Diversified Top-\(r\) Community Search in Geo-Social Network: A K-Truss Based Model

Renjie Sun
East China Normal University
renjie.sun@stu.ecnu.edu.cn

Yanping Wu
Zhejiang Gongshang University
yanpingw.zjgsu@gmail.com

Xiaoyang Wang*
Zhejiang Gongshang University
xiaoyangw@zjgsu.edu.cn

ABSTRACT
Given a graph \(G\) and a query node \(q\), the community search problem aims to retrieve the important communities whose nodes are closely related to \(q\). Recently, plenty of research is conducted to find cohesive subgraphs on geo-social networks, where each node is associated with a location. However, the properties of diversified communities on geo-social networks are neglected by previous studies. To meet this gap, in this paper, we propose a new problem, called diversified DBTruss search, which aims to find \(r\) cohesive subgraphs that can maximize the number of nodes covered. The DBTruss is a subgraph of \(G\), which satisfies both structure cohesiveness and spatial constraint. For the problem, we develop two algorithms, i.e., TD Algorithm and BD Algorithm, with approximation ratio guarantee. Finally, experiments on real-world datasets are conducted to demonstrate the advantages of proposed techniques.

1 INTRODUCTION
As an important tool, community search aims to retrieve the connected subgraphs based on a query request [7]. Recently, due to the location equipped systems [9], many research has been conducted on the community search problem in geo-social networks [4, 8]. Most existing works in geo-social networks are driven by size constraint or other defined constraints. However, the obtained subgraphs are usually highly overlapping, i.e., the diversify property is neglected, which significantly reduces the usefulness of information contained in results.

To fill the gap, in this paper, we propose and investigate the diversified top-\(r\) Diameter Bound \(k\)-truss (DBTruss) problem in geo-social networks. Specifically, given a query node \(q\) in a geo-social graph \(G\), diameter restriction \(d\) and support constraint \(k\), a subgraph of \(G\) is a DBTruss if it is 1) a \(k\)-truss containing query node \(q\), and 2) enclosed by a circle with diameter bound \(d\). Given an integer \(r\), the diversified DBTruss problem aims to find \(r\) DBTrusses that can cover the most number of nodes in \(G\). Figure 1 shows a geo-social network with 19 users. Given \(k=4\) and \(r=2\), there are 3 DBTrusses that can be recommended to Bill, i.e., 3 colored circles. Without considering the diversity property, the DBTrusses covered by orange and pink colors will be returned because of larger size. However, by considering the diversity property, the DBTrusses in purple and pink areas will be retrieved, due to coverage of more users.

Challenges and contributions. To the best of our knowledge, we are the first to investigate the diversified top-\(r\) DBTruss problem. The main challenges of the problem are two folds. Firstly, due to the uncertainty of the diameter bounded circle where \(k\)-truss is located, it is time-consuming to enumerate all possible circles. Secondly, we show that the problem is NP-hard. In this paper, TD Algorithm is first presented with \(1−1/e\) approximate guarantee to ensure the quality of results. In addition, BD Algorithm is further presented with \(1/4\) approximate ratio guarantee to speedup the computation. Experiments on real datasets from various domains are conducted to evaluate the performance of the proposed techniques.

Related work. Community or cohesive subgraph search are widely studied [6, 7, 10]. Different cohesive subgraph model are proposed in the literature, such as \(k\)-core, \(k\)-truss and clique [2, 3]. Liu et al. [6] study the problem of community search over large directed graphs. [10] considers the influence of nodes when searching the communities. However, the geo-location of users are not considered. Recently, some works investigate the community search problem by considering the spatial information (e.g., [5, 8, 11]). The most close work to us is [8], which investigates the problem of searching RB-\(k\)-core by considering both spacial and social constraints. However, they focus on finding all RB-\(k\)-cores, but the diversified characters are ignored and each result is treated individually. That is, it aims to identify top-\(r\) answers which are most relevant to a user query. Moreover, the \(k\)-core model may not well portray the strength of connections in a community. Thus, the techniques in the above research cannot support the problem in this paper.

2 PRELIMINARY
We consider a geo-social network \(G = (V, E)\) as undirected graph, where \(V (|V| = n)\) and \(E (|E| = m)\) denote the set of nodes and edges in \(G\), respectively. For each node \(u \in V\), it associates a location \(loc(u) = (x_u, y_u)\), which represents its position in 2-dimensional space. In this paper, we use Euclidean distance to measure the distance between two nodes, \(u\) and \(v\), denoted by \(d(u, v)\).

We use \(S = (V_S, E_S)\) to denote a subgraph of \(G\), where \(V_S \subseteq V\) and \(E_S \subseteq E\). A triangle is a cycle with 3 edges. Given a subgraph \(S\), we use \(sup(e, S)\) to denote the support of edge \(e\) in \(S\), i.e., the number of triangles that contain \(e\) in \(S\). For simplicity, we omit \(S\) in the notations if the context is self-evident.

**Definition 1 (\(k\)-truss).** Given a graph \(G\), a subgraph \(S\) is the \(k\)-truss of \(G\), denoted as \(T_k\), if it satisfies: 1) for each \(e \in E_S\),...
In this paper, we leverage the concept of $k$-truss to measure the cohesiveness of a subgraph. Compared with the $k$-core model, $k$-truss emphasizes not only the engagement of each node but also the strength of nodes connection. Considering the distance constraint, we introduce the concept of minimum covering circle, which is defined as follows.

**Definition 2 (Minimum covering circle (MCC)).** Given a node set $S$ in 2-dimensional space, the MCC of $P$ is a circle, which can cover all nodes in $P$ with the smallest diameter.

We use $O(c, d)$ to denote a circle, where $c$ and $d$ denote the center and diameter of the circle, respectively. As discussed, in geo-social networks, a community is more preferred if it is densely-connected and spatially close. Based on the concepts of $k$-truss and MCC, we define the diameter-bound $k$-truss model as follows.

**Definition 3 (Diameter-bound $k$-truss (DBTruss)).** Given a geo-social network $G = (V, E)$, a support constraint $k$ and a diameter bound $d$, a subgraph $S$ is a diameter-bound $k$-truss, denoted as $T^d_k$, if it satisfies the following constraints: (1) Support: For each edge $e \in E_k$, the support of the subgraph $S$ is $k - 2$; (2) No isolated node in $S$; (3) Any supergraph $T'$ of $T^d_k$ cannot meet the previous constraints.

Theorem 2.2. The objective function $\text{cov}(D)$ is monotonic and submodular.

Proof. Given two DBTruss sets $A_1, A_2 \subseteq T^d_k(q)$ and $A_1 \subset A_2$, we prove the theorem as follows. Monotonic. Since all nodes in $\text{cov}(A_1)$ must belong to $\text{cov}(A_2)$, we have $|\text{cov}(A_1)| \leq |\text{cov}(A_2)|$. Therefore, the function is monotonic. Submodular.

**Algorithm 1: TD Algorithm**

1. **Input:** $G$: a geo-social network, $q$: query node, $k$: support constraint, $d$: distance constraint, $r$: number constraint
2. **Output:** $D$: the top-$r$ DBTrusses
3. **Input:** $T^d_k(q)\leftarrow \emptyset, T^k_k$ connected $k$-truss containing $q$ in $G$
4. for each $v \in T^k_k$ do
5. for each $u \in T^k_k$ and $d(u, v) \leq d$ do
6. if $d(u, v) = d$ then
7. $O(c, d)\leftarrow$ the MCC of node set $\{u, v\}$
8. $T^d_k(q)\leftarrow T^d_k(q) \cup \{T^d_k\}$
9. else
10. for each $w \in T^k_k$ and $w \neq u, v$ do
11. $O(c, d_m)\leftarrow$ the circle determined by three nodes $u, v$ and $w$
12. if $d_m \leq d$ then
13. $T^d_k(q)\leftarrow T^d_k(q) \cup \{T^d_k\}$
14. for $i = 1$ to $r$ do
15. $T^*\leftarrow \arg\max_{T^d_k(q) \subseteq \text{cov}(T) \setminus \text{cov}(D)} |\text{cov}(T) \setminus \text{cov}(D)|$
16. $D\leftarrow D \cup \{T^*\}$
17. return $D$.

Given $T \in T^d_k(q) \setminus A_2$, we say the function $|\text{cov}(D)|$ is submodular if $|\text{cov}(A_1 \cup \{T\})| - |\text{cov}(A_1)| \geq |\text{cov}(A_2 \cup \{T\})| - |\text{cov}(A_2)|$. Note that, $\text{cov}(A_1) \subset \text{cov}(A_2)$ according to monotonicity. For simplicity, let $X = |\text{cov}(A_1 \cup \{T\})| - |\text{cov}(A_1)|$ and $Y = |\text{cov}(A_2 \cup \{T\})| - |\text{cov}(A_2)|$. The relationship between $\text{cov}(T)$, $\text{cov}(A_1)$, and $\text{cov}(A_2)$ there are four cases. Case 1: $\text{cov}(T) \cap \text{cov}(A_2) = \emptyset$. Case 2: $\text{cov}(T) \cap \text{cov}(A_2) \neq \emptyset$ and $\text{cov}(T) \cap \text{cov}(A_1) = \emptyset$. Case 3: $\text{cov}(T) \cap \text{cov}(A_1) \neq \emptyset$ and $\text{cov}(T) \cap \text{cov}(A_2) = \emptyset$. Case 4: $\text{cov}(T) \cap \text{cov}(A_1) \neq \emptyset$ and $\text{cov}(T) \cap \text{cov}(A_2) = \text{cov}(T) \cap \text{cov}(A_2)$. Then, we prove each case fit the submodular.

In the algorithm, we first initialize top-$r$ results set $D$ and DBTruss set $T^d_k(q)$ as empty, and compute the connected $k$-truss.
\( T_k \) that contains \( q \) in \( G \) (Line 1). Then, for each pair of nodes \( u, v \in T_k \) with distance \( d(u, v) \) no larger than \( d \), we compute all the DBTrusses in Lines 2-13. It has two cases: 1) if \( d(u, v) = d \), we obtain the MCC of node set \( \{u, v\} \) and put the connected \( k \)-truss contains \( q \) into \( O(c, d) \) to \( T^d_k(q) \) (Lines 4-7); 2) for each node \( w \in T_k \) where \( w \neq u, v \), we compute the circle \( O(c, d_m) \) determined by \( \{u, w\} \) and put the connected \( k \)-truss contains \( q \) in \( O(c, d_m) \) into \( T^d_k(q) \) (Lines 9-13). After that, we iteratively select a \( k \)-truss in \( T^d_k(q) \) with the largest number of nodes that are not in the current optimal result (Lines 14-16). Finally, we return the best result \( D \).

Though TD Algorithm can significantly reduce the computation cost, it still has some limitations: 1) TD Algorithm needs to verify \( O(n^3) \) MCC with time complexity \( O(n^{m+1}) \) in each iteration may be involved.

3.2 BD Algorithm

To deal with the above issues, in this section, we propose BD Algorithm to accelerate the computation based on novel structures. Before introducing the details, we first present some related properties as follows.

Lemma 3.1. Suppose there are circles that consist of two boundary nodes \( u, v \) with diameter \( d \). We can determine the number of these circles as follows: 1) if \( d(u, v) = d \), we can determine there is a unique circle; 2) if \( d(u, v) < d \), we can find two circles satisfying the constraint.

The correctness of Lemma 3.1 is easy to verify, and we omit the proof here. According to the lemma, we can obtain at most two circles with diameter \( d \) by a pair of nodes. We use \( R(u, D) \) to denote the set of DBTrusses that contains \( u \), i.e., \( R(u, D) = \{T_k|u \in T_k \cap T_k \in D\} \). For simplicity, we omit \( D \) when the context is self-evident. Intuitively, the fewer the number of common nodes in top-\( r \) DBTrusses \( D \) is, the number of nodes covered by \( D \) will be larger. Next, we introduce a new concept.

Definition 4 (Unique Set). Given a set of DBTrusses \( D_k \) and \( T^d_k \) in \( D_k \), the unique set of \( T^d_k \) denoted as \( U(T^d_k, D) \), is the set of nodes in \( T^d_k \) but not in any other DBTrusses.

We call each node \( v \in U(T^d_k, D) \) as a unique node of \( T^d_k \). Based on it, we define set \( T^d_k \) in \( D_k \) with the least unique nodes as the minimum cover truss of \( D_k \), denoted as \( T_{min}(D) \). For simplicity, we omit \( D \) when the context is self-evident.

BD Algorithm. To improve the efficiency, we turn to the method by determining circles with two boundary nodes and a diameter, i.e., using two nodes instead of three nodes. After that, we enumerate DBTrusses enclosed in circles. Then, the newly generated DBTruss will join the result set \( D \), if the size of \( D \) is less than \( r \). Otherwise, we use it to replace the minimum cover truss of \( D \) and update result set information. In addition, we build an effective structure to maintain the top-\( r \) results and key information. The details are shown in Algorithm 2.

In Algorithm 2, we first initialize \( D \) as empty and compute the connected \( k \)-truss \( T_k \) that contains \( q \) in \( G \) as candidate set (Line 1). Then, we obtain the circles and DBTrusses based on two boundary nodes and a diameter in Lines 2-8. In detail, for each pair of nodes \( u, v \in T_k \) with \( d(u, v) \leq d \), we enumerate all candidate circles with diameter \( d \) and preserve them in \( C_g(u, v) \) (Lines 2-5). After that, for each circle \( O(c, d) \) in \( C_g(u, v) \), we compute the connected \( k \)-truss for nodes enclosed in \( O(c, d) \) and use Maintain Procedure to determine if \( T^d_k \) can be added into diversified top-\( r \) results \( D \) (Lines 6-8). For the Maintain Procedure, it has two cases. For each newly enumerated DBTruss \( T^d_k \), when the size of top-\( r \) DBTrusses set is less than \( r \), it invokes Change Procedure to directly push \( T^d_k \) into \( D \) and update the index (Lines 11-13). Otherwise, we use \( T^d_k \) to replace the minimum cover truss of the current top-\( r \) result when it satisfies the inequality in Lines 14-16. The procedure first computes the value of unique set \( U(T^d_k, D) \) for updated \( D \), i.e., the size of \( X \) in Line 14. Then, if \( |X| > |U(T_{min}(D))| + \frac{|\text{cost}(D)|}{|D|} \), it invokes Change Procedure to remove \( T_{min}(D) \) from \( D \), push \( T^d_k \) into \( D \) and update the information (Lines 15-16). The inequality in Line 15 is proved with \( 1/4 \) approximate ratio. The algorithm terminates until all nodes in \( T_k \) are visited and finally return \( D \).

Change Procedure. To reduce time consumption, we use an index to maintain the information within Change Procedure, including the size of coverage, i.e., \( |\text{cost}(D)| \), the number of DB-Truss containing \( v \) for each \( v \in \text{cov}(D) \), i.e., \( |R(v, D)| \), and the unique set for each \( T \in D \), i.e., \( U(T, D) \). For simplicity, we use \( |R(o)| \) and \( U(T) \) instead. For Change Procedure, we first update \( D \) by removing the minimum cover truss \( T_{min}(D) \) and inserting \( T^d_k \). Then, we update information for nodes in \( T^d_k \) and \( T_{min}(D) \). We first update the value of \( |R(o)| \). Note that, the index needs to be considered only when the last or current value of \( |R(o)| \) equals 1, since unique nodes will only be changed in this circumstance. In detail, there are four cases as follows:

- If \( |R(o)| \) decreases from 1 to 0, it implies \( o \) will be removed from \( D \) because it is the unique node of \( T_{min}(D) \), i.e., \( o \notin \text{cov}(D) \setminus (T_{min}(D) \cup T^d_k) \). Thus, \( |\text{cost}(D)| \) decreases by 1.
- If \( |R(o)| \) increases from 0 to 1, it indicates \( o \) will join \( D \) because it is the unique node of \( T^d_k \), i.e., \( o \notin \text{cov}(D) \setminus T^d_k \). Thus, \( |U(T^d_k)| \) and \( |\text{cost}(D)| \) both increase by 1.
- If \( |R(o)| \) decreases from 2 to 1, it implies \( o \) becomes the unique node of \( T^d_k \), i.e., \( T^d_k \) is the only one that contains \( o \) by \( T_{min}(D) \) elimination. Thus, \( |U(T^d_k)| \) increases by 1.
- If \(|R(o)|\) increases from 1 to 2, it indicates \(o\) is not a unique node of \(T'\) anymore, where \(T'\) is the only one that contains \(o\) before inserting \(T_k\). Thus, \(\vert U(T') \vert \) decreases by 1.

The space complexity is linear to \(D\), i.e., \(O(\sum_{T \in \delta(D)} |T'_k|)\). Each step of \textsc{Maintain Procedure} and \textsc{Change Procedure} takes \(O(\vert T'_k \vert)\) time while the enumeration phase takes \(O(n^2m^{1.5})\). The total time complexity of \textsc{BD Algorithm} is \(O(n^2m^{1.5} + |TD|)\) with 1/4 approximate ratio as shown in Theorem 3.2.

**Theorem 3.2.** Let \(D'\) be the optimized solution and \(D\) be the answer of \textsc{BD Algorithm} for the diversified DBTruss search problem, we have \([\text{cov}(D)] \geq 1/4 \cdot [\text{cov}(D')]\).

Proof. We prove the theorem based on the theoretical result in [1]. Given a set \(D = \{T_1, T_2, \ldots\}\) and an integer \(r\), we set \(D_r = \{T_1, T_2, \ldots, T_r\}\) with \(1 \leq s \leq r\). For any \(s > r\), we construct \(D_r\) from \(D_{s-1}\) as follows. Suppose \(D_r = D_{s-1} \cup \{T_s\}\), if \([\text{cov}(D_s)] > (1 + 1/r)[\text{cov}(D_{s-1})]\), we have \(D_r = D_s\). Otherwise, \(D_r = D_{s-1}\). It can be guaranteed that \(D_r\) is a 1/4-approximation solution of the max-r-cover problem on \(\mathcal{H}_r\). Our problem has the same setting as the above problem. Therefore, we only need to prove \(\vert U(T, D') \vert > \vert U(T_{\text{min}}(D), D) \vert\) equals \([\text{cov}(D'_k)] > (1 + 1/r)[\text{cov}(D_{s-1})]\) when a new \(T_s\) joins. As shown in Figure 2, the relationship of \(D, D', T, T_{\text{min}}\) is illustrated by using 7 subsets, i.e., \(A, B, E, F, J, I, H\). We can observe that \([\text{cov}(D'_k)] > (1 + 1/r)[\text{cov}(D_{s-1})]\) is equivalent to \(|A| + |B| + |E| + |F| + |H| + |I| > (1 + 1/r)(|A| + |B| + |E| + |F| + |H| + |I|)\), i.e., \(|H| + |I| > |H| + |I| + 1/4 \cdot [\text{cov}(D)]\). Since \(D = r\), we have \(\vert U(T, D') \vert > \vert U(T_{\text{min}}(D), D) \vert + [\text{cov}(D')]/|D|\). The theorem holds. □

### 4 EXPERIMENTS

#### Algorithms

To best of our knowledge, there is no existing work for the diversified DBTruss search problem. In the experiments, we implement and evaluate the proposed two algorithms, i.e., TD algorithm and BD algorithm.

#### Datasets

We employ two real-world geo-social networks, i.e., Brightkite (197K) and Gowalla (456K) where the number after each dataset represents the corresponding number of edges. The details are public available on SNAP\(^1\). We vary \(d\) from 1.0 to 3.0 and \(k\) from 4 to 8. We set \(r = 5, d = 2.0\) and \(k = 6\) as the default value. All algorithms are implemented in standard C++.

#### Efficiency evaluation

To evaluate the efficiency, we report the response time of TD and BD by varying \(k\) and \(d\). The results are shown in Figure 3. In Figures 3(a) and 3(d), we can see BD significantly outperforms TD by a wide margin, which achieves up to 2 orders of magnitude speedup. Also, both of them decrease with increasing \(k\), because the size of communities becomes smaller with tighter support constraint. By varying \(d\) in Figures 3(b) and 3(e), we find that the response time of two methods both increase. This is because the size of result increases.

#### Effectiveness evaluation

Figures 3(c) and 3(f) show the number of nodes for two algorithms when varying \(d\). As observed, the number of nodes covered increases when \(d\) increasing because the size of community becomes larger for larger \(d\). Moreover, TD significantly outperforms BD due to better approximate ratio guarantee. Obviously, though TD can achieve better effectiveness, its time complexity is larger. Thus, users can make a trade-off between efficiency and effectiveness when selecting algorithms.

### 5 CONCLUSION

In this paper, we conduct the first attempt to study the diversified DBTruss search problem in geo-social networks. Due to the NP-hardness of the problem, two algorithms with approximation ratio guarantee are developed to strive for the trade-off between efficiency and effectiveness. Finally, experiments are conducted on real geo-social networks to demonstrate the effectiveness and efficiency of proposed model and strategies.

### ACKNOWLEDGMENTS

This work was supported by NSFC 61802345, ZJNSF LQ20F020007 and ZJNSF LY21F020012.

### REFERENCES


[6] Qing Liu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. Truss-based Community Search over Large Directed Graphs. In *SIGMOD.*


\(^1\)http://snap.stanford.edu/