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

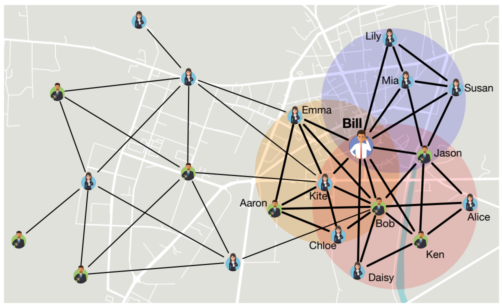


Figure 1: Motivation example

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 $e \in E_S$,*

*Corresponding author

$sup(e, S) \geq k-2$; 2) no isolated node in S; 3) any supergraph S' of S cannot meet the previous constraints.

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 P 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_k^d, if it satisfies the following constraints: (1) Support: For each $e \in E_S$, $sup(e, S) \geq k-2$; (2) Spatial closeness: The MCC diameter of V_S is no larger than d; (3) Connectivity: The subgraph S is connected; (4) Maximal: Any supergraph of S does not satisfy the first three constraints.*

Given a query node q, we use $\mathcal{T}_k^d(q)$ to denote the set of all DBTrusses that contain q. For a subset $D \subseteq \mathcal{T}_k^d(q)$, we use $cov(D)$ to denote the set of nodes covered by D, i.e., $cov(D) = \bigcup\{V_S | S \in D\}$. In real applications, the cardinality of $\mathcal{T}_k^d(q)$ could be very large, and it may not be easy for users to explore the results. An intuitive idea is to retrieve the top-r DBTrusses with the largest size. However, as discussed, these results may be highly overlapped and fail to reveal the properties of $\mathcal{T}_k^d(q)$. Therefore, we introduce the diversified DBTruss search problem as follows.
Problem statement. Given a geo-social graph G, three parameters k, d, and r, and a query node $q \in V$. Then, the problem of *diversified DBTruss search* aims to find a set D^* of r DBTrusses from $\mathcal{T}_k^d(q)$, which can maximize the number of nodes covered, i.e.,

$$D^* = \underset{D \subseteq \mathcal{T}_k^d(q) \wedge |D|=r}{\arg\max} \quad |cov(D)| \tag{1}$$

Theorem 2.1. *The diversified DBTruss search problem is NP-hard.*

Proof. To solve the problem, we have to enumerate all the DBTrusses of q, i.e, $\mathcal{T}_k^d(q)$. Then, we can reduce r-set cover problem [1] to the diversified DBTruss search problem by treating each $S \in \mathcal{T}_k^d(q)$ as a set. The r-set cover problem is to find r subsets from a collection of subsets such that the union of selected r subsets can cover as many elements as possible. The r-set cover problem is NP-hard. Therefore, our problem is also NP-hard. □

Theorem 2.2. *The objective function $|cov(D)|$ is monotonic and submodular.*

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

Algorithm 1: TD Algorithm

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

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

□

3 OUR SOLUTION

Naively, we can traverse all the combinations and return the optimal result. However, the approach is not affordable even on small graphs. To accelerate the processing, our framework consists of two main steps: 1) enumerate all DBTrusses $\mathcal{T}_k^d(q)$, and 2) select r DBTrusses that can maximize the number of covered nodes in the graph. Based on this framework, we propose the two effective solutions, i.e., TD Algorithm and BD Algorithm, in this paper.

3.1 TD Algorithm

Note that, three nodes can determine a unique circle, if they are on the boundary. A circle can also be determined by two nodes, when they are the endpoints of its diameter. Based on it, we propose the TD Algorithm. Specifically, it first enumerates all candidate circles and obtains all MCCs. Then, it greedily selects the best DBTruss with the largest marginal gain in each iteration. The algorithm terminates until r DBTrusses are found. The details are shown in Algorithm 1.

In the algorithm, we first initialize top-r results set D and DBTruss set $\mathcal{T}_k^d(q)$ 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 lager than d, we compute all the DBTruss in Lines 2-13. It has two cases: 1) if $d(u, v) = d$, we obtain the MCC $O(c, d)$ of node set $\{u, v\}$ and put the connected k-truss contains q into $O(c, d)$ into $\mathcal{T}_k^d(q)$ (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 $\{u, v, w\}$ and put the connected k-truss contains q in $O(c, d_m)$ into $\mathcal{T}_k^d(q)$ (Lines 9-13). After that, we iteratively select a k-truss in $\mathcal{T}_k^d(q)$ 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 $O(n^3 m^{1.5} + r \cdot \sum_{T_k^d \in \mathcal{T}_k^d(q)} |T_k^d|)$; 2) During the greedy procedure, redundant computation 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., $R(u, D) = \{T_i | u \in T_i \wedge T_i \in D\}$. 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 and $T_k^d \in D$, the unique set of T_k^d, denoted as $\mathcal{U}(T_k^d, D)$, is the set of nodes in T_k^d but not in any other DBTrusses.*

We call each node $v \in \mathcal{U}(T_k^d, D)$ as a unique node of T_k^d. Based on it, we define set $T_k^d \in D$ with the least unique nodes as the minimum cover truss of D, 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 $d(u, v) \le d$, we enumerate all candidate circles with diameter d and preserve them in $C_d(u, v)$ (Lines 2-5). After that, for each circle $O(c, d)$ in $C_d(u, v)$, we compute the connected k-truss for nodes enclosed in $O(c, d)$ and

Algorithm 2: BD ALGORITHM

```
Input   : G: a geo-social network, q: query node, k: support
          constraint, d: distance constraint, r: number constraint
Output  : D: the top-r DBTrusses
1  $D \leftarrow \varnothing$; $T_k \leftarrow$ connected $k$-truss containing $q$ in $G$;
2  for each $v \in T_k$ do
3      for each $u \in T_k$ do
4          if $u \neq v$ and $d(u, v) \le d$ then
5              $C_d(u, v) \leftarrow$ all circles with diameter $d$ and bounded
               by $u, v$;
6              for each $O(c, d) \in C_d(u, v)$ do
7                  $T_k^d \leftarrow$ connected $k$-truss contains $q$ enclosed
                   $O(c, d)$;
8                  MAINTAIN($T_k^d, D$);
9  return $D$;
10 procedure MAINTAIN($T_k^d, D$)
11 if $|D| < r$ then
12     CHANGE($\varnothing, T_k^d, D$);
13     return;
14 $X \leftarrow \{v \in T_k^d \mid |R(v)| = 0$ or $|R(v)| = 1 \wedge v \in T_{min}\}$;
15 if $|X| > |\mathcal{U}(T_{min}, D)| + \frac{|cov(D)|}{|D|}$ then
16     CHANGE($T_{min}, T_k^d, D$);
```

use MAINTAIN PROCEDURE to determine if T_k^d 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_k^d, if the size of top-r DBTrusses set is less than r, it invokes CHANGE PROCEDURE to directly push T_k^d into D and update the index (Lines 11-13). Otherwise, we use T_k^d to replace the minimum cover truss of the current top-r result when it satisfies the inequation in Lines 14-16. The procedure first computes the value of unique set $\mathcal{U}(T_k^d, D')$ for updated D', i.e., the size of X in Line 14. Then, if $|X| > |\mathcal{U}(T_{min}, D)| + \frac{|cov(D)|}{|D|}$, it invokes CHANGE PROCEDURE to remove T_{min} from D, push T_k^d into D and update the information (Lines 15-16). The inequation 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., $|cov(D)|$, the number of DBTruss containing v for each $v \in cov(D)$, i.e., $|R(v, D)|$, and the unique set for each $T \in D$, i.e., $\mathcal{U}(T, D)$. For simplicity, we use $|R(v)|$ and $\mathcal{U}(T)$ instead. For CHANGE PROCEDURE, we first update D by removing the minimum cover truss T_{min} and inserting T_k^d. Then, we update information for nodes in $T_k^d \cup T_{min}$. We first update the value of $|R(v)|$. Note that, the index needs to be considered only when the last or current value of $|R(v)|$ equals 1, since unique nodes will only be changed in this circumstance. In detail, there are four cases as follows:

- If $|R(v)|$ decreases from 1 to 0, it implies v will be removed from D because it is the unique node of T_{min}, i.e., $v \notin cov(D \backslash \{T_{min}\} \cup \{T_k^d\})$. Thus, $|cov(D)|$ decreases by 1.
- If $|R(v)|$ increases from 0 to 1, it indicates v will join D because it is the unique node of T_k^d, i.e., $v \notin cov(D \backslash \{T_k^d\})$. Thus, $|\mathcal{U}(T_k^d)|$ and $|cov(D)|$ both increase by 1.
- If $|R(v)|$ decreases from 2 to 1, it implies v becomes the unique node of T', where T' is the only one that contains v by T_{min} elimination. Thus, $|\mathcal{U}(T')|$ increases by 1.

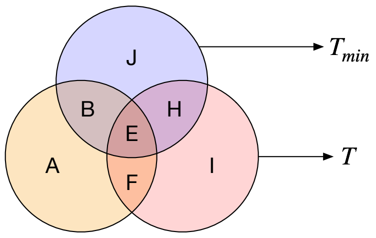


Figure 2: Illustration of Theorem 3.2

- If $|R(v)|$ increases from 1 to 2, it indicates v is not a unique node of T' anymore, where T' is the only one that contains v before inserting T_k^d. Thus, $|\mathcal{U}(T')|$ decreases by 1.

The space complexity is linear to D, i.e., $O(\sum_{T_k^d \in D} |T_k^d|)$. Each step of Maintain Procedure and Change Procedure takes $O(|T_k^d|)$ time while the enumeration phase takes $O(n^2 m^{1.5})$. The total time complexity of BD Algorithm is $O(n^2 m^{1.5} + |T_k^d|)$ 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 BD Algorithm for the diversified DBTruss search problem, we have $|cov(D)| \geq 1/4 \cdot |cov(D^*)|$.*

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

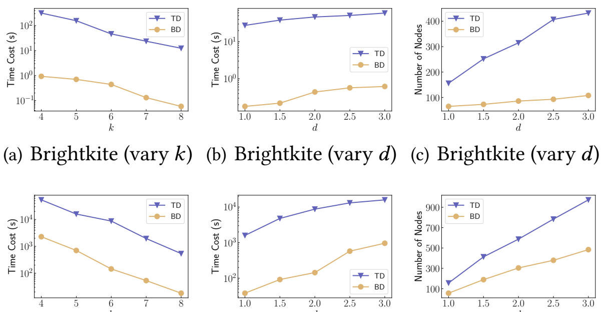
4 EXPERIMENTS

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.

[1]http://snap.stanford.edu/



(a) Brightkite (vary k) (b) Brightkite (vary d) (c) Brightkite (vary d)

(d) Gowalla (vary k) (e) Gowalla (vary d) (f) Gowalla (vary d)

Figure 3: Experiment results

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

[1] Giorgio Ausiello, Nicolas Boria, Aristotelis Giannakos, Giorgio Lucarelli, and Vangelis Th Paschos. 2011. Online maximum k-coverage. In *FCT*.
[2] Chen Chen, Yanping Wu, Renjie Sun, and Xiaoyang Wang. 2021. Maximum signed θ-clique identification in large signed graphs. *TKDE* (2021).
[3] Chen Chen, Qiuyu Zhu, Renjie Sun, Xiaoyang Wang, and Yanping Wu. 2021. Edge Manipulation Approaches for K-core Minimization: Metrics and Analytics. *TKDE* (2021).
[4] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, Jeffrey Xu Yu, and Jianxin Li. 2020. Finding Effective Geo-social Group for Impromptu Activities with Diverse Demands. In *KDD*.
[5] Yixiang Fang, Reynold Cheng, Xiaodong Li, Siqiang Luo, and Jiafeng Hu. 2017. Effective community search over large spatial graphs. *PVLDB* (2017).
[6] Qing Liu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. Truss-based Community Search over Large Directed Graphs. In *SIGMOD*.
[7] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *KDD*.
[8] Kai Wang, Xin Cao, Xuemin Lin, Wenjie Zhang, and Lu Qin. 2018. Efficient computing of radius-bounded k-cores. In *ICDE*.
[9] Xiaoyang Wang, Ying Zhang, Wenjie Zhang, Xuemin Lin, and Muhammad Aamir Cheema. 2015. Optimal spatial dominance: an effective search of nearest neighbor candidates. In *SIGMOD*.
[10] Yanping Wu, Jun Zhao, Renjie Sun, Chen Chen, and Xiaoyang Wang. 2020. Efficient Personalized Influential Community Search in Large Networks. In *APWeb-WAIM*.
[11] Qijun Zhu, Haibo Hu, Cheng Xu, Jianliang Xu, and Wang-Chien Lee. 2017. Geo-social group queries with minimum acquaintance constraints. *VLDB Journal* (2017).