Algorithms in the Public-Private Model to Sub-Additive functions

Brian Brubach, Soheil Ehsani, Karthik A Sankararaman

University of Maryland
{bbrubach,ehsani,kabinav}@cs.umd.edu

Abstract

In this work we consider the Public-Private model of graphs introduced by Chierichetti et al [1]. In the first part, we briefly introduce the model and give the description of the shortest-path algorithm as described in their paper. In the second part we describe algorithms for Community Detection(Densest Subgraph problem), Max-Cut and Vertex Cover . We also briefly explain how the above algorithms can be extended to a small class of sub-additive functions. Finally, we give details of the experimental analysis of this algorithm run on real-world social network data.

I. The Public-Private Model

The *public-private* model was introduced in [1] to capture graph problems on social networks. In this model we have a known public graph $G = (V, E)$ and for each node $u \in V$ we have a hidden private graph $G_u = (V, E_u)$. For each $(v, w) \in E_u$ both v and w must be at most a distance of two from u in the union graph $G \cup G_u$. Without loss of generality, we can also assume $G \cap G_u = \emptyset$.

In social networks, nodes represent people/users and edges represent connections between them (e.g. friendship, shared group membership, etc.). Here we differentiate between edges which are known to the public, E, and those which are hidden by the user, E_u. For example, a Facebook user can hide their friend list while other users allow their friend lists to be publicly available. A 2012 study [2] showed that out of 1.4 million New York City Facebook users, 52.6% hid their friend list.

To see how this simple public-private model captures phenomena in social networks, we consider a few examples. If a user u hides their friend list, then the private graph G_u is a star centered at u. Similarly, if u is part of a private group, then G_u will include a clique containing all members of the group. Note that in some online social networks, such as Facebook, two users can be in the same group even if they are not "friends" and we put an edge between them since they can interact with each other in the group.

When solving problems in this graph model, we assume the public graph G is known in advance and a private graph G_u for some user u will be revealed. The solution we find should apply to the union graph $G \cup G_u$. Observe that it could be very time consuming to compute some function on the entire union graph each time a user reveals their public graph. So we wish to do some preprocessing once on the public graph to answer queries efficiently when private graphs are revealed. Specifically, our goal is to preprocess the public graph G using $poly(|E|)$ time and $\tilde{O}(|V|)$ space. Then, when G_u is revealed, we should answer queries in time/space that is $\tilde{O}(|E_u|)$ and $poly(\lg |V|)$.

II. All Pairs Shortest Path

In this section, we will describe the algorithm for the *All Pairs Shortest Path*(APSP) problem in the Public-Private model. APSP is an important problem in social networks since many learning

algorithms over graphs use it as a feature. For example, a common feature that is often used in this context is the likelihood of a person A following celebrity B. This can be interpreted well by the distance on the social network from other celebrities in the related area. In [1], they give an efficient approximation algorithm to this problem using sampling techniques. We would like to remark that, even though it is possible to obtain an exact solution to this problem in $O(n^3)$ it is too slow for large social networks.

The central idea of the algorithm is to use the sampling based algorithm due to Das Sharma *et al* [3]. In this algorithm, they first generate $O(\log^2 n)$ random subsets. They use this sample to estimate distance between any two vertices. in the following subsection, we will give a brief description of their algorithm.

I. A $O(\log n)$ approximation to the APSP in Offline Graphs

In Algorithm 1, we give the details of the algorithm. The running time of the algorithm is $O(m \log^2 n)$ because of the following observations.

Observation II.1. *For a given set S_i, we can compute the element u_i corresponding to vertex u for all the vertices in the graph by performing one BFS.*

The observation is true because, we can connect all the vertices in S_i to a new vertex d_i and start a BFS from this vertex. Since, per set we need $O(m)$ time, we have $O(\log n)$ subsets samples $O(\log n)$ times, giving a total running time of $O(m \log^2 n)$.

Algorithm 1 A $O(\log n)$ approximation to the APSP in Offline Graphs

▷ Sample with replacement $r = \lfloor \log n \rfloor + 1$ subsets of vertices $S_0, S_1, \ldots, S_r$, such that the subset S_i contains 2^i vertices
▷ For any vertex u, let $SKETCH(u)$ denote the vertices $u_0, u_1, \ldots, u_r$ such that the closest vertex from u to set S_i is u_i
▷ Repeat the above process of generating $r + 1$ subsets $O(\log n)$ times independently. For each iteration, append the $SKETCH(u)$ to the $SKETCH(u)$ from the previous iterations. Hence, at the end, the size of $SKETCH(u)$ is of size $O(r \log n)$
▷ Define $COMMONSKETCH(u, v) = SKETCH(u) \cap SKETCH(v)$. Compute $dist(u, v)$ by the following expression: $\min\{dist(u, w) + dist(w, v) | w \in COMMONSKETCH(u, v)\}$

To prove the correctness of the above algorithm, we prove the following theorem.

Theorem II.1. *The estimated distance $\overline{d}(u, v)$ given by Algorithm 1 is a $O(\log n)$ approximation to $d(u, v)$ with high probability*

Proof. (sketch) Let d denote the optimal value of dist(u, v). We will now show that with high probability there exists a $w \in COMMONSKETCH(u, v)$ such that w is at a distance of at most $d \log n$ from both u and v. Hence, this gives that $\overline{d}(u, v)$ is a $O(\log n)$ approximation to $d(u, v)$.

Let U_r denote the set of points that are at a distance of $r * d$ from vertex u. Similarly, let V_r denote the set of points that are at a distance of $r * d$ from v. Firstly, we can make the following observation

Observation II.2. $U_r \cup V_r \subseteq U_{r+1} \cap V_{r+1}$.

Proof. This is because for a point in $U_r \cup V_r$, the distance from either u or v is at most $r * d + d$ which is $(r+1) * d$. □

Suppose there is a set S_i such that there is exactly one vertex z such that, $z \in U_r \cup V_r$ and $z \in U_r \cap V_r$ then clearly z is in $COMMONSKETCH(u, v)$. Now we will show that with high probability such a set S_i exists.

Consider the quantity $\frac{|U_r \cap V_r|}{|U_r \cup V_r|}$. We will consider the following two cases

- For some r in range $1, 2, \ldots, \log n$, $\frac{|U_r \cap V_r|}{|U_r \cup V_r|} > \frac{1}{2}$. Now, consider the vertex $z \in U_r \cup V_r$ which satisfies the property that it is the only vertex from $U_r \cup V_r$ which is in $U_r \cap V_r$. To this end, define the event E to be z is the only vertex in $U_r \cup V_r$ which is also in $U_r \cup V_r$. Hence, the probability of event E happening is atleast $\frac{1}{2e}$.
- For all r in range $1, 2, \ldots, \log n$, $\frac{|U_r \cap V_r|}{|U_r \cup V_r|} \leq \frac{1}{2}$: From Observation above, we know that $U_r \cup V_r \subseteq U_{r+1} \cap V_{r+1}$. Hence, $\frac{|U_r \cap V_r|}{|U_r \cup V_r|} \leq \frac{1}{2}$ implies that for all r in range $1, 2, \ldots, \log n$ $|U_r \cup V_r| > 2|U_{r-1} \cup V_{r-1}|$. Since $|U_0 \cup V_0| = 2$ we have $|U_{\log n} \cup V_{\log n}| > n$ which is a contradiction.

Since we know that for a single sampling of sets $S_0, S_1, \ldots, S_r$ with constant probability there exists a vertex in the $COMMONSKETCH(u, v)$ which is at a distance at most $d \log n$ from both u and v, using Chernoff bounds we can amplify this probability to a negligibly close quantity to 1 with $\log n$ independent trials. □

II. Extending the above Offline Algorithm to the Public-Private model

In the Public-Private model, we rephrase the APSP question as follows. Given a public graph and a private query (u, G_u) we want to give output to dist(u, *) for all vertices * in the graph. The algorithm first computes the distances on the public graph using the offline algorithm. When a query arrives, it considers the following three cases and returns the minimum of them.

- $dist_{G \cup G_u}(u, v)$ does not need to use any edges from the private graph. In this case, we do not need to do any computation.
- $dist_{G \cup G_u}(u, v)$ uses exactly one edge from the private graph. In this case we have $dist_{G \cup G_u}(u, v) = 1 + dist_G(w, v)$ where $w \in N(u)$.
- $dist_{G \cup G_u}(u, v)$ uses exactly two edges from the private graph. In this case we have $dist_{G \cup G_u}(u, v) = 2 + dist_G(w, v)$ where $w \in N(N(u))$.

Since we have the assumption that every vertex in the private graph is at a distance of atmost 2 from u, the above cases is exhaustive. The proof of correctness follows immediately from the proof of correctness of the offline algorithm.

Note that at the end of offline computation, we only need to store the $SKETCH(q)$ for every vertex $q \in V$. For an online query, we need $O(|E_u| \log^2 n)$ time for each of the cases two and three above. Hence, the total running time in the Online phase is $O(|E_u| \log^2 n)$.

References

[1] Flavio Chierichetti, Alessandro Epasto, Ravi Kumar, Silvio Lattanzi, and Vahab Mirrokni. Efficient algorithms for public-private social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 139–148. ACM, 2015.

[2] Ratan Dey, Zubin Jelveh, and Keith W. Ross. Facebook users have become much more private: A large-scale study. In *PerCom Workshops*, pages 346–352. IEEE Computer Society, 2012.

[3] Atish Das Sarma, Sreenivas Gollapudi, Marc Najork, and Rina Panigrahy. A sketch-based distance oracle for web-scale graphs. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 401–410, 2010.