Network Science Project Report

Haoyu Zhao,2016012390

1 Introduction

1.1 Introduction to the Problem

In this project, we study a problem that is related to both online decision problem, or more specificly, the multi-armed bandit problem, and the information synchronization problem. The problem is useful in real life, especially in distrubuted computing and network field. For example, the distributed file storage system may need to synchronize the file data through the communication network, and in a multi-agent setting, each agent needs to transfer the information to coordinate the action. However, usually we do not know the exact cost of connecting between 2 nodes in a communication network before we actually choose to connect, and moreover, we may not know the distrubution of that cost, so we need to make decision and modify it online.

1.2 Basic Settings and Problem Formulation

In this section, we state our problem and the basic settings formally. Given a graph $G(V, E)$ as a communication network, each node $v \in V$ has it's own information I_v. Then we want to synchronize the information in the network, i.e. each node know the information of all other nodes. Each time the information goes from u to v, u can transfer all the information to v.

Our problem is the online repeated version of this information synchronization problem. We know the graph topology $G(V, E)$, but we do not know the weight of each edge. There is a random variable X_e which denote the weight of edge e and we do not know the distribution of X_e at first. Assume that we want to synchronize the information in many rounds, and in each round, the weight of each edge will not change. If we choose edge e in round t, then we can know the value of X_e in round t.

In this project, we focus on 2 different aspect of the problem. The first aspect is that we want to minimize the cost of the information synchronization process. Under this setting, the weight of each edge e in round t is the cost of this edge in round t, which is not affected by the number of times we use this edge to transfer the information in this round and by the amount of information we transfer during one usage of the edge. Formally, let $X_{e,t}$ denote the cost of edge e in round t, what we want to do is to select S_t in each round t, such that the expectation of the total cost

$$\mathbb{E} \sum_{e \in S_T} X_{e,t},$$

is minimized, with the constraint that the information synchronization can be completed successfully.

The other aspect is that we want to minimize the time of the information synchronization process. Under this setting, the weight of each edge e in round t is the time that information transfer from 1 endpoint of that edge to another. We also assume that the in each round, the time to transfer the information does not change. Formally, let $X_{e,t}$ denote the time needed by edge e in round t, what we want to do is to choose a sequence of edges that synchronize the information as fast as possible(allow transfer the information parallel on different edges).

1.3 Hardness of the Problem

1.3.1 Exploration vs. Exploitation

How to balance the exploration and exploitation is a general problem in multi-armed bandit problem. To get more accurate estimation of each random variable, we should observe that random variable as more time as possible. However, if we choose the random variable that is 'bad' ,i.e. it may cost a lot to choose that variable, we should not observe that random variable. So there is a seemingly contradiction between the exploration and exploitation, and we should decide the tradeoff between these 2 aspects.

1.3.2 Exponential Number of Feasible Action

Given a graph with n edges, there are 2^n number of possible actions in total, although some of them violate the constraints. If we just consider the spanning tree of a complete graph, the number of spanning trees is also exponential compared with the number of edges. So given the time horizon of our repeated online information synchronization problem, we may not try each feasible action once. So how to estimate the random variable is crucial.

1.3.3 Different Kinds of Measurements

Different kinds of measurements of the information synchronization also lead to a challenge of the problem. Because when we want to optimize the objective function under different measurements, such as cost, time or a combination of them, we will often lead to different algorithm and models. So it may hard to combine different kind of model under a general framework and get theoretical guarantee of the algorithm performance.

1.4 Related Works

Online learning and multi-armed bandit is an old topic which has much result in this field. Thompson [1933] first study the multi-armed bandit problem and Robbins [1985] first formulize the multi-armed bandit problem. Our problem is related to a branch of the problem named stochastic multi-armed bandit. The stochastic MAB is just the problem to study the tradeoff between exploration and exploitation, and there are many results toward this topic. To deal with the tradeoff between exploration and exploitation, Lai and Robbins [1985] introduced the famous upper confidence bound to balance the ratio between exploration and exploitation. Agrawal [1995] analysis the above strategy and get a result of $O(\log n)$ regret bound, and Auer et al. [2002] analysis the previous strategy with bounded random variable.

To deal with the problem of exponential number of arms in the multi-armed bandit, Chen et al. [2013] generalize the original stochastic MAB problem as the combinatorial multi-armed bandit problem. In that paper, the authors give an algorithm that generalize the upper-confidence bound strategy, called combinatorial upper-confidence bound to deal with the exponential number of bounds. Chen et al. [2014b] extend the combinatorial MAB problem into the probabilistic triggered arm case and Chen et al. [2014a] study the combinatorial pure exploration problem. However, these work make the assumptions that the random variables are bounded(mostly with support $[0, 1]$) and the cost function just depend on the mean of each random variable.

As for the information synchronization problem, we just use a number to represent the information, so it can be viewed as a combinatorial problem and our information synchronization model is much related to the packet routing problem. The most simple routing scheme is the shortest path routing, which is studied in Ahn and Ramakrishna [2002], Wang and Crowcroft [1992]. However, finding the all pair shortest path may use many edges and it may not be reasonable when the edges have cost. So in this project, we focus on another routing mechanism, called spanning tree routing. In the spanning tree routing, Huang et al. [2006] studies the minimum spanning tree scheme and Christofides et al. [1981] studies the centering problem and the relaxation of shortest path problem. These works enlight us to dive deeper into the spanning tree mechanism. Moreover, Waxman [1988] studied the multi-point connection routing, which is more related with our problem compared with other works.

2 Methods to Solve the Problem

2.1 General Methods

To tackle the problem of the unknown distribution of the edges, we apply the framework in multi-arm bandit. However, in the network information syncronization setting, the number of possible result will be exponentially large with respect to the number of nodes. Fortunately, Chen et al. [2013] gave a framework of combinatorial multi-armed bandit to deal with the problem of exponentially number of 'arms' when the random variables have support $[0, 1]$. We apply the algorithm in that framework, modify the algorithm to let it fit the unbounded case, which use the empirical mean as a parameter to adjust the 'exploration rate', and combine it with the network information synchronization problem. The algorithm is shown in **Algorithm 1**

Algorithm 1 Algorithm to solve the online information syncronization problem

Input: The graph structure(without weight) $G(V, E)$, and the algorithm $\mathcal{A}$ we want to run online.
Output: The action of each round

```
procedure ALG(G(V,E), 𝒜)
    μ̂_i ← the empirical expectation of edge i.
    T_i ← the number of time the edge i is used.
    Every time we use an edge i, we will update the empirical expectation μ̂_i and the counter T_i.
    for t = 1, 2, ..., |E| do
        Play an instance from all possible instances which contains the egde i.
    end for
    for t = |E| + 1, ... do
        μ̄_i ← μ̂_i − μ̂_i · sqrt(3 ln t / 2T_i).        ▷ Change the exploration rate by empirical mean.
        Play an instance which minimize the result of algorithm 𝒜 when each edge has weight μ̄_i.
    end for
end procedure
```

2.2 Minimize the Total Cost

In this section, we consider the setting when we want to minimize the cost of information synchronization in the network. Now suppose that the random variable X_i on edge i represents the cost of select edge i. Under this setting, we just consider the total cost but not consider the other measurements. To make the model simple, we assume that if we choose an edge, we will pay for the cost for setting up the connection, and the cost is independent of how much information we transfer or how many times we transfer the information.

This setting is reasonable in many real life cases when the information synchronization does not have constraints on time but the cost of communication is large, especially when the information synchronization does not happen so often and the cost of synchronization is high, for example synchronize the data between distributed databases. We have the following simple theorem under this setting, and we assume that the random variables that represent the cost on each edge are positive.

Theorem 2.1. *Under the senerio of minimizing the total cost while synchronizing the information, the edges that are chosen by the optimal strategy will form a spanning tree.*

Proof. This theorem is really simple. First because we want to synchronize the information for all nodes, the chosen edges will let the graph to be connected. Then because all of the edges have positive cost, then if there is a cycle it must be non-optimal. So the optimal strategy will choose a spanning tree of the graph. □

Now it is obvious that the best strategy tries to find the minimum expectation cost spanning tree, which is the spanning tree that minimize the total cost in the long term. We have the following algorithm, see **Algorithm 2**, which follows the framework **Algorithm 1**.

Algorithm 2 Algorithm to solve the problem under the min cost setting

Input: The graph structure(without weight) $G(V, E)$.

Output: The action of each round, where the edge selected in each round forms a spanning tree of all the nodes and then the algorithm tries to minimize the long term total cost of the spanning tree.

```
procedure Alg(G(V, E))
    μ̂_i ← the empirical expectation of edge i.
    T_i ← the number of time the edge i is used.
    Every time we use an edge i, we will update the empirical expectation μ̂_i and the counter T_i.
    for t = 1, 2, ..., |E| do
        Find an arbitrary spanning tree that contains edge t.
    end for
    for t = |E| + 1, ... do
        μ̄_i ← μ̂_i − μ̂_i · sqrt(3 ln t / (2T_i)).
        Find the minimum spanning tree with μ̄_i as the weight on edge i.
    end for
end procedure
```

Now the only problem is how to find the minimum spanning tree of a graph. The problem is simple and we can use Kruskal's algorithm. Then given a spanning tree of the graph, we can transfer the information through the edges of the tree and we can synchronize the infromation.

2.3 Minimize the Maximum Time

In this subsection, we consider the case when we want to minimize the time when we synchronize all the information. Here we assume that the weight of each edge $e = (u, v)$ represent the time that information goes from u to v or v to u.

2.3.1 A very simple case

Now we start from a simple case of our problem, we add an assumption about our problem.

Assumption: The cost of each edge is a constant, and we know the constant.

Under this setting, the fastest way to complete the information synchronization is that $\forall u, v \in V$, u send the information to v through the shortest path between u, v and v send the information to u along the shortest path from v to u. However, finding the union of the shortest path of all pair of points usually has $O(n^2)$ edges, and it is very wasting if every edge just transfer little information.

Here we use another method to transfer the information, which is an approxiamtion of the theoretical optimal solution, and use the minimum possible number of edges. We need the following definition.

Definition 2.1. (Center) Given a graph $G(V, E)$, we call a vertex $v \in V$ the center of graph G, such that

$$v = \arg\min_{u \in V} \{\max_{u' \in V} \mathrm{dist}(u, u')\},$$

where $\mathrm{dist}(u, u')$ denotes the shortest distance from u to u'.

Now given $G(V, E)$, denote $v \in V$ as the center of graph G, then denote T as the shortest path tree rooted at v. The shortest path tree only has $|V| - 1$ edges so it is really saving when we no not have so much cost. Moreover, we have the following theorem which shows that given the center and the shortest path tree rooted at the center, the strategy is not so bad.

Theorem 2.2. *If $v \in V$ is the center of graph $G(V, E)$ and T is the shortest path tree of G rooted at v, then consider the following strategy, all of the node transfer it's own information to v through tree T, and after* *collecting all the information, the node v transfer all of the information to each nodes. Then this strategy is a 2 approximation with respect to the time, i.e. it cost at most twice the time of the optimal strategy.*

Proof. The proof is really simple. Denote $d = \min_{u\in V}\{\max_{u'\in V} \operatorname{dist}(u, u')\}$, which is the maximum distance from the center v to any other nodes. Then denote $d' = \max_{u,u'} \operatorname{dist}(u, u')$, there dist denote the shortest path length between u, u'. Then we know that d is the time required by our strategy, and d' is the time required by the optimal strategy. Then we have

$$d' = \operatorname{dist}(u, u') \leq \operatorname{dist}(u, v) + \operatorname{dist}(v, u') \leq d + d = 2d.$$

□

Now given the theorem, we know that if we can find the center of the graph and then figure out the shortest path tree rooted at the center, then the solution is a good approximation of the optimal solution, and the number of edges is minimized.

Now the only problem is how to find the center and the shortest path tree. First we call the 'All Pair Shortest Path' algorithm to find the shortest distance between every 2 nodes, and we denote the distance matrix D, such that $d_{ij} = \operatorname{dist}(v_i, v_j)$. Then we find $c = \arg\min_i \max_j d_{ij}$, then c is the center of the graph. At last we call the shortest path algorithm to find the shortest path tree rooted at c.

We know that 'All Pair Shortest Path' problem can be solved in $O(n^3)$ by the Floyd algorithm, and the shortest path tree can be found by Dijkstra algorithm, and the total running time is $O(n^3)$.

2.3.2 A not so simple case

Now we consider a not so simple case, which we have the assumption that the each random variable is a constant but we do not know the constant. Then our solution is to call the algorithm framework (**Algorithm 1**), and turned it into the online version, see **Algorithm 3**.

Algorithm 3 Algorithm to solve the problem under the min time setting

Input: The graph structure(without weight) $G(V, E)$.

Output: The action of each round, where the edge selected in each round forms a spanning tree of all the nodes and then the algorithm tries to find the minimize the information transfer time in the long term.

procedure ALG($G(V, E)$)
  $\hat{\mu}_i \leftarrow$ the empirical expectation of edge i.
  $T_i \leftarrow$ the number of time the edge i is used.
  Every time we use an edge i, we will update the empirical expectation $\hat{\mu}_i$ and the counter T_i.
  for $t = 1, 2, \ldots, |E|$ **do**
    Find an arbitrary spanning tree that contains edge t.
  end for
  for $t = |E| + 1, \ldots$ **do**
    $\bar{\mu}_i \leftarrow \hat{\mu}_i - \hat{\mu}_i \cdot \sqrt{\frac{3 \ln t}{2T_i}}$.
    Find the center of the graph G and the corresponding shortest path tree rooted at the center with $\bar{\mu}_i$ as the weight on edge i.
  end for
end procedure

2.3.3 Another not so simple case

Now we consider the case that the distribution of each random variable on the edge is known. Similar as the previous discussion, we also want to find a 'center' of the graph and a spanning tree rooted at the center, such that the spanning tree minimize the largest distance from the center we choose. Formally, we want to find a spanning tree T rooted at v such that

$$\mathbb{E}[\max_{u,T} \text{dist}(v,u)],$$

is minimized, where the distance function dist denote the shortest path from v to u on the tree T.

Note that computing the exact spanning tree that minimize the objective function may be difficult, and we try an approximate solution. We compute the mean weight of each edge and we find the center and the corresponding shortest path tree rooted at the center with respect to the mean of each weight. Our solution is approximate because of the following fact of the objective functions,

$$\mathbb{E}[\max_{u,T} \text{dist}(v,u)] \geq \max_{u,T} \mathbb{E}[\text{dist}(v,u)],$$

since the max function is convex. Then the algorithm is the same as the most simple case, given the mean of each edge.

2.3.4 The general case

Then in the general case, we do not know the distribution of the random variables. We also use the approximate solution as shown in the previous case. Then combined with the online settings, we can thus reuse the algorithm when the random variables are constants, see **Algorithm 3**.

3 Simulation

3.1 Minimize the Total Cost

We test the result of our method on a complete graph with 20 nodes. We first test the situation when the weight random variable of each edge is a uniform random variable. We plot the total cost of our action, the best action, and the difference between them. The best action is the action that is optimal is the long term, i.e.$\mathbb{E}[\sum_{e\in T} x_e]$ is minimized. We also plot the average difference, as shown is **Figure 1**. We also simulate the case when the random variable of weight on each edge is $m_i \cdot x_i$ where m_i, x_i are standard Weibull distribution with shape parameter 0.8, and the results are shown in **Figure 2**.

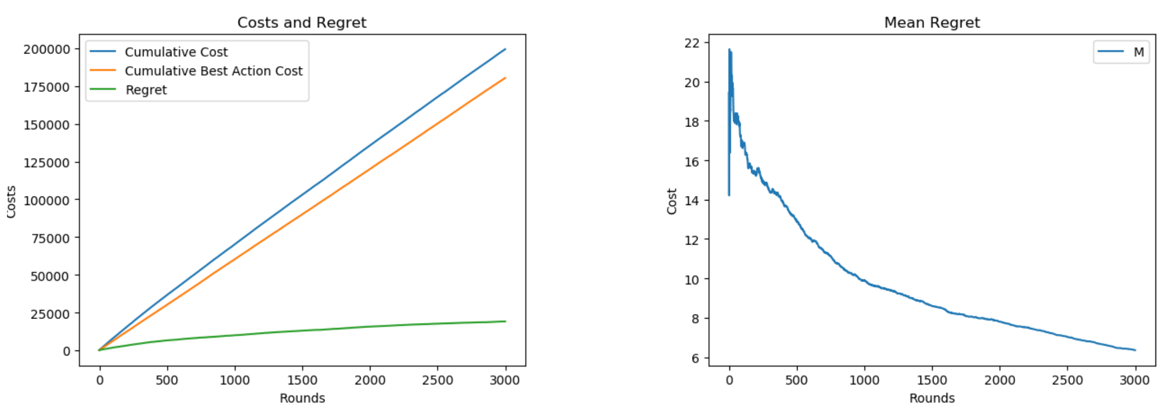

Figure 1: The simulation result on a complete graph with 20 nodes. The weight of each edge is a uniform random variable with support $[m_i - 3, m_i + 3]$, where m_i is chosen uniformly from $[3, 5]$. The left figure shows the cost of the best action, our action, and the difference between them(regret). The right figure shows the average regret through time.

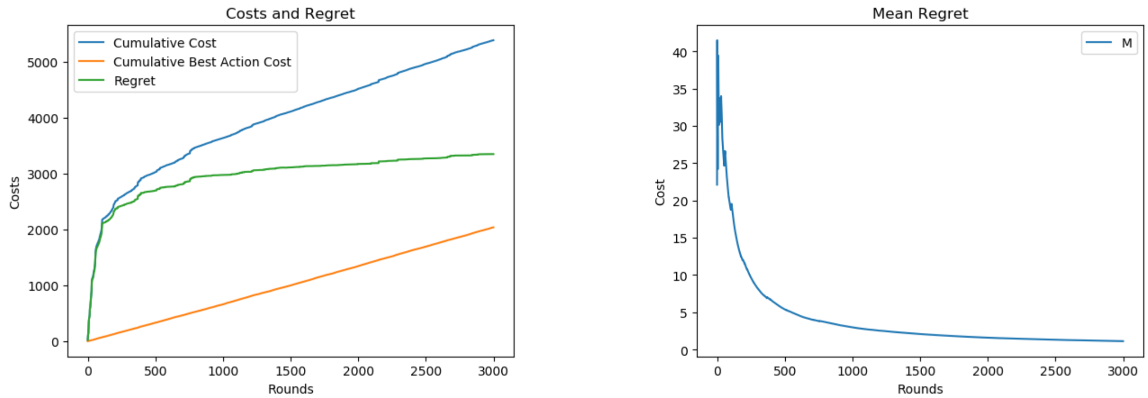

Figure 2: The simulation result on a complete graph with 20 nodes. The weight of each edge is random variable $m_i \cdot x_i$, where m_i, x_i are standard Weibull distrubution with parameter 0.8. The left figure and the right figure shows the same things as in the previous figure.

3.2 Minimize the Maximum Distance

3.2.1 The case when the time is constant but unknown

We test the result of our method on a complete graph with 20 nodes. We test the situation when the weight random variable of each edge is a uniform random variable, and the results are shown in **Figure 3**. We also simulate the case when the random variable of weight on each edge is $m_i \cdot x_i$ where m_i, x_i are standard Weibull distribution with shape parameter 2 and 0.7, and the results are shown in **Figure 4,Figure 5**.

We also compare our algorithm with the original algorithm in Chen et al. [2013] with Weibull distribution with parameter 2 and 0.7, which are showwn in **Figure ??**

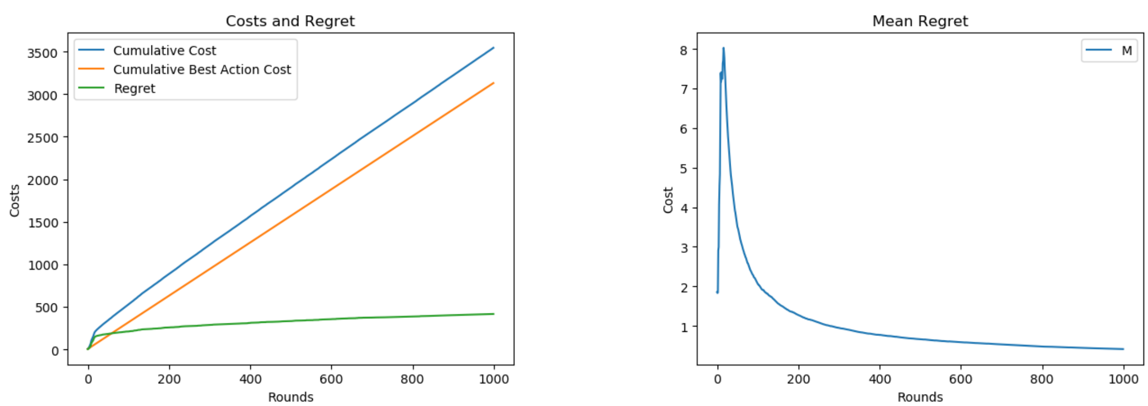

Figure 3: The simulation result on a complete graph with 20 nodes. The weight of each edge is a constant m_i, which is chosen uniformly from $[1, 5]$. The left figure and the right figure shows the same things as in the previous figure.

3.2.2 The general case

We test our algorithm on the general case. Because our algoithm just find the best solution in the long term approximately, so it is not reasonble to test the difference between our action and the approximate solution. Here we test on the 'cheated' best action, which knows the exact value of the random variables in each round and find the best solution on each round. The test results are shown in **Figure 8**.

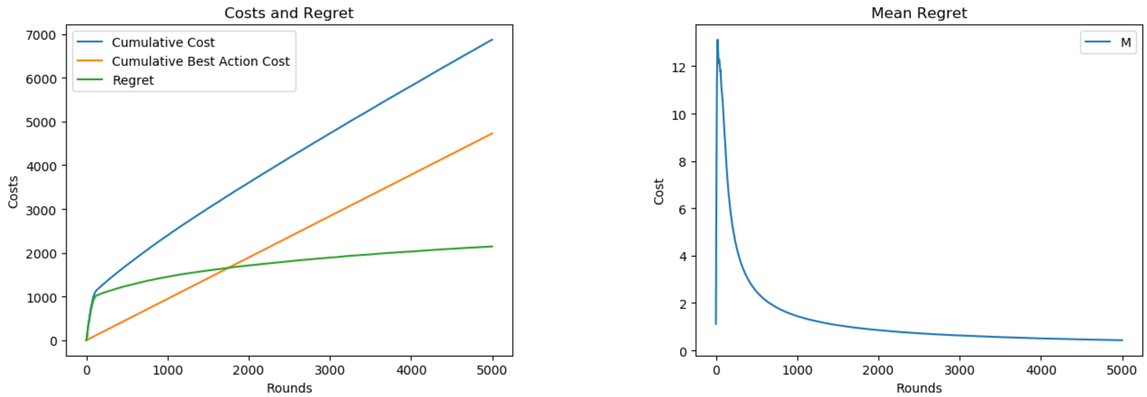

Figure 4: The simulation result on a complete graph with 20 nodes. The weight of each edge is random variable $m_i \cdot x_i$, where m_i, x_i are standard Weibull distrubution with parameter 2. The left figure and the right figure shows the same things as in the previous figure.

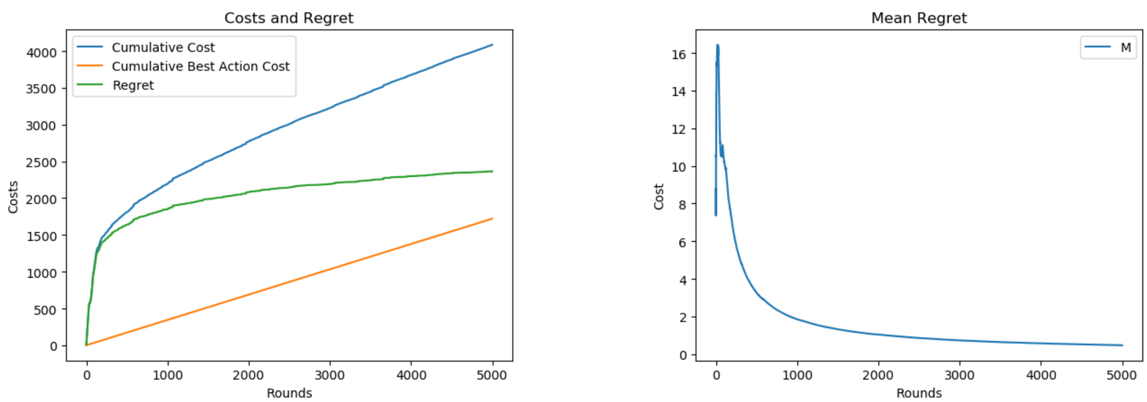

Figure 5: The simulation result on a complete graph with 20 nodes. The weight of each edge is random variable $m_i \cdot x_i$, where m_i, x_i are standard Weibull distrubution with parameter 0.7. The left figure and the right figure shows the same things as in the previous figure.

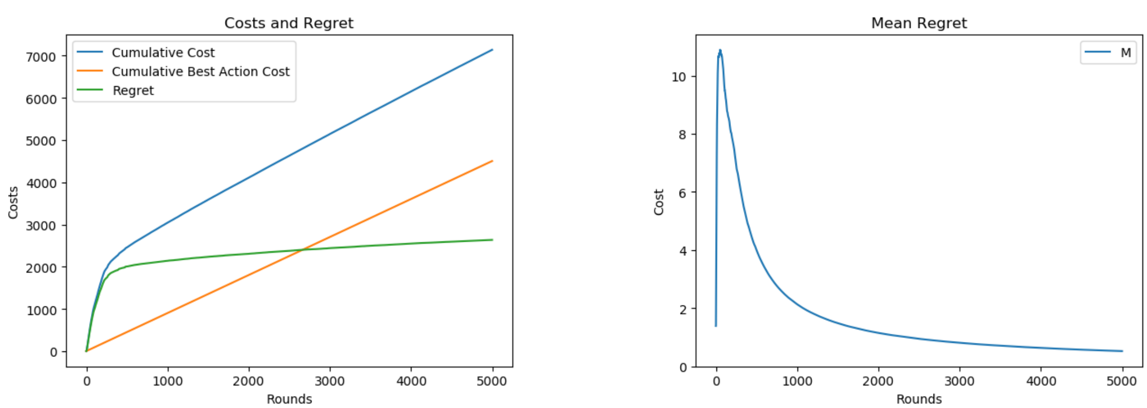

Figure 6: The simulation result on original algorithm by Chen et al. [2013] with Weibull 2 distribution.

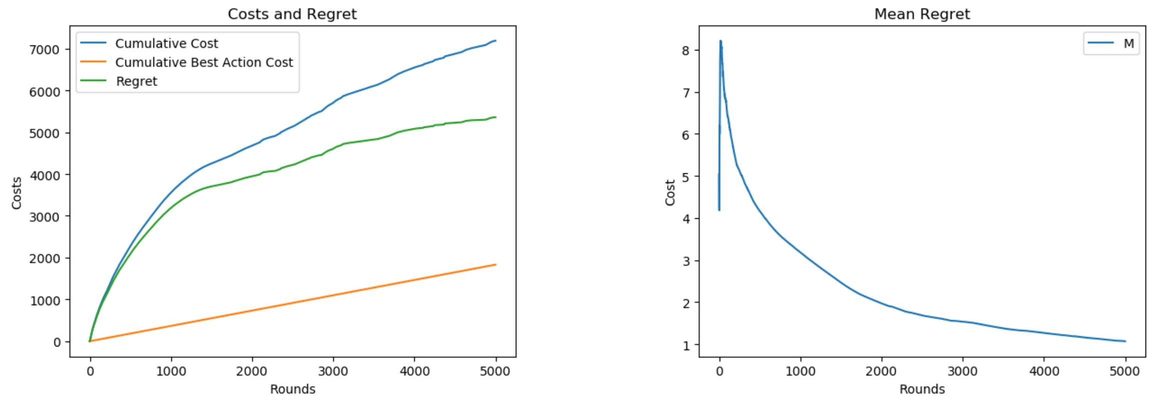

Figure 7: The simulation result on original algorithm by Chen et al. [2013] with Weibull 0.7 distribution.

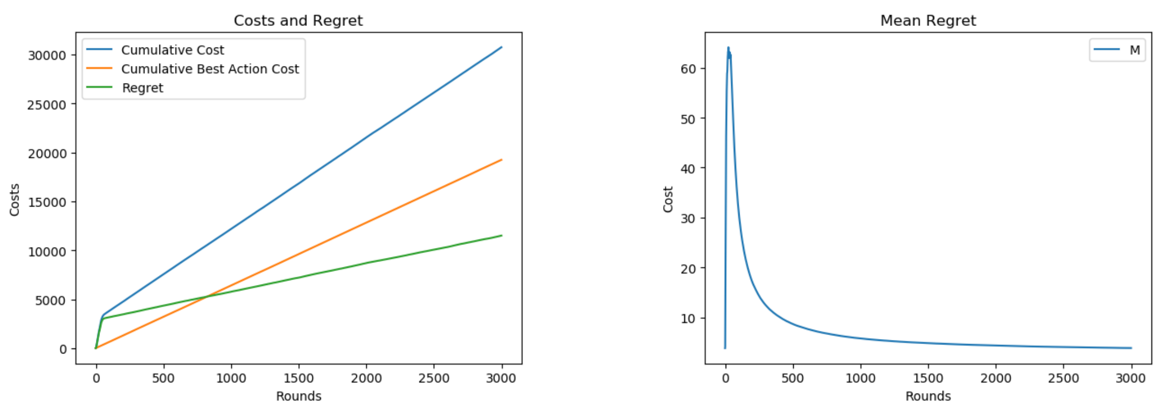

Figure 8: The simulation result on a complete graph with 20 nodes. The weight of each edge is a uniform random variable with support $[m_i - 3, m_i + 3]$, where m_i is chosen uniformly from $[4, 8]$. The left figure shows the cost of the best action which 'cheats' by find the best action each round, our action, and the difference between them. The right figure shows the average differnece through time.

3.3 Analysis and Further Discussion

First from the simulation results, we find that the algorithm will converge to the best action except for the general case. We think that this is due to the fact that the previous cases only cares about the mean of each edge, which we try to approximate in our algorithm. But in the general case when the solution is not only based on the mean of each random variable but also the distribution of each random variable, our method may not converge to the best action.

But the simulation results shows that our algorithm may also approximate the ‘cheating’ solution to some constant, which may depend on the graph topology and the distribution of the random variable. Then combine our previous argument that when the information transfer through the shortest path tree rooted at the center, the time is at most 2 times of the optimal strategy. So we can say that our strategy can solve this general case approximately.

Moreover, we can find that the speed of convergence are differnet. When the distrubution has larger variance or the random variable is long tail, the speed of convergence is slow.

From the comparision with the original framework in Chen et al. [2013], we find that our algorithm converge faster in the case when the distribution is long tailed or have larger variance. We conjecture that this is due to the fact that we use the empirical mean as a coefficient to adjust the rate of ‘exploration’, which will find the best action faster when the random variables have large variance.

4 Conclusion

In this report, we study the online information synchronization problem, with unknown weight of the communication graph. We modify the framework in Chen et al. [2013] and let it fit our problem settings. We study the problem from 2 different aspect, one from the minimum cost, and the other from the minimum time spent during the synchronization process. In the minimum cost version, we show that our method can tackle this problem successfully by simulation. However under the minimum time spent setting, our algorithm can successfully solve the problem when the random variables are constant and can only get a approximate solution of the general case.

Acknowledgement

The source code of the report and the simulation is stored at

`https://github.com/haoyuzhao123/network-science-project`,

which is private now and will be made public after the deadline of the project.

References

Rajeev Agrawal. Sample mean based index policies with o(log n) regret for the multi-armed bandit problem. 27, 12 1995.

Chang Wook Ahn and Rudrapatna S Ramakrishna. A genetic algorithm for shortest path routing problem and the sizing of populations. *IEEE transactions on evolutionary computation*, 6(6):566–579, 2002.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002. ISSN 0885-6125. doi: 10.1023/A:1013689704352. URL `https://doi.org/10.1023/A:1013689704352`.

Shouyuan Chen, Tian Lin, Irwin King, Michael R. Lyu, and Wei Chen. Combinatorial pure exploration of multi-armed bandits. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 379–387, Cambridge, MA, USA, 2014a. MIT Press. URL `http://dl.acm.org/citation.cfm?id=2968826.2968869`.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 151–159, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `http://proceedings.mlr.press/v28/chen13a.html`.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *CoRR*, abs/1407.8339, 2014b. URL `http://arxiv.org/abs/1407.8339`.

Nicos Christofides, Aristide Mingozzi, and Paolo Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282, 1981.

Guangyan Huang, Xiaowei Li, and Jing He. Dynamic minimal spanning tree routing protocol for large wireless sensor networks. In *Industrial Electronics and Applications, 2006 1ST IEEE Conference on*, pages 1–5. IEEE, 2006.

T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1): 4–22, March 1985. ISSN 0196-8858. doi: 10.1016/0196-8858(85)90002-8. URL `http://dx.doi.org/10.1016/0196-8858(85)90002-8`.

Herbert Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933. ISSN 00063444. URL `http://www.jstor.org/stable/2332286`.

Zheng Wang and Jon Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *ACM SIGCOMM Computer Communication Review*, 22(2):63–71, 1992.

Bernard M Waxman. Routing of multipoint connections. *IEEE journal on selected areas in communications*, 6(9):1617–1622, 1988.