Identifying Top-k Structural Hole Spanners in Large-Scale Social Networks

Mojtaba Rezvani† Weifa Liang† Wenzheng Xu†‡ Chengfei Liu¶

† Australian National University, Canberra, ACT 0200, Australia
‡ Sichuan University, Chengdu, P. R. China
¶ Swinburne University of Technology, Melbourne, VIC, Australia
mojtaba.rezvani@anu.edu.au, wliang@cs.anu.edu.au, wenzheng.xu3@gmail.com, cliu@swin.edu.au

ABSTRACT

Recent studies have shown that in social networks, users who bridge different communities, known as structural hole spanners, have great potentials to acquire available resources from these communities and gain access to multiple sources of information flow. Structural hole spanners are crucial in many applications such as community detections, diffusion controls, and viral marketing. In spite of their importance, not much attention has been paid to them. Particularly, how to characterize the structural hole spanner properties and how to devise efficient yet scalable algorithms to find them are fundamental issues. In this paper, we formulate the problem as the top-k structural hole spanner problem. Specifically, we first provide a generic model to measure the quality of structural hole spanners, by exploring their properties, and show that the problem is NP-hard. We then devise efficient and scalable algorithms, by exploiting the bounded inverse closeness centralities of vertices and making use of articulation points of the network. We finally evaluate the performance of the proposed algorithms through extensive experiments on real and synthetic datasets, and validate the effectiveness of the proposed model. Our experimental results demonstrate that the proposed model can capture the characteristics of structural hole spanners accurately, and the proposed algorithms are very promising.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications; J.4 [**Social and Behavioral Sciences**]: Miscellaneous

Keywords

Social networks, Top-k structural hole spanners, Linear-time algorithms, Inverse closeness centrality, Articulation points

1. INTRODUCTION

The last decade experienced an exponential growth of a variety of large-scale networks such as social networks, citation networks, biological networks, wireless networks, etc. Thus, there are high demands for developing efficient algorithms to explore some unique properties of such networks. Most social networks exhibit the so-called community structure property, that is, the vertices in a network can be grouped into different sets of cohesive groups (communities) [10], where vertices in the same community share similar attributes. The communities play a significant role in information diffusion within the network, information within a community circulates very quickly and diffuses to other communities through community boundaries or bridges. On the other hand, there is a consensus among social scientists [7] that a person who plays a bridge role between different communities can acquire more potential resources from these communities and has more control over the information that is being transmitted. Burt [7] studied social structures of many organizations and introduced the notion of *structural holes* as positions that can bridge diverse groups and bring benefits to the beholder. It is shown that information within a single community tends to be homogeneous. Non-redundant information is often obtained through the contacts between different communities [23]. Therefore, a person who develops relations with people from multiple communities will gain more benefits. *Structural hole spanners* were studied initially by Lou *et al.* [19], as a few people who fill the structural holes can bridge different communities. For example, a community in an academic collaboration network represents the group of people with the similar research interests, and people (structural hole spanners) who bridge different communities are more potent to combine ideas from different research groups and create interdisciplinary works. Structural hole spanners have a wide range of applications. For example, in community detection, identifying central hubs that connect different groups can help isolate and identify communities [2, 29]. In Epidemic diseases and rumors spreading, quarantining structural hole spanners can stop the spread of infection and rumors into other communities [6, 12, 20]. In viral marketing, the most influential structural hole spanners can speed-up the new product marketings to different groups [15, 26, 25, 31].

Since structural hole spanners are fundamental in many applications, several models have been proposed for it [11, 17, 19, 24]. For example, Lou *et al.* [19] introduced a model for structural hole spanners, and proposed two algorithms based on the model, by assuming that communities are given already. However, their work relies on communities while finding communities in a large-scale network is painstaking. Moreover, the quality of the solution delivered by their algo-

CIKM'15, October 19–23, 2015, Melbourne, Australia.
ACM 978-1-4503-3794-6/15/10.
DOI: http://dx.doi.org/10.1145/2806416.2806431.

rithm is determined by the chosen communities. There are other studies that aim to discover the structural hole spanners from a social network, using the topological structure of the network. Goyal *et al.* [11] considered a structural hole spanner as a vertex that lies on a large number of shortest paths, which is similar to betweenness centrality. Tang *et al.* [24] formulated structural hole spanners as the vertices which lie on a large number of shortest paths of length two only. These models however failed to capture some essential properties of structural hole spanners, which is illustrated by Fig. 1. As can be seen, vertex v_1, rather than vertex v_2, lies on a large number of shortest paths of length two, vertex v_2, instead of vertex v_1 is a better structural hole spanner as it bridges more communities.

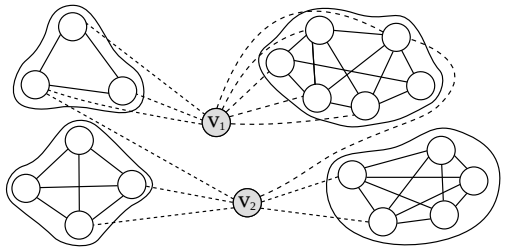


Figure 1: Illustration of structural hole spanners; each closed area represents a community, and vertices v_1, v_2 represent structural hole spanners that span multiple communities.

One of the implications of structural hole spanners is that they bridge different communities and the shortest paths between those communities go through them. Therefore, their removal will increase the length of shortest path between other vertices. For example, vertex v_1 in Fig. 1 plays a key role in the shortest paths between the nodes in different communities and its removal can significantly change the length between the other nodes, while the impact of the removal of other vertices on shortest paths is insignificant. In this paper, we propose a model based on the *mean distance* of the network [5] for modeling the structural hole spanners, which is the average of the lengths of all pairs of vertices in the network. We consider the structural hole spanners problem as a set of vertices whose removal will result in the maximum increase on the mean distance of the network, and we term *the top-k structural holes problem* as the problem of finding a set of k vertices whose removal will make the increase on the mean distance maximized. To the best of our knowledge, this is the first time that a novel, top-k structural hole spanner problem is formulated and its NP-hardness is proven. Unlike most existing works that assume that either all communities are given or rely on community detection algorithms to find them first, our model relies on the network topological structure only.

The main contributions of this paper are as follows. We study the top-k structural hole spanner problem in a large-scale social network. We first formulate the problem as an optimization problem and show its NP-hardness. We then devise two efficient, yet scalable algorithms, by exploiting the small-world phenomenon and using the bounded inverse of closeness centrality of vertices and using articulation points in the network. We finally evaluate the performance of the proposed algorithms by extensive experiments on real and synthetic datasets. Experimental results show that the structural hole spanners delivered by the proposed algorithms can connect more and larger communities in comparison with that by other existing methods in real datasets. Moreover, using a synthetic datasets, we show that the proposed algorithms can accurately find the structural hole spanners. Furthermore, our evaluations show that the proposed algorithms outperform the other heuristics in terms of accuracy and running time.

The rest of this paper is organized as follows. Section 2 introduces basic notations, and the problem definition. Section 3 shows the NP-hardness of the problem. Section 4 proposes algorithms for the problem. Section 5 evaluates the performance of the proposed algorithms, using real and synthetic datasets. Section 6 reviews related works on structural hole spanners, and Section 7 concludes the paper.

2. PRELIMINARIES AND DEFINITIONS

2.1 Network Model

A social network can be modeled as an undirected connected graph $G = (V, E)$, where V is the set of vertices representing individuals and E is the set of edges representing the relationships between individuals. Let $n = |V|$ and $m = |E|$. The degree of a vertex v is the number of its neighbors, denoted by $\deg(v)$. Maximum degree of vertices in G is denoted by $\Delta(G)$.

Given two vertices $u, v \in V$, the *vertex connectivity* $\kappa^G(u, v)$ in G between them is the minimum number of vertex-disjoint paths. Vertices u and v are referred to as *k-vertex-connected* if they are still connected after the removal of no more than k vertices from G. A graph G is *k-vertex-connected* if any pair of vertices in it is k-vertex-connected. A vertex is an *articulation point* of G if its removal will disconnect the graph.

As G is an unweighted graph, we assume that each edge has a weight of 1, and we term each edge $e \in E$ as a *real edge*. The distance d^G_{uv} between two vertices u and v in G is the length of the shortest path between them. We assume that $d^G_{vv} = 0$ for any vertex $v \in V$. Given a subset V_S of V, let $G[V \setminus V_S]$ be the induced subgraph of G by the vertices in $V \setminus V_S$. We abbreviate $G[V \setminus V_S]$ by $G \setminus V_S$. The *inverse closeness centrality* of a vertex v in G is the average distance between vertex v and other vertices [5], i.e.,

$$c(v) = \frac{\sum_{u \in V} d^G_{uv}}{n-1}. \tag{1}$$

The mean distance of a graph G thus is defined as follows.

$$c(G) = \frac{\sum_{v \in V} c(v)}{|V|} = \frac{\sum_{v \in V} \sum_{u \in V} d^G_{uv}}{(n-1)|V|} = \frac{\sum_{v \in V} \sum_{u \in V} d^G_{uv}}{n(n-1)}. \tag{2}$$

The sum of lengths of all pairs shortest paths in G is

$$C(G) = \sum_{u \in V} \sum_{v \in V} d^G_{uv} = n(n-1)c(G). \tag{3}$$

Note that if G is disconnected, to make the mean distance of G still be valid, the distance between two vertices not in the same connected component is defined by a sufficiently large value ζ to avoid the infinite distance. This value should be larger than the sum of lengths of all pairs of shortest paths in any connected component of G, e.g., $\zeta = n^3$, as the upper bound on the sum of lengths of all pairs shortest paths in a n-vertex graph is no more than $\zeta/3$ [22].

2.2 Problem Definition

Given a social network $G = (V, E)$ and a positive integer k, the top-k *structural hole spanner problem* in G is to find a subset of vertices V_S ($V_S \subset V$) with $|V_S| = k$, such that the removal of the vertices in V_S from G will result in the maximum increase on the sum of the lengths of all pairs of shortest paths among the vertices in the induced subgraph $G \setminus V_S$, i.e., the problem objective is to

$$\max_{V_S \subset V,\ |V_S|=k} \{C(G \setminus V_S) - C(G)\}, \tag{4}$$

which is equivalent to

$$\max_{V_S \subset V,\ |V_S|=k} \{C(G \setminus V_S)\}. \tag{5}$$

Since communities are dense, the distance between vertices within each community is small and the member removal does not change the distance between the other members considerably. In contrast, the removal of top-k structural hole spanners in a network will result in the maximum number of communities disconnected in comparison with other k-vertex removals, thereby significantly increasing the mean distance of network. Fig. 1 illustrates the impact of removal of structural hole spanners on the distances. Specifically, the proposed model captures three important characteristics of structural hole spanners.

1. Given an individual u who bridges multiple communities and another individual v who contacts with people only in his/her community, individual u is considered by the model to be a better structural hole spanner than individual v, since the connections among individuals within the communities to which individual v belongs are strong, and the absence of v only slightly increases the distance among other individuals in the network. In contrast, individual u connects people who are in different communities, thus the absence of u can dramatically increase the distance between them, as they are loosely connected.

2. Given an individual u who bridges large communities and an individual v who bridges small communities, individual u is considered to be a better structural hole spanner than individual v, since the removal of u will disconnect more people in the network.

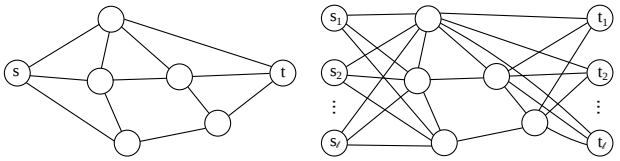
3. Given an individual u who bridges many communities and an individual v who bridges only a few, individual u is considered to be a better structural hole spanner, since the removal of u can increase the distance between more communities (even if they are smaller).

3. NP-HARDNESS

In this section we show that the top-k structural hole spanner problem is NP-hard by a reduction from an NP-hard problem - the *Most Vital Node Problem* (MVNP) [4], defined as follows. Given an undirected graph $G = (V \cup \{s, t\}, E)$, a pair of nodes s and t, and a positive integer k, assume that there is no edge in G connecting vertices s and t and the vertex connectivity $\kappa^G(s, t)$ between vertices s and t is no less than $k+1$, the problem is to find a subset V_S of V with $|V_S| = k$ such that the length of the shortest path between s and t in subgraph $G[(V \setminus V_S) \cup \{s, t\}]$ of G is maximized. The rest is to show that the problem is NP-hard by a reduction from the MVNP by the following theorem.

THEOREM 1. *The top-k structural hole spanner problem is NP-hard.*

SKETCH OF THE PROOF. Given an instance of MVNP in an undirected graph $G = (V \cup \{s, t\}, E)$ with $n = |V \cup \{s, t\}|$, a pair of vertices s and t in G, and a positive integer k, an instance of top-k structural hole spanner problem in another undirected graph $G' = (V \cup S \cup T, E')$ can be constructed as follows. Let $l = 4n^6$. Sets of vertices S and T are obtained by duplicating vertices s and t, each l times, i.e., $S = \{s_1, s_2, \ldots, s_l\}$ and $T = \{t_1, t_2, \ldots, t_l\}$. For any two different vertices $u, v \in V$, an edge (u, v) is added to E' if an edge $(u, v) \in E$. For each vertex $v \in V$, l edges $(v, s_1), (v, s_2), \ldots,$ and (v, s_l) (or $(v, t_1), (v, t_2), \ldots,$ and (v, t_l)) are added to E' if edge (v, s) (or (v, t)) is contained in E. The construction of G' is illustrated in Fig. 2. Clearly, it can be verified that $\kappa^{G'}(s_j, t_j) = \kappa^G(s, t)$ for any vertex $s_j \in S$ and any vertex $t_j \in T$ and $d^{G'}_{uv} = d^G_{uv}$ for every pair of vertices u and v.



(a) G (b) G'

Figure 2: G' is constructed from G by replicating vertices s and t and their incident edges l times.

The MVNP in $G = (V \cup \{s, t\}, E)$ can be reduced to the structural hole spanner problem in G' as follows. We first show that the optimal solution to the problem in G' does not contain any vertex s_i or t_i. We prove so by showing that for every given solution V_S of the problem for G', if V_S contains a vertex s_i or t_j, we can construct another set V'_S such that the mean distance of $G' \setminus V'_S$ is strictly larger than $G \setminus V_S$. Since, $\kappa^{G'}(s_i, t_j) > k$, therefore, there exist at least one path from s_i to t_j in $G' \setminus V_S$, thus we replace one s_i or t_j in V_S by a vertex that lies on the shortest path between s_i and t_j in $G \setminus V_S$ to obtain V'_S. We then prove that every feasible solution V_S of the MVNP problem is optimal if and only if it is the optimal solution for the problem in G', otherwise, we show a contradiction to the optimality by finding another set V'_S such that $d^{G \setminus V'_S}_{st} > d^{G \setminus V_S}_{st}$.

Due to space limit, the detailed version of the proof is deferred to the full version of this paper. □

4. ALGORITHMS FOR TOP-k STRUCTURAL HOLE SPANNER PROBLEM

In this section, we devise efficient algorithms for the top-k structural hole spanner problem. We first consider a greedy approach for the problem and improve the efficiency by simplifying the objective function. We start with a basic algorithm, using the inverse closeness centrality of vertices. We then develop a faster algorithm, by exploring the bounded inverse closeness centrality of vertices. We finally propose a fast, scalable algorithm by utilizing both articulation points and the bounded inverse closeness centrality of vertices.

4.1 The basic Algorithm

A structural hole spanner in a social network usually spans multiple communities, thus the sum of distances between the

spanner and the other vertices should not be larger than the sum of distances between an ordinary vertex and the other members in the network. The mean distance of the network after the removal of a vertex $v \in V$ thus is

$$c(G \setminus \{v\}) = \frac{n(n-1)c(G) - 2\sum_{u \in V} d_{uv}^G}{(n-1)(n-2)} + \frac{\sum_{u,w \in V} (d_{uw}^{G \setminus \{v\}} - d_{uw}^G)}{(n-1)(n-2)}, \quad (6)$$

where the value of $n(n-1)c(G)$ is the same for every vertex in G. If we only consider the first term in Eq. (6) as the dominant term, it can be implied that the shorter the distance between v and others, the more likely vertex v is to maximize the mean distance in the graph - the inverse closeness centrality of a vertex. We will use this metric as a measure to find the top-k structural hole spanners in G. Specifically, the algorithm proceeds iteratively. The set of hole spanners V_S is empty initially, within each iteration, a new hole spanner $v \in V \setminus V_S$ is found and added to V_S, if its inverse closeness centrality $c(v)$ is the smallest among vertices in $V \setminus V_S$. This procedure continues until the number of vertices in V_S becomes k. The detailed algorithm is described as follows.

Algorithm 1 `ICC`

Input: $G = (V, E), k$
Output: The set V_S of top-k structural hole spanners

```
1: V_S ← ∅;
2: build a priority queue Q of top k hole spanners with the key
   of each element in Q; is its inverse closeness centrality;
3: for each vertex v ∈ V do
4:     calculate the inverse closeness centrality c(v) of v;
5:     if |Q| < k then
6:         add v to Q;
7:     else if c(v) is less than the largest key in Q then
8:         remove the largest key element from Q;
9:         add v to Q;
10: V_S ← Q.
```

We refer to Algorithm 1 based the Inverse Closeness Centrality of vertices as Algorithm `ICC` for short.

The dominant running time of Algorithm 1 is to find a single source shortest path tree for each source vertex $v \in V$, which takes $O(m+n)$ time, using the BFS traversal on G. Algorithm 1 thus takes $O(nm + n \log k) = O(mn)$ time, where the $\log k$ factor in the second term is the time of each priority operation in priority queue Q. Despite Algorithm 1 is efficient, its time complexity is still quite high for a large-scale network that contains millions or billion of vertices. A challenging question then is whether this time complexity can be further significantly improved, e.g. a linear-time complexity, while the solution quality is not inversely compromised. In the following we answer this question affirmatively by devising two efficient algorithms for the problem.

4.2 Algorithm based on the bounded inverse closeness centrality

Most real world social networks follow two important facts: one is the sparsity. The number of neighbors of each vertex is constant, which does not proportionally grow with the network size [3]; another follows the small world law: the expected distance between any pair of vertices is a small constant, not proportional to the network size [16, 30]. Thus, instead of finding the single source shortest path tree for each vertex that includes all vertices in G, it suffices to find a *partial shortest path tree* for the vertex that reaches up to a given level of neighbors, where the neighbors of a vertex is its *level-1 neighbors*, the neighbors of its neighbors is its level-2 neighbors, and so on. We term the partial shortest path tree spanning up to level-l neighbors of v as the *l-bounded shortest tree* $T_l(v)$, and the *l-bounded inverse closeness centrality* of v thus is defined as

$$c_l(v) = \sum_{u \in T_l(v)} d_{uv}^G / (n-1). \quad (7)$$

We here adopt the similar metric as Algorithm 1, the only difference between this algorithm and Algorithm 1 is to choose K vertices with top-K *largest* l-bounded inverse closeness centrality, rather than the k vertices with top-k *smallest* inverse closeness centralities in Algorithm 1, assuming that $K \geq k$. The rationale behind is that if a vertex (as a source) can reach a larger portion of vertices in a network within a small distance l, then its average distance to other vertices is shorter. To explore the diversity among vertices and to mitigate two neighbors to be chosen as the top-k structural hole spanners at the same time, the number of candidates K for the top-k hole spanners can be larger than k, e.g., $K = ck$ ($c \geq 1$). We then calculate the inverse closeness centralities of these K vertices in G, and choose the top-k smallest ones as the top-k structural hole spanners of the network. Specifically, the proposed algorithm proceeds as follows.

Algorithm 2 `BICC`

Input: $G = (V, E), k, K, l$
Output: The set of top-k structural hope spanners V_S

```
1: build a priority queue H with the bounded inverse closeness
   as the key of each element in H;
2: build a priority queue V_S with the inverse closeness as the
   key of each element in V_S;
3: for each vertex v ∈ V do
4:     calculate the bounded inverse closeness centrality c_l(v) of
   v, using BFS search;
5:     if |H| < K then
6:         add v to H;
7:     else if c_l(v) is larger than the smallest key in H then
8:         remove the smallest key element from H;
9:         add v to H;
10: for each vertex v ∈ H do
11:     calculate the inverse closeness centrality c(v) of v;
12:     if |V_S| < k then
13:         add v to V_S;
14:     else if c(v) is less than the largest key in V_S then
15:         remove the largest key element from V_S;
16:         add v to V_S;
    return V_S.
```

It first identifies K vertices with top-K largest l-bounded inverse closeness centralities, starting at each vertex $v \in V$, using the BFS traversal on G. Assume that $c_l(v)$ is the sum of the lengths of shortest paths from each vertex within the l-neighborhood of vertex v. Let H be the set of top-K vertices with top-K largest bounded inverse closeness centralities. It then calculates the inverse closeness centrality $c(v)$ of v, for each vertex $v \in H$, using the BFS technique on G. It finally identifies the k vertices from the K chosen vertices with top-k smallest inverse closeness centralities of the vertices. We refer to Algorithm 2 based the Bounded Inverse Closeness Centrality of vertices as algorithm `BICC` for short. The rest is to analyze its time complexity by the following theorem.

THEOREM 2. *Given an undirected connected graph $G = (V, E)$ with constant maximum degree and positive integers k and l, there is a fast, scalable algorithm, Algorithm 2, for the bounded inverse closeness centrality in G, which takes $O(m+n)$ time, where $n = |V|$ and $m = |E|$.*

PROOF. Assume that G is represented by adjacency lists of its vertices. Following Algorithm 2, it first constructs a partial shortest path tree $T_l(v)$ rooted at v for each vertex $v \in V$ using BFS, which takes $O(\sum_{i=1}^{l} d_{max}^i) = O(d_{max}^{l+1})$ time, where d_{max} is the maximum degree of vertices in G. It then identifies the top-K vertices with the largest bounded inverse closeness centrality, which takes $O(\log K)$ time for each vertex insertion into the priority queue H. Therefore, it takes $O(nd_{max}^{l+1}) = O(m+n)$ time for finding the set H as both d_{max} and l are small constants, in comparison with the network size n. Identifying set V_S takes $O(K(n+m)+K\log k)$ time, due to the BFS search in G for each candidate vertex in H, and the addition of the candidate vertex to set V_S, where V_S is maintained as a priority queue. Therefore, the time complexity of Algorithm 2 is $O(K(n+m)+K\log K) = O(m+n)$ as $K = ck$ usually is constant. □

Note that in real social networks the number of neighbours of an individual is a small constant which is not proportional to the network size.

4.3 A fast and scalable algorithm

So far, we have provided an algorithm based on the inverse closeness centrality and devised an efficient algorithm by approximating the inverse closeness centrality of each vertex, through the introduction of l-bounded inverse closeness centrality concept. In the following, we take the second term of Eq. (6) into account and devise another efficient algorithm which further speeds up the running time in practice, by exploring articulation points of G.

One of the instinct properties of structural hole spanners in most real social networks is their tendency to connect multiple isolated communities. Such hole spanners are referred to the *articulation points* in graph theory. Thus, a top-k structural hole spanner usually is an articulation point too. However, the number of articulation points in real social networks is quite large, e.g., the number of articulation points in each network of Table 1 is at least 10% of the number of vertices in the network. How to identify top-k structural hole spanners from all articulation points in a large-scale network is a challenging issue. In the following, we shall devise a fast yet scalable algorithm for the top-k structural hole spanner problem, by exploring the articulation points and using the bounded inverse closeness centrality of vertices.

LEMMA 1. *[22] Let G be an unweighted graph $G = (V, E)$ with $n = |V|$ vertices, then the sum of lengths of all pairs shortest paths in G is no more than $n^3/3$.*

Given two vertices u and v that are not in the same connected component of G, we assume that there is a *virtual edge* in G between them with weight larger than the sum of lengths of all pairs of shortest paths in G, and we assign this virtual edge with a weight $w(u,v) = cn^3$ with $c \geq 1/3$. For the sake of convenience, we set $c = 1$ in the rest of discussion. Assume that v is an articulation point in G, we distinguish into two cases as follows.

Case one: if the removal of v results in two connected components CC_1 and CC_2. Let CC_i contain n_i vertices. The weighted sum of all virtual edges resulting from the removal of v is $n_1 \times (n-n_1)cn^3 + n_2 \times (n-n_2)cn^3 = cn^3(nn_1 + nn_2 - n_1^2 - n_2^2)$. Clearly, when $n_1 \approx n_2$, the weighted sum is maximized. This implies that an articulation point is likely to be a top-k hole spanner if its removal results in two large connected components.

Case two: if the removal of v results in l connected components $CC_1, CC_2, \ldots, CC_l$ with $l > 2$. Let CC_i contain n_i vertices ($1 \leq i \leq l$). Let $n = \sum_{i=1}^{l} n_i$. Then, the weighted sum of virtual edges between vertices in CC_i and CC_j is $\sum_{i=1}^{l}\sum_{j=1}^{l} n_i n_j cn^3 = cn^3 \sum_{i=1}^{l} n_i(n-n_i)$, which is maximized when all components have roughly equal sizes.

Following the analysis of these two cases, it can be seen that an articulation point in G is likely to be one of top-k structural hole spanner if its inverse closeness centrality is maximized, which approximately equals the weighted sum of virtual edges resulting from its removal, since the sum of all pairs shortest paths in each connected component is much less than the weight of each virtual edge. We now propose a fast, scalable algorithm by exploring the inverse closeness centralities of articulation points. Specifically, the algorithm consists of two stages. Let A be the set of articulation points in G. If $|A| < k$, the algorithm will proceed the second stage after the first stage. Within the first stage, there is a number of iterations. An articulation point within each iteration will be chosen as a top-k hole spanner. In the second stage, Algorithm 2 will be invoked to find the rest of top-k structural hole spanners. The detailed algorithm is described in Algorithm 3.

Algorithm 3 `AP_BICC`

Input: $G = (V, E), k, K, l$
Output: The set of top-k structural hole spanners V_S

```
1: build a priority queue V_S of top-K approximate inverse closeness candidates with the key of each element in V_S;
2: let A be the set of articulation points in G, which can be found by invoking Procedure 1;
3: for each vertex v ∈ A do
4:     find the approximate inverse closeness centrality c'(v);
5:     if |V_S| < k then
6:         add v to V_S;
7:     else if c'(v) is larger than the smallest key in V_S then
8:         remove the smallest key element from V_S;
9:         add v to V_S;
10: if |V_S| < k then /* the number of APs is less than k */
11:     U ← V \ A;
12:     k' ← k − |V_S|;
13:     build a priority queue Q of K elements with the key of each element in U being its l-bounded inverse closeness centrality;
14: while |V_S| ≠ k do
15:     extract the the element v with largest key from Q;
16:     add v to V_S;
    return V_S.
```

We refer to Algorithm 3 based on Articulation Points and Bounded Inverse Closeness Centrality of vertices as Algorithm `AP_BICC` for short. We now show the articulation point finding and their approximate inverse closeness centrality, $c'(v)$, for each $v \in A$ can be efficiently calculated, using a Depth-First Search (DFS) traversal on G.

Let v be an articulation point of G, in the DFS tree construction starting from a vertex v, assume that $u_1, u_2, \ldots, u_p$ be the children of vertex v in the DFS tree. Let V_i be the set of vertices in the subtree T_i rooted at u_i and CC_i the connected component of G induced by the vertices in V_i with

$1 \leq i \leq p$. Let CC_0 be the connected component containing the ancestors of v in the DFS tree. Following the DFS search property, all edges in G can be partitioned into "tree edges" and "non-tree edges", respectively. And all non-tree edges are "back edges", which means that one endpoint of the edge is a descendant while another endpoint of the edge is a proper ancestor of v in the DFS tree. Clearly, there is no edges between any two connected components CC_i and CC_j with $i \neq j$ and $1 \leq i, j \leq p$, by the DFS traversal property. If there is a back edge between a vertex in V_i and a vertex in CC_0, then both CC_i and CC_0 are the same connected component when the removal of v from G, $1 \leq i \leq p$. An illustration of this case is shown in Fig 3.

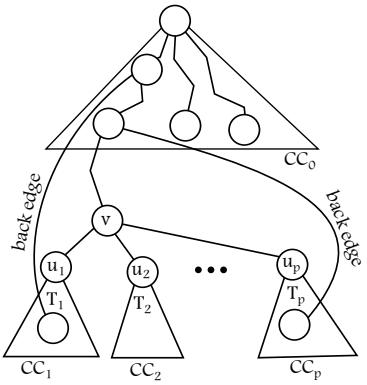


Figure 3: An illustration of exploring an articulation point v and its p children $u_1, u_2, \ldots, u_p$ during a DFS traversal on G.

Assume that there are p' CCs among the p CCs derived from p children of v have back edges. Then, the removal of v will result in $p - p'$ CCs. For the sake of convenience, we assume that these $p-p'$ CCs are $CC'_1, CC'_2, \ldots, CC'_{p-p'}$ with each having n'_i vertices. The approximate inverse closeness centrality of v then is

$$c'(v) \approx \sum_{i=1}^{p-p'} |CC'_i| \cdot (n - |CC'_i| - 1) \cdot n^3. \quad (8)$$

The linear-time procedure of detecting each articulation point and the calculation of its approximate inverse closeness centrality is then detailed as follows. A vertex v is identified as an articulation point of G if a subtree rooted at one of its children does not contain any back edges. The induced subgraph by the set of vertices in this subtree is a connected component after the removal of vertex v from G. The number of vertices contained in each such connected component is the number of descendants of that child in the DFS tree. To keep track of the number of descendants of each vertex when performing the DFS traversal on G and to identify those children of the vertex without any back edges, the approximate inverse closeness centrality of v (as an articulation point) can be easily calculated. The detailed implementation of this is given in Procedure 1 and Procedure 2.

THEOREM 3. *Given a graph $G(V, E)$ with constant maximum degree and an integer $k > 0$, Algorithm 3 takes time $O(n \log k + m) = O(m + n)$ as k is a constant.*

PROOF. Following Algorithm 3, the detection of all articulation points and the calculation of their approximate inverse closeness centralities takes $O(n + m)$ time by Procedure 1.

Procedure 1 Articulation points and their approximate inverse closeness centrality calculation

```
1: for each vertex u ∈ V do
2:     /* the number of children of u */
3:     u.child ← 0;
4:     /* each vertex has 3 colors white/grey/black */
5:     u.color ← white;
6:     c'(u) ← 0;
7: time ← 0;
8: for each vertex u ∈ V do
9:     if u.color == white then
10:        call Modified-DFS(G,u) /* Procedure 2 */;
```

Procedure 2 Modified-DFS(G, u)

```
1: u.color ← black;
2: time ← time + 1;
3: /* the discovered time of vertex u */
4: u.discovered ← time;
5: /* the smallest discovered time of any neighbor of u's descendants (through a back-edge) */
6: u.lowest ← time;
7: /* the number of vertices in CC_0 after removing u */
8: cc_0 ← n;
9: /* the number of descendants of u in DFS tree */
10: u.descendant ← 0;
11: for all (u, v) ∈ E do
12:     if u.color == white then
13:         u.color ← grey;
14:         v.π ← u /* u is the parent of v */;
15:         u.child ← u.child + 1;
16:         call Modified-DFS(G,v);
17:         u.descendant ← u.descendant + v.descendant;
18:         u.lowest ← min(u.lowest, v.lowest);
19:         if (v.lowest ≥ u.discovered) OR (u is root AND u.child > 1) then
20:             /* v will be disconnected without u */
21:             c'(u) ← c'(u)+(v.descendant×(n−v.descendant−1));
22:             cc_0 ← cc_0 − v.descendant /* subtree of v is not part of CC_0 */;
23:     else if v ≠ u.π then
24:         u.lowest ← min(u.lowest, v.discovered);
25: c'(u) ← c'(u) + (cc_0 × (n − cc_0 − 1)).
```

For each vertex u, its adjacency list is traversed exactly once and the number of descendants and children are calculated in the post-traversal in DFS. The maintenance of the priority queue Q takes $O(|A| \log k) = O(n \log k)$ time, where A is the set of all articulation points in G. The total amount of time for calculating the number of descendants of each vertex in the DFS tree is $O(n)$. Thus, the time complexity of Algorithm 3 is $O(n + m)$. □

5. PERFORMANCE EVALUATION

In this section we evaluate the performance of the proposed algorithms for the structural hole spanner problem, using different datasets. We start with the experimental environment settings, we then investigate the effectiveness of the proposed models of structural hole spanners, compared with other models using both real and synthetic datasets. We finally study the performance of the proposed algorithms and the impacts of parameters on the performance using the datasets in Table 1 and in the end, we discuss the results.

5.1 Experimental environment setting

To evaluate the performance of the proposed algorithms, we adopt the real-world datasets, which are listed in Table 1,

where GR-QC is the collaboration network from arXiv[1], covering collaborations between authors of papers submitted to General Relativity and Quantum Cosmology category. Epinions is an online social network of a general consumer review site Epinions[2]. The Twitter dataset was obtained from [19]. Email-EuAll is the anonymous email network of a large European research institution for an 18-month period [18]. The DBLP-2011 dataset is the collaboration network obtained from the DBLP web site[3], and the LiveJournal dataset describes the social network of free on-line blogging community[4].

Recall that Algorithms 1, 2, 3 are denoted by ICC, BICC and AP_BICC, respectively. To evaluate their performance on the mentioned datasets, we will use the following state-of-the-art algorithms for benchmark purposes.

- Algorithm PathCount [11] is similar to betweenness centrality and assigns each vertex a score that is the average number of shortest paths (between all pairs of vertices) on which the vertex lies, then selects the top-k vertices with the highest scores.
- Algorithm 2-Step [24] assigns each vertex a score that is the number of pairs of its neighbors without edges between them, then selects the top-k highest scores.
- Algorithm PageRank [21] assigns each vertex v a PageRank score $r(v)$ that is the visiting probability of v by a random surfer, $r(v) = 1/n$ initially. The algorithm then updates $r(v)$ with a new value $r(v) = (1-\alpha)/n + \alpha \sum_{(u,v)\in E} r(u)/\deg(u)$, where $\alpha = 0.85$ is the random jump parameter. It finally chooses the top-k vertices with the highest PageRank scores.
- Algorithm MaxD [19] is to find a set of k vertices such that the minimum cut of communities will be reduced significantly, after removing these vertices, assuming that l communities are given. For any pair of communities, the algorithm selects $\lceil 2k/(l(l-1)) \rceil$ vertices as structural hole spanners using a greedy strategy. In each round, it chooses the vertex whose removal will result in a maximum decrease of the minimum cut.
- Algorithm HIS [19] assigns each vertex v a score that simulates the likelihood of v as a structural hole spanner across the given subset of communities, assuming that l communities are given.

Notice that all our experiments were conducted based on a Linux desktop with GenuineIntel Core i7-3370 (3.40GHz) CPU and 8GB main memory.

Dateset	$\|V\|$	$\|E\|$	% APs	$\Delta(G)$	Diam
GR-QC	5,242	28,980	15%	81	17
Epinions	75,879	508,837	14%	1,551	14
Twitter	92,180	188,971	14%	233	26
Email-euAll	265,214	420,045	2%	7,636	14
DBLP-2011	986,324	6,707,236	9%	979	12
LiveJournal	5,363,260	79,023,142	16%	2,469	14

Table 1: Six different real datasets, where APs stands for Articulation Points in the corresponding network and Diam stands for Diameter.

[1]http://arxiv.org/
[2]http://epinions.com/
[3]http://www.informatik.uni-trier.de/~ley/db/
[4]http://livejournal.com/

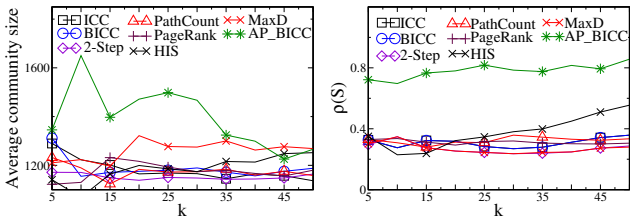
5.2 Effectiveness of the proposed model

We first evaluate the effectiveness of the proposed model using the definition proposed by Burt [7] such as: (1) the size of communities that each individual spans, (2) the number of communities and (3) the number of neighbours of that individuals. Burt [7] suggested that a good structural hole spanner is connected to many communities, but to be influential, the ratio of the number of its communities to the number of its neighbors should be large. This definition implies a metric for evaluating the structural hole spanners in a setting where the communities are given in advance. Given a graph $G = (V, E)$, suppose S is the set of structural holes found by an algorithm, then, the quality of the solution S is

$$\rho(S) = \frac{\sum_{v\in S} \frac{\#\text{ of communities that } v \text{ is connected to}}{\deg(v)}}{|S|}. \quad (9)$$

We evaluate the performance of different algorithms using this metric in order to find out the degree to which our model maps the real structural hole spanners. We use the DBLP dataset which has been used for the same purpose in [19]. The communities in this network are publication venues, e.g, journal or conference; authors who published to a certain journal or conference form a community. We evaluate two algorithms by MaxD and HIS proposed by Lou *et al.* [19].

Fig. 4(a) shows that Algorithm AP_BICC significantly outperforms all the other algorithms in terms of the average community size by varying k. Fig. 4(a) verifies our claim in Section 2.2 that our model can identify the vertices connecting with larger communities. Similarly, it can be observed in Fig. 4(b) that algorithm AP_BICC outperforms the other algorithms in the benchmark at least 50%, using the metric in Eq. (9). In a nutshell, Algorithm AP_BICC can guarantee to find efficient structural hole spanners connecting to larger communities in this dataset while the other algorithms produce the results with less number of communities compared to the number of their neighbors. Also, the running time of Algorithm AP_BICC is a few milliseconds, while Algorithm MaxD takes minutes and Algorithm HIS takes a few seconds.



(a) Average community size (b) Community membership

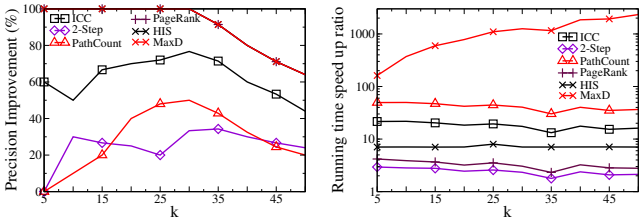
Figure 4: Effectiveness of different algorithms on dataset DBLP using different quality metrics.

5.3 Performance on synthetic datasets

We then evaluate the quality of structural hole spanners found by different algorithms. We generate a random graph of 2^{11} vertices using SSCA method[5]. SSCA generates cliques of random size with average size 2^7 and random inter-clique edges. We then place a ground-truth structural hole spanner s_i for every clique i. For every edge (u, v), such that u is in clique i and v is in clique j, we replace it with two edges (u, s_i) and (s_i, v). Fig. 5 shows the performance improvement made by Algorithm AP_BICC in empirical results.

[5]http://www.cse.psu.edu/~madduri/software/GTgraph/

Fig. 5 demonstrates that the solution accuracy delivered by Algorithm AP_BICC is at least 20% of the others for $k > 5$. The reason for such significant improvement is that algorithm AP_BICC finds more meaningful structural hole spanners that play a significant role in the connectivity of communities (cliques) and vertices in the network.



(a) Precision improvement (b) Speed up ratio

Figure 5: Running time and precision improvement by Algorithm AP_BICC using synthetic dataset.

5.4 Performance on real datasets

We thirdly evaluate the performance of the proposed algorithms ICC, BICC, and AP_BICC, against benchmark structure-based algorithms PathCount, PageRank, and 2-Step against different datasets listed in Table 1. We avoid comparing community-based algorithms MaxD and HIS, since using any community detection method is subject to unfairness.

Fig. 6 and Fig. 7 show the performance of different algorithms in terms of the optimization objective in Eq. (4), i.e., $C(G \setminus S) - C(G)$ and their running times. Specifically, it can be seen from Fig. 6(a) that Algorithm AP_BICC significantly outperforms all the other algorithms by at least 100% for dataset GR-QC, while its running time is only 6% of the best of the three benchmark algorithms as shown in Fig. 7(a). It can be observed from Fig. 6(e) to Fig. 6(f) that Algorithm AP_BICC has the similar performance and running time for other datasets. That is, it outperforms all the other algorithms at least 50% while its running time is only around 7% and 65% of the fastest benchmark algorithm for datasets DBLP-2011 and LiveJournal, respectively. Fig. 6(b) and Fig. 6(d) show that it is superior in comparison with benchmark algorithms such as PageRank and PathCount on both performance and running time for datasets Epinions and Email-EuAll when k is small ($k < 20$). With the increase of k, the performance gap between Algorithm AP_BICC and other algorithms increases, too. Furthermore, it can be seen from Fig. 7(b) and Fig. 7(d) that the running time of Algorithm AP_BICC is less than 7% of the fastest benchmark Algorithm PageRank and 0.01% of Algorithms ICC and PathCount for dataset DBLP-2011 as shown in Fig. 7(e). Moreover, Algorithm AP_BICC significantly outperforms Algorithm BICC, which means that the second term in Eq. (6) is dominant. Furthermore, since the number of articulation points is large, Algorithm AP_BICC never reaches to the second phase, causing it to run considerably faster than Algorithm BICC.

5.5 Impact of parameters on the performance

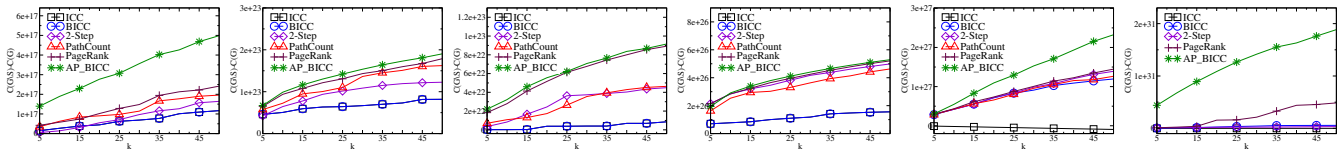
We finally evaluate the impact of parameters on the performance of the proposed Algorithms AP_BICC and BICC. As the number of articulation points in each mentioned dataset is far larger than k, the second stage of Algorithm AP_BICC, the bounded inverse closeness centrality of vertices will not be invoked. Therefore, we only investigate the impact of parameters l and K on the performance of algorithm BICC.

Fig. 8 plots the performance curves of Algorithm BICC by varying the value of l, based on two representative large datasets LiveJournal and DBLP-2011 in Table 1. As the experimental results indicate that algorithm BICC will deliver the similar performance for other datasets, we focus only on these two large datasets. Fig. 8(b) implies that when l is small with $2 \leq l \leq 6$, Algorithm BICC has the best performance for dataset LiveJournal. It also exhibits similar behavior for dataset DBLP-2011 as shown in Fig. 8(a). However, its performance degrades chromatically when $l = 6$, i.e., its performance decreases by at least 80% compared with its performance when $l = 2$. This implies that its performance dramatically drops with the growth of l. The reason behind is that algorithm BICC always chooses the top-K vertices with the largest l-bounded inverse closeness $c_l(v)$. By continuously increasing the value of l, the value of $c_l(v)$ will be closer to the value of its inverse closeness centrality $c(v)$. Since the top-K vertices with the "largest" l-bounded inverse closeness centrality (that is indeed almost the same as inverse closeness centrality for large l) will be chosen, the algorithm performance drops. Fig. 8 suggests that the value of l in practice should be small, otherwise the quality of the solution is not promising, this further verifies the small-world phenomenon that causes l-bounded closeness centrality to be very close to inverse closeness centrality, since the small-world phenomenon states that the largest distance in the network is expected to be a constant. Fig. 8(d) plots the running times of algorithm BICC for different k using dataset LiveJournal, from which it can be seen that it takes more time on finding a solution with the growth of l. Specifically, its running time when $l = 2$ is a tiny fraction of its running time when $l = 10$ (0.001%). Fig. 8(c) further shows that it has the similar behavior for dataset DBLP-2011.

Fig. 9(a) shows that the performance of algorithm BICC for dataset DBLP is stable, with the increase of K. The rationale behind is that social networks follow the small-world law that individuals can reach each other with a few number of hops, thus the l-bounded inverse closeness centrality of vertices can approximately represent closeness centrality. Fig. 9(b) demonstrates that algorithm BICC leads to a better performance for dataset LiveJournal with the growth of K. Fig. 9(d) shows its running time, using different values of K for dataset LiveJournal, from which it can be seen that the running time of algorithm BICC linearly increases, with the growth of K. The reason is that by increasing the value of K, algorithm BICC will examine more candidate vertices.

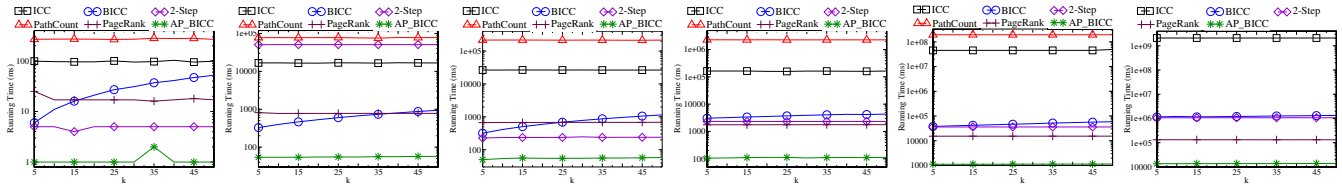
5.6 Discussion

We now turn to the discussion of the experimental results. Fig. 6 shows that in co-authorship datasets GR-QC and DBLP-2011, the performance gain of Algorithm AP_BICC over Algorithm PageRank is significant. The reason is that Algorithm AP_BICC finds the individuals who bridge different communities, while Algorithm PageRank detects the individuals with high reputation. In other words, within co-authorship networks, opinion leaders have a high reputation as they collaborate with many people in their own community, however, the structural hole spanners connect different communities and their absence is more tangible. Similarly, in dataset LiveJournal, bloggers produce contents within a few subjects gain high reputation. The bloggers who publish in multiple subjects however, have a wider perspective, more sources of information and hold more significant positions.



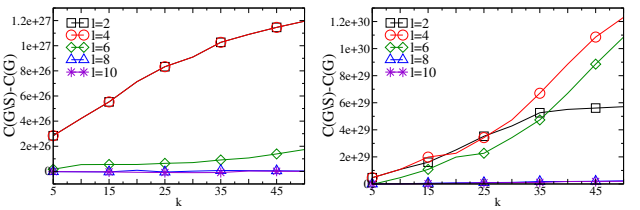
(a) GR-QC (b) Epinions (c) Twitter (d) Email-EuAll (e) DBLP-2011 (f) LiveJournal

Figure 6: Performance of various algorithms, using different datasets.



(a) GR-QC (b) Epinions (c) Twitter (d) Email-EuAll (e) DBLP-2011 (f) LiveJournal

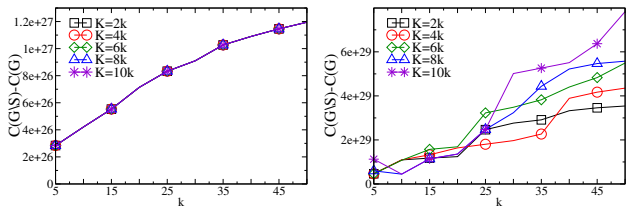
Figure 7: Running times of various algorithms, using different datasets.



(a) DBLP-2011 (Performance) (b) LiveJournal (Performance)

(c) DBLP-2011 (Time) (d) LiveJournal (Time)

Figure 8: Impact of parameter l on performance of algorithm BICC.



(a) DBLP-2011 (Performance) (b) LiveJournal (Performance)

(c) DBLP-2011 (Time) (d) LiveJournal (Time)

Figure 9: Impact of parameter K on performance of algorithm BICC.

Moreover, Fig. 6 shows that the performance of Algorithm AP_BICC is similar to PageRank for dataset Email-EuAll. Dataset Email-EuAll is the email network of an organization in which high level managers gain a high reputation. The high level managers communicate with the operation managers in each separate division and act as the only communication link between each separate division. Thus, they are also structural hole spanners. A similar explanation can be applied to dataset Twitter, where people who have higher reputation are directly followed by a larger portion of users and their content is forwarded by their followers.

6. RELATED WORK

Over last few years, the size of real networks has increased enormously, developing efficient algorithms for finding influential individuals in such large-scale networks with unique properties such as structural hole spanners has become a challenging task. Moreover, building accurate models that truly reflect the properties of structural hole spanners is crucial to identify such individuals. Nevertheless, researchers take lots of effort towards this aim.

The notion of structural hole spanners was first introduced by Burt [7] to find the key employees in organizations for integrating operations across functional and business boundaries. This concept later was further refined in [1, 8, 9]. A few studies have exploited the concept of structural holes in order to design strategic games for network formation [11, 17]. Goyal *et al.* [11] presented a network formation model that a vertex u serves as an intermediary between many vertices. However, this strategy leads to the star network and real networks do not follow a star topology. In order to tackle this problem, Kleinberg *et al.* [17] designed a game by building a model of the payoffs that arise from filling structural holes. This payoff is a decreasing function of the number of paths with length two between each pair of neighbors to avoid the star topology. One of the limitations of the model presented by Kleinberg *et al.* [17] is that this model needs careful tuning of parameters such as the link maintenance cost that is not easily achieved in large-scale networks. Another line of research in computer science is to find structural hole spanners in order to incorporate them in contagion, and can be divided in two categories as follows.

Structural-based Models: Goyal *et al.* [11] formulated a structural hole spanner as a vertex that resides on more shortest paths between different pairs of vertices. Since counting the number of shortest paths in large networks is time-consuming, Tang *et al.* [24] proposed to only count the number of shortest paths with length two on which a vertex lies. In this model, any shortest path of length greater than two will be ignored, thus the model suggests candidates that are connected to smaller rather than larger, richer and more influential communities. A fairly common case under this model is its failure of finding good quality structural

hole spanners when a vertex is densely connected to two communities. For example, in Fig. 1, vertex v_1 forms more length-2 paths than that of v_2, while v_2 is connected to more communities and is a better structural hole based on Burt's theory [7]. However, this model suggests v_1 as a better structural hole spanner. In order to address this problem, Ugander *et al.* [28] defined the *structural diversity* of an individual as the number of connected components in its contact neighborhood, which is a similar notion as structural hole spanners, and studied the role of structural diversity in contagion of information within real social networks. Huang *et al.* [14] studied the top-k structural diversity search in large networks and developed efficient algorithms for massive dynamic networks. However, only a small number of vertices in each community can be part of contact neighborhood, and they can form multiple connected components. Similarly, Tong *et al.* [27] defined the gateway-ness of a vertex v, proportional to the paths between source vertices S and target vertices T, on which v lies. In addition, each path is given a score, which is inversely proportional to its length. Yang *et al.* [31] studied the role of structural holes in diffusion.

Community-based Models: Lou *et al.* [19] proposed a model to find structural holes in a network, assuming that communities in the network are given. The objective in their model is to maximize a utility function that measures the degree to which vertices span communities. One instantiation is to find a set of vertices whose removal leads to the maximum decreases on the number of inter-community edges. One major concern about this model is that communities usually are not known, thus the quality of the solution relies on the quality of communities found. Moreover in Fig. 1, the removal of v_1 decreases the number of inter-community edges by 8 and the removal v_2 decreases the inter-community edges by 6. Therefore, this model implies v_1 as the preferred structural hole spanner. However, v_2 should be a better structural hole as it bridges more communities.

7. CONCLUSION

In this paper we studied the top-k structural hole spanner problem in a large-scale social network. We first proposed a novel model to measure the quality of structural holes. We then formulated a novel top-k structural hole spanner problem and showed its NP-hardness. We thirdly devised two fast yet scalable linear-time algorithms for the problem by using both the bounded inverse closeness centrality of vertices and articulation points of the network. We finally evaluated the performance of the proposed algorithms and validated the effectiveness of the proposed model through extensive experiments on real and synthetic datasets. Experimental results demonstrated that the proposed model can capture the characteristics of structural hole spanners, and the proposed algorithms outperform several other heuristics.

8. ACKNOWLEDGMENTS

This work is supported by the grant of Australian Research Council Discovery Project No. DP120102627.

9. REFERENCES

[1] G. Ahuja. Collaboration Networks, Structural Holes, and Innovation: A Longitudinal Study. *Administrative science quarterly*, vol. 45, pp. 425–455, 2000.

[2] R. Andersen, and K. J. Lang. Communities from Seed Sets. *WWW'06*, pp. 223–232, ACM, 2006.

[3] A-L. Barabási, and R. Albert. Emergence of Scaling in Random Networks. *Science 286*, pp. 509–512, 1999.

[4] A. Bar-Noy, S. Khuller, and B. Schieber. The Complexity of Finding Most Vital Arcs and Nodes. *Univ. of Maryland Institute for Advanced Computer Studies Report No. UMIACS-TR-95-96*, 1998.

[5] M. A. Beauchamp. An Improved Index of Centrality. *Behavioral Science* , vol. 10, pp. 161–163, 1965.

[6] C. Budak, D. Agrawal, and A. El Abbadi. Limiting the Spread of Misinformation in Social Networks. *WWW'11*, pp. 665–674, ACM, 2011.

[7] R. S. Burt. Structural holes: The Social Structure of Competition. *Harvard university press*, 1992.

[8] R. S. Burt. Structural Holes and Good Ideas. *American Journal of Sociology*, vol. 110, pp. 349–399, 2004.

[9] R. S. Burt. Secondhand Brokerage: Evidence on the Importance of Local Structure for Managers, Bankers, and Analysts. *Academy of Management Journal*, vol. 50, pp. 119–148, 2007.

[10] N. Girvan, and M. E. Newman. Community Structure in Social and Biological Networks. *Proc. Natl. Acad. Sci.* , vol. 99, pp. 7821–7826, 2002.

[11] S. Goyal, and F. Vega-Redondo. Structural Holes in Social Networks. *Journal of Economic Theory*, vol. 137, pp. 460–492, Elsevier, 2007.

[12] A. Guille, H. Hacid, C. Favre, and D. A. Zighed. Information Diffusion in Online Social Networks: A Survey. *ACM SIGMOD Record*, vol. 42(2), pp. 17–28, 2013.

[13] J. Hopcroft, T. Lou, and J. Tang. Who Will Follow You Back? : Reciprocal Relationship Prediction. *CIKM'11*, pp. 1137–1146, ACM, 2011.

[14] X. Huang, H. Cheng, R. Li, L. Qin, and J. X. Yu. Top-K Structural Diversity Search in Large Networks. *VLDB'13*, vol. 6, pp. 1618–1629, 2013.

[15] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the Spread of Influence through a Social Network. *SIGKDD'03*, pp. 137–146, ACM, 2003.

[16] J. Kleinberg. Navigation in a Small World. *Nature*, vol. 406, pp. 845–845, 2000.

[17] J. Kleinberg, S. Suri, É. Tardos, and T. Wexler. Strategic Network Formation with Structural Holes. *EC'08*, pp. 284–293, 2008.

[18] J. Leskovec, A. Krevl. SNAP Datasets: Stanford Large Network Dataset Collection. *http://snap.stanford.edu*, 2014.

[19] T. Lou, and J. Tang. Mining Structural Hole Spanners through Information Diffusion in Social Networks. *WWW'13*, pp. 825–836, 2013.

[20] M. Marathe, and A. K. S. Vullikanti. Computational Epidemiology. *SIGKDD'14*, pp. 1969–1969, 2014.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *Stanford InfoLab*, 1999.

[22] J. Plesník. On the Sum of All Distances in a Graph or Digraph. *Journal of Graph Theory*, vol. 8, 1984.

[23] E. J. Rinia, T. N. Van Leeuwen, E. E. Bruins, H. G. Van Vuren, and A. F. Van Raan. Citation Delay in Interdisciplinary Knowledge Exchange. *Scientometrics*, vol. 51, pp. 293–309, Springer, 2001.

[24] J. Tang, T. Lou, and J. Kleinberg. Inferring Social Ties Across Heterogenous Networks. *WSDM'12*, 2012.

[25] J. Tang, S. Wu, and J. Sun. Confluence: Conformity Influence in Large Social Networks. *SIGKDD'13*, pp. 347–355, 2013.

[26] Y. Tang, X. Xiao, and Y. Shi. Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency. *SIGMOD'14*, pp. 75–86, ACM, 2014.

[27] H. Tong, S. Papadimitriou, C. Faloutsos, P. S. Yu, and T. Eliassi-Rad. Gateway Finder in Large Graphs: Problem Definitions and Fast Solutions. *Information retrieval*, vol. 15, pp. 391–411, Springer, 2012.

[28] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg. Structural Diversity in Social Contagion. *Proc. Natl. Acad. Sci.* , vol. 109, pp. 5962–5966, 2012.

[29] L. Wang, T. Lou, J. Tang, and J. E. Hopcroft. Detecting Community Kernels in Large Social Networks. *ICDM'11*, 2011.

[30] D. Watts, and S. Strogatz. Collective Dynamics of 'Small-World' Networks. *Nature*, vol. 393, pp. 440–442, 1998.

[31] Y. Yang, J. Tang, C. Leung, Y. Sun, Q. Chen, J. Li, and Q. Yang. RAIN: Social Rrole-Aware Information Diffusion. *AAAI'15*, 2015.