Network Science Project Report

Haoyu Zhao,2016012390

Abstract

1 Introduction

1.1 Introduction to the Problem

1.2 Related Works

1.3 Basic Settings and Problem Formulation

1.4 Hardness of the Problem

1.4.1 Exploration vs. Exploitation

1.4.2 Exponential Number of Arms

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, [1] gave a framework of combinatorial multi-armed bandit to deal with the problem of exponentially number of 'arms'. We apply the framework, and the algorithm in that framework, 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($G(V, E), \mathcal{A}$)
    $\hat{\mu}_i \leftarrow$ 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**
        Play an instance from all possible instances which contains the egde i.
    end for
    for $t = |E| + 1, \ldots$ **do**
        $\bar{\mu}_i \leftarrow \hat{\mu}_i - \sqrt{\frac{3 \ln t}{2T_i}}$.
        Play an instance which minimize the result of algorithm $\mathcal{A}$ when each edge has weight $\bar{\mu}_i$.
    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. In this setting, we just consider the total cost but not consider the other measurements. 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. 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 all of the nodes 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 find the minimize the long term total cost of the spanning tree.

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**
  - $\bar{\mu}_i \leftarrow \hat{\mu}_i - \sqrt{\frac{3 \ln t}{2T_i}}$.
  - Find the minimum spanning tree with $\bar{\mu}_i$ 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 Distance

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 1: 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, and 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

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

where $\operatorname{dist}(u, u')$ 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

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

$\square$

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

2.3.3 Another not so simple case

2.3.4 The general case

3 Simulation

3.1 Minimize the Total Cost

3.2 Minimize the Maximum Distance

3.3 Analysis and Further Discussion

4 Conclusion

References

[1] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: general framework, results and applications," 2013.