Multi-Armed Bandits
Project Report

Learning with Semi-Bandit Feedback

Report by:
1. **A**bdul Bakey Mir (CS20D400)
2. **A**dwait Pramod Parsodkar (CS20D404)
3. **A**nirudh Sharma (EE18B036)

1. Introduction

The online combinatorial optimization is a special case of online linear optimization in which the learner chooses a subset of arms from a set of arms. The subset of arms chosen by the learner at time 't' is referred to as the action taken at 't'. The action taken by the learner often has to satisfy certain constraints such as the number of arms chosen not exceeding some value 'K'. At the same time, the adversary fixes a loss vector, and the loss suffered by the learner is the sum of losses associated with the chosen arms. It might also be the case that the adversary directly provides the total loss instead of breaking the loss down across the various arms. Depending on what kind of inputs are received by the adversary for the computation of loss, the difficulty of the problem varies.

The online combinatorial optimization problem encompasses a larger variety of problems, such as online least cost path problem, minimum spanning tree (MST) problem and maximum-weighted bipartite matching. In section 2 we discuss how the online least cost path problem can be mapped to the online combinatorial optimization setting.

We now describe the organization of the report. Section 2 defines the combinatorial optimization problem, the semi bandits setting, and the linear generalization assumption. The approaches from the assigned paper along with their regret analysis are explained in section 3. The associated experiments and their results are respectively presented in section 4 and section 5.

2. Background

2.1 Combinatorial Optimization

The combinatorial optimization problem can be represented as a triple (E,**A**,w) where
E = Set of L arms,
A = Set of actions, where an action is one or more arms satisfying some constraint,
w : E → R, is a function that assigns real valued weights to the arms in E.
The total weight of an action is essentially given by the sum of the weights of the component arms.

Mathematically, $f(A, w) \;=\; \sum_{e \in A} w(e).$

The optimal action to be taken for a value of w is called the optimal action, and is denoted as A*.
Consider the online least cost path problem, which we shall refer to as OLCP, in which the set of arms correspond to the set of all the edges in the graph. In the problem of finding the cheapest path from the source node S to the target node T, an action corresponds to a legal path from S to T. Optional constraints can be applied to this

2.2 Semi-Bandits Setting

Depending on what kind of loss value we get, the setup can be classified into full information, semi-bandits or full-bandits setting. In the full information setting, irrespective of any action chosen by the learner, the adversary provides a loss vector indicating the loss corresponding to each arm. On the other hand, the learner is provided with losses corresponding to only the arms in the chosen action in the semi-bandit setting. In the full-bandit setting, the learner is provided with the total loss suffered due to the choice of the action.

2.3 Linear Generalization

In many real world applications concerning combinatorial optimization problems, the cost of an arm is dependent (possibly imperfectly) on some of its pre-known feature values. In the OLCP, the cost of a path might depend on features such as the traffic on the path, the quality of the path and the length of the path. Note that these feature values corresponding to all the actions are known to the algorithm, and are assumed to be constant throughout the run of the algorithm. The features values corresponding to the various arms can be expressed in a matrix, which we shall refer to as Φ, such that the i^{th} row of this matrix denotes the feature vector of the i^{th} arm.

The linear generalization assumption considers a linear dependence of the cost of an action on its feature values, where the weights to be given to each feature is expressed in a vector denoted by θ. The goal now boils down to figuring out the entries in θ.

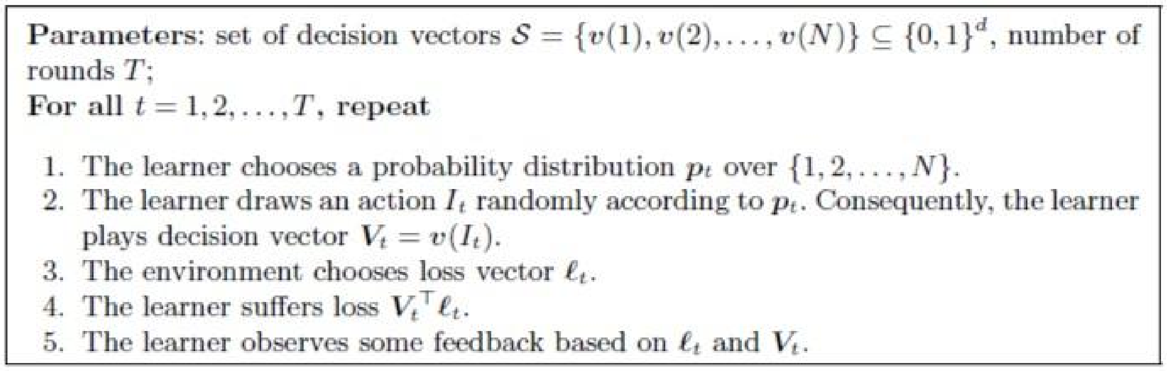
As a result of this, the cost of a path can be approximated as follows $w(e) = \Phi_e \theta$

3. Approaches

3.1 Approach Used in paper 1 (Efficient Algorithm for learning with semi-bandit feedback)

This paper considers the problem of online combinatorial optimization under semi-bandit feedback. This paper assumes that the decision set is finite represented by set $S = \{ v(1) , v(2) , ..., v(N)\} \subseteq \{0, 1\}^d$ and the elements of the decision set can be described with d-dimensional binary vectors with at most m non-zero entries that correspond to at most m arms that can be selected in a particular round.

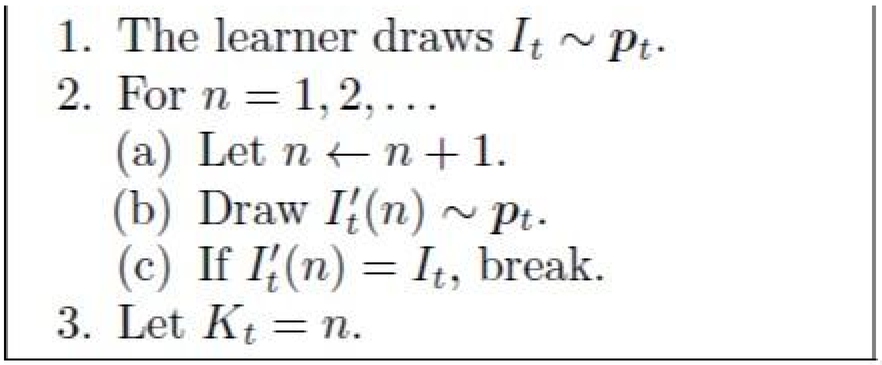
Formally the paper introduces the protocol of online combinatorial optimization as below:



```
Parameters: set of decision vectors $\mathcal{S} = \{v(1), v(2), \dots, v(N)\} \subseteq \{0,1\}^d$, number of rounds $T$;
For all $t = 1, 2, \dots, T$, repeat
    1. The learner chooses a probability distribution $p_t$ over $\{1, 2, \dots, N\}$.
    2. The learner draws an action $I_t$ randomly according to $p_t$. Consequently, the learner plays decision vector $V_t = v(I_t)$.
    3. The environment chooses loss vector $\ell_t$.
    4. The learner suffers loss $V_t^\top \ell_t$.
    5. The learner observes some feedback based on $\ell_t$ and $V_t$.
```

This paper uses Follow-The-Perturbed-Leader (FPL) based approach to solve the combinatorial optimization problem under semi-bandit feedback. Let us first consider the normal bandit setting where $d = N$ and $m = 1$. In each time step the learner specifies a distribution p_t over the arms where $p_{t,i} = P[I_t = i | F_{t-1}]$ where F_{t-1} is the history of the learners observations. Almost all bandit algorithms rely on feeding some loss estimates to black box algorithms like Hedge etc. A commonly used form of loss estimate for the i^{th} arm in t^{th} round is $\hat{l}_{t,i} = \frac{l_{t,i}}{p_{t,i}} I\{I_t = i\}$. This is an unbiased estimate for the true loss which is a nice property but these forms of loss estimates can be used when we can maintain p_t explicitly. For many good algorithms like FPL maintaining an explicit probability vector is notoriously hard.

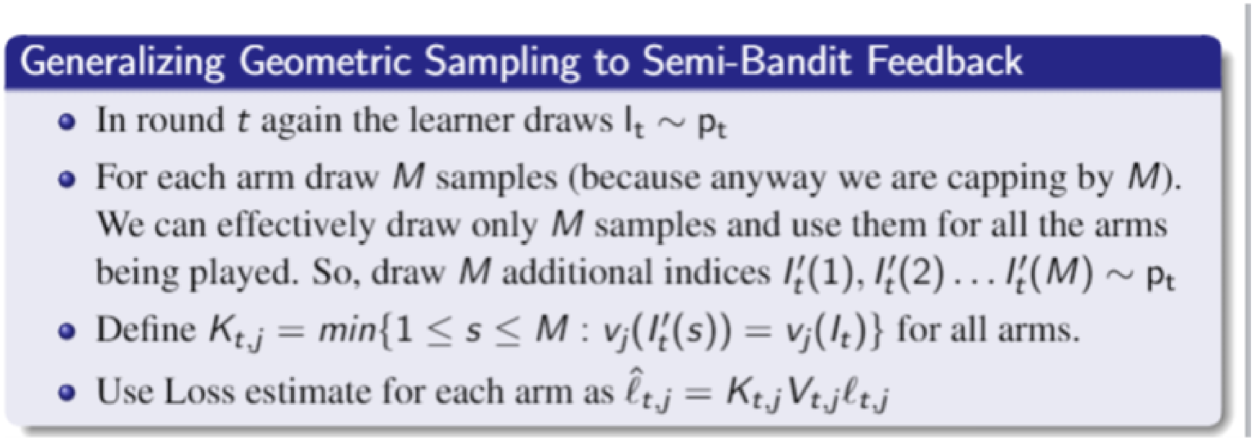
To circumvent this problem the authors have introduced a procedure called **Geometric Resampling** and then use a form of loss estimate based on this.



```
1. The learner draws $I_t \sim p_t$.
2. For $n = 1, 2, \dots$
    (a) Let $n \leftarrow n + 1$.
    (b) Draw $I'_t(n) \sim p_t$.
    (c) If $I'_t(n) = I_t$, break.
3. Let $K_t = n$.
```

Clearly, K_t is a geometrically distributed random variable given I_t, F_{t-1}. The loss estimate used for the i^{th} arm in t^{th} round is $\hat{l}_{t,i} = l_{t,i} I\{I_t = i\} K_t$. This is also an unbiased estimate for the losses. But this sampling process has a problem that it can be unbounded so cap the number of samples by M and use $\hat{K}_t = min\{K_t, M\}$This introduces some bias but it does not hurt much if M is chosen carefully.

Now let us generalize this Geometric Resampling to semi-bandit setting.



Generalizing Geometric Sampling to Semi-Bandit Feedback

- In round t again the learner draws $\mathrm{I_t} \sim \mathrm{p_t}$
- For each arm draw M samples (because anyway we are capping by M). We can effectively draw only M samples and use them for all the arms being played. So, draw M additional indices $I_t'(1), I_t'(2) \dots I_t'(M) \sim \mathrm{p_t}$
- Define $K_{t,j} = min\{1 \leq s \leq M : v_j(I_t'(s)) = v_j(I_t)\}$ for all arms.
- Use Loss estimate for each arm as $\hat{\ell}_{t,j} = K_{t,j} V_{t,j} \ell_{t,j}$

Since $V_{t,j}$ is non-zero only for those arms for which the loss is observed. Hence the above loss estimates are well defined.

3.2 Algorithm

```
Input: $\mathcal{S} = \{v(1), v(2), \ldots, v(N)\} \subseteq \{0,1\}^d$, $\eta \in \mathbb{R}^+$, $M \in \mathbb{Z}^+$;
Initialization: $\widehat{L}(1) = \cdots = \widehat{L}(d) = 0$;
for $t$=1,...,$T$ do
    Draw $Z(1), \ldots, Z(d)$ independently from distribution Exp($\eta$);
    Choose action $I = \underset{i \in \{1,2,\ldots,N\}}{\arg\min} \left\{ v(i)^\top \left( \widehat{L} - Z \right) \right\}$;
    $K(1) = \cdots = K(d) = M$;
    $k = 0$;                                   /* Counter for reoccurred indices */
    for $n$=1,...,$M$-1 do                     /* Geometric Resamplig */
        Draw $Z'(1), \ldots, Z'(d)$ independently from distribution Exp($\eta$);
        $I(n) = \underset{i \in \{1,2,\ldots,N\}}{\arg\min} \left\{ v(i)^\top \left( \widehat{L} - Z' \right) \right\}$;
        for $j$=1,...,$d$ do
            if $v(I(n))(j) = 1$ & $K(j) = M$ then
                $K(j) = n$;
                $k = k + 1$;
                if $k = \|v(I)\|_1$ then break;     /* All indices reoccurred */
            end
        end
    end
    for $j$=1,...,$d$ do  $\widehat{L}(j) = \widehat{L}(j) + K(j) v(I)(j) \ell(j)$ ;      /* Update */
end
```

This Geometric Resampling procedure is now coupled with FPL to produce an algorithm called as **FPL-with-GR** for the semi-bandit setting given below:

In the above algorithm the distribution p_t is explicitly specified by Z_t. At each step we play the decision vector whose index is given by $I = argmin\{V(i)^T(\hat{L} - Z)\}$. Then the losses for each of the arms being played is given estimated using Geometric Resampling and the losses for the non-played arms is 0. The loss for each arm is then added to its current cumulative loss and the algorithm proceeds to the next round with this loss.

3.3 Regret Bound and Comparison with State of Art

The total expected regret of **FPL-with-GR** satisfies $R_n \leq \frac{m(\log d\, 1)}{\eta} + \eta m d T + \frac{dT}{eM}$ under semi-bandit information. In particular if we set $\eta = \sqrt{(\log d\; + 1)/dT}$ and $M \geq \frac{\sqrt{dT}}{em\sqrt{\log d + 1}}$ the regret can be upper bounded as $R_n \leq 3m\sqrt{dT(\log d\; + 1)} = O(m\sqrt{dT\log d}$.

In the full-information setting the authors have provided a tighter upper bound for **FPL** as $R_n \leq \frac{m(\log d + 1)}{\eta} + \eta m C_T$ where $C_T = \sum_{t=1}^{T} E[V_t^T l_t]$. In particular if we set $\eta = \sqrt{(\log d\; + 1)/mT}$ the regret can be upper bounded as $R_n \leq 2\, m^{\frac{3}{2}}\sqrt{T(\log d\; + 1)}$

The optimal regret one can achieve in the semi-bandit case is $O(\sqrt{mdT})$ which has been shown to be achieved by Online-Stochastic-Mirror-Descent (OSMD) or Follow-The-Regularized-Leader(FTRL) based methods. This algorithm **FPL-with-GR** is off by a factor of $O(\sqrt{m\log d})$.

The optimal regret one can achieve in the full-information setting is $O(m\sqrt{T\log d})$ which is shown to be achieved by component Hedge . This algorithm **FPL-with-GR** is here off by a factor of $O(\sqrt{m})$.

So, it is conclusive that in terms of regret bound this paper has not achieved much but the algorithms that are shown to have optimal regret are not computationally efficient. OSMD/FTRL methods can be efficiently implemented by convex programming if the convex hull of the decision set can be described by a polynomial number of constraints. Similarly methods based on exponential weighting of each decision vector can only be efficiently implemented only for a few handful of decision sets.So there is a lack of general efficient algorithms for online combinatorial optimization.

Let us now have a look at the expected Running time of **FPL-with-GR.** For each coordinate that the original arm had 1, we take sampling until we get 1 in the same coordinate again. (Here we do not assume cutoff). Let $q_{t,k} = E[V_{t,k} \mid F_{t-1}]$. At time step t for a given coordinate k with 1, the expected number of samples is $\frac{1}{q_{t,k}}$ while the probability of coordinate k being 1 is $q_{t,k}$. So, the expected number of samples per round is $\sum_{k=1}^{d} q_{t,k}\frac{1}{q_{t,k}} = d$. So, for T rounds the expected running time is dT which is highly feasible.

3.4 Approach for Paper 2 (Efficient Learning in Large Scale Combinatorial Bandits)

This paper focuses on large scale bandits and tries to develop robust algorithms that can be used even when the number of arms is very huge. The setup used in this paper is formalized below:

3.5 Setup

This paper uses a general combinatorial setup as shown in section []. This paper also uses linear generalization. Without loss of generality we assume rank of generalization matrix Φ is d. This paper separates linear generalization in two cases:

3.5.1 Coherent Learning Case:

The weight received for each arm is an exact linear combination of its features. In other words $\bar{w} \in span[\Phi]$.

3.5.2 Agnostic Learning Case

The weight received for each arm is not an exact linear combination of its features. In other words $\bar{w} \notin span[\Phi]$but is close to $span[\Phi]$.

3.6 Performance Metric Used in this Paper

The learning algorithm choses a subset of arms A_t at round t and the loss suffered by an algorithm at this round $R_t = f(A^*, w_t) - f(A_t, w)$ where $A^* = ORACLE(E, A, \bar{w})$.

If $\bar{w}$ is fixed then we measure the performance w.r.t the cumulative regret of the learning algorithm in T rounds as $R_T = \sum_{t=1}^{T} E[R_t \mid \bar{w}]$where expectation is over random weights and possible randomization in the algorithm.

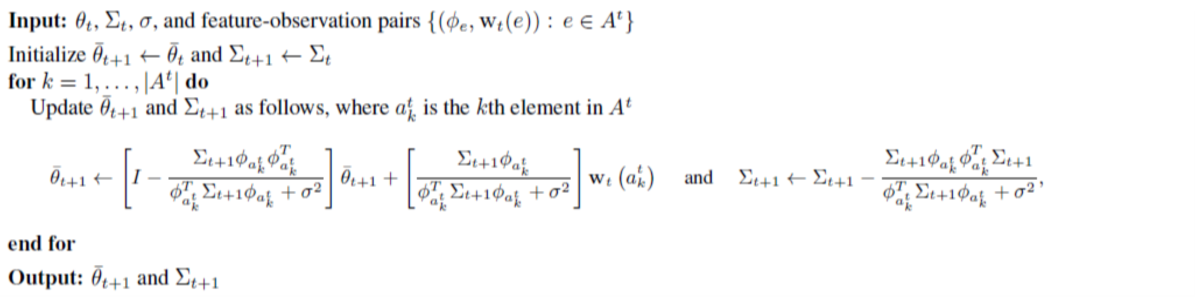
If $\bar{w}$ is also randomly generated then performance is measured w.r.t cumulative regret of learning algorithm as $R_T = \sum_{t=1}^{T} E[R_t]$ where expectation is over random weights, possible randomization in the algorithm and also over $\bar{w}$. This is the Bayes regret bound.

We also define $\theta^* = argmin_\theta \|\bar{w} - \Phi\theta\|$. Since $rank[\Phi] = d$, θ^* is uniquely defined. Also in coherent learning case $\bar{w} = \Phi\theta^*$.

3.7 Algorithm

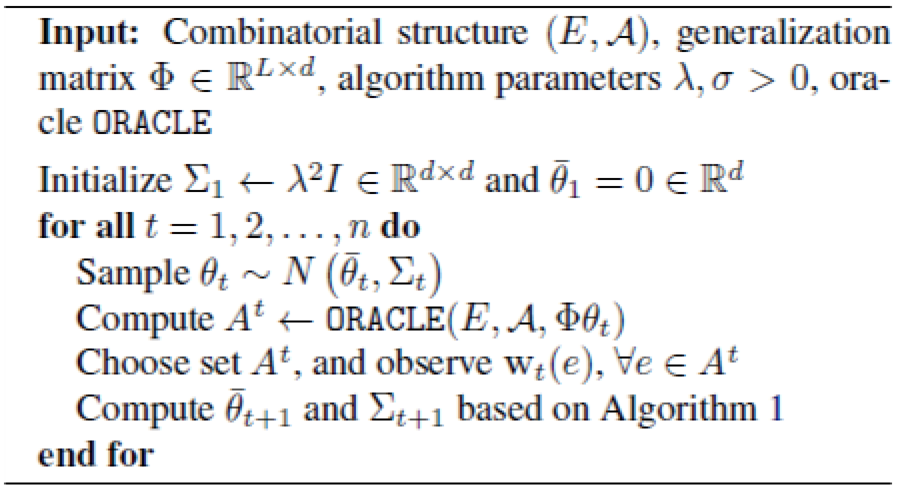
This paper introduces two algorithms **Combinatorial Linear Thompson Sampling** and **Combinatorial Linear UCB.** Both these algorithms use Kalman filtering as a parameter estimation technique. Let us discuss that first:

3.7.1 Kalman Filtering (Algorithm 1)



```
**Input:** $\theta_t$, $\Sigma_t$, $\sigma$, and feature-observation pairs $\{(\phi_e, \mathrm{w}_t(e)) : e \in A^t\}$
**Initialize** $\bar{\theta}_{t+1} \leftarrow \bar{\theta}_t$ and $\Sigma_{t+1} \leftarrow \Sigma_t$
**for** $k = 1, \dots, |A^t|$ **do**
    Update $\bar{\theta}_{t+1}$ and $\Sigma_{t+1}$ as follows, where $a_k^t$ is the $k$th element in $A^t$
    $\bar{\theta}_{t+1} \leftarrow \left[I - \frac{\Sigma_{t+1}\phi_{a_k^t}\phi_{a_k^t}^T}{\phi_{a_k^t}^T \Sigma_{t+1}\phi_{a_k^t} + \sigma^2}\right]\bar{\theta}_{t+1} + \left[\frac{\Sigma_{t+1}\phi_{a_k^t}}{\phi_{a_k^t}^T \Sigma_{t+1}\phi_{a_k^t} + \sigma^2}\right]\mathrm{w}_t\left(a_k^t\right)$ and $\Sigma_{t+1} \leftarrow \Sigma_{t+1} - \frac{\Sigma_{t+1}\phi_{a_k^t}\phi_{a_k^t}^T\Sigma_{t+1}}{\phi_{a_k^t}^T \Sigma_{t+1}\phi_{a_k^t} + \sigma^2}$,
**end for**
**Output:** $\bar{\theta}_{t+1}$ and $\Sigma_{t+1}$
```

3.7.2 Combinatorial Linear Thompson Sampling



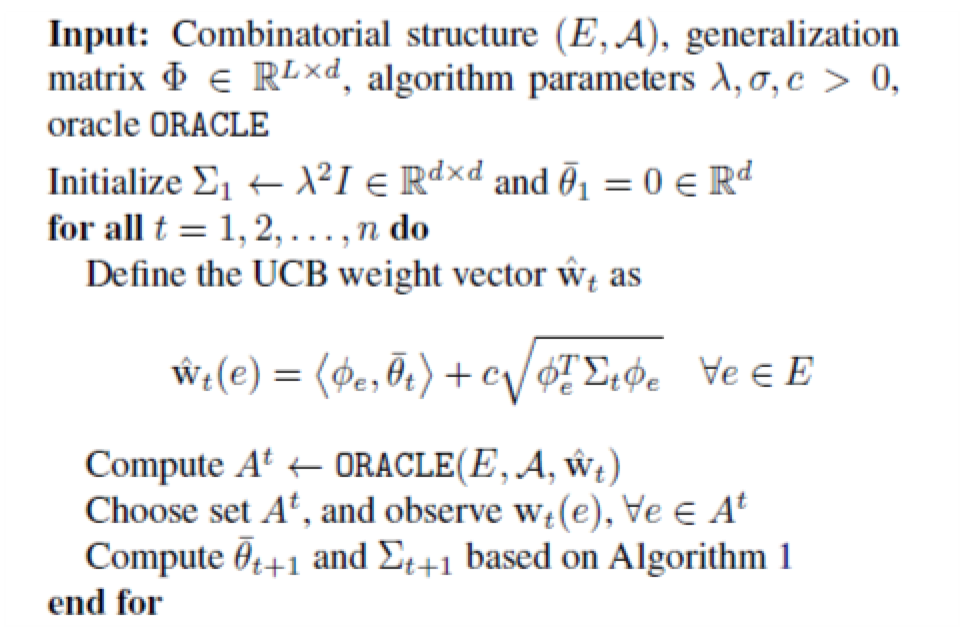
```
**Input:** Combinatorial structure $(E, \mathcal{A})$, generalization matrix $\Phi \in \mathbb{R}^{L \times d}$, algorithm parameters $\lambda, \sigma > 0$, oracle ORACLE
Initialize $\Sigma_1 \leftarrow \lambda^2 I \in \mathbb{R}^{d \times d}$ and $\bar{\theta}_1 = 0 \in \mathbb{R}^d$
**for all** $t = 1, 2, \dots, n$ **do**
    Sample $\theta_t \sim N\left(\bar{\theta}_t, \Sigma_t\right)$
    Compute $A^t \leftarrow \mathtt{ORACLE}(E, \mathcal{A}, \Phi\theta_t)$
    Choose set $A^t$, and observe $\mathrm{w}_t(e)$, $\forall e \in A^t$
    Compute $\bar{\theta}_{t+1}$ and $\Sigma_{t+1}$ based on Algorithm 1
**end for**
```

Here λis the inverse regularization parameter and smaller λ makes the covariance matrix $\sum_t$ closer to 0. So, a smaller λ means narrower prior and will lead to insufficient exploration and degrades the performance of CombLinTS. σ controls the decrease rate of covariance matrix $\sum_t$. So, a large σ will lead to slow learning and a smaller σ will make the algorithm quickly converge to some sub-optimal coefficient vector.

3.7.3 Regret Bound of Combinatorial Linear Thompson Sampling (CombLinTS)

If $\bar{w} = \Phi\theta^*$, the prior on θ^* is $N(0, \lambda^2 I)$, the noises $w_t(e) - \bar{w}(e)$are i.i.d sampled from $N(0, \sigma^2)$ then CombLinTS guarantees $R_{Bayes}(T) = \tilde{O}(K\lambda\sqrt{dT\,min\{ln(L), d\}})$. The conditions ensure it is a coherent gaussian case. The $\tilde{O}$ notation hides the logarithmic factors. The regret bound is a minimum of two bounds. The first bound is L-dependent as $O(\sqrt{ln(L)})$. The second bound is L-independent but is $\tilde{O}(d)$ instead of $\tilde{O}(\sqrt{d})$.

3.7.4 Combinatorial Linear UCB (CombLinUCB)



```
**Input:** Combinatorial structure $(E, \mathcal{A})$, generalization
matrix $\Phi \in \mathbb{R}^{L\times d}$, algorithm parameters $\lambda, \sigma, c > 0$,
oracle ORACLE
Initialize $\Sigma_1 \leftarrow \lambda^2 I \in \mathbb{R}^{d\times d}$ and $\bar{\theta}_1 = 0 \in \mathbb{R}^d$
for all $t = 1, 2, \ldots, n$ do
    Define the UCB weight vector $\hat{w}_t$ as
        $\hat{w}_t(e) = \langle \phi_e, \bar{\theta}_t \rangle + c\sqrt{\phi_e^T \Sigma_t \phi_e} \quad \forall e \in E$
    Compute $A^t \leftarrow$ ORACLE$(E, \mathcal{A}, \hat{w}_t)$
    Choose set $A^t$, and observe $w_t(e), \forall e \in A^t$
    Compute $\bar{\theta}_{t+1}$ and $\Sigma_{t+1}$ based on Algorithm 1
end for
```

Here also λ is the inverse regularization parameter. σ controls the rate of decrease of the covariance matrix. The constant c controls the degree of optimism. If c is too small the algorithm might converge to some sub-optimal coefficient vector due to insufficient exploration. On the other hand too large a c will lead to excessive exploration and slow learning.

3.7.5 Regret Bound of Combinatorial Linear UCB

Assuming P is a subset of $[0, 1]^L$,the stochastic P. Then for the coherent learning case $\bar{w} = \Phi\theta$ we have :

For any λ, σ and $\delta \in (0, 1)$ and any c satisfying $c \geq \frac{1}{\sigma}\sqrt{d\,ln\left(1 + \frac{nK\lambda^2}{d\sigma^2}\right) + 2\,ln\left(\frac{1}{\delta}\right)} + \frac{||\theta^*||_2}{\lambda}$

$$R_t \le 2cK\lambda\sqrt{\frac{dn\,ln\,(1+\frac{nK\lambda^2}{d\sigma^2})}{ln\,(1+\frac{\lambda^2}{\sigma^2})}} + nK\lambda$$

In particular setting $\lambda = \sigma = 1$, $\delta = \frac{1}{nK}$ and c as its upper bound we have $R_n = \widetilde{O}\,(Kd\sqrt{n})$

3.7.6 Comparison with State of the Art

Standard results have been set for the no generalisation case. Standard results have been set for the no generalisation case. As we already discussed for the semi-bandit case with no generalization we have a lower bound of $\Omega(\sqrt{LKT})$. The no generalisation condition is represented by $\Phi = I$. Since here $L = d$ the best result we can have is $\Omega(\sqrt{KdT})$.

For **CombLinTS** we are off by a factor of $\widetilde{O}(\sqrt{K\,min\,\{\,\ln(L)\,,\,d\}}\,.$
For **CombLinUCB** we are off by a factor of $\widetilde{O}(\sqrt{Kd})$.

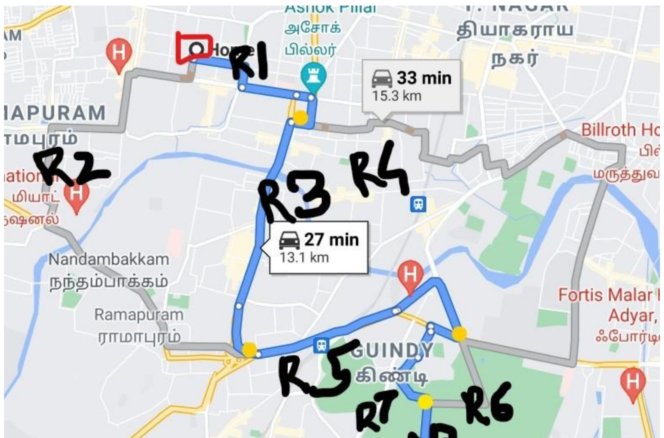
Although these algorithms are off by some factors from the state of the art, what we have achieved here is the L-Independent bound. For **CombLinUCB** we have achieved a pure L-Independent bound and in case of **CombLinTS** we get an L-Independent bound for very large L. Also, it has been established that $\Omega(\sqrt{K}\,)$ and $\Omega(\sqrt{d}\,)$ are due to linear generalization and we can get away with them.

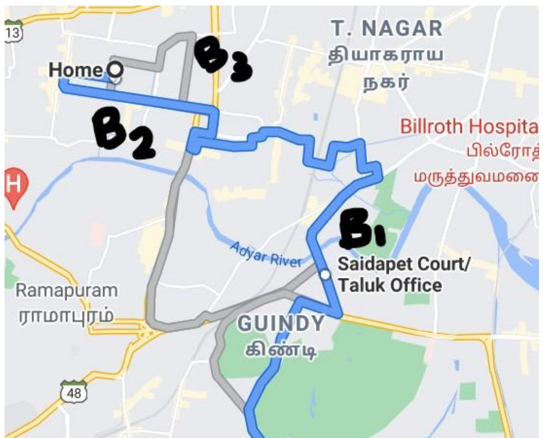
4. Experiments

Experiment 1: Shortest Path from Home to IITM

The experiment performed with the three algorithms follows a Least cost path problem setup. The graph includes S (Anirudh's home) as the starting point, and T (IITM) as the target node. The graph contains 11 edges (R-1 through R-8 + B-1 through B-3; R indicating roads, and B indicating a direct bus tour), and hence a path is represented as a 11-D binary vector. We consider 7 paths from S to T.

The attributes associated with each edge are estimated using Google Maps. A score associated with the length of an edge is taken by the estimated mean travel time, and the reliability score has been estimated using the standard deviation in the estimated time to travel over the edge. These are shown in the following table. Notice that a path is represented as a binary column vector in the table.
Regrets were averaged over 20 experiments (due to lack of sufficient computational power), each with 1000 time rounds.





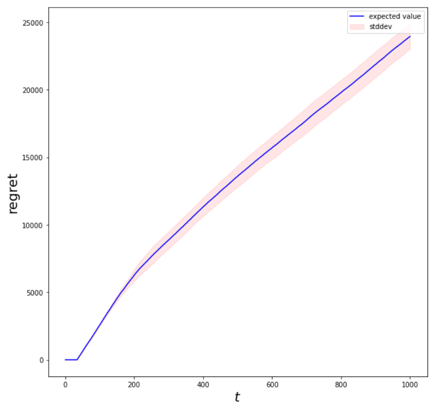
Path	Mean Time (min)	Std dev (min)	Allowed Combination						
			Route 0	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6
R1	5	1	1	1	0	0	0	0	0
R2	9	1.4	0	0	1	1	0	0	0
R3	4	1	0	1	0	0	0	0	0
R4	10	0.5	1	0	0	0	0	0	0
R5	4	1.5	0	1	1	1	0	0	0
R6	4	1	1	0	0	1	0	0	0
R7	4	1	0	1	1	0	0	0	0
R8	15	5	1	1	1	1	0	0	0
B1	69	10	0	0	0	0	1	0	0
B2	72	10	0	0	0	0	0	1	0
B3	77	10	0	0	0	0	0	0	1

Experiment 2: Effect of number of arms on algorithm

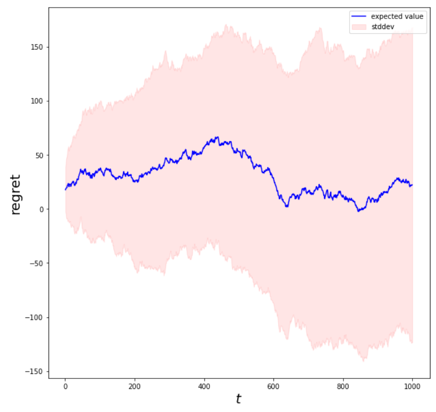
We created coherent learning cases randomly and varying the number of arms(L). We observed regret bound for cases L= 100 and L =1000, without varying anything else. To make sure the weights of arms are a linear combination of a set of feature vectors (coherent learning), we first generated the generalization matrix ϕ using gaussian randomness of deviation 5 around 0, an exact coefficient vector θ^* using standard uniform distribution between 0 and 1, and generated the average weight vector as $\bar{w} = \phi.\theta^*$.
Similar to the previous experiment, regrets were averaged over 20 experiments for L=1000 and 1000 experiments for L=100 (due to lack of sufficient computational power), each with 1000 time rounds.

5. Results

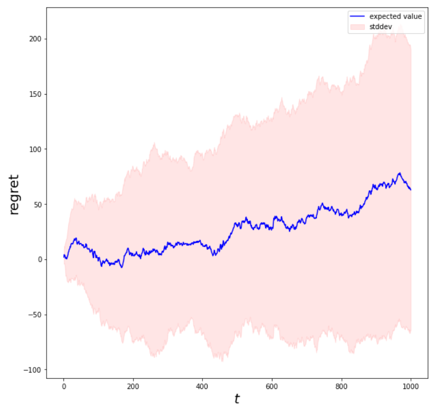
Results for Experiment 1:



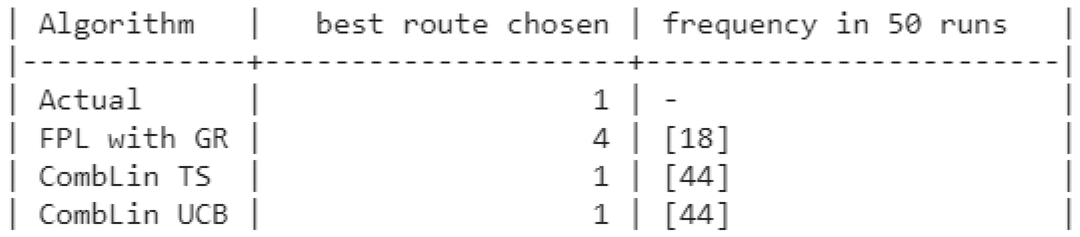
FPL with GR: Regret over trip to IIT



CombLinTS: Regret over trip to IIT



CombLinUCB: Regret over trip to IIT

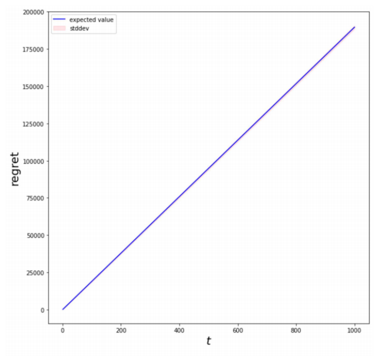


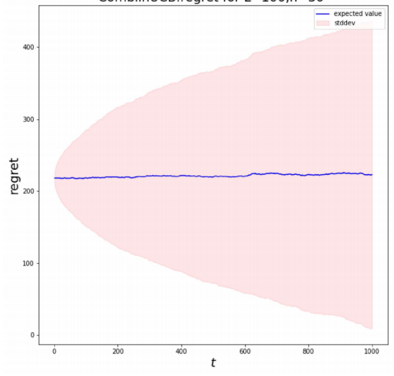
```
| Algorithm    |   best route chosen | frequency in 50 runs    |
|--------------+---------------------+-------------------------|
| Actual       |                   1 | -                       |
| FPL with GR  |                   4 | [18]                    |
| CombLin TS   |                   1 | [44]                    |
| CombLin UCB  |                   1 | [44]                    |
```

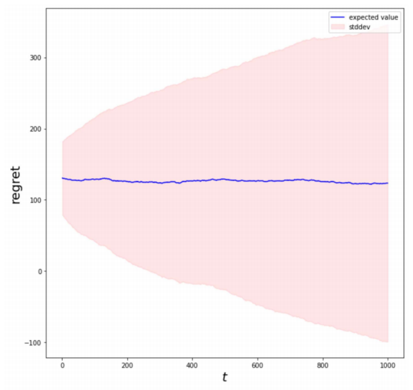
Except for the FPL algorithm, the rest gave an excellent regret bound and best route selection. Despite scrutinizing the algorithm in Paper 1 with the code implementation, we couldn’t explain this large regret bound and disappointing performance of FPL with geometric resampling.

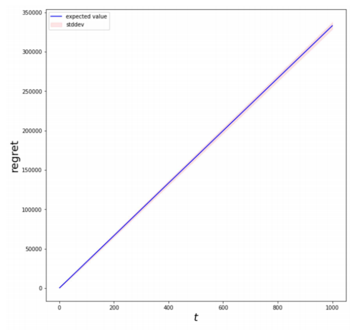
Results for Experiment 2:

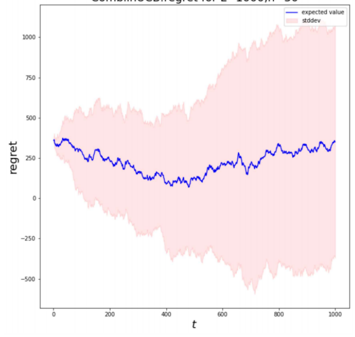
The following plot shows the variation of cumulative Regret with ‘t’ for different algorithms and ‘L’ values.

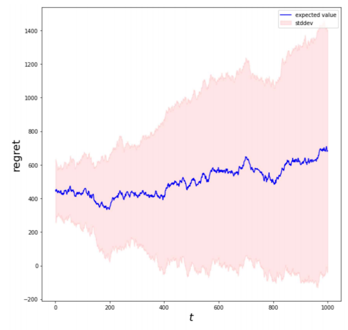

FPL with GR: regret for L=100,n=30

ComblinUCB:regret for L=100,n=30


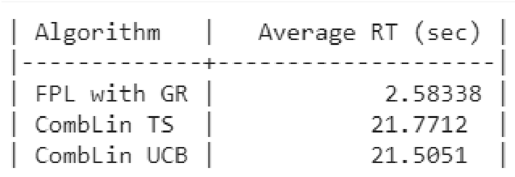

ComblinTS:regret for L=100,n=30


FPL with GR: regret for L=1000,n=30

ComblinUCB:regret for L=1000,n=30



ComblinTS:regret for L=1000,n=30

For d = 10, and L being varied from 100 to 1000, the mean regret bound has not changed much for CombLinUCB as expected (completely L independent).
But for CombLinTS, we observe a marginally significant increase in the regret bound. This is explained from our $\sqrt{(min(d, lnL)}$ factor in the regret bound. $ln100 = 4.6 < d$ and $ln\,1000 = 6.9 < d$. Hence we have the factor of regret bound $\sqrt{(ln\,L)}$ for both L =100 and 1000, and the fact that the bound established in the paper for CombLinTS has been left for further analysis.
As in experiment 1, the behaviour of the FPL has been unexplainable and disappointing. But we note that FPL with GR has been efficient in runtime. For the case of L=1000 here, the runtimes are as follows:


```
| Algorithm    |   Average RT (sec) |
|--------------+--------------------|
| FPL with GR  |            2.58338 |
| CombLin TS   |           21.7712  |
| CombLin UCB  |           21.5051  |
```

6. Conclusion

The first paper has been successful at providing the first general efficient algorithm for online combinatorial optimization under semi-bandit feedback. It has thrown two problems wide open, one is whether the gaps mentioned above can be closed for FPL style algorithms , other is how to develop efficient algorithms for combinatorial optimization under full bandit feedback.

The second paper has given two algorithms **CombLinTS** and **CombLinUCB** which are the first algorithms that have L-independent bounds in coherent linear generalization assumption setting which is highly useful for real life problems e.g Online advertisements(millions of users and products).
As potential directions for future work, the derivation of the regret bounds for the agnostic learning case, and how to extend the results to combinatorial semi bandits with non-linear generalization are left open.

The three algorithms in the mentioned papers have also been implemented.

7. References

1. Wen, Zheng, Branislav Kveton, and Azin Ashkan. "Efficient learning in large-scale combinatorial semi-bandits." International Conference on Machine Learning. PMLR, 2015.
2. Neu, Gergely, and Gábor Bartók. "An efficient algorithm for learning with semi-bandit feedback." International Conference on Algorithmic Learning Theory. Springer, Berlin, Heidelberg, 2013.

8. Contribution

All the members have contributed roughly equally with respect to reading of papers, discussion of ideas and implementation strategy, preparation of the slides and writing of the report. To specify the exclusive contributions, Abdul has had a careful look into the regret analysis parts, Adwait referred to external resources for understanding the intuition of Kalman Filtering, and Anirudh has taken the lead in implementation and dataset creation.