



A Multi-Arm Bandit Approach To Subset Selection Under Constraints

Ayush Deva, Kumar Abhishek, Sujit Gujar

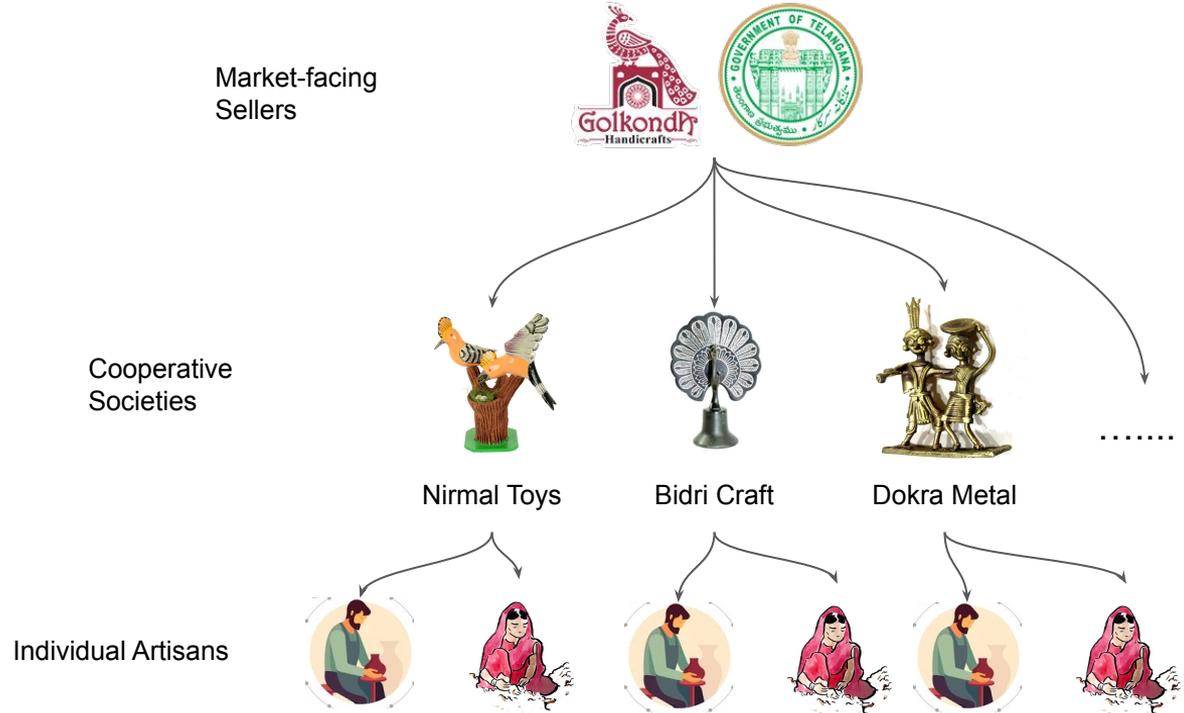
ayushdeva97@gmail.com | kumar.abhishek@research.iiit.ac.in | sujit.gujar@iiit.ac.in



Motivation

Cooperative Societies, such as those of artisans, where the quality and perceived value of each produce may vary greatly, it becomes important to carefully manage the inventory and generate supply accordingly.

The problem faced is hence about deciding what quantity of each type of product to produce / procure so as to maximize the overall revenue of the cooperative(s).



Subset Selection and Its Applications

Such problems fall under the wider category of **'Subset Selection'** and is faced not only by cooperative societies but also in other familiar marketplaces such as **E-commerce Platforms** and **Supermarkets** etc.

In all scenarios, the broader goal is to maximize revenue whilst ensuring certain quality standards.

The Amazon logo, consisting of the word "amazon" in a lowercase, black, sans-serif font with a curved orange arrow underneath it pointing from the 'a' to the 'z'.

Which seller's products to display to a user for a particular searched keyword.

The Coop logo, featuring the word "coop" in a lowercase, orange, sans-serif font where the letters are interconnected.

How many apples to procure from each of the regional farmers.

How to Quantify Quality?

- Quality is usually a **subjective** measure.
- For consumer products, a good proxy to quality is how **likely** the product is to be liked (read : purchased) by a consumer.
- In our setting, where an agent produces multiple units of a produce, the quality of an individual unit is **stochastic**, especially in artistic and farm products.
- We refer to to the **expected quality** of the units produced by an agent as ‘the quality of the agent’ (q_i).
- Hence, if the product has a selling price of ‘R’, and the expected probability of sale, q_i , the **expected revenue** it brings = $R \cdot q_i$

Ensuring expected average quality ~ Ensuring average quality

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,

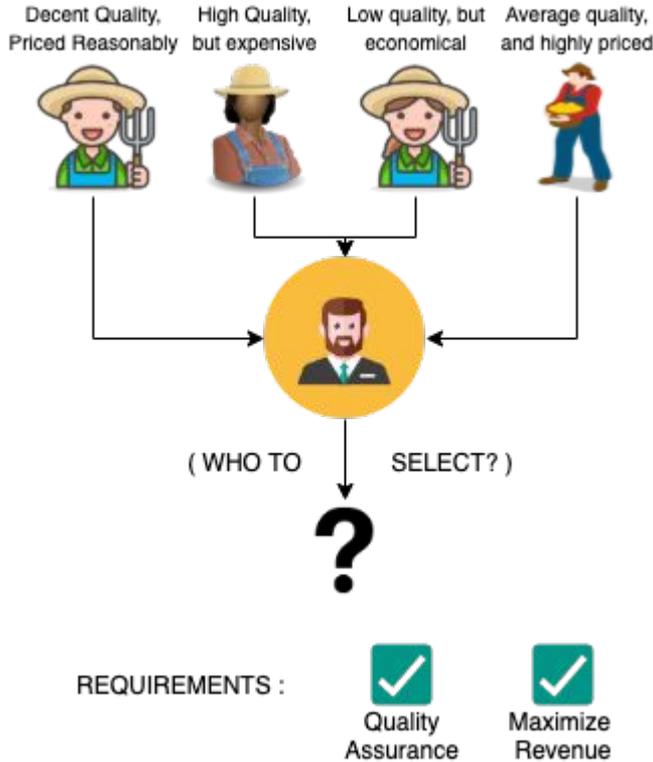
$$\begin{aligned} \mathcal{P}(V < \alpha - \epsilon \mid E[V] \geq \alpha) &\leq \mathcal{P}(V < E[V] - \epsilon) \\ &= \mathcal{P}(V - E[V] < -\epsilon) \leq \exp(-2\epsilon^2 m) \end{aligned}$$

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

Q_{ij} : Bernoulli RV (Realized Quality of j^{th} produce of agent i)

x_i : No. of units procured/selected from agent i

Problem Formulation



N agents, each associated with
capacity k_i ,
cost c_i , and
quality q_i .

Expected utility to planner upon selecting an agent is equal to $Rq_i - c_i$, where R is a constant.

Average Quality Threshold α

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

Dynamic Programming Based Solution

Agents are segregated into four categories :

- High Quality, Low Cost (S_1) ✓
- Low Quality, High Cost (S_4) ✗
- High Quality, High Cost (S_3) ?
- Low Quality, Low Cost (S_2) ?

We formulate it as a dynamic programming problem where selecting each unit brings in a revenue of $R^*(q_i - c_i)$ at an expense of $(\alpha - q_i)$, which is the loss in quality.

Hence, the problem is to maximize the revenue while making sure that the total expense is within *budget*.

Algorithm 1 DPSS

```

1: Inputs:  $N, \alpha, R$ , costs  $\mathbf{c} = \{c_i\}_{i \in N}$ , qualities  $\mathbf{q} = \{q_i\}_{i \in N}$ 
2: Output: Quantities procured  $\mathbf{x} = (x_1, \dots, x_n)$ 
3: Initialization:  $\forall i \in N, r_i = Rq_i - c_i, z = 0$ 
4: Segregate  $S_1, S_2, S_3, S_4$  as described in Section 3.4
5:  $\forall i \in S_1, x_i = 1; z = z + r_i; d = \sum_{i \in S_1} (q_i - \alpha)$ 
6:  $\forall i \in S_4, x_i = 0$ 
7:  $G = S_2 \cup S_3; \forall i \in G, d_i = q_i - \alpha$ 
8: function DP( $i, d^{te}, x^{te}, \mathbf{x}^*, z^{te}, z^*$ )
9:   if  $i == |G|$  and  $d^{te} < 0$  then return  $\mathbf{x}^*, z^*$ 
10:  if  $i == |G|$  and  $d^{te} \geq 0$  then
11:    if  $z^{te} > z^*$  then
12:       $z^* = z^{te}; \mathbf{x}^* = \mathbf{x}^{te}$ 
13:    return  $\mathbf{x}^*, z^*$ 
14:     $\mathbf{x}^*, z^* = \text{DP}(i + 1, d^{te}, [x^{te}, 0], \mathbf{x}^*, z^{te}, z^*)$ 
15:     $\mathbf{x}^*, z^* = \text{DP}(i + 1, d^{te} + d_i, [x^{te}, 1], \mathbf{x}^*, z^{te} + r_i, z^*)$ 
16:  return  $\mathbf{x}^*, z^*$ 
17:  $\mathbf{x}^G, z^G = \text{DP}(0, d, [ ], [ ], 0, 0)$ 
18:  $\forall i \in G, x_i = x_i^G$ 
19: return  $\mathbf{x}$ 

```

What if qualities are unknown?

- An artisan's quality of work may be hard to estimate until its products are procured and sold in the market.
- An E-commerce platform may not know its sellers' quality at the time of registration
- The planner needs to carefully learn the qualities by procuring units from the agents across multiple round in a way that minimizes the loss of revenue.

This is an example of **Exploration vs Exploitation** trade-off.

- This type of sequential learning problem is often formulated as Multi-Armed Bandit (MAB) problem.
- Here, each agent is modelled as an arm associated with an unknown stochastic parameter: quality.
- Since the planner can select multiple arms in each round (say, a day), and gets to observe the sale of all the procured products, this problem falls under the category of **Combinatorial MAB (CMAB) with semi-bandit feedback**.

Isn't there something that already addresses this?

Combinatorial Multi-Armed Bandit: General Framework, Results and Applications

Wei Chen

Microsoft Research Asia, Beijing, China

WEIC@MICROSOFT.COM

Yajun Wang

Microsoft Research Asia, Beijing, China

YAJUNW@MICROSOFT.COM

Yang Yuan

Computer Science Department, Cornell University, Ithaca, NY, USA

YANGYUAN@CS.CORNELL.EDU

- Assumes the availability of a feasible set of subsets (which they refer to as 'super arm') to select from
- In our setting, the feasible set (that satisfies the Quality Constraint) is unknown and needs to be learnt as well.

A quality assuring, cost optimal multi-armed bandit mechanism for expertsourcing



Shweta Jain^{a,*}, Sujit Gujar^{b,1}, Satyanath Bhat^{c,2}, Onno Zoeter^{d,3}, Y. Narahari^a

^a Indian Institute of Science, Bangalore, India

^b International Institute of Information Technology, Hyderabad, India

^c Institute of Operations Research and Analytics, NUS Business School, Singapore, Singapore

^d Booking.com, Amsterdam, Netherlands

- The objective function depends only on the agents' cost (which are known) and not the qualities of the agents.
- In our setting, both the constraint and the objective function depends on the unknown parameter.

(DP)SS - UCB

We propose a UCB-based framework that assumes the availability of an offline Subset Selection Algorithm (SSA) which outputs the optimal solution for the qualities and cost of agents, and the quality threshold provided to it as input.

When DPSS is used as the SSA, we refer to it as **DPSS-UCB**.

We prove that our framework achieves the following two properties:

1. Constraint Satisfaction
2. Sublinear Regret

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: $S^t = \text{SSA}(\{(\hat{q}_i^t)^+\}_{i \in N}, c, \alpha + \epsilon_2, R)$
 - 11: Observe qualities $X_i^j, \forall i \in S^t$ and update w_i^t, \hat{q}_i^t
 - 12: $t \leftarrow t + 1$
-

\hat{q}_i^t : Empirical Mean of Observed Qualities of arm i until round t

w_i^t : No. of rounds arm i was chosen until round t .

ϵ_2 : Constant, higher the value, sooner you want to ensure the quality constraint (in lieu of slightly higher revenue)

Ensuring Quality Constraints

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.

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

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

Ensuring Qualities during exploration is difficult and counter-productive.

We provide PAC bounds on DPSS satisfying QC after ' τ ' rounds.

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,

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

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

$$\frac{\sqrt{3 \ln t}}{\sqrt{2w_{min}^t}} \leq \frac{\sqrt{3 \ln T}}{\sqrt{2\tau}}.$$

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

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

LEMMA 3. $\forall t > \tau$

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

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

$$\begin{aligned} \mathcal{P}(E[Y^t] < \alpha - \epsilon_1 \mid Y^t \geq \alpha) &\leq \mathcal{P}(Y^t \geq E[Y^t] + \epsilon_1) \\ &\leq \exp(-\epsilon_1^2 w^t). \end{aligned}$$

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, $w^t \geq t$. Thus, $\forall t > \tau$

$$\mathcal{P}\left(q_{av}^t < \alpha - \epsilon_1 \mid \frac{1}{s^t} \left(\sum_{i \in S^t} (\hat{q}_i^t) \geq \alpha\right)\right) \leq \exp(-\epsilon_1^2 t). \quad \square$$

Regret Analysis

Definition

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

$$Reg^t(A) = \begin{cases} (r_q(S^*) - r_q(S^t)) & \text{if } S^t \text{ satisfies QC} \\ L & \text{otherwise.} \end{cases}$$

where $S^* = \operatorname{argmax}_{S \in S_f} r_q(S)$ and $L = \max_{S \in S_f} (r_q(S^*) - r_q(S))$ is some constant. Here, S_f are the feasible subsets which satisfies QC;

$$S_f = \{S | S \subseteq N \text{ and } \sum_{i \in S} x_i q_i \geq q_{av}\}.$$

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

$$Reg(A) = \sum_{t=1}^T Reg^t(A). \quad (2)$$

Part I : Before ' τ ' rounds

Let us assume that it always violates the QC, hence,

$$\sum_{t=1}^{\tau} Reg^t(A) \leq L \cdot \tau \leq \frac{L \cdot 3 \ln T}{2\epsilon_2^2}$$

Part II : After ' τ ' rounds when QC is violated

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

$$\begin{aligned} \sum_{t \geq \tau} \sigma L &= \sum_{t \geq \tau} L e^{(-\epsilon_1^2 t)} \leq \frac{L e^{(-\epsilon_1^2 \tau)}}{1 - e^{(-\epsilon_1^2)}} \\ &\sim O\left(\frac{1}{T^a}\right), \text{ where } a = \frac{3\epsilon_1^2}{2\epsilon_2^2}. \end{aligned}$$

Part III : After ' τ ' rounds when QC is not violated

Here, we adopt our proof from [Chen et al. '13'](#).

We show that the utility function considered in our problem setting follows both the required properties, namely, monotonicity and bounded smoothness. Using the same proof, we can show that the regret incurred here is also of $O(\ln T)$.

Greedy Approach

DPSS is $O(2^n)$ which makes it difficult to scale when n is large.

We propose a greedy based algorithm, GSS, that runs in polynomial time, $O(n/\log(n))$, and provides an approximate solution to our ILP.

GSS can also be used as a SSA in the SS-UCB framework, which we refer to as GSS-UCB.

Algorithm 3 GSS

```
1: Inputs:  $N, \alpha, R$ , costs  $\mathbf{c} = [c_i]$ , qualities  $\mathbf{q} = [q_i]$ 
2: Output: Quantities procured  $\mathbf{x} = (x_1, \dots, x_n)$ 
3: Initialization:  $\forall i \in N, r_i = Rq_i - c_i$ 
4: Segregate  $S_1, S_2, S_3, S_4$  as described in Section 3.4
5:  $\forall i \in S_1, x_i = 1; d = \sum_{i \in S_1} (q_i - \alpha)$ 
6:  $\forall i \in S_4, x_i = 0$ 
7:  $L_2 = \text{sort}(S_2)$  on decreasing order of  $\frac{r_i}{\alpha - q_i}$ 
8:  $L_3 = \text{sort}(S_3)$  on increasing order of  $\frac{r_i}{\alpha - q_i}$ 
9:  $p = 0, q = 0$ 
10: while  $d > 0$  and  $p < |S_2|$  do
11:    $i = L_2[p];$ 
12:   if  $\alpha - q_i \leq d$  then  $x_i = 1, d = d - \alpha - q_i, p++ = 1$ 
13:   else  $x_i = \frac{d}{\alpha - q_i}, d = 0$ 
14: while  $p < |S_2|$  and  $q < |S_3|$  do
15:    $i = L_2[p], j = L_3[q]$ 
16:    $a = \frac{r_i}{\alpha - q_i}, b = \frac{r_j}{\alpha - q_j}$ 
17:   if  $a \leq b$  then break;
18:    $w_1 = \min((1 - x_i)(\alpha - q_i), (1 - x_j)(q_j - \alpha))$ 
19:    $x_i += \frac{w_1}{\alpha - q_i}, x_j += \frac{w_1}{q_j - \alpha}$ 
20:   if  $x_i == 0$  then  $p++;$ 
21:   if  $x_j == 0$  then  $q++;$ 
22: if  $0 < x_j < 1$  then  $x_j = 1$ 
23: return  $\lfloor \mathbf{x} \rfloor$ 
```

How Fast and How Approximate?

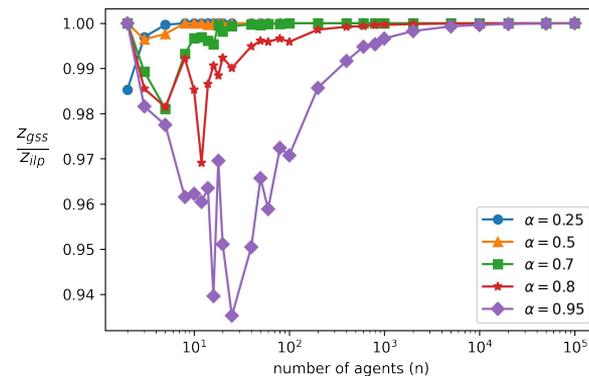
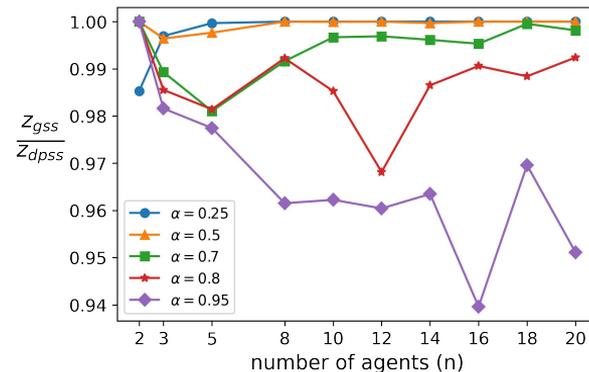
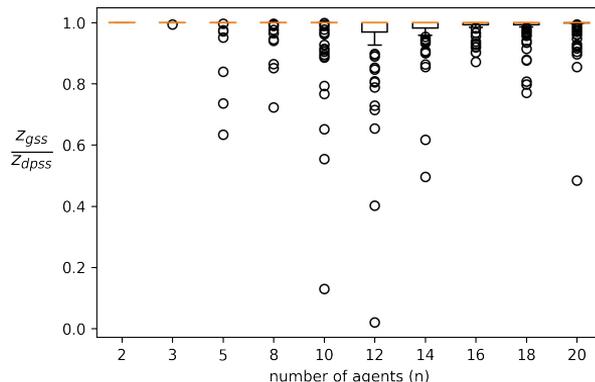
We compare the empirical performance of GSS with DPSS and an ILP solver - the COIN-OR Branch and Cut Solver (CBC).

n	$t_{dpss} : t_{gss}$	$t_{ilp} : t_{gss}$
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

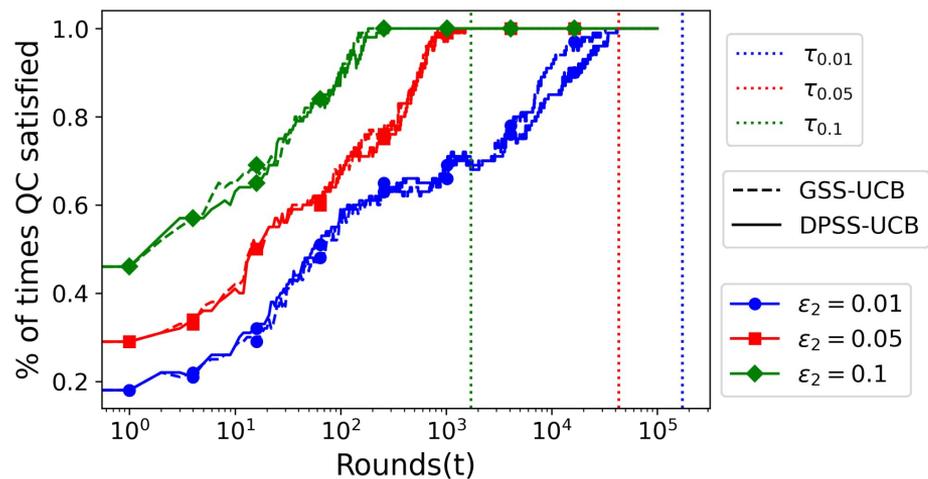
n	$t_{ilp} : t_{gss}$
25	66.7
50	58.3
100	52.7
400	43.1
1000	31.8
5000	31.6
10000	34.5
50000	45
100000	56.8

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

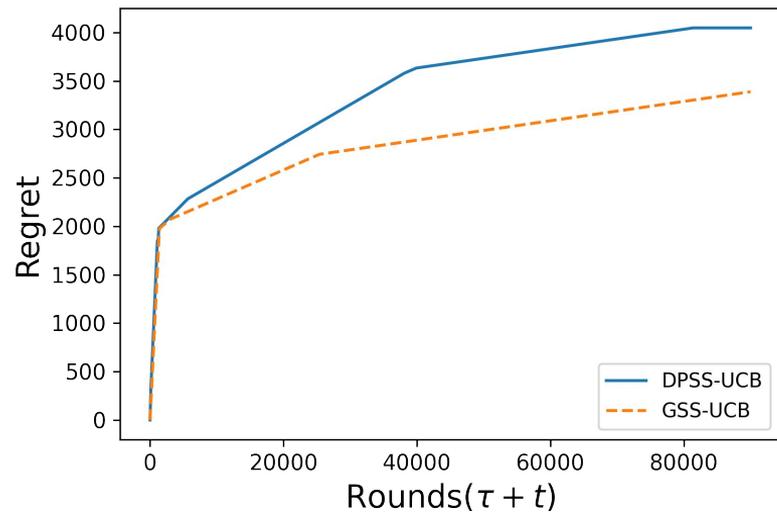
The theoretical approximation limit can be arbitrarily small but in practice it gives an average approximation ratio greater than 0.9



Experimental Results



Constraint Satisfaction



Regret Incurred

Possible Extensions To This Research

1. Dynamic Pool of Agents
2. An agent selected in a particular round is not available for the next few round possibly due to the lead time in procuring the units. *Sleeping Bandits*.
3. Inclusion of strategic agents where a planner also needs to elicit the cost of production truthfully. *Mechanism Design*

