Stochastic Network Utility Maximization with Unknown Utilities: Multi-Armed Bandits Approach

Arun Verma and Manjesh K. Hanawal
Industrial Engineering and Operations Research
Indian Institute of Technology Bombay, India
{v.arun, mhanawal}@iitb.ac.in

***Abstract*—In this paper, we study a novel Stochastic *Network Utility Maximization* (NUM) problem where the utilities of agents are unknown. The utility of each agent depends on the amount of resource it receives from a network operator/controller. The operator desires to do a resource allocation that maximizes the expected total utility of the network. We consider threshold type utility functions where each agent gets non-zero utility if the amount of resource it receives is higher than a certain threshold. Otherwise, its utility is zero (hard real-time). We pose this NUM setup with unknown utilities as a regret minimization problem. Our goal is to identify a policy that performs as 'good' as an oracle policy that knows the utilities of agents. We model this problem setting as a bandit setting where feedback obtained in each round depends on the resource allocated to the agents. We propose algorithms for this novel setting using ideas from Multiple-Play Multi-Armed Bandits and Combinatorial Semi-Bandits. We show that the proposed algorithm is optimal when all agents have the same utility. We validate the performance guarantees of our proposed algorithms through numerical experiments.**

***Index Terms*—Network Utility Maximization, Multi-Armed Bandits, Combinatorial Semi-Bandits, Resource Allocation**

I. Introduction

Network Utility Maximization (NUM) is an approach for resource allocation among multiple agents such that the total utility of all the agents (network utility) is maximized. In its simplest form, NUM solves the following problem:

$$\underset{\boldsymbol{x}}{\text{maximize}} \sum_{i=1}^{K} U_i(x_i)$$
$$\text{subject to} \quad \sum_{i=1}^{K} x_i \leq C$$

where $U_i(\cdot)$ denotes the utility of agent i, variable $\boldsymbol{x} = (x_1, x_2, \ldots, x_K) \in \mathbb{R}_+^K$ denote the allocated resource vector, and $C \in \mathbb{R}_+$ is amount of resource available. Utilities define the satisfaction level of the agents, which depend on the amount of resource they are allocated. A resource could be bandwidth, power, or rates they receive. Since the seminal work of Kelly [1], there has been a tremendous amount of work on NUM and its extensions. NUM is used to model various resource allocation problems and improve network protocols based on its analysis. The nature of utility functions is vital in the analysis of the NUM problem and assumed to be known or can be constructed based on the agent behavior model and operator cost model. However, agent behavior models are often difficult to quantify. In this work, we study the NUM problem where the utilities of the agent are unknown and stochastic.

The earlier NUM problems considered deterministic settings. Significant progress has been made to extend the NUM setup to take into account the stochastic nature of the network and agent behavior [2]. For both the static and stochastic networks, the works in the literature often assume that the utility functions are smooth concave functions and apply Karush-Kuhn-Tucker conditions to find the optimal allocation. However, if the utility functions are unknown, these methods are useful only once the utilities are learned. Many of the NUM variants with full knowledge of utilities aim to find an optimal policy that meets several constraints like stability, fairness, and resource [3], [4], [5]. In this work, we only focus on resource constraint due to limited divisible resource (bandwidth, power, rate).

Since learning an arbitrary utility function is not always feasible, we assume the utilities belong to a class of 'threshold' type functions. Specifically, we assume that the utility of each agent is stochastic with some positive mean only when it is allocated a certain minimum resource. We refer to the minimum resource required by an agent as its 'threshold' and the mean utility it receives when it is allocated resource above the threshold as its 'mean reward.' Thus the expected utility of each agent is defined by two parameters – a threshold and a mean reward. Such threshold type utilities correspond to hard resource requirements. For example, an agent can transmit and obtain a positive rate (reward) only if its power or bandwidth allocation is above a certain amount.

In each round, the operator allocates a resource to each agent and observes the utilities the agent obtains. The goal of the operator is to allocate resource such that the expected network utility is maximized. We pose this problem as a Multi-Armed Bandit (MAB) problem where the operator corresponds to a learner, agents to arms, and utilities to rewards. The learner's goal is to learn a policy that minimizes the difference between the best achievable expected network utility with full knowledge of the agent utilities and that obtained by the learner under the same resource constraint with the estimated utilities of agents.

The reward structure in the MAB formulation of the NUM problem is different from the standard MAB problem. Hence one cannot directly apply the standard MAB algorithms to the NUM setting. Unlike standard MAB setup where the reward depends on the arm played, in the NUM setup, the reward obtained in each round depends on the resource allocated by

the learner. The learner observes the utility of an agent only when it allocates resource above its threshold. Otherwise, it gets no reward on the utilities of the agents. Further, in the NUM setup, the learner may observe utilities of more than one agent in each round depending on how many agents receive resource above their corresponding thresholds.

A good policy for the NUM setting needs to learn the expected utility for each agent, i.e., the thresholds, and mean rewards. We first consider the case where the threshold for each agent is the same and then consider the case where the thresholds could be different. For both cases, we develop a policy based on Thompson Sampling that achieves sub-linear regret. Our contributions can be summarized as follows:

- In Section II, we give a novel model for Online Network Utility Maximization (ONUM) with unknown utilities.
- In Section III, we study the symmetric case where the threshold is the same for all the agents. Using the concept of 'allocation equivalent,' we develop an optimal algorithm named ONUM-ST by exploiting connection with Multiple-Play Multi-Armed Bandits to our setting.
- In Section IV, we study a more general asymmetric case where the threshold for agents could be different. We develop an efficient algorithm named ONUM-DT by exploiting its connection to Combinatorial Semi-Bandits.
- We empirically validate the performance of our algorithms via experiments on synthetic problems in Section V.

A. Related Work

NUM has been an active area of research in the past two decades. Many of its variants are developed for resource allocation in networking. We refer the readers to [6],[7] for an informative tutorial and survey on this subject. In this Section, we discuss works that look into learning aspects in NUM.

NUM in a multi-agent network is studied with partially observable channel states in [8]. The authors assume that the channel states are Markovian and exploit the memory in the channel to maximize a known concave function of time average reward using the framework of Restless Bandits. Stochastic Multi-Armed Bandits (MAB) [9], [10] are applied in distributed optimization in networks. In cognitive radio networks (CRNs) with multiple agents, the MAB setup is used to maximize network throughput in a distributed setting [11], [12], [13], [14], [15]. The fairness issues while maximizing the network utility using the MAB setting is studied in [16].

Our MAB formulation of NUM involves solving a combinatorial 0-1 knapsack problem. Bandits with Knapsacks studied in [17] also require solving a knapsack problem in each round. However, in their model, resource gets consumed in every round, unlike ours. Also, in Bandits with Knapsacks, the resource allocation does not affect the reward observed. [18], [19] also assume some threshold model for rewards. However, in their model, an agent receives a reward only if the sampled reward from its associated distribution is above some threshold. Whereas in our setup, the threshold corresponds to the minimum resource required. Resource allocation with semi-bandits feedback [20], [21], [22] study a related but less general setting where the reward is observed in each round irrespective of the amount of resource allocated. Whereas in our setting, it is not the case as the reward is zero if the minimum requirement of the resource is not satisfied. The adaptive resource allocation problem is also studied in loss setting with censored feedback by [23], where no loss values are observed from arms that receive more resource than their associated thresholds. In this work, we consider a reward setting, and our algorithms differ from that in [23] as [23] first estimate the threshold value associated with the arms and then estimate the mean losses of arms. Whereas our goal is to maximize total reward, and our algorithms jointly estimate both threshold and mean reward of the arm.

Depending on the resource allocated in each round, we observe the reward from a subset of agents who get their minimum required resource. Such combinatorial aspects of arms play are widely studied as Combinatorial (Semi-)Bandits in [24], [25], [26], [27]. Though these works are not directly related to our setup, we make explicit connections of our algorithms to the algorithms given in [25] and [27].

II. Problem Setting

We consider an online version of the NUM problem where utilities of the agents are unknown, and the network operator aims to reach the optimal resource allocation via sequential allocations. Let K denote the number of agents, and C denotes the amount of divisible resource (bandwidth, power). The operator assigns a fraction of resource to each agent, and the utility of agents depends on the amount of resource they receive. Utility for agent $i \in [K]$ where $[K] := \{1, 2, \ldots, K\}$, is stochastic and drawn from a fixed distribution ν_i with support in $[0, 1]$ and mean $\mu_i \in [0, 1]$ in each round, provided it receives a certain minimum amount of resource, otherwise its utility is zero. For each $i \in [K]$, let $\theta_i \in [0, C]$ denote the minimum resource required for agent i to obtain non-zero utility. Then, for each agent $i \in [K]$ utility is parameterized as (θ_i, μ_i) such that agent i receives mean utility μ_i if it is allocated at least θ_i fraction of resource, otherwise its utility is zero.

The resource allocated to the agents decides the reward observed by the operator. If the allocated resource is at least θ_i for agent $i \in [K]$, the operator observes the realization of the utility obtained by the agent drawn from the distribution ν_i. Otherwise, zero utility is obtained by the agent.

In the following, we assume that each $\nu_i, i \in [K]$ is a Bernoulli distribution with parameter μ_i. It is a challenging setting as the operator can't know whether sufficient resource is allocated to an agent whenever the agent receives zero utility. Because with Bernoulli utility the agent $i \in [K]$ can receive zero utility even if it is allocated minimum required resource with probability $(1 - \mu_i)$. Whereas this probability is very small (almost zero) if the utility distribution is continuous.

Following the terminology of Multi-Armed Bandits (MAB), henceforth we refer to agents as arms, operator as learner and utility as a reward. Let $\boldsymbol{x} := \{x_i : i \in [K]\}$, where $x_i \in [0, C]$, denotes the resource allocated to arm i. An allocation vector $\boldsymbol{x}$ is said to be feasible if $\sum_{i=1}^{K} x_i \leq C$. The set of all feasible

allocations is denoted as $\mathcal{A}_C$. For any $\boldsymbol{x} \in \mathcal{A}_C$, mean reward from arm i is non-zero only if $x_i \geq \theta_i$. The goal of the learner is to find a feasible resource allocation that maximizes the network utility.

The available resource may be allocated to multiple arms in our setup. However, the reward from each arm may not be observed depending on the amount of resource allocated to them. Hence we have semi-bandit feedback in each round, and we refer to this setup as Online Network Utility Maximization (ONUM). The vectors $\boldsymbol{\mu} := \{\mu_j\}_{i\in[K]}$ and $\boldsymbol{\theta} := \{\theta_i\}_{i\in[K]}$ are unknown and identify an instance of ONUM problem. Henceforth we identify an ONUM instance as $P = (\boldsymbol{\mu}, \boldsymbol{\theta}, C) \in [0,1]^K \times \mathbb{R}_+^K \times \mathbb{R}_+$ and denote collection of ONUM instances as $\mathcal{P}_{\text{ONUM}}$. For simplicity of discussion, we assume that arms are indexed according to their decreasing mean rewards, i.e., $\mu_1 \geq \mu_2, \ldots, \geq \mu_K$, but the algorithms are not aware of this ordering. We refer to the first M arms as *top-M* arms. For instance $P \in \mathcal{P}_{\text{ONUM}}$, the optimal allocation can be computed as the following 0-1 knapsack problem:

$$\boldsymbol{x}^\star \in \arg\max_{\boldsymbol{x}\in\mathcal{A}_C} \sum_{i=1}^{K} \mu_i \mathbb{1}_{\{x_i \geq \theta_i\}}.$$

The interaction between a learner and the environment that governs rewards for the arms is as follows: In the round t, the environment generates a reward vector $(Y_{t,1}, Y_{t,2}, \ldots, Y_{t,K}) \in \{0,1\}^K$, where $Y_{t,i}$ denotes the true reward for arm i in round t. The sequence $(Y_{t,i})_{t\geq 1}$ is generated i.i.d. with the common mean $\mathbb{E}[Y_{t,i}] = \mu_i$ for each $i \in [K]$. The learner selects a feasible allocation $\boldsymbol{x}_t = \{x_{t,i} : i \in [K]\}$ and observes reward vector $Y'_t = \{Y_{t,i} : i \in [K]\}$, where $Y'_{t,i} = Y_{t,i}\mathbb{1}_{\{x_{t,i}\geq\theta_i\}}$ and collects reward $r_t(\boldsymbol{x}_t) = \sum_{i\in[K]} Y'_{t,i}$. A policy of the learner is to select a feasible allocation in each round based on the observed reward such that the cumulative reward is maximized. The performance of a policy that makes allocation $\{\boldsymbol{x}_t\}_{t\geq 1}$ in round t is measured in terms of expected (pseudo) cumulative regret for T rounds given by

$$\mathbb{E}[\mathcal{R}_T] = T\sum_{i=1}^{K} \mu_i \mathbb{1}_{\left\{x_i^\star \geq \theta_i\right\}} - \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{K} Y_{t,i}\mathbb{1}_{\{x_{t,i}\geq\theta_i\}}\right].$$

A good policy must have sub-linear cumulative regret, i.e., $\mathbb{E}[\mathcal{R}_T]/T \to 0$ as $T \to \infty$.

A. Allocation Equivalent

Next, we define the notion of treating a pair of thresholds for the given loss vector and resource to be 'equivalent.'

Definition 1 (Allocation Equivalent)**.** *For any reward vector $\boldsymbol{\mu}$ and fix amount of resource Q, two threshold vectors $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ are* allocation equivalent *iff the following holds:*

$$\max_{\boldsymbol{x}\in\mathcal{A}_C} \sum_{i=1}^{K} \mu_i \mathbb{1}_{\{x_i\geq\theta_i\}} = \max_{\boldsymbol{x}\in\mathcal{A}_C} \sum_{i=1}^{K} \mu_i \mathbb{1}_{\left\{x_i\geq\hat{\theta}_i\right\}}.$$

Such equivalence allows us to find the threshold vector within fix error tolerance, which has the same total mean reward reduction as a true threshold vector has.

III. Same Threshold for All Arms

We first focus on the special case of the online network utility maximization problem where $\theta_i = \theta_s$ for all $i \in [K]$. With abuse of notation, we denote an instance of ONUM with the same threshold as $(\boldsymbol{\mu}, \theta_s, C)$ where $\theta_s \in [0, C]$ is the value of the same threshold. Note that even though the threshold is the same, the mean rewards can be different across the arms. Though θ_s can be any value in the interval $[0, C]$, an allocation equivalent to it can be restricted to a finite set. Our next result shows that the search for an allocation equivalent can be confined to a set consisting of K elements.

Lemma 1. *Let $\theta_s \in [0, C]$, $M = \min\{\lfloor C/\theta_s \rfloor, K\}$ and $\hat{\theta}_s = C/M$. Then θ_s and $\hat{\theta}_s$ are allocation equivalent. Further, $\hat{\theta}_s \in \Theta$ where $\Theta = \{C/K, C/(K-1), \cdots, C\}$.*

Proof. The proof is a straight forward adaption of Lemma 1 in [23] by allowing the threshold to be any value in $[0, C]$ where C can be greater than 1. The case when $\lfloor C/\theta_s \rfloor \geq K$ is trivial. Let consider the case when $\lfloor C/\theta_s \rfloor < K$. Using the definition of M, we have $M \leq C/\theta_s$ and $\theta_s \leq C/M \doteq \hat{\theta}_s$. Hence $\hat{\theta}_s \geq \theta_s$. Therefore allocation of $\hat{\theta}_s$ or θ_s fraction of resource allocation to an arm achieves the same mean reward. Further, for both instances $(\boldsymbol{\mu}, \theta_s, C)$ and $(\boldsymbol{\mu}, \hat{\theta}_s, C)$, the optimal allocations collect reward from the top-M arms and no reward from the remaining arms. Hence the mean reward collected from the optimal allocations in both the instances results in the same total mean reward. It completes the proof of first part of lemma. Since all arms have same threshold, learner has to equally distribute resource among selected top-M arms. As $M \in \{1, \ldots, K\}$ and $\hat{\theta}_s \leq C$, the desired value of $\hat{\theta}_s$ is the one of element in set $\Theta = \{C/K, C/(K-1), \cdots, C\}$. □

Once the threshold is known, the optimal allocation of a learner is to allocate $\hat{\theta}_s$ amount of resource to each of the top-M arms where $M = C/\hat{\theta}_s$. Lemma 1 shows that an allocation equivalent $\hat{\theta}_s$ for any instance $(\boldsymbol{\mu}, \theta_s, C)$ is one of value in a finite set Θ. Once allocation equivalent is known, the problem reduces to identifying the top-M arms and then allocating $\hat{\theta}_s$ amount of resource to each one of them to maximize the total mean reward. The latter part is equivalent to solving Multiple-Play Multi-Armed Bandits, as discussed next.

A. Multiple-Play Multi-Armed Bandits (MP-MAB)

In stochastic Multiple-Play Multi-Armed Bandits, a learner can play a subset of arms in each round. The selected subset of arms is of fixed size (know) and also known as superarm [28]. The mean reward of a superarm is the sum of the mean reward of its constituent arms. In every round, a learner selects a superarm and observes the reward from each selected arm (semi-bandit feedback). The goal of the learner is to select a superarm that has the maximum mean reward. In MP-MAB, a policy selects a superarm in each round based on the previous reward information. The performance of any policy is measured in terms of regret. The regret is the difference between cumulative reward collected by playing optimal superarm and that collected by the policy in each round.

Lower bound: Due to the equivalence between the MP-MAB and ONUM problem with the (known) same threshold, the lower bound for MP-MAB is also a lower bound for the ONUM problem with the same threshold. Therefore, the following lower bound given for a strongly consistent algorithm by Theorem 3.1 in [28] is also a lower bound on the ONUM problem with known same threshold:

$$\lim_{T\to\infty} \frac{\mathbb{E}[\mathcal{R}_T]}{\log T} \geq \sum_{i\in[K]\setminus[M]} \frac{\mu_M - \mu_i}{d(\mu_i, \mu_M)} \tag{1}$$

where $d(p, q)$ is the Kullback-Leibler (KL) divergence between two Bernoulli distributions with parameter p and q.

Once the threshold is known, any algorithm that works well for the MP-MAB also works well for the ONUM. Hence one can apply algorithms like ESCB [24] and MP-TS [25] once an allocation equivalent is found for θ_s. MP-TS uses Thompson Sampling, whereas ESCB uses UCB and kl-UCB type indices. We can adapt any of these algorithms for our setting. But we use MP-TS to our as it gives better empirical performance compare to ESCB and has been shown to obtain optimal regret bound for Bernoulli reward distributions.

B. Algorithm ONUM-ST

We develop an algorithm named Online Network Utility Maximization with the Same Threshold (ONUM-ST). It exploits the result of Lemma 1 to learn an allocation equivalent of threshold and adapts MP-TS to minimize the regret. The pseudo-code of ONUM-ST is given in Algorithm 1. ONUM-ST works as follows: it takes K, C, δ and ϵ as input where δ is the confidence on the correctness of estimated allocation equivalent and ϵ is such that $\mu_K \geq \epsilon > 0$. We set $\Theta = \{C/K, C/(K-1), \ldots, C\}$ as in Lemma 1. The elements of Θ are in increasing order, and each element is a candidate for allocation equivalent of θ_s (line 2). We also set the prior distribution for the mean reward of each arm as the Beta distribution $\beta(1, 1)$. For each arm $i \in [K], S_i$ represents the number of rounds when the reward is 1, and F_i represents the number of rounds when the reward is 0 whenever the arm i receives resource above its threshold.

ONUM-ST finds a threshold $\hat{\theta}_s$ that is a allocation equivalent to θ_s with high probability (at least $1-\delta$) using binary search over the set Θ. The search begins by taking $\hat{\theta}_s$ to be the middle element in Θ (line 5). Let $S_i(t)$ and $F_i(t)$ denote the values of S_i and F_i in the starting of the round t. In round t, a sample $\hat{\mu}_i$ is drawn from $\beta(S_i(t), F_i(t))$ for each arm $i \in [K]$ independent of other arms (line 5). $\hat{\mu}_i$ values are ranked as per their decreasing values and each of the top-$(C/\hat{\theta}_s)$ (denoted as set A_t in line 6) arms are allocated $\hat{\theta}_s$ amount of resource and their rewards are observed (line 7). After knowing allocation equivalent of threshold, only S_i and F_i are updated for each arm $i \in A_t$ (line 17).

Before knowing allocation equivalent (line 8), if a reward 1 is observed at any of the arms in the set A_t (line 9), it implies that the current value of $\hat{\theta}_s$ is possibly an overestimate of $\hat{\theta}_s$. So all candidates larger than $\hat{\theta}_s$ in set Θ are removed, and the search is repeated in the remaining half of the elements by starting with the middle element (line 10). The success and failure counts are also updated as $S_i = S_i + Y_{t,i}, F_i = F_i + 1 - Y_{t,i} + Z_i$ for each arm $i \in A_t$, and for all $k \in K \setminus A_t$ only failure count is updated as $F_k = F_k + Z_k$ (line 11). The variable $Z_i,\ \forall i \in [K]$ keeps track of how many times 0 is observed for arm i before 1 is observed on it when it is allocated resource. It is reset to zero once a reward 1 is observed for any arm in set A_t. Variable $Z_i, i \in [K]$ allow us to distinguish the zeros observed when the arm receives over and under allocation of resource.

Algorithm 1 ONUM-ST

1: **Input:** K, C, δ, ϵ
2: Initialize Θ as in Lemma 1, $W_c = 0, l = 0, u = K, j = \lceil u/2 \rceil, \forall i \in [K] : S_i = 1, F_i = 1, Z_i = 0$
3: $W_\delta = \log(\log_2(K)/\delta)/(\log(1/(1-\epsilon)))$
4: **for** $t = 1, 2, \ldots,$ **do**
5: Set $\hat{\theta}_s = \Theta[j]$ and $\forall i \in [K] : \hat{\mu}_i \leftarrow \beta(S_i, F_i)$
6: $A_t \leftarrow$ set of top-$(C/\hat{\theta}_s)$ arms from estimates $(\hat{\mu}_i)$
7: $\forall i \in A_t$: allocate $\hat{\theta}_s$ resource and observe $Y_{t,i}$
8: **if** $j \neq u$ **then**
9: **if** $Y_{t,a} = 1$ for any $a \in A_t$ **then**
10: Set $u = j, j = u - \lfloor (u-l)/2 \rfloor, W_c = 0$
11: $\forall i \in A_t$: set $S_i = S_i + Y_{t,i}, F_i = F_i + 1 - Y_{t,i} + Z_i$, $\forall k \in [K] \setminus A_t : F_k = F_k + Z_k, \forall i \in [K] : Z_i = 0$
12: **else**
13: Set $W_c = W_c + 1$, and $\forall i \in A_t, Z_i = Z_i + 1$
14: If $W_c = W_\delta$ then set $l = j, j = l + \lceil (u-l)/2 \rceil$, $W_c = 0, \forall i \in [K] : Z_i = 0$
15: **end if**
16: **else**
17: $\forall i \in A_t : S_i = S_i + Y_{t,i}, F_i = F_i + 1 - Y_{t,i}$
18: **end if**
19: **end for**

If reward 0 is observed for all arms in the set A_t, Z_i is incremented by 1 for each arm $i \in A_t$ and variable W_c is incremented by 1 (line 13). Variable W_c counts the number of rounds for which reward is not observed on all the arms that are allocated resource. If W_c equals W_δ, then with high probability $\hat{\theta}_i$ is possibly an underestimate of allocation equivalent. So all candidates smaller than the current value of $\hat{\theta}_s$ in set Θ are removed, and the search is repeated, starting with the middle element in the remaining half. W_c as well as $Z_i,\ \forall i \in [K]$ are reset to 0 (lines 14). Resetting Z_i values to zero once the number of zeros observed reaches W_δ ensures that they do not add to F_i values when the resource is over-allocated.

Since Θ has a finite size, the search for an allocation equivalent of $\hat{\theta}$ terminates in the finite number of rounds with high probability. Once this happens, the algorithm allocates a resource to $C/\hat{\theta}_s$ arms (from Lemma 1) in the subsequent rounds and observes their reward samples, i.e., a fixed number of arms are played (multiple-play) in each round. Also, the $(C/\hat{\theta}_s)$ arms selected corresponds to top arms with the highest estimated means (line 7), which are generated based on Thompson Sampling. Hence after finding allocation equivalent of θ_s, our algorithm is the same as MP-TS. We leverage this observation to adapt the regret bounds of MP-TS.

C. Analysis of ONUM-ST

When $\hat{\theta}_s$ is an overestimate, and no reward is observed for consecutive W_δ rounds, then $\hat{\theta}_s$ will be increased. Such increment leads to an incorrect estimate of $\hat{\theta}_s$. Hence, the value of W_δ is set such that the probability of having the wrong allocation equivalent is upper bounded by δ. Let T_{θ_s} denote number of rounds needed to find an allocation equivalent of θ_s in Θ. Our next result gives a high probability bound on it.

Lemma 2. *Let $(\boldsymbol{\mu}, \theta_s, C)$ be an instance such that $\mu_K \geq \epsilon > 0$. Then with probability at least $1-\delta$, the number of rounds needed by ONUM-ST to find the allocation equivalent of θ_s is upper bounded as*

$$T_{\theta_s} \leq \frac{\log(\log_2(K)/\delta)}{\log\left(1/(1-\epsilon)\right)} \log_2(K).$$

This result extends Lemma 2 in [23]. The proof follows by binary search arguments and noting that one can come out of an under-allocation with high probability by observing the arms for a sufficiently large number of rounds. The detailed proof is given in APPENDIX. Once the allocation equivalent of θ_s is found, the regret of ONUM-ST in the subsequent rounds, denoted by $\mathcal{R}_T^s$ is upper bounded as given in Theorem 1.

Theorem 1. *Let $(\boldsymbol{\mu}, \theta_s, C) \in \mathcal{P}_{ONUM}$ such that $\mu_M > \mu_{M+1}$. The expected regret of ONUM-ST in T rounds after identifying allocation equivalent of θ_s is upper bounded as*

$$\mathbb{E}\left[\mathcal{R}_T^s\right] \leq O\left((\log T)^{2/3}\right) + \sum_{i\in[K]\setminus[M]} \frac{(\mu_M - \mu_i)\log T}{d(\mu_i, \mu_M)}.$$

As $\hat{\theta}_s$ is allocation equivalent to θ_s, the instances $(\boldsymbol{\mu}, \theta_s, C)$ and $(\boldsymbol{\mu}, \hat{\theta}_s, C)$ is having the same mean reward. After knowing $\hat{\theta}_s$, the expected regret of ONUM-ST is the same as solving a MP-MAB instance. Therefore, we can directly use Theorem 1 of [25] to get the above regret bounds by setting $L = M$.

The assumption $\mu_M > \mu_{M+1}$ ensures that KL divergence in the bound is well defined. It is also equivalent to assume that the set of top-M arms is unique. For a instance $(\boldsymbol{\mu}, \boldsymbol{\theta}, C) \in \mathcal{P}_{\text{ONUM}}$ and any feasible allocation $\boldsymbol{x} \in \mathcal{A}_C$, we define the sub-optimality gap as $\Delta_x = \sum_{i=1}^K \mu_i\left(\mathbb{1}_{\left\{x_i^\star \geq \theta_i\right\}} - \mathbb{1}_{\{x_i \geq \theta_i\}}\right)$. The maximum regret incurred in a round is $\Delta_m = \max_{\boldsymbol{x}\in\mathcal{A}_C} \Delta_x$.

Theorem 2. *Let $(\boldsymbol{\mu}, \theta_s, C) \in \mathcal{P}_{ONUM}$, $\mu_K \geq \epsilon > 0$, $\mu_M > \mu_{M+1}$, $W_\delta = \log(\log_2(K)/\delta)/\log(1/(1-\epsilon))$, and $T > W_\delta \log_2(K)$. Then with probability at least $1-\delta$, the expected regret of ONUM-ST is upper bounded as*

$$\mathbb{E}\left[\mathcal{R}_T\right] \leq W_\delta \log_2(K)\Delta_m + O\left((\log T)^{2/3}\right) + \sum_{i\in[K]\setminus[M]} \frac{(\mu_M - \mu_i)\log T}{d(\mu_i, \mu_M)}.$$

Proof. We divide the cumulative regret of ONUM-ST into the two parts: regret before finding a correct allocation equivalent $(\hat{\theta}_s)$ and regret after knowing allocation equivalent. The $\hat{\theta}_s$ estimation happens in T_{θ_s} rounds and returns a correct allocation equivalent with probability at least $1-\delta$. As Δ_m is the maximum regret that can be incurred in any round, the maximum regret incurred for estimating allocation equivalent is upper bounded by $\Delta_m T_{\theta_s}$. Given that $\hat{\theta}_s$ is correct, Theorem 1 gives the regret incurred after knowing $\hat{\theta}_s$. Hence the expected regret of ONUM-ST is a sum of these two regret bounds, and it holds with probability at least $1-\delta$. □

Note that the assumption $\mu_K \geq \epsilon > 0$ is only required to guarantee that the allocation equivalent of the threshold is found in finite time. This assumption is not required to get the upper bound on expected regret after knowing the allocation equivalent.

Corollary 1. *Let assumptions in Theorem 2 hold and set $\delta = T^{-(\log T)^{-\alpha}}$ in ONUM-ST such that $\alpha > 0$. Then the expected regret of ONUM-ST is upper bounded as*

$$\mathbb{E}\left[\mathcal{R}_T\right] \leq O\left((\log T)^{1-\alpha}\right) + O\left((\log T)^{2/3}\right) + \sum_{i\in[K]\setminus[M]} \frac{(\mu_M - \mu_i)\log T}{d(\mu_i, \mu_M)}.$$

Proof. The bound follows from Theorem 2 by setting $\delta = T^{-(\log T)^{-\alpha}}$ where $W_\delta = O(\log T)^{1-\alpha}$ and unconditioning the expected regret obtained in Theorem 2. □

Corollary 2. *The ONUM-ST is asymptotically optimal.*

The first term in the regret bound of Corollary 1 corresponds to the number of rounds needed to find an allocation equivalent, and the rest of it corresponds to the expected regret after knowing allocation equivalent. The proof of Corollary 2 follows by comparing the above bound with the lower bound in Eq. 1.

IV. Different Thresholds for All Users

Now we consider a general case where the threshold may not be the same for all arms. The first difficulty with this setup is to estimate the threshold for each of the arms. Unfortunately, we do not have a result equivalent of Lemma 1 so that the search space can be restricted to a finite set. We need to search over the entire $[0, C]$ interval for each arm. The second difficulty is to find an optimal allocation which need not be just allocating resource to top M arms. To see this, consider a problem instance $(\boldsymbol{\mu}, \boldsymbol{\theta}, C)$ with $\boldsymbol{\mu} = (0.9, 0.6, 0.4)$, $\boldsymbol{\theta} = (0.6, 0.55, 0.45)$, and $C = 1$. The optimal allocation is $\boldsymbol{x}^\star = (0, 0.55, 0.45)$ with no resource allocated to the top arm. Our first result gives the optimal allocation for an instance with different thresholds in $\mathcal{P}_{\text{ONUM}}$. Let $KP(\boldsymbol{\mu}, \boldsymbol{\theta}, C)$ denote a 0-1 knapsack problem with capacity C and K items where item i has weight θ_i and value μ_i.

Proposition 1. *Let $P = (\boldsymbol{\mu}, \boldsymbol{\theta}, C) \in \mathcal{P}_{ONUM}$. Then the optimal allocation for P is a solution of $KP(\boldsymbol{\mu}, \boldsymbol{\theta}, C)$ problem.*

Assigning θ_i resource to arm i increases the total mean reward by an amount μ_i. As the goal is to allocate a resource such that the total mean reward is maximized, i.e., $\max_{\boldsymbol{x}\in\mathcal{A}_C} \sum_{i\in[K]} \mu_i \mathbb{1}_{\{x_i \geq \theta_i\}}$. It is equivalent to solving a 0-1 knapsack with capacity C where item i has weight θ_i and value μ_i.

Let $l = C - \sum_{i:x_i^\star \geq \theta_i} \theta_i$ for an instance $P := (\boldsymbol{\mu}, \boldsymbol{\theta}, C)$, where r is the leftover resource after doing optimal allocation

and recall that $\boldsymbol{x}^\star = (x_1^\star, \ldots, x_K^\star)$ is the optimal allocation. Define $\gamma := l/K$. Note that any problem instance having $\gamma = 0$ becomes a 'hopeless' problem because the only threshold vector that is allocation equivalent to $\boldsymbol{\theta}$ is $\boldsymbol{\theta}$ itself, i.e., $x_i^* = \theta_i,\ \forall i \in [K]$, which needs θ_i values to be estimated with full accuracy to obtain the optimal allocation. But if $\gamma > 0$, then optimal allocation can be found with a small error in the estimates of θ_i, as shown in the next result.

Lemma 3. *Let $\gamma > 0$, $C \geq \gamma + \min_{i\in[K]} \theta_i$, and $\forall i \in [K], \hat{\theta}_i \in [\theta_i, \theta_i + \gamma]$. Then for any $\boldsymbol{\mu} \in [0,1]^K$, the instances $(\boldsymbol{\mu}, \boldsymbol{\theta}, C)$ and $(\boldsymbol{\mu}, \hat{\boldsymbol{\theta}}, C)$ are allocation equivalent.*

Let $L^\star = \{i : x_i^\star \geq \theta_i\}$ and $l = C - \sum_{i:x_i^\star \geq \theta_i} \theta_i$. Since $l < \min_{i\in K\setminus L^\star} \theta_i$, no reward can be obtained from any arm $i \in [K] \setminus L^*$. If the leftover resource l is uniformly distributed among all the arms i.e., increasing resource of each by an amount $\gamma = l/K$ for each arm, the optimal total mean reward still remains same. If threshold estimate of each arm $i \in [K]$ lies in $[\theta_i, \theta_i + \gamma]$, then by using Theorem 3.2 of [29], $KP(\boldsymbol{\mu}, \boldsymbol{\theta}, C)$ and $KP(\boldsymbol{\mu}, \hat{\boldsymbol{\theta}}, C)$ have the same optimal solution because of the total mean reward observed for instance $(\boldsymbol{\mu}, \boldsymbol{\theta}, C)$ and instance $(\boldsymbol{\mu}, \hat{\boldsymbol{\theta}}, C)$ is same.

Once the allocation equivalent of $\boldsymbol{\theta}$ is known, the problem is equivalent to solving a $KP(\boldsymbol{\mu}, \hat{\boldsymbol{\theta}}, N)$ which is equivalent to solving a Combinatorial Semi-Bandits [27] as shown in [23]. Combinatorial Semi-Bandits is the generalization of Multiple-Play Multi-Armed Bandits, where the size of superarms need not be identical in each round.

We develop an algorithm named Online Network Utility Maximization with the Different Threshold (ONUM-DT). It exploits result of Lemma 3 to find allocation equivalent and minimizes the regret using an algorithm from Combinatorial Semi-Bandits. The pseudo-code of ONUM-DT is given in Algorithm 2. ONUM-DT works as follows: it takes K, C, δ, ϵ and γ as input. We initialize the prior distribution of each arm as the Beta distribution $\beta(1,1)$ which is same as in ONUM-ST. For every arm $i \in [K]$, a binary search is performed over the interval $[0, C]$ and the variables $\hat{\theta}_i$, $\hat{\theta}_{t,i}$, $\hat{\theta}_{l,i}, \hat{\theta}_{u,i}, \hat{\theta}_{g,i}$ are tracked where $\hat{\theta}_i$ is the estimated value of θ_i, $\hat{\theta}_{t,i}$ is the current estimate of θ_i and initialized by C/K; $\hat{\theta}_{u,i}$ and $\hat{\theta}_{l,i}$ denote the upper and lower bound of the binary search region for arm i; and $\hat{\theta}_{g,i}$ indicates whether current estimate lies in the interval $[\theta_i, \theta_i + \gamma]$ (line 2). Z_i keeps count of consecutive 0 on arm i when it is allocated resource. Z_i changes to 0 either after observing a reward or if no reward is observed for consecutively W_δ rounds. Let $S_i(t)$ and $F_i(t)$ denote the value of S_i and F_i at the start of round t. In round t, for each $i \in [K]$ an independent sample of $\hat{\mu}_{t,i}$ is drawn from $\beta(S_i(t), F_i(t))$ (line 5).

ONUM-DT finds allocation equivalent of $\boldsymbol{\theta}$ by doing binary search for all i. We say that threshold estimate of arm i is good, i.e., $\hat{\theta}_i \in [\theta_i, \theta_i + \gamma]$ is checked by condition $\hat{\theta}_{u,i} - \hat{\theta}_{l,i} \leq \gamma$. If the condition satisfies, then the estimated threshold of the arm is within the desired tolerance, and it is indicated by setting $\hat{\theta}_{g,i} = 1$. Otherwise it is set to 0. If threshold estimate of arm i is good, we set $\hat{\theta}_i = \hat{\theta}_{u,i}$ (line 12). $\hat{\theta}_i$ represents the threshold estimate of arm i that is used after having $\hat{\theta}_{g,i} = 1,\ \forall i \in [K]$.

Algorithm 2 ONUM-DT

1: **Input:** $K, C, \delta, \epsilon, \gamma$
2: Initialize: $\forall i \in [K] : \hat{\theta}_i = C, \hat{\theta}_{1,i} = C/K, \hat{\theta}_{l,i} = 0, \hat{\theta}_{u,i} = C, \theta_{g,i} = 0, S_i = 1, F_i = 1, Z_i = 0$
3: Set $W_\delta = \log(K \log_2(\lceil 1 + C/\gamma \rceil)/\delta)/\log(1/(1-\epsilon))$
4: **for** $t = 1, 2, \ldots,$ **do**
5: $\quad \forall i \in [K] : \hat{\mu}_{t,i} \leftarrow \text{Beta}(S_i, F_i)$
6: $\quad$ **if** $\theta_{g,j} = 0$ for any $j \in [K]$ **then**
7: $\quad\quad \forall i \in [K]$, update $\hat{\theta}_{t,i}$ using Eq. (2). Allocate $\hat{\theta}_{t,i}$ resource to arm i and observe $Y_{t,i}$
8: $\quad\quad$ **for** $i = \{1, 2, \ldots, K\}$ **do**
9: $\quad\quad\quad$ **if** $\theta_{g,i} = 0$ and $\hat{\theta}_{t,i} > 0$ **then**
10: $\quad\quad\quad\quad$ If $Y_{t,i} = 1$ then set $\hat{\theta}_{u,i} = \hat{\theta}_{t,i}, S_i = S_i + 1, F_i = F_i + Z_i, Z_i = 0$ otherwise $Z_i = Z_i + 1$
11: $\quad\quad\quad\quad$ If $Z_i = W_\delta$ then set $\hat{\theta}_{l,i} = \hat{\theta}_{t,i}, Z_i = 0$
12: $\quad\quad\quad\quad$ If $\hat{\theta}_{u,i} - \hat{\theta}_{l,i} \leq \gamma$ then set $\theta_{g,i} = 1$ and $\hat{\theta}_i = \hat{\theta}_{u,i}$
13: $\quad\quad\quad$ **else if** $\theta_{g,i} = 1$ and $\hat{\theta}_{t,i} = \hat{\theta}_i$ **then**
14: $\quad\quad\quad\quad$ Set $S_i = S_i + Y_{t,i}$ and $F_i = F_i + 1 - Y_{t,i}$
15: $\quad\quad\quad$ **end if**
16: $\quad\quad$ **end for**
17: $\quad$ **else**
18: $\quad\quad A_t \leftarrow \text{Oracle}\big(KP(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\theta}}, C)\big)$
19: $\quad\quad \forall i \in A_t$, allocate $\hat{\theta}_i$ resource and observe $Y_{t,i}$. Update $S_i = S_i + Y_{t,i}$ and $F_i = F_i + 1 - Y_{t,i}$
20: $\quad$ **end if**
21: **end for**

If $\theta_{g,i} = 0$ (line 6) for some i, $\hat{\theta}_{t,i}$ is updated (line 7) after computing the following events:

$$B_i(t) = \left\{ \hat{\theta}_{t,i} \leq C - \sum_{\substack{j\in[K]:\theta_{g,j}=0 \\ \hat{\mu}_{t,j}/\hat{\theta}_{t,j} > \hat{\mu}_{t,i}/\hat{\theta}_{t,i}}} \hat{\theta}_{t,j} \right\},$$

$$G_i(t) = \left\{ \hat{\theta}_{t,i} \leq C - \sum_{j\in[K]:\theta_{g,j}=0} \hat{\theta}_{t,j} - \sum_{\substack{k\in[K],\theta_{g,k}=1 \\ \hat{\mu}_{t,k}/\hat{\theta}_{t,k} > \hat{\mu}_{t,i}/\hat{\theta}_{t,i}}} \hat{\theta}_{t,k} \right\},$$

and $E_\theta = \{\forall i \in [K] : \theta_{g,i} = 1\}$.

Event E_θ states that each arm has a good threshold estimate, which means ONUM-DT found the allocation equivalent for $\boldsymbol{\theta}$. In round t, event $B_i(t)$ is defined for arm i with $\theta_{g,i} = 0$ and indicates whether it can get resource or not. Event $G_i(t)$ is defined for arm with $\theta_{g,i} = 1$ and indicates if it can get resource. By construction, event $B_i(t)$ does not happen for arms having a good threshold estimate, and event $G_i(t)$ does not happen for arms having a bad threshold estimate. The arms having the highest empirical reward to resource ratio, i.e., $\hat{\mu}_j/\hat{\theta}_{t,i}$ gets resource first followed by second highest. The resource is first allocated among arms having a bad threshold estimate to find allocation equivalent as soon as possible. The leftover resource is allocated to arms with a good threshold

estimate to increase the reward. In round t, the $\hat{\theta}_{t,i}$ for arm i is updated as follows:

$$\hat{\theta}_{t,i} = \begin{cases} \hat{\theta}_{u,i} & \text{if } E_\theta \text{ or } G_i(t) \text{ happens} \\ \frac{\hat{\theta}_{l,i}+\hat{\theta}_{u,i}}{2} & \text{if } B_i(t) \text{ happens} \\ 0 & \text{Otherwise} \end{cases} . \quad (2)$$

If $\theta_{g,i} = 0$ for any arm, $\hat{\theta}_{t,i}$ resource is allocated to each arm $i \in [K]$ and reward $Y_{t,i}$ is observed (line 7). If reward 1 is observed for arm i with $\hat{\theta}_{g,i} = 0$, then the upper bound of threshold is $\hat{\theta}_{t,i}$, i.e, $\theta_{u,i} = \hat{\theta}_{t,i}$ (line 10). The success and failure counts are also updated as $S_i = S_i+1, F_i = F_i+Z_i$, and Z_i is reset to 0. If reward 0 is observed after allocating positive resource, Z_i is incremented by 1. If 0 reward is observed for successive W_δ rounds for arm i that have bad threshold estimate then it means that $\hat{\theta}_{t,i}$ is an underestimate of θ_i. So, lower bound of threshold to $\hat{\theta}_{t,i}$, i.e, $\theta_{l,i} = \hat{\theta}_{t,i}$ and Z_i is reset to 0 (line 11). For any arm i having good threshold estimate and $\hat{\theta}_i = \hat{\theta}_{u,i}$, its success and failure counts are updated as $S_i = S_i + Y_{t,i}, F_i = F_i + 1 - Y_{t,i}$ (line 14).

Once we have good threshold estimate for all arms, we could adapt to an algorithm that works well for Combinatorial Semi-Bandits, like SDCB [26] and CTS [27]. SDCB uses the UCB type index, whereas CTS uses Thompson Sampling. We adapt the CTS to our setting due to its better empirical performance. Oracle uses $KL(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\theta}}, C)$ to identify the arms in the round t where the learner has to allocate resource (denoted as set A_t in line 18). $\hat{\theta}_i$ resource is allocated to each arm $i \in A_t$ and reward $Y_{t,i}$ is observed. Then $S_i = S_i + Y_{t,i}, F_i = F_i + 1 - Y_{t,i}$ are updated (line 19).

A. Analysis of ONUM-DT

The value of W_δ in ONUM-DT is set such that the probability of threshold estimate does not lie in $[\theta_i, \theta_i + \gamma]$ for all arms is upper bounded by δ. Our next result gives an upper bound on the number of rounds required to obtain the allocation equivalent $\hat{\boldsymbol{\theta}}$ with high probability.

Lemma 4. *Let $(\boldsymbol{\mu}, \boldsymbol{\theta}, C) \in \mathcal{P}_{ONUM}$ such that $\gamma > 0$ and $\mu_K \geq \epsilon > 0$. Then with probability at least $1-\delta$, the number of rounds needed by* ONUM-DT *to find an allocation equivalent of $\boldsymbol{\theta}$ is upper bounded as*

$$T_{\theta_d} \leq \frac{K \log(K \log_2(\lceil 1 + C/\gamma \rceil)/\delta)}{\log(1/(1-\epsilon))} \log_2(\lceil 1 + C/\gamma \rceil).$$

Let Δ_x and Δ_m be defined as in Section III-C. Let $\gamma > 0$, $S_x = \{i : x_i \geq \theta_i\}$ for any feasible allocation a, $K_{max} = \max_{x \in \mathcal{A}_C} |S_x|$, and $k^\star = \min_{x^\star \in \mathcal{A}_C} |S_{x^\star}|$. Note that we redefine $W_\delta = \log(K \log_2(\lceil 1 + C/\gamma \rceil)/\delta)/\log(1/(1-\epsilon))$. We need the following results to prove the regret bounds.

Theorem 3. *Let $\hat{\boldsymbol{\theta}}$ be allocation equivalent to $\boldsymbol{\theta}$ for instance $(\boldsymbol{\mu}, \boldsymbol{\theta}, C)$. After knowing $\hat{\boldsymbol{\theta}}$, the expected regret of ONUM-DT in T rounds is upper bounded by $\left(\sum_{i \in [K]} \max_{S_x : i \in S_x} \frac{8|S_x| \log T}{\Delta_x - 2(k^{\star 2}+2)\eta}\right) + \left(\frac{K K_{max}^2}{\eta^2} + 3K\right)\Delta_m + \alpha_1 \left(\frac{8\Delta_m}{\eta^2}\left(\frac{4}{\eta^2}+1\right)^{k^\star} \log \frac{k^\star}{\eta^2}\right)$ for any η such that $\forall \boldsymbol{x} \in \mathcal{A}_C$, $\Delta_x > 2(k^{\star 2}+2)\eta$ and α_1 is a problem independent constant.*

Note that once the estimated $\hat{\boldsymbol{\theta}}$ is allocation equivalent to $\boldsymbol{\theta}$, the ONUM problem with the different thresholds is equivalent to solving a Combinatorial Semi-Bandits problem. The proof follows by verifying Assumptions $1-3$ of [27] for the Combinatorial Semi-Bandits setup and then applying their regret bounds. Assumption 1 states that the mean reward of a superarm only depends on the mean rewards of its constituting arms, and distributions of the arms are independent (Assumptions 3). Both these assumptions hold for our case. We next proceed to verify Assumption 2. For a fixed allocation $\boldsymbol{x} \in \mathcal{A}_C$, the mean reward collected from vector $\boldsymbol{\mu}$ is given by $r(S, \boldsymbol{\mu}) = \sum_{i \in S} \boldsymbol{\mu}_i$ where $S = \left\{i : x_i \geq \hat{\theta}_i\right\}$. For any two reward vectors $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$, we have

$$\begin{aligned} r(S, \boldsymbol{\mu}) - r(S, \boldsymbol{\mu}') &= \sum_{i \in S} (\mu_i - \mu_i') \\ &= \sum_{i \in [K]} \mathbb{1}_{\{x_i \geq \hat{\theta}_i\}} (\mu_i - \mu_i') \quad \left(\text{as } \sum_{i \in S} \mu_i = \sum_{i \in [K]} \mu_i \mathbb{1}_{\{x_i \geq \hat{\theta}_i\}}\right) \\ &\leq \sum_{i \in [K]} (\mu_i - \mu_i') \leq \sum_{i \in [K]} |\mu_i - \mu_i'| = B \parallel \boldsymbol{\mu} - \boldsymbol{\mu}' \parallel_1 \end{aligned}$$

where $B = 1$. After knowing the allocation equivalent, the allocation to each arm remains the same in every subsequent round ($\hat{\theta}_i$ resource is allocated to arm $i \in A_t$). By using Theorem 1 of [27] with parameter $B = 1$, we get the regret bounds of Theorem 3.

Theorem 4. *Let $(\boldsymbol{\mu}, \boldsymbol{\theta}, C) \in \mathcal{P}_{ONUM}$ such that $\gamma > 0$, $\mu_K \geq \epsilon > 0$, and $T > T_{\theta_d}$. Then with probability at least $1-\delta$, the expected regret of ONUM-DT is upper bounded by $\Delta_m K W_\delta \log_2(\lceil 1 + C/\gamma \rceil) + \left(\sum_{i \in [K]} \max_{S_x : i \in S_x} \frac{8|S_x| \log T}{\Delta_x - 2(k^{\star 2}+2)\eta}\right) + \left(\frac{K K_{max}^2}{\eta^2} + 3K\right)\Delta_m + \alpha_1 \left(\frac{8\Delta_m}{\eta^2}\left(\frac{4}{\eta^2}+1\right)^{k^\star} \log \frac{k^\star}{\eta^2}\right)$.*

The first term of regret bound corresponds to the regret incurred for finding the correct allocation equivalent with high probability. The number of rounds needed to find the correct allocation equivalent is T_{θ_d}. As Δ_m is the maximum regret that can be incurred in any round, the maximum regret incurred for estimating allocation equivalent is upper bounded by $\Delta_m T_{\theta_d}$. The other terms correspond to the regret incurred after knowing the allocation equivalent. Once an allocation equivalent is known, the expected regret incurred is upper bounded as given in Theorem 3. Hence the expected regret of ONUM-DT is a sum of these two regret bounds, and it holds with probability at least $(1-\delta)$.

Corollary 3. *Assume technical conditions stated in Theorem 4 hold. Set $\delta = 1/T$ in ONUM-DT. Then the expected regret of ONUM-DT is upper bounded by $\Delta_m K W_\delta \log_2(\lceil 1 + C/\gamma \rceil) + \left(\sum_{i \in [K]} \max_{S_x : i \in S_x} \frac{8|S_x| \log T}{\Delta_x - 2(k^{\star 2}+2)\eta}\right) + \left(\frac{K K_{max}^2}{\eta^2} + 3K\right)\Delta_m +$*

$\alpha_1 \left(\frac{8\Delta_m}{\eta^2} \left(\frac{4}{\eta^2} + 1 \right)^{k^\star} \log \frac{k^\star}{\eta^2} \right)$ *where* $W_\delta = \log(KT \log_2(\lceil 1 + C/\gamma \rceil))/\log(1/(1-\epsilon))$.

The above bound follows from Theorem 4 with $\delta = 1/T$ and unconditioning the expected regret obtained in Theorem 3.

V. Experiments

We evaluate the performance of ONUM-ST and ONUM-DT empirically on three synthetically generated instances. In instance 1, the threshold is the same for all arms, whereas, in instances 2 and 3, thresholds vary across arms. We ran the algorithm for $T = 10000$ rounds in all the simulations. All the experiments are repeated 50 times, and the regret curves are shown with a 95% confidence interval. The vertical line on each curve shows the confidence interval. The following empirical results validate sub-linear bounds for our algorithms. The details about the problem instances are as follows:

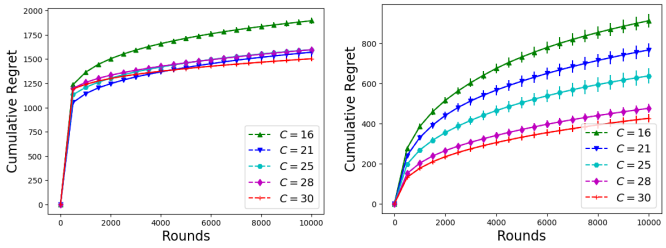
Instance 1 **(Identical Threshold):** It has $K = 50, C = 20$, $\theta_s = 0.7, \delta = 0.1$ and $\epsilon = 0.1$. The mean reward of arm $i \in [K]$ is $0.25 + (i-1)/100$.

Instance 2 **(Different Thresholds):** It has $K = 5, C = 2$, $\delta = 0.1$, $\epsilon = 0.1$ and $\gamma = 10^{-3}$. The mean reward vector is $\boldsymbol{\mu} = [0.9,\ 0.89, 0.87, 0.6, 0.3]$ and the corresponding threshold vector is $\boldsymbol{\theta} = [0.7, 0.7, 0.7, 0.6, 0.35]$.

Instance 3 **(Different Thresholds):** It has $K = 10, C = 3$, $\delta = 0.1$, $\epsilon = 0.1$ and $\gamma = 10^{-3}$. The mean reward vector is $\boldsymbol{\mu} = [0.9, 0.8, 0.42, 0.6, 0.5, 0.2, 0.11, 0.7, 0.3, 0.98]$ and the corresponding threshold vector is $\boldsymbol{\theta} = [0.6, 0.55, 0.3, 0.46, 0.34, 0.2, 0.07, 0.3, 0.25, 0.8]$.

We considered two different reward distributions of arms: 1) Bernoulli, where the rewards of arm i are Bernoulli distributed with parameter μ_i, and 2) Uniform, where the rewards of arm i is uniformly distributed in the interval $[\mu_i - 0.1, \mu_i + 0.1]$. For any continuous reward distribution with support in $(0, 1]$, the value of W_δ is set to 1 because the reward is observed with probability 1 when the allocated resource is above its threshold on any arm. For the Bernoulli distribution, the value of W_δ is 38 for instance 1, 62 for instance 2 and 69 for instance 3. Hence, we observe less regret for uniformly distributed rewards than Bernoulli distributed rewards. This difference is more significant when the arms have different thresholds.

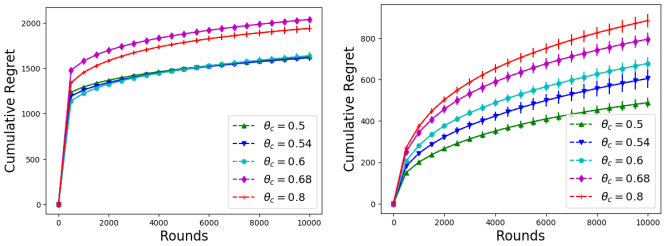
Experiments with the same threshold: We perform two different experiments on problem instance 1 using ONUM-ST. First, we varied the amount of resource C while keeping other parameters unchanged. With more resource, the learner can allocate resource to more arms. Hence learner can observe rewards from more arms in each round, which leads to faster learning and low cumulative regret, as shown in Fig. (1b) for the uniformly distributed rewards. For the uniform distribution we use binarization trick [30] to apply ONUM-ST: when a real-valued reward $Y_{t,i} \in (0, 1]$ is observed, the algorithm is updated with a fake binary reward that is drawn from Bernoulli distribution with parameter $Y_{t,i}$, i.e., $Y^f_{t,i} \sim Ber(Y_{t,i}) \in 0, 1$. The different amount of resource has different optimal allocation and sub-optimality gap. Hence with large W_δ value for Bernoulli distributed rewards, we may not observe similar behavior (less regret with more resource) as shown in Fig. (1a).



(a) Bernoulli Distributed Reward (b) Uniform Distributed Reward

Fig. 1: Regret of ONUM-ST.

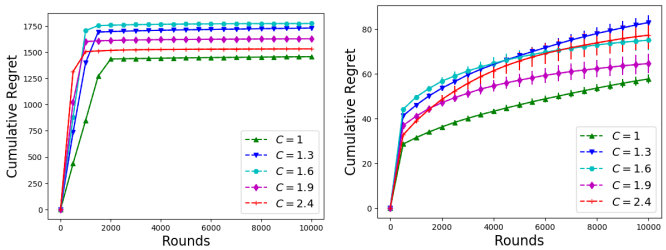
Second, we varied the threshold θ_s while keeping other parameters unchanged. As a smaller threshold allows the allocation of resource to more arms, we observe that a smaller threshold leads to faster learning due to more feedback. These trends are shown in Fig. (2a) and (2b) for Bernoulli and uniformly distributed rewards, respectively.



(a) Bernoulli Distributed Reward (b) Uniform Distributed Reward

Fig. 2: Regret of ONUM-ST.

Experiments with different thresholds: We evaluate the performance of ONUM-DT on problem instances 2 and 3. We varied the amount of resource C while keeping other parameters unchanged. As the thresholds are different across arms, an increase in the resource may lead to a selection of a different set of arms leading to different sub-optimality gaps. Hence, it does not show the same behavior (less regret with more resource) as observed for the same threshold. But we

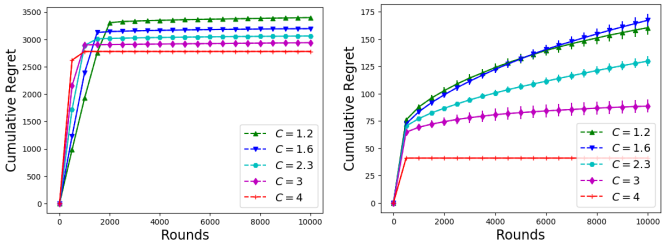


(a) Bernoulli Distributed Reward (b) Uniform Distributed Reward

Fig. 3: Regret of ONUM-DT.

observe that the allocation equivalent is learned faster as the reward of more arms can be observed simultaneously with

more resource. These observations are shown in Figs. (3a) and (3b) generated on instance 2 for Bernoulli and uniformly distributed rewards on instance 2, and same is repeated in Figs. (4a) and (4b) on instance 3. We run experiment 200 times for uniformly distributed rewards (Figs. (3b) and (4b)) on instance 2 and 3 as confidence intervals overlapped for 50 runs.



(a) Bernoulli Distributed Reward (b) Uniform Distributed Reward

Fig. 4: Regret of ONUM-DT.

VI. Conclusion and Future Extensions

We proposed a novel framework for Online Network Utility Maximization (ONUM) with unknown utilities. We focused on threshold type utilities where each agent gets non-zero utility only when its allocated resource is higher than some threshold. The goal is to assign resource among agents such that the total expected utility is maximized. We considered two variants of the problem depending on whether thresholds are identical across the arms (symmetric) or not (asymmetric). Using the concept of 'allocation equivalent,' and its connection to Multiple-Play Multi-Armed Bandits, we developed an optimal algorithm named ONUM-ST for the symmetric case. For the asymmetric case, we established that it is connected to a more general Combinatorial Semi-Bandits setup and developed an algorithm named ONUM-DT. Both algorithms achieve logarithm regret.

In our work, we assumed that a lower bound of the mean utilities is known, and it is also required knowledge of horizon T to achieve logarithms regret. It would be interesting to see if logarithm regret can be achieved without such assumptions.

Appendix

Proof of Lemma 2. The proof is adapted from Lemma 2 of [23] by allowing $\theta_s \in [0, C]$. Note that when $\hat{\theta}_s > \theta_s$, it can happen that no reward is observed for consecutive W_δ rounds and leads to incorrect estimation of θ_s. We want to set W_δ such a way that the probability of occurring of such event is upper bounded by δ.

Let $E_{\hat{\theta}_s}$ be the event that no reward is observed on $C/\hat{\theta}_s$ arms for W_δ consecutive rounds when $\hat{\theta}_s > \theta_s$. As $(1-\mu_i)$ is the probability of not observing reward at arm i, the probability of the event $E_{\hat{\theta}_s}$ is bounded as follows:

$$\mathbb{P}\left\{E_{\hat{\theta}_s} \text{ occurs } |\hat{\theta}_s \text{ 1st used at } T_{\hat{\theta}_s}\right\} = \prod_{w=T_{\hat{\theta}_s}}^{T_{\hat{\theta}_s}+W_\delta-1} \prod_{i\in A_w} (1-\mu_i)$$

As rewards are i.i.d., $\mu_K \geq \epsilon > 0$ and $\hat{\theta}_s \in [0, C]$, we have

$$\leq \prod_{w=T_{\hat{\theta}_s}+1}^{T_{\hat{\theta}_s}+W_\delta} (1-\epsilon)^{\frac{C}{\hat{\theta}_s}} = (1-\epsilon)^{\frac{CW_\delta}{\hat{\theta}_s}} \leq (1-\epsilon)^{W_\delta}$$

Since we are doing binary search, the algorithm goes through at most $\log_2(K)$ overestimates of θ_s.

$$\mathbb{P}\left\{E_{\hat{\theta}_s} \text{ for any overestimated } \hat{\theta}_s\right\} \leq (1-\epsilon)^{W_\delta} \log_2(K)$$

We bound the probability of making mistake by δ and get,

$$(1-\epsilon)^{W_\delta} \log_2(K) \leq \delta \implies (1-\epsilon)^{W_\delta} \leq \delta/\log_2(K)$$

Taking log both side, we have

$$W_\delta \log(1-\epsilon) \leq \log(\delta/\log_2(K))$$
$$\implies W_\delta \geq \frac{\log(\log_2(K)/\delta)}{\log(1/(1-\epsilon))}$$

W_δ is set as above so that ONUM-ST finds correct allocation equivalent with probability at least $1-\delta$ in $W_\delta \log_2(K)$ rounds.

Proof of Lemma 4. The proof is adapted from Lemma 4 of [23] by allowing $\theta_i \in [0, C]$. For any arm $i \in [K]$, we want $\hat{\theta}_i \in [\theta_i, \theta_i + \gamma]$ so we divide interval $[0, C]$ into a discrete set $\Theta \doteq \{0, \gamma, 2\gamma, \ldots, C\}$ and note that $|\Theta| = \lceil 1 + C/\gamma \rceil$.

Let $E_{\hat{\theta}_i}$ be the event that no reward is observed for consecutive W_δ rounds when $\hat{\theta}_i$ is overestimated. As $(1-\mu_i)$ is the probability of not observing reward for arm i and $\mu_K \geq \epsilon$, the probability of happening $E_{\hat{\theta}_i}$ is bounded by δ as follows:

$$\mathbb{P}\left\{E_{\hat{\theta}_i} \text{ happens}\right\} = (1-\mu_i)^{W_\delta} \leq (1-\epsilon)^{W_\delta}$$

Since we are doing binary search, the algorithm goes through at most $\log_2(|\Theta|)$ overestimates of θ_i.

$$\mathbb{P}\left\{E_{\hat{\theta}_i} \text{ happens for any overestimate}\right\} \leq (1-\epsilon)^{W_\delta} \log_2(|\Theta|)$$

Next, we will bound the probability of making mistake for any of the arm. That is given by

$$\mathbb{P}\left\{\exists i \in [K], E_{\hat{\theta}_i} \text{ happens for any overestimate}\right\}$$
$$\leq \sum_{i=1}^{K} \mathbb{P}\left\{E_{\hat{\theta}_i} \text{ happens for any overestimate}\right\}$$
$$\leq K(1-\epsilon)^{W_\delta} \log_2(|\Theta|)$$

We bound the probability of making mistake by δ and get,

$$K(1-\epsilon)^{W_\delta} \log_2(|\Theta|) \leq \delta \implies (1-\epsilon)^{W_\delta} \leq \delta/K \log_2(|\Theta|)$$

Taking log both side, we have

$$W_\delta \log(1-\epsilon) \leq \log(\delta/K \log_2(|\Theta|))$$
$$\implies W_\delta \geq \frac{\log(K \log_2(|\Theta|)/\delta)}{\log(1/(1-\epsilon))}$$

We set $W_\delta = \log(K \log_2(|\Theta|)/\delta)/\log(1/(1-\epsilon))$. Therefore, the minimum rounds needed for each arm i to correctly find $\hat{\theta}_i$ with probability at least $1 - \delta/K$ is upper bounded by $W_\delta \log_2(|\Theta|)$. Using union bound, all $\hat{\theta}_i \in [\theta_i, \theta_i + \gamma]$ are correctly estimated with probability at least $1 - \delta$ in $KW_\delta \log_2(|\Theta|)$ rounds where $|\Theta| = \lceil 1 + C/\gamma \rceil$.

ACKNOWLEDGMENTS

Arun Verma would like to thank travel support from LRN Foundation and COMSNETS Association. Manjesh K. Hanawal would like to thank the support from SEED grant (16IRCCSG010) from IIT Bombay, INSPIRE faculty fellowships from DST and Early Career Research (ECR) Award from SERB, Government of India.

REFERENCES

[1] F. P. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, pp. 33–37, 1997.

[2] Y. Yi and M. Chiang, "Stochastic network utility maximisation—a tribute to kelly's paper published in this journal a decade ago," *European Transactions on Telecommunications*, vol. 19, no. 4, pp. 421–442, 2008.

[3] M. J. Neely, "Delay based network utility maximization," in *IEEE INFOCOM*, 2010.

[4] A. Eryilmaz and I. Koprulu, "Discounted-rate utility maximization (drum): A framework for delay-sensitive fair resource allocation," in *IEEE WiOpt*, 2017.

[5] A. Sinha and E. Modiano, "Network utility maximization with heterogeneous traffic flows," in *IEEE WiOpt*, 2018.

[6] D. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[7] D. Palomar and M. Chiang, "Alternative distributed algorithms for network utility maximization: Framework and applications," *IEEE Transaction on Automatic Control*, vol. 52, no. 12, pp. 2254–2269, 2007.

[8] C.-P. Li and M. J. Neely, "Network utility maximization over partially observable markovian channels," *Perform. Eval.*, vol. 70, pp. 528–548, 2013.

[9] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2–3, pp. 235 –256, 2002.

[10] S. Bubeck and N. Cesa-Bianchi, *Regret analysis of stochastic and nonstochastic multi-armed bandit problems*, 2012.

[11] A. Anandkumar, N. Michael, A. Tang, and A. Swami, "Distributed algorithms for learning and cognitive medium access with logarithmic regret," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 4, pp. 731–745, 2011.

[12] N. Nayyar, D. Kalathil, and R. Jain, "Regret-optimal learning in decentralized multi-player multi-armed bandits," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 597–606, 2016.

[13] L. Besson and E. Kaufmann, "Multi-player bandits models revisited," in *Algorithmic Learning Theory (ALT)*, 2018.

[14] H. Tibrewal, S. Patchala, M. Hanawal, and S. Darak, "Distributed learning and optimal assignment in multiplayer heterogeneous networks," in *IEEE INFOCOM*, 2019.

[15] A. Verma, M. Hanawal, and R. Vaze, "Distributed algorithms for efficient learning and coordination in ad hoc networks," in *IEEE WiOpt*, 2019.

[16] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," in *IEEE INFOCOM*, 2019.

[17] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, "Bandits with knapsacks," *Journal of the ACM (JACM)*, vol. 65, no. 3, p. 13, 2018.

[18] J. D. Abernethy, K. Amin, and R. Zhu, "Threshold bandits, with and without censored feedback," in *Advances In Neural Information Processing Systems*, 2016, pp. 4889–4897.

[19] L. Jain and K. Jamieson, "Firing bandits: Optimizing crowdfunding," in *International Conference on Machine Learning*, 2018, pp. 2211–2219.

[20] T. Lattimore, K. Crammer, and C. Szepesvári, "Optimal resource allocation with semi-bandit feedback," in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2014, pp. 477–486.

[21] T. Lattimore, K. Crammer, and C. Szepesvári, "Linear multi-resource allocation with semi-bandit feedback," in *Advances in Neural Information Processing Systems*, 2015, pp. 964–972.

[22] Y. Dagan and C. Koby, "A better resource allocation algorithm with semi-bandit feedback," in *Proceedings of Algorithmic Learning Theory*, 2018, pp. 268–320.

[23] A. Verma, M. Hanawal, A. Rajkumar, and R. Sankaran, "Censored semi-bandits: A framework for resource allocation with censored feedback," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 499–14 509.

[24] R. Combes, M. S. T. M. Shahi, A. Proutiere *et al.*, "Combinatorial bandits revisited," in *Advances in Neural Information Processing Systems*, 2015, pp. 2116–2124.

[25] J. Komiyama, J. Honda, and H. Nakagawa, "Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays," in *International Conference on Machine Learning*, 2015, pp. 1152–1161.

[26] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, and P. Lu, "Combinatorial multi-armed bandit with general reward functions," in *Advances in Neural Information Processing Systems*, 2016, pp. 1659–1667.

[27] S. Wang and W. Chen, "Thompson sampling for combinatorial semi-bandits," in *International Conference on Machine Learning*, 2018, pp. 5101–5109.

[28] V. Anantharam, P. Varaiya, and J. Walrand, "Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part I," *IEEE Transactions on Automatic Control*, vol. 32, no. 11, pp. 968–976, 1987.

[29] M. Hifi and H. Mhalla, "Sensitivity analysis to perturbations of the weight of a subset of items: The knapsack case study," *Discrete Optimization*, vol. 10, no. 4, pp. 320–330, 2013.

[30] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Conference on Learning Theory*, 2012, pp. 39–1.