Report for The Course of Wireless Communication: Community-based Task Allocation Optimization in Social Network

5141109021 Lu, Yifei 515030910109 Xue, Fei

May 27, 2018

1 Abstract

In this report, we propose a Crowdsourcing Social Network model and a corresponding heuristic algorithm for selecting proper set of workers in Social Network. The experiment is based on the dblp dataset. Yifei Lu mainly contributes to the problem discovery and model designing, and Fei Xue mainly contributes to algorithm designing and experiment. The study is collaborated with Gehua Qing and instructerd by Professor Xiaofeng Gao.

2 Model Introduction

In Crowdsourcing field, how to choose the best workers for tasks is always a hot problem. For example, David is a committee member of an international artificial intelligence conference, and he is in charge of the edition of proceeding. After the main conference, he intends to invite several celebrated scholars to write the preface. However, David cannot simply invite the most famous scholars, since some of them may not have co-author experiences with each other and are not willing to get involved. He has to invite a relatively small community of scholars, in which all of them not only enjoy great academic reputation but also are socially familiar, e.g. they had co-authored papers in artificial intelligence track. Therefore, David studies the scholar social community and wants to select a team of authors to satisfy his requirement. Previously, the above scenario leads to the study of Team Formation (TF).In this section, we first browse the structure of the Crowdsourcing Social Network. Then we introduce the definition of ability and communication cost of a team. At the end of the section, we give the detailed description of the problem.

2.1 Crowdsourcing Social Network

We describe the Crowdsourcing Social Network using a graph with heterogeneous weighted vertices and edges. As shown in Figure 1.

There are five vertices in Figure 1, and the number attaching the letter means its weight. The weight of each vertices represent the ability level. The

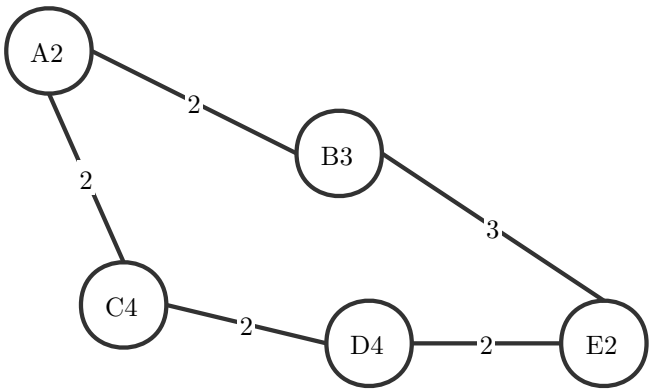

Figure 1: Crowdsourcing Social Network

weight of each edge represent the communication distance of two end points. For example, in Figure 1 C and D have the highest ability level, and B and E have more distant relationship than D and E.

2.2 Team Ability

If we choose some vertices as a team, the team ability means the sum of the weight of the whole vertices in the team.

Definition 1 *(Team Ability). For a team $T = \{v_1, v_2, \cdots, v_k\}$, the team ability (TA) means the sum of the weight of the whole vertices in the team:*

$$TA = \sum_{i=1}^{k} a_i,$$

for a_i is the ability level of vertices v_i.

2.3 Communication Cost

We describe the communication cost of a team using the notion of graph diameter. Given an original graph $G(V, E)$, we could get the shortest path of any two vertices in V. As for a subgraph $G'(V', E')$, the diameter D of G' means the largest shortest path of V', whose value has been calculated in G.

For example, in Figure 2, we choose A, C, D, E as a team, which can be represented by G'. In original graph G, the shortest communication distance of A and E is 5, of which the path passes B. Thus the diameter D of G' is 5, not 6(A-C-D-E), although B is not in G'.

2.4 Problem Definition

After introducing some requisite notions, we could give a detailed definition of the maximum ability bounded cost team formation problem (MATF).

Definition 2 *(MATF problem). Given a social graph with reputation for every vertex and weight for every edge, and a threshold for communication cost, find a*

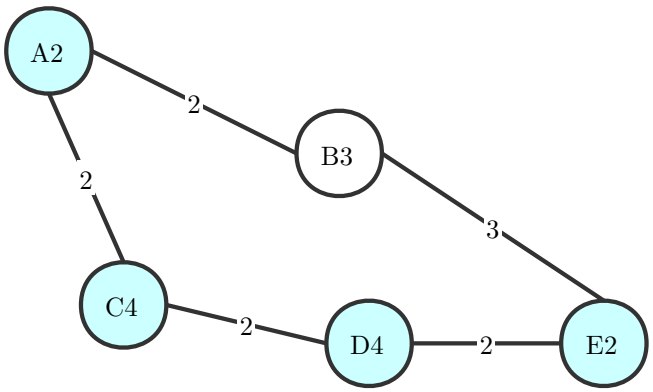

Figure 2: An Example of A Team

team of vertices which maximize the sum of ability while ensuring the diameter of selected team lower than threshold.

3 Algorithm Introduction

In this section, we introduce a heuristic algorithm to solve the MATF problem. The algorithm has three steps: the reconstruction of the graph, duplication of the graph and searching the solution.

3.1 Graph Reconstruction

Assume that the threshold of the communication cost is λ. We use Algorithm 1 to reconstruct the original graph: link any two vertices whose diameter is less than λ while delete the edge of others whose diameter is more than λ.

Algorithm 1: GRAPHRECONSTRUCTION

Input: The original graph $G(V, A, E, W)$, $|V| = n$, diameter matrix $\mathcal{C}_{n\times n}$ and threshold λ

```
Initialize a new graph G^λ(V, A, E^λ), E^λ = ∅
for v_i ∈ V do
    for v_j ∈ V\v_i do
        if c_ij ∈ C_{n×n}, c_ij ≤ λ then
            Add and edge to (v_i, v_j) to E^λ
return G^λ
```

After the reconstruction of the graph showing in Algorithm 1, there will be no edge between any two vertices whose diameter is more than λ, two of which can not both exist in our solution. Which means, the solution area limits in the

set of the complete subgraph of G^{λ}. Note that the edge of G^{λ} has no weight attribute.

3.2 Graph Duplication

After doing the Algorithm 1, we do not need to care about the weight of the edges any more. In this subsection, we duplicate the G^{λ} to eliminate the influence of the weight of vertices.

Algorithm 2: GRAPHDUPLICATION

Input: The reconstructed graph $G^{\lambda}(V, A, E^{\lambda})$

Initialize a new graph $G^{R_0\lambda}(V^R, E^R), V^R = E^R = \emptyset$
for $v_i \in V$ *that* $a_i > 1$ **do**
    $k = r_i$
    Duplicate the number of k new vertices, and each is weighted 1, denoted as $\mathcal{V}_i = \{v_i^1, v_i^2, \ldots, v_i^k\}$
    $V^R = V^R \cap \mathcal{V}_i$
for $v_i \in V$ *that* $a_i > 1$ **do**
    for $v_i^j \in \mathcal{V}_i$ **do**
        for $v_i^q \in \mathcal{V}_i \backslash v_i^j$ **do**
            Add an edge(v_i^j, v_i^q) to E^{λ}
    for $v_i^j \in \mathcal{V}_i$ **do**
        for $e_p \in E^{\lambda}$ **do**
            if e_p *connects* v_i *and any other vertex* v_{ep} **then**
                Add an edge(v_i^j, v_{ep}) to E^{λ}
return $G^{R_0\lambda}$

After the duplication showing in Algorithm 2, we split any v_i with weight a_i into a_i vertexes without weight and add edges to any two of them. An example is showed in Figure 3.

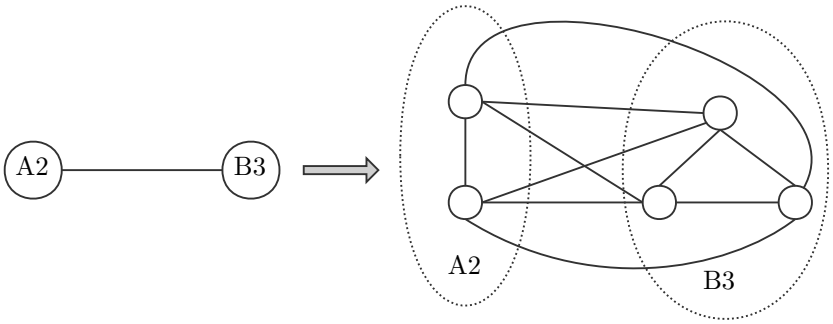

Figure 3: GRAPHDUPLICATION

3.3 Searching Solutions

The algorithm shown in Algorithm 3 works as the basic structure of our heuristic algorithm. First, we run GRAPHRECONSTRUCTION to setup a new graph. Then, we use GRAPHDUPLICATION to eliminate the influence from the weights of vertices. As a result , in the new graph, the weight of every vertex is one. So the problem will transfer to find a clique in graph which has maximum members. Then we randomly(or sequentially)select one vertex from V as a searching starting point. Then we construct a set of vertices named Cand, which is the potential candidate vertices that can be selected (initially, all these vertices must be adjacent to v, since we aim to find a clique containing v). Please notice that every vertex in Cand is fully connected to vertices in C, i.e. there is a complete bipartite matching graph between Cand and C. Next, when there are candidates in Cand, we select the best vertex(which can make upper bound of newly gained size of clique largest). Here comes the first prunning operation, since when we try to add u into our clique, the upper bound of newly gained size of clique is $|C|+|N(u)\cap Cand|+1$ at most (note that if we add u to our clique, then all the following added vertices must be adjacent to u and other vertices in the original Cand). This because after adding u, all the newly added vertices in Cand would also be added to C later. Thus, the clique would grow by at most $1+|N(u)\cap Cand\cap N(N(u)\cap Cand)\cap N(u)\cap Cand\cap\ldots|\leq 1+|N(u)\cap Cand|$.We can directly drop v if the upper bound of current growing clique is less than current best result. If we do not prune the growing clique, we can say that we get a bigger clique with size of $|C|+1$, and by the adjacent constraint we mentioned above, we have to update the candidate set by let $Cand = Cand\cap N(u)$. Thus, we get a new bipartite graph between Cand and C.The algorithm will stop when all the vertexes in V have been traversed.

Algorithm 3: SOLUTION SEARCHING

Input: The original graph $G(V, A, E, W)$ and threshold λ

$G^{\lambda} = \text{GraphReconstruction}(G(V, A, E, W), \lambda)$
$G^{R \circ \lambda} = \text{DuplicatedGraph}(G^{\lambda})$
$C^* = \emptyset$
for $v \in V$ **do**
  $C = C \cup \{v\}$
  $Cand = N(v)$
  while $Cand \neq \emptyset$ **do**
    $u = \arg\max_{u \in Cand} |N(u) \cap Cand|$
    if $|C| + |N(u) \cap Cand| + 1 \leq |C^*|$ **then**
      break
    $C = C \cup \{u\}$
    $Cand = Cand \backslash \{u\}$
    $Cand = Cand \cap N(u)$
  if $|C| > |C^*|$ **then**
    $C^* = C$
return C^*

4 Experiment

For this experiment, we use PC with i7-4720hq and 8GB DDR3 RAM in matlab platform and dblp database. In the following figures, lambda is the edge threshold, and time is the time cost for running the algorithm. Utility is the ability of the final solution.

From Figure 4, we can see that as lambda increase, the utility will also increase. This is because if the threshold is higher, more edge will be added to new graph. As a result, the solution and its utility will be larger.

From Figure 5, we can see that as lambda increase, the time will also increase. This is because if the threshold is higher, more edge will be added to new graph. As a result, the calculated amount will be larger. So the time cost increase.

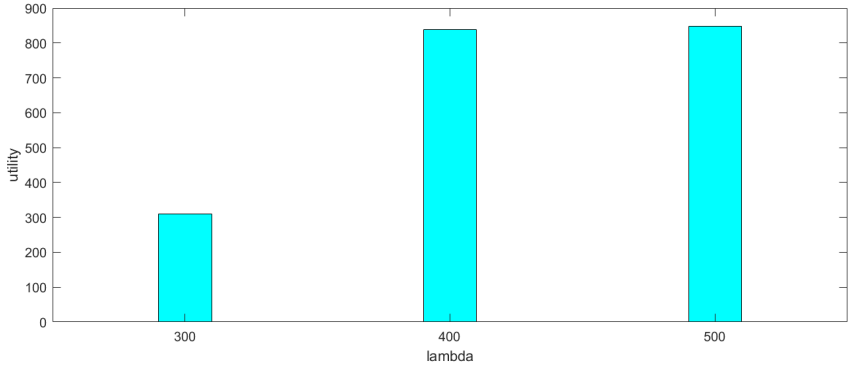

Figure 4: Utility Increases with Lambda

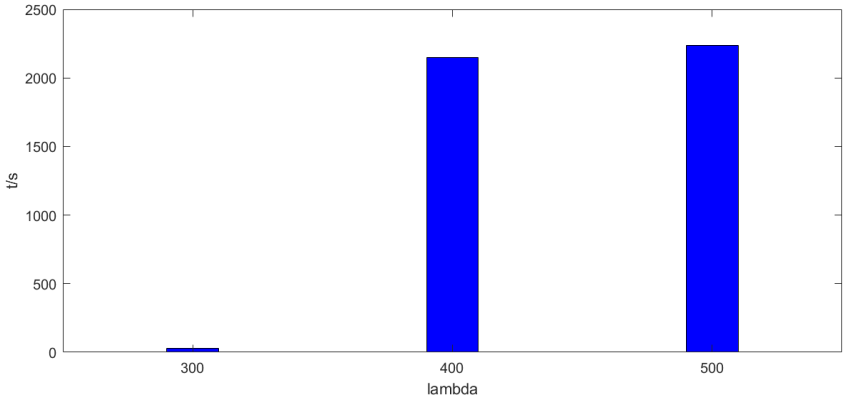

Figure 5: Time increases with Lambda