in the standard bandit setting [2, 4, 5, 22] and finds its applications in dynamic pricing, crowdsourcing, etc. (see [2, 4]). Typically, in BwK, the objective is to learn the optimal agent(s) under a budgetary constraint (e.g., a limited number of selections) that depends solely on the agents' cost. However, we consider a setting where the selected subset needs to satisfy a quality constraint that depends on the learned qualities.

The closest work to ours is Jain et al. [22] where the authors present an assured accuracy bandit (AAB) framework where the objective is to minimize cost while ensuring a target accuracy level in each round. While they do consider a constraint setting similar to ours, the objective function in [22] depends only on the agents' cost and not on the qualities of the agents that are unknown. Hence, it makes our setting different and more generalizable with respect to both AAB and CMAB as in our setting, both the constraint and the utility function depend on the unknown parameter.

3 SUBSET SELECTION WITH KNOWN QUALITIES OF AGENTS

Here we assume that the agents' quality is known and consider the problem where a central planner C needs to procure multiple units of a particular product from a fixed set of agents. Each agent is associated with the quality and cost of production. C's objective is to procure the units from the agents such that the average quality of all the units procured meets a certain threshold. We assume that there is no upper limit to the number of units it can procure as long as the quality threshold is met.

In Section 3.1, we define the notations required to describe our model, formulate it as an integer linear program (ILP) in Section 3.3, and propose a solution to it in Section 3.4.

3.1 Model and Notations

(1) There is a fixed set of agents $N = \{1; 2; \colon \colon \colon ; n\}$ available for selection for procurement by planner C.
(2) Agent i has a cost of production, c_i, and capacity, k_i.
(3) The quality of the j^{th} unit of produce by agent i is denoted by Q_{ij}, which we model as a Bernoulli random variable.
(4) For any agent i, the probability that Q_{ij} is 1 is defined by q_i, i.e., $E[Q_{ij}] = q_i$ for any unit j procured from agent i. q_i is also referred to as the quality of the agent in the rest of the paper.

(5) The utility for C to procure a single unit of produce from agent i is denoted by r_i, which is equal to its expected revenue[1] minus the cost of production, i.e., $r_i = Rq_i - c_i$, where R is the proportionality constant.
(6) The quantity of products procured by C from the i^{th} agent is given by x_i.
(7) The average quality of products procured by C is therefore equal to $\frac{\sum_{i \in N} \sum_{j=1}^{x_i} Q_{ij}}{\sum_{i \in N} x_i}$.
(8) We define $q_{av} = \frac{\sum_{i \in N} x_i q_i}{\sum_{i \in N} x_i}$, which is the expected average quality of the units procured by C.
(9) C needs to ensure that the average quality of all the units procured is above a certain threshold, $\alpha \in [0; 1]$.
(10) The total utility of C is given by, $z = \sum_{i \in N} x_i r_i$.

Usually, an individual unit's quality Q_{ij} may not be quantifiable and can only be characterized by observing whether it was sold. Hence, we model it as a Bernoulli random variable.

3.2 Ensuring Quality Constraints

In our setting, average quality (Section 3.1, point 7) is dependent on Q_{ij}, which is stochastic in nature. In such a stochastic framework, it is more natural to work with expected terms than on a sequence of realized values. Towards this, we show that by ensuring our quality constraint on expected average quality, q_{av}, instead, we can still achieve approximate constraint satisfaction with a high probability. Formally, we present the following lemma,

LEMMA 1. *The probability that average quality is less than $\alpha - \epsilon$ given that $q_{av} \geq \alpha$, can be bounded as follows:*

$$\mathcal{P}\left(\frac{\sum_{i \in N} \sum_{j=1}^{x_i} Q_{ij}}{\sum_{i \in N} x_i} < \alpha - \epsilon \mid q_{av} \geq \alpha \right) \leq \exp(-2\epsilon^2 m));$$

where $m = \sum_{i \in N} x_i$, and ϵ is a constant.

PROOF. Let, $V = \frac{\sum_{i \in N} \sum_{j=1}^{x_i} Q_{ij}}{\sum_{i \in N} x_i}$

$$E[V] = \frac{\sum_{i \in N} \sum_{j=1}^{x_i} E[Q_{ij}]}{\sum_{i \in N} x_i} = \frac{\sum_{i \in N} q_i x_i}{\sum_{i \in N} x_i} = q_{av}$$

Therefore,

$$\mathcal{P}\ V < \alpha - \epsilon \mid E[V] \geq \alpha \ \leq \mathcal{P}\ V < E[V] - \epsilon)$$
$$= \mathcal{P}\ V - E[V] < -\epsilon) \leq exp(-2\epsilon^2 m)$$

The last line follows from the Hoeffding's inequality [20].

From the above lemma, we show that by ensuring $q_{av} \geq \alpha$, we can achieve probably approximate correct (PAC) results on our constraint. Hence, for the rest of the paper, we work with $q_{av} \geq \alpha$ as our quality constraint (QC).

[1]We assume expected revenue to be proportional to the quality of the product. It is a reasonable assumption as if q_i is the probability of the product being sold and R is the price of the product, its expected revenue would be Rq_i

3.3 Integer Linear Program (ILP)

When the qualities of the agents are known, the planner's subset selection problem can be formulated as an ILP where it needs to decide on the number of units, x_i, to procure from each agent i so as to maximize its utility (objective function) while ensuring the quality and capacity constraints. The optimization problem can be described as follows:

$$\begin{aligned} \max_{x_i} \quad & \sum_{i \in N} (Rq_i - c_i)x_i \\ \text{s.t. } & q_{av} = \frac{\sum_{i \in N} q_i x_i}{\sum_{i \in N} x_i} \\ & q_{av} \geq \alpha \\ & 0 \leq x_i \leq k_i \qquad \forall i \in N \\ & x_i \in \mathbb{Z} \qquad \forall i \in N \end{aligned} \tag{1}$$

3.4 Dynamic Programming Based Subset Selection (DPSS)

In order to solve the ILP, we propose a dynamic programming based algorithm, called DPSS. For ease of exposition, we consider $k_i = 1$, i.e., each agent has a unit capacity of production. This is a reasonable assumption that doesn't change our algorithm's results, since, for an agent with $k_i > 1$, we can consider each unit as a separate agent, and the proofs and discussion henceforth follows.

Formally, the algorithm proceeds as follows:

(1) Divide the agents into one of the four categories:
  (a) S_1: Agents with $q_i \geq \alpha$ and $r_i \geq 0$
  (b) S_2: Agents with $q_i < \alpha$ and $r_i \geq 0$
  (c) S_3: Agents with $q_i \geq \alpha$ and $r_i < 0$
  (d) S_4: Agents with $q_i < \alpha$ and $r_i < 0$
(2) Let $x = \{x_i\}_{i \in N}$ be the selection vector, where $x_i = 1$ if the i^{th} agent is selected and 0 otherwise.
(3) Since an agent in S_1 has a positive utility and above threshold quality, $x_i = 1, \forall i \in S_1$. Let $d = \sum_{i \in S_1} (q_i - \alpha)$ be the excess quality accumulated.
(4) Similarly, all units in S_4 have a negative utility and below threshold quality. Hence, $x_i = 0, \forall i \in S_4$.
(5) Let G be the set of the remaining agents (in S_2 and S_3). For each agent $i \in G$, we define $d_i = q_i - \alpha$. Thus, we need to select the agents $i \in G$ that maximizes the utility, such that $\sum_{i \in G} x_i d_i \leq d$.
(6) For agents in G, select according to the DP function defined in Algorithm 1 (Lines [8-16]). Here, d^{te} denotes the access quality accumulated before choosing the next agent and $x^t e$ refers to the selections made so far in the DP formulation.

4 SUBSET SELECTION WITH UNKNOWN QUALITIES OF AGENTS

In the previous section, we assumed that the qualities of the agents, q_i, are known to C. We now consider a setting when q_i are *unknown* beforehand and can only be learned by selecting the agents. We model it as a CMAB problem with semi-bandit feedback and QC.

Algorithm 1 DPSS

Inputs: N, α, R, costs $c = \{c_i\}_{i \in N}$, qualities $q = \{q_i\}_{i \in N}$
Output: Quantities procured $x = (x_1, :::, x_n)$
Initialization: $\forall i \in N, r_i = Rq_i - c_i, z = 0$
Segregate S_1,S_2,S_3,S_4 as described in Section 3.4
$\forall i \in S_1, x_i = 1; z = z + r_i; d = \sum_{i \in S_1} (q_i - \alpha)$
$\forall i \in S_4, x_i = 0$
$G = S_2 \cup S_3$; $\forall i \in G; d_i = q_i - \alpha$
function DP($i; d^{te}; x^{te}; x^\star; z^{te}; z^\star$)
  if $i == |G|$ and $d^{te} < 0$ **then return** $x^\star; z^\star$
  if $i == |G|$ and $d^{te} \geq 0$ **then**
    if $z^{te} > z^\star$ **then**
      $z^\star = z^{te}; x^\star = x^{te}$
    return $x^\star; z^\star$
  $x^\star; z^\star = DP(i + 1; d^{te}; [x^{te}; 0]; x^\star; z^{te}; z^\star)$
  $x^\star; z^\star = DP(i + 1; d^{te} + d_i; [x^{te}; 1]; x^\star; z^{te} + r_i; z^\star)$
  return $x^\star; z^\star$
$x^G; z^G$ = DP(0,d,[ ],[ ],0,0)
$\forall i \in G; x_i = x_i^G$
return x

4.1 Additional Notations

We introduce the additional notations to model our problem. Similar to our previous setting, we assume that we are given a fixed set of agents, N, each with its own average quality of produce, q_i and cost of produce, c_i. Additionally, our algorithm proceeds in discrete rounds $t = 1; :::; T$. For a round t:

- Let $x^t \in \{0; 1\}^n$ be the selection vector at round t, where $x_i^t = 1$ if the agent i is selected in round t and $x_i^t = 0$ if not.
- The algorithm selects a subset of agents, $S^t \subseteq N$, referred to as a super-arm henceforth, where $S^t = \{i \in N | x_i^t = 1\}$. Let s^t be cardinality of selected super-arm, i.e., $s^t = |S^t|$.
- Let w_i^t denote the number of rounds an agent i has been selected until round t, i.e., $w_i^t = \sum_{y \leq t} x_i^y$.
- For each agent $i \in S^t$, the planner, C, observes its realized quality X_i^j, where $j = w_i^t$ and $\mathrm{E}[X_i^j] = q_i$. For an agent $i < S_t$, we do not observe its realized quality (semi-bandit setting).
- The empirical mean estimate of q_i at round t, is denoted by $\hat{q}_i^t = \frac{1}{w_i^t} \sum_{j=1}^{w_i^t} X_i^j$. The upper confidence bound (UCB) estimate is denoted by $(\hat{q}_i^t)^+ = \hat{q}_i^t + \sqrt{\frac{3 \ln t}{2 w_i^t}}$.
- Utility to C at round t is given by: $r_q(S^t) = \sum_{i \in S^t} Rq_i - c_i$, where $q = \{q_1; q_2; :::; q_n\}$ is the quality vector.
- The expected average quality of selected super-arm at round t is given by: $q_{av}^t = \frac{1}{s^t} \sum_{i \in S^t} q_i$.

Following from Lemma 1, we continue to work with expected average quality instead of realized average quality.

4.2 SS-UCB

In this section, we propose an abstract framework, SS-UCB, for subset selection problem with quality constraint. SS-UCB assumes that there exist an offline subset selection algorithm, SSA, (e.g., DPSS), which takes a vector of qualities, q', and costs, c', along with the target quality threshold, α', and proportionality constant, R, as an input and returns a super-arm which satisfies the quality constraint (QC) with respect to q' and α'.

SS-UCB runs in two phases: (i) Exploration: where all the agents are explored for certain threshold number of rounds, τ ; (ii) Explore-exploit: We invoke SSA (line 10, Algorithm 2) with $\{(\hat{q}_i^t)^+\}_{i \in N}$, $\{c_i\}_{i \in N}$, $\alpha + \epsilon_2$ and R as the input parameters and select accordingly. We invoke SSA with a slightly higher target threshold, $\alpha + \epsilon_2$, so that our algorithm is more conservative while selecting the super-arm in order to ensure QC with a high probability (discussed in Section 4.3). As we shall see in Section 4.3, the higher the value of ϵ_2, the sooner the SSA satisfies QC with a high probability but it comes with the cost of loss in utility. Thus, the value of ϵ_2 must be appropriately selected based on the planner's preferences.

We refer to the algorithm as DPSS-UCB when we use DPSS (Algorithm 1) as SSA in the SS-UCB framework. We show that DPSS-UCB outputs the super-arm that satisfies the QC with high probability (w.h.p) after a certain threshold number of rounds, τ, and incurs a regret of $O(\ln T)$.

Algorithm 2 SS-UCB

1: **Inputs:** N, α, ϵ_2, R, costs $c = \{c_i\}_{i \in N}$
2: For each agent i, maintain: $w_i^t, \hat{q}_i^t, (\hat{q}_i^t)^+$
3: $\tau \leftarrow \frac{3 \ln T}{2\epsilon_2^2}$; $t = 0$
4: **while** $t \leq \tau$ (**Explore Phase**) **do**
5: Play a super-arm $S^t = N$
6: Observe qualities $X_i^j; \forall i \in S^t$ and update $w_i^t, \hat{q}_i^t$
7: $t \leftarrow t + 1$
8: **while** $t \leq T$ (**Explore-Exploit Phase**) **do**
9: For each agent i, set $(\hat{q}_i^t)^+ = \hat{q}_i^t + \sqrt{\frac{3 \ln t}{2 w_i^t}}$
10: S^t = SSA ($\{(\hat{q}_i^t)^+\}_{i \in N}; c; \alpha + \epsilon_2$,R)
11: Observe qualities $X_i^j; \forall i \in S^t$ and update $w_i^t, \hat{q}_i^t$
12: $t \leftarrow t + 1$

4.3 Ensuring Quality Constraints

We provide Probably Approximate Correct (PAC) [14, 18] bounds on DPSS-UCB satisfying QC after τ rounds:

THEOREM 1. *For $\tau = \frac{3 \ln T}{2\epsilon_2^2}$, if each agent is explored τ number of rounds, then if we invoke DPSS with target threshold $\alpha + \epsilon_2$ and $\{(\hat{q}_i^t)^+\}_{i \in N}$ as the input, the QC is approximately met with high probability.*

$$\mathcal{P}\left(q_{av}^t < \alpha - \epsilon_1 \mid \frac{1}{s^t} \tilde{\sum_{i \in S^t}} (\hat{q}_i^t)^+ \geq \alpha + \epsilon_2; t > \tau \right) \leq exp(-\epsilon_1^2 t):$$

where ϵ_1 is the tolerance parameter and refers to the planner's ability to tolerate a slighty lower average quality than required.

Henceforth, a super-arm will be called *correct* if it satisfies the QC approximately as described above.

PROOF. The proof is divided into two parts. Firstly, we show that for each $t > \tau$ round, the average value of $(\hat{q}_i^t)^+$ and that of $\hat{q}_i^t$ of the agents i in selected super-arm S^t is less than ϵ_2. Secondly, we show that if the average of $\hat{q}_i^t$ is guaranteed to be above the threshold, then the average of q_i over the selected agents would not be less than $\alpha - \epsilon_1$ with a high probability.

LEMMA 2. *The difference between the average of $(\hat{q}_i^t)^+$ and the average of $\hat{q}_i^t$ over the agents i in S^t is less than ϵ_2, $\forall t > \tau$.*

PROOF. We have,

$$\frac{1}{s^t}\sum_{i\in S^t}(\hat{q}_i^t)^+ - \hat{q}_i^t = \frac{1}{s^t}\sum_{i\in S^t}\sqrt{\frac{3\ln t}{2w_i^t}} \le \sqrt{\frac{3\ln t}{2w_{min}^t}}:$$

where $w_{min}^t = \min_i w_i^t$. Since, for $t < \tau$, we are exploring all the agents, thus, $w_i^t = \tau$. Now, since $w_i^t \ge w_i^\tau$, $\forall t > \tau$, thus, we claim that $w_{min}^t \ge \tau$ for $t > \tau$. Hence,

$$\sqrt{\frac{3\ln t}{2w_{min}^t}} \le \frac{\sqrt{3\ln T}}{\sqrt{2\tau}}:$$

For $\tau = \frac{3\ln T}{2\epsilon_2^2}$, we have,

$$\frac{1}{s^t}\sum_{i\in S^t}(\hat{q}_i^t)^+ - \hat{q}_i^t \le \epsilon_2:$$

LEMMA 3. $\forall t > \tau$

$$\mathcal{P}\left(q_{av}^t < \alpha - \epsilon_1 \mid \frac{1}{s^t}\sum_{i\in S^t}\hat{q}_i^t) \ge \alpha\right) \le exp(-\epsilon_1^2 t):$$

PROOF. Let $Y^t = \frac{1}{s^t}\sum_{i\in S^t}\hat{q}_i^t$. Since $\mathrm{E}[\hat{q}_i^t] = \mathrm{E}[X_i^j] = q_i$, $\mathrm{E}[Y^t] = q_{av}^t$. Hence, we have,

$$\mathcal{P}(E[Y^t] < \alpha - \epsilon_1) \mid Y^t \ge \alpha) \le \mathcal{P}\ Y^t \ge E[Y^t] + \epsilon_1) \le exp(-\epsilon_1^2 w^t):$$

where $w^t = \sum_{i\in S^t} w_i^t$, i.e., total number of agents selected till round t. Since we pull atleast one agent in each round, we can say that, $w^t \ge t$. Thus, $\forall t > \tau$

$$\mathcal{P}\left(q_{av}^t < \alpha - \epsilon_1 \mid \frac{1}{s^t}\sum_{i\in S^t}\hat{q}_i^t) \ge \alpha\right) \le exp(-\epsilon_1^2 t):$$

From Lemma 2 and Lemma 3, the proof follows.

4.4 Regret Analysis of DPSS-UCB

In this section, we propose the regret definition for our problem setting that encapsulates the QC. We then upper bound the regret incurred by DPSS-UCB to be of the order $O(\ln T)$.

We define regret incurred by an algorithm A on round t as follows:

$$Reg^t(A) = \begin{cases} (r_{\mathbf{q}}(S^?) - r_{\mathbf{q}}(S^t)) & \text{if } S^t \text{ satisfies QC} \\ L & \text{otherwise:} \end{cases}$$

where $S^? = argmax_{S\in S_f} r_{\mathbf{q}}(S)$ and $L = \max_{S\in S_f}(r_{\mathbf{q}}(S^?) - r_{\mathbf{q}}(S))$ is some constant. Here, S_f are the feasible subsets which satisfies QC; $S_f = \{S | S \subseteq N and \frac{\sum_{i\in S} x_i q_i}{\sum_{i\in S} x_i} \ge q_{av}\}$.

Hence, the cumulative regret in T rounds incurred by the algorithm is:

$$Reg(A) = \sum_{t=1}^{T} Reg^t(A): \tag{2}$$

We now analyse the regret when the algorithm, A, is DPSS-UCB.

$$\begin{aligned} Reg(A) &= \sum_{t=1}^{\tau} Reg^t(A) + \sum_{t=\tau+1}^{T} Reg^t(A) \\ &\le L\cdot\tau + \sum_{t=\tau+1}^{T} Reg^t(A) \\ &\le \frac{L\cdot 3\ln T}{2\epsilon_2^2} + \sum_{t=\tau+1}^{T} Reg^t(A): \end{aligned}$$

Since our algorithm ensures that S^t satisfies the approximate QC for $t > \tau$ with a probability greater than $1-\sigma$, where $\sigma = exp(-\epsilon_1^2 t)$, we have,

$$\mathbb{E}[Reg(A)] \le \frac{L\cdot 3\log T}{2\epsilon_2^2} + \underbrace{\sum_{t\ge\tau}(1-\sigma)(r_{\mathbf{q}}(S^?) - r_{\mathbf{q}}(S^t))}_{Re\partial_u(T)} + \sigma L\Big]: \tag{3}$$

where $S^t \in S_f$.

Now,

$$\sum_{t\ge\tau}\sigma L = \sum_{t\ge\tau} Le^{(-\epsilon_1^2 t)} \le \frac{Le^{(-\epsilon_1^2\tau)}}{1-e^{(-\epsilon_1^2)}} \sim O\left(\frac{1}{T^a}\right) \text{; where } a = \frac{3\epsilon_1^2}{2\epsilon_2^2}:$$

Now we bound the cumulative regret incurred after $t > \tau$ rounds when QC is satisfied, i.e., $Reg_U(T)$. Here we adapt the regret proof given by Chen et al. [10]. We highlight the similarities and differences of our setting with theirs and use it to bound $Reg_U(T)$.

Bounding $Reg_U(T)$:

Chen et al. [10] have proposed CUCB algorithm to tackle CMAB problem which they prove to have an upper bound regret of $O(\ln T)$. Following is the CMAB problem setting considered in [10]:

- There exists a constrained set of super-arms $\chi \subseteq 2^N$ available for selection.
- There exists an offline (η, ν)-approximation oracle, $(\eta, \nu \le 1)$ s.t. for a given quality vector $\mathbf{q}'$ as input, it outputs a super-arm, S, such $\mathcal{P}(r_{\mathbf{q}'}(S) \ge \eta \cdot opt_{\mathbf{q}'}) \ge \nu$, where $opt_{\mathbf{q}'}$ is the optimal reward for quality vector $\mathbf{q}'$ as input.
- Their regret bounds hold for any reward function that follows the properties of monotonicity and bounded smoothness (defined below).
- Similar to our setting, they assume a semi-bandit feedback mechanism.

Now, we state the reasons to adopt the regret analysis provided by Chen et al. [10] to bound $Reg_U(T)$

(1) We have shown that after τ rounds, we get the constrained set of super-arms, χ, i.e., the set of super-arms that satisfies QC), which forms a well defined constrained set, to select from in future rounds ($t > \tau$).
(2) We remark here that the utility function considered in our problem setting follows both the required properties, namely,

(i) *Monotonicity:* The expected reward of playing any super-arm $S \in \chi$ is monotonically non-decreasing with respect to the quality vector, i.e., let $\boldsymbol{q}$ and $\overline{\boldsymbol{q}}$ be two quality vectors such that $\forall i \in N$, $q_i \leq \overline{q}_i$, we have $r_{\boldsymbol{q}}(S) \leq r_{\overline{\boldsymbol{q}}}(S)$ for all $S \in \chi$. Since our reward function is linear, it is trivial to note that it is monotone on qualities.
(ii) *Bounded Smoothness:* There exists a strictly increasing (and thus invertible) function $f(:)$, called bounded smoothness function, such that for any two quality vectors $\boldsymbol{q}$ and $\overline{\boldsymbol{q}}$, we have $r_{\boldsymbol{q}}(S) - r_{\overline{\boldsymbol{q}}}(S) \leq \mathrm{f}(\Lambda)$ if $\max_{i \in S} q_i - \overline{q}_i \leq \Lambda$. As our reward function is linear in qualities, $f(\Lambda) = nR \times \Lambda$ is the bounded smoothness function for our setting, where n is the number of agents.

(3) *Oracle*: Analogous to the oracle assumption in [10], we have assumed the existence of an algorithm SSA (Section 4.2). For DPSS-UCB, we use DPSS (Algorithm 1) as our SSA . As DPSS provides exact solution, it acts as an $(\eta; \nu)$- approximate oracle for DPSS-UCB with $\eta = 1 = \nu$.

However, to ensure χ consists of all the correct super-arms, we need one additional property that should be satisfied, namely ϵ-seperatedness property.

DEFINITION 1. *We say* $\boldsymbol{q} = (q_1; q_2; : : : ; q_n)$ *satisfies* ϵ*-seperatedness if* $\forall S \subseteq N$, $U(S) = \frac{1}{S} \sum_{i \in S} q_i$ *s.t.* $U(S) < (\alpha - \epsilon; \alpha)$

This suggests that there is no super-arm $S \in \chi$, such that $\alpha - \epsilon \leq \frac{1}{|S|} \sum_{i \in S} q_i^t \leq \alpha$. It is important for DPSS-UCB to satisfy ϵ_1-seperatedness because if there exists such a super-arm, for which the average quality is between $(\alpha - \epsilon_1, \alpha)$, DPSS-UCB will include it in χ due to tolerance parameter ϵ_1 while it would violate the QC.

THEOREM 2. *If qualities of the agents satisfy* ϵ_1*-seperatedness, then* $Reg_U(T)$ *is bounded by* $O(\ln T)$.

PROOF. Following from the proof in [10], we define some parameters. A super-arm, S is bad if $r_{\boldsymbol{q}}(S) < opt_{\boldsymbol{q}}$. Define S_B as the set of bad super-arms. For a given underlying agent $i \in [n]$, define:

$$\Delta^i_{\min} = \mathrm{opt}_{\boldsymbol{q}} - \max\{r_{\boldsymbol{q}}(S) | S \in S_B; i \in S\}$$
$$\Delta^i_{\max} = \mathrm{opt}_{\boldsymbol{q}} - \min\{r_{\boldsymbol{q}}(S) | S \in S_B; i \in S\}:$$

Using the same proof as in [10], we can show that, V_T, the expected number of times we play a sub-optimal agent till round T, is upper bounded as:

$$V_T \leq n(l_T) + \sum_{t=\tau}^{\infty} \frac{2n}{t^2} \leq n(l_T) + \sum_{t=1}^{\infty} \frac{2n}{t^2}$$
$$\leq \frac{6n \cdot \ln T}{(f^{-1}(\Delta_{\min}))^2} + \frac{\pi^2}{3} \cdot n:$$

where $l_T = \frac{6 \ln T}{(f^{-1}(\Delta_{\min}))^2}$. Hence, we can bound the regret as:-

$$Reg_U(T) \leq V_T \cdot \Delta_{\max} \leq \frac{6 \cdot \ln T}{(f^{-1}(\Delta_{\min}))^2} + \frac{\pi^2}{3} \cdot n \cdot \Delta_{\max}$$
$$= \frac{6 \cdot \ln T}{(\frac{\Delta_{\min}}{R})^2} + \frac{\pi^2}{3} \cdot n \cdot \Delta_{\max}:$$

Substituting the results of Theorem 2 in Equation 3, we prove that DPSS-UCB incurs a regret of $O(\ln T)$.

5 GREEDY APPROACH

In the previous sections, we propose a framework and dynamic programming based algorithm to solve our subset selection problem for both when the agents' quality is known and not. Since DPSS explores all the possible combinations of the selection vector and the utility associated with it, the complexity of DPSS is of $O(2^n)$, which makes it difficult to scale when n is large.

To overcome this limitation, we propose a greedy based approach to our problem. When the quality of agents are known, we propose GSS that runs in polynomial time, $O(n \log n)$, and provides an approximate solution to our ILP. Then, we use GSS as our SSA in the SS-UCB framework and propose GSS-UCB as an alternate algorithm to DPSS-UCB in the setting where the qualities of the agents are unknown.

5.1 Greedy Subset Selection (GSS)

Greedy algorithms have been proven effective to provide approximate solutions to ILP problems such as 0-1 knapsack. They do so by solving linearly relaxed variants of an ILP, such as fractional knapsack, and removing any fractional unit from its solution. We propose a similar algorithm for our subset selection problem by allowing $x_i \in [0; 1]$. However, we cannot simply remove fractional units from our solution, as it may lead to QC violation. Consider the following example:

Given $n = 2$ agents with qualities, $\boldsymbol{q} = [0{:}6; 0{:}9]$, $\boldsymbol{c} = [10; 100]$ and $\alpha = 0{:}7$. Allowing fractional units to be taken, the optimal solution would be to take $x_1 = 1; x_2 = 0{:}5$ units of the two agents. Removing fractional units would lead to selecting only the first agent, which violates the QC. Towards this, we include an additional step (Line 22, Algorithm 3) in our algorithm that ensures that QC is not violated. Formally, the algorithm proceeds as follows:

(1) Divide the agents into the four categories, namely, S_1,S_2,S_3,S_4, as described in Section 3.4.
(2) Select all agents in S_1. Let $d = \sum_{i \in S_1} (q_i - \alpha)$ be the excess quality accumulated and as before, drop all agents in S_4.
(3) For agents in S_2, sort them in the decreasing order of revenue gained per unit loss in quality ($\frac{r_i}{\alpha - q_i}$). Similarly, for agents in S_3, sort them in the increasing order of revenue lost per unit gain in quality ($\frac{r_i}{\alpha - q_i}$).
(4) Select units (could be fractional) from agents from S_2 until the total loss of quality is no more than d. Essentially, we use the agents in S_2 to increase revenue while ensuring average quality is above the threshold.
(5) For agents in S_2 with remaining fractional units, we pair them up with an equivalent fractional unit of an agent in S_3 that balances the loss in average quality.
(6) When the revenue gained per unit loss in quality from the first non-exhausted agent in S_2 is less than the revenue lost per unit gain of quality from the first non-exhausted agent in S_3, terminate the algorithm. An agent is exhausted if the unit produce is completely selected.
(7) For any agent in S_3 with a fractional unit, take the complete unit instead. For all other agents, remove any fractional units selected.

Algorithm 3 GSS

Inputs: N, α, R, costs $c = [c_i]$, qualities $q = [q_i]$
Output: Quantities procured $x = (x_1; : : : ; x_n)$
Initialization: $\forall i \in N, r_i = Rq_i - c_i$
Segregate S_1,S_2,S_3,S_4 as described in Section 3.4
$\forall i \in S_1, x_i = 1; d = \lceil \sum_{i \in S_1} (q_i - \alpha) \rceil$
$\forall i \in S_4, x_i = 0$
$L_2 = sort(S_2)$ on decreasing order of $\frac{r_i}{\alpha - q_i}$
$L_3 = sort(S_3)$ on increasing order of $\frac{r_i}{\alpha - q_i}$
$p = 0; q = 0$
while $d > 0$ and $p < |S_2|$ **do**
    $i = L_2[p]$;
    if $\alpha - q_i \leq d$ **then** $x_i = 1, d = d - \alpha - q_i, p+ = 1$
    else $x_i = \frac{d}{\alpha - q_i}, d = 0$
while $p < |S_2|$ and $q < |S_3|$ **do**
    $i = L_2[p], j = L_3[q]$
    $a = \frac{r_i}{\alpha - q_i}, b = \frac{r_j}{\alpha - q_j}$
    if $a \leq b$ **then break**;
    $w_1 = \min((1 - x_i)(\alpha - q_i); (1 - x_j)(q_j - \alpha))$
    $x_i + = \frac{w_1}{\alpha - q_i}, x_j + = \frac{w_1}{q_j - \alpha}$
    if $x_i == 0$ **then** $p + +$;
    if $x_j == 0$ **then** $q + +$;
if $0 < x_j < 1$ **then** $x_j = 1$
return $\lfloor x \rfloor$

5.2 Approximation Ratio

While GSS is computationally more efficient than DPSS, it is important to note that it may not always return the optimal subset of agents. We show for the following example, that GSS doesn't have a constant approximation w.r.t. the optimal solution:

Consider $n = 3$ agents with qualities, $q = [1{:}00; 0{:}98; 0{:}97]$ and $c = [R-\epsilon; \frac{78R}{100}; \frac{47R}{100}]$. Hence, $r = [\epsilon; \frac{R}{5}; \frac{R}{2}]$, where R is some constant as discussed before such that $r_i = Rq_i - c_i$. If $\alpha = 0{:}99$, the value of $\frac{r_i}{\alpha - q_i}$ for the third agent is higher than that of the second, but only a fractional unit can pair with the first agent. Hence, according to GSS, we only select the first agent giving us a utility of ϵ, whereas the optimal utility is equal to $\epsilon + \frac{R}{5}$ corresponding to choosing the first and the third agent. Thus, the approximation ratio is $\frac{\epsilon}{\epsilon + \frac{R}{5}}$. Since ϵ can take an arbitrary small value, the approximation ratio between the utility achieved by GSS and DPSS can be arbitrary small.

However, through experiments, we show that in practice, GSS gives close to optimal solutions at a huge computational benefit that allows us to scale our framework for a large number of agents, such as in an E-commerce setting.

5.3 GSS-UCB

When we use GSS as the SSA in our SS-UCB framework, we refer to the algorithm as GSS-UCB. While the regret analysis may not necessarily hold, as GSS does not have a constant approximation, we still show that in practice, it works as good as DPSS-UCB in both (i) achieving constraint satisfaction after τ rounds and (ii) the regret incurred thereafter. We show this via experiments, as discussed in Section 6.

6 EXPERIMENTAL ANALYSIS

6.1 Subset Selection With Known Qualities

In this section, we compare the performance of GSS with DPSS in the setting where quality of the agents is known. In Figure 1a, we compare the ratio of the utility achieved by GSS ($z_{\partial ss}$) to the utility achieved by DPSS (z_{dpss}) while ensuring the QC is met. In Figure 2, we present a box plot of the distribution of the ratios of these utilities over 1000 iterations for $\alpha = 0{:}7$. To compare the performance of GSS for much larger values of n, we compare it against the utility achieved by an ILP solver (z_{ilp}), namely, the COIN-OR Branch and Cut Solver (CBC) [15] since the computational limitations of DPSS made it infeasible to run experiments for large values of n. The results for the same are presented in Figure 1b. Lastly, in Table 1, we compare the ratio of the time taken by GSS ($t_{\partial ss}$) with respect to DPSS (t_{dpss}) and the ILP solver (t_{ilp}) for different values of n with α being set to 0:7.

6.1.1 Setup. For different values of n, the number of agents and α, the quality threshold, we generate agents with q_i and c_i both $\sim U[0; 1]$. For Figure 1a and 1b, we average our results over 1000 iterations for each (n, α) pair, while in Figure 2, we plot the distribution of the ratios obtained in each of the 1000 iterations for different values of n with α set to 0:7. We use $R = 1$ for all our experiments.

6.1.2 Results and Discussion. As can be seen from Figures 1a and 1b, the average ratio of both ($\frac{z_{\partial ss}}{z_{dpss}}$) and ($\frac{z_{\partial ss}}{z_{ilp}}$) lies approximately between [0.94,1.0], with a median of 1:0 for almost all values of n and only a few outliers and a few rare instances when the ratio drops below 0.2 as evident from Figure 2. This indicates that GSS performs almost as good as DPSS in practice with an exponentially improving computational performance in terms of time complexity with respect to DPSS and an almost 50x improvement over the ILP solver as well. This establishes the efficacy of GSS for practical use at scale.

n	$t_{dpss} : t_{\partial ss}$	$t_{ilp} : t_{\partial ss}$
2	5.5	70
5	15.7	64
8	32.6	63.7
10	54.3	58.6
12	106.3	67.6
14	284.4	65.3
16	897.1	60.2
18	3109.7	63.1
20	11360.6	68.1

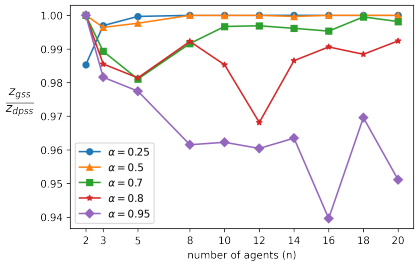
n	$t_{ilp} : t_{\partial ss}$
25	66.7
50	58.3
100	52.7
400	43.1
1000	31.8
5000	31.6
10000	34.5
50000	45
100000	56.8

Table 1: Computational Performance of GSS w.r.t. to DPSS and ILP

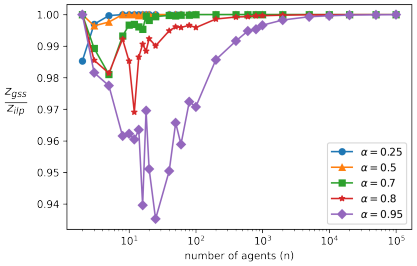
6.2 Subset Selection With Unknown Qualities

In this section, we present experimental results of DPSS-UCB and GSS-UCB towards the following:

(1) Constraint Satisfaction: As discussed in section 4.3, DPSS-UCB satisfies the QC approximately with high probability



(a) w.r.t DPSS



(b) w.r.t. ILP

Figure 1: Performance of GSS on different values of α

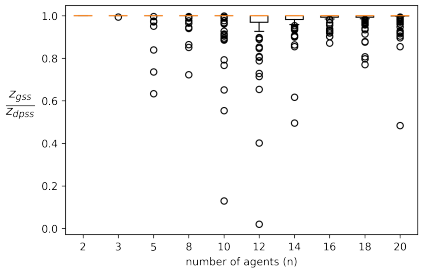


Figure 2: GSS vs DPSS ratio distribution

after $\tau = \frac{3 \ln T}{2\epsilon_2^2}$ rounds. Here, $\alpha + \epsilon_2$ is the target constraint of the agent when α is the required average quality threshold. Towards this, we plot the average number of iterations where DPSS-UCB and GSS-UCB returns a subset that satisfies QC at each round in our experiment for different values of ϵ_2.

(2) Regret incurred for $t > \tau$: We show that the regret incurred by our algorithm for $t > \tau$, follows a curve upper bounded by $O(\ln T)$. Towards this we plot the cumulative regret vs. round t, where $\tau < t \leq T$.

6.2.1 Setup. To carry out these experiments, we generated $n = 10$ agents with both $q_i, c_i \sim U[0; 1]$. We chose $\alpha = 0{:}7$ as for a higher value of α the number of super-arms satisfying QC is very low and hardly much to learn whereas for a low value, the number of super-arms that satisfy QC is very high but practically of not much interest. In Figure 4, we perform the experiment over a varied range of values of ϵ_2, whereas in Figure 3, we set $\epsilon_2 = 0{:}01$. We average our results for 1000 iterations of each experiment. For example, in Figure 4, a value of 0:4 at some round t, would denote that in 40% of the iterations, the QC was satisfied at round t. For both the experiments, $R = 1$ and $T = 100000$.

6.2.2 Discussion. Higher the value of ϵ_2, higher is the target constraint and thus more conservative is our algorithm in selecting the subset of agents. Therefore, we achieve correctness quickly, which is evident from Figure 4. In all three cases, the algorithm achieves correctness in close to 100% of the iterations, after $\frac{3 \ln T}{2\epsilon_2^2}$ rounds (indicated by the vertical dotted line), which justifies our value of τ. Similarly, the regret incurred by DPSS-UCB for $t > \tau$ follows a curve upper bounded by $O(\ln T)$. The regret incurred by GPSS-UCB is slightly lower than DPSS-UCB which further establishes the efficacy of our greedy approach.

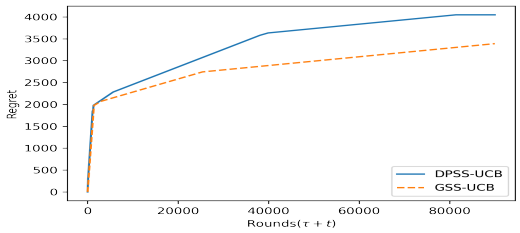


Figure 3: Regret incurred for $t > \tau$

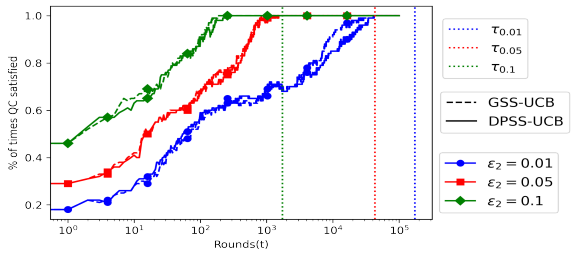


Figure 4: Constraint Satisfaction at each round

7 CONCLUSION AND FUTURE WORK

In this paper, we addressed the class of problems where a central planner had to select a subset of agents that maximized its utility while ensuring a quality constraint. We first considered the setting where the agents' quality is known and proposed DPSS that provided an exact solution to our problem. When the qualities were unknown, we modeled our problem as a CMAB problem with semi-bandit feedback. We proposed SS-UCB as a framework to address this problem where both the constraint and the objective function depend on the unknown parameter, a setting not considered previously in the literature. Using DPSS as our SSA in SS-UCB, we proposed DPSS-UCB that incurred a $O(\ln T)$ regret and achieved constraint satisfaction with high probability after $\tau = O(\ln T)$ rounds. To address the computational limitations of DPSS, we proposed GSS for our problem that allowed us to scale our framework to a large number of agents. Via simulations, we showed the efficacy of GSS.

The SS-UCB framework proposed in this paper can be used to design and compare other approaches to this class of problems that find its applications in many fields. It can also easily be extended to solve for other interesting variants of the problem such as (i) where the pool of agents to choose from is dynamic with new agents entering the setting, (ii) where an agent selected in a particular round is not available for the next few rounds (sleeping bandits) possibly due to lead time in procuring the units, a setting which is very common in operations research literature. Our work can also be extended to include strategic agents where the planner needs to design a mechanism to elicit the agents' cost of production truthfully.

REFERENCES

[1] Kontogeorgos Achilleas and Semos Anastasios. 2008. Marketing aspects of quality assurance systems: The organic food sector case. *British Food Journal* 110, 8 (2008), 829–839.

[2] Shipra Agrawal and Nikhil R Devanur. 2014. Bandits with concave rewards and convex knapsacks. In *Proceedings of the fifteenth ACM conference on Economics and computation*. 989–1006.

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

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

[5] Ashwinkumar Badanidiyuru, John Langford, and Aleksandrs Slivkins. 2014. Resourceful contextual bandits. In *Conference on Learning Theory*. 1109–1134.

[6] Arpita Biswas, Shweta Jain, Debmalya Mandal, and Y. Narahari. 2015. A Truthful Budget Feasible Multi-Armed Bandit Mechanism for Crowdsourcing Time Critical Tasks. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (Istanbul, Turkey) *(AAMAS '15)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1101–1109. http://dl.acm.org/citation.cfm?id=2772879.2773291

[7] Sébastien Bubeck and Nicolo Cesa-Bianchi. 2012. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Machine Learning* 5, 1 (2012), 1–122.

[8] Shouyuan Chen, Tian Lin, Irwin King, Michael R Lyu, and Wei Chen. 2014. Combinatorial Pure Exploration of Multi-Armed Bandits. In *Advances in Neural Information Processing Systems 27*. 379–387.

[9] Wei Chen, Wei Hu, Fu Li, Jian Li, Yu Liu, and Pinyan Lu. 2016. Combinatorial multi-armed bandit with general reward functions. In *Advances in Neural Information Processing Systems*. 1659–1667.

[10] Wei Chen, Yajun Wang, and Yang Yuan. 2013. Combinatorial Multi-Armed Bandit: General Framework and Applications. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 28)*, Sanjoy Dasgupta and David McAllester (Eds.). PMLR, Atlanta, Georgia, USA, 151–159. http://proceedings.mlr.press/v28/chen13a.html

[11] Wei Chen, Yajun Wang, Yang Yuan, and Qinshi Wang. 2016. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *The Journal of Machine Learning Research* 17, 1 (2016), 1746–1778.

[12] Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, et al. 2015. Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*. 2116–2124.

[13] Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, and marc lelarge. 2015. Combinatorial Bandits Revisited. In *Advances in Neural Information Processing Systems 28*. 2116–2124.

[14] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. 2002. PAC bounds for multi-armed bandit and Markov decision processes. In *International Conference on Computational Learning Theory*. Springer, 255–270.

[15] John J. Forrest, Stefan Vigerske, Haroldo Gambini Santos, Ted Ralphs, Lou Hafer, Bjarni Kristjansson, jpfasano, EdwinStraver, Miles Lubin, rlougee, jpgoncal1, h-i gassmann, and Matthew Saltzman. 2020. *coin-or/Cbc: Version 2.10.5*. https://doi.org/10.5281/zenodo.3700700

[16] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. 2010. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *IEEE Symposium on New Frontiers in Dynamic Spectrum*. 1–9.

[17] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. 2012. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking* (2012), 1466–1478.

[18] David Haussler. 1990. *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory.

[19] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. 2013. Adaptive task assignment for crowdsourced classification. In *International Conference on Machine Learning*. 534–542.

[20] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Amer. Statist. Assoc.* 58, 301 (1963), 13–30.

[21] Shweta Jain, Satyanath Bhat, Ganesh Ghalme, Divya Padmanabhan, and Y. Narahari. 2016. Mechanisms with learning for stochastic multi-armed bandit problems. *Indian Journal of Pure and Applied Mathematics* 47, 2 (01 Jun 2016), 229–272. https://doi.org/10.1007/s13226-016-0186-3

[22] Shweta Jain, Sujit Gujar, Satyanath Bhat, Onno Zoeter, and Y Narahari. 2018. A quality assuring, cost optimal multi-armed bandit mechanism for expertsourcing. *Artificial Intelligence* 254 (2018), 44–63.

[23] David R Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*. 1953–1961.

[24] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. 2015. Tight regret bounds for stochastic combinatorial semi-bandits. In *Artificial Intelligence and Statistics*. 535–543.

[25] Jason D Papastavrou, Srikanth Rajagopalan, and Anton J Kleywegt. 1996. The dynamic and stochastic knapsack problem with deadlines. *Management Science* 42, 12 (1996), 1706–1718.

[26] Robert P Rooderkerk and Harald J van Heerde. 2016. Robust optimization of the 0–1 knapsack problem: Balancing risk and return in assortment optimization. *European Journal of Operational Research* 250, 3 (2016), 842–854.

[27] Prabhakant Sinha and Andris A Zoltners. 1979. The multiple-choice knapsack problem. *Operations Research* 27, 3 (1979), 503–515.

[28] Aleksandrs Slivkins. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* (2019).

[29] Milé Terziovski, Danny Samson, and Douglas Dow. 1997. The business value of quality management systems certification. Evidence from Australia and New Zealand. *Journal of operations management* 15, 1 (1997), 1–18.

[30] William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25, 3/4 (1933), 285–294.

[31] Long Tran-Thanh, Sebastian Stein, Alex Rogers, and Nicholas R Jennings. 2014. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence* 214 (2014), 89–111.

[32] Long Tran-Thanh, Matteo Venanzi, Alex Rogers, and Nicholas R Jennings. 2013. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 901–908.

[33] GJ Zaimai. 1989. Optimality conditions and duality for constrained measurable subset selection problems with minmax objective functions. *Optimization* 20, 4 (1989), 377–395.